RC 70 COMPUTER

MAINTENANCE MANUAL

RC REDCOR

CORPORATION

Volume 1 $10.00

RC 70 COMPUTER

MAINTENANCE MANUAL

January 1970

R
C REDCOR
CORPORATION

7800 Deering Avenue, P.O. Box 1031, Canoga Park, California 91304

No. M-7001A

RELATED PUBLICATIONS

Title	Publication No.
RC 70 Computer, Reference Manual	M-5000
RC 70 Computer, Operator's Manual	M-5001
RC 70 Assembler, Reference Manual	M-5002
RC 70 FORTRAN IV, Reference Manual	M-5003
RC 70 Computer, Interface Design Manual	M-7000

CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

APPENDIX

ILLUSTRATIONS

ILLUSTRATIONS (Cont.)

TABLES

TABLES (Cont.)

SECTION I
GENERAL DESCRIPTION

1.1 INTRODUCTION

This manual contains information for installing, operating, and servicing the RC 70 Digital Computer manufactured by REDCOR Corporation, Canoga Park, California. The manual consists of two separate volumes, with volume I containing the information required for normal servicing and maintenance, and volume II containing supporting engineering data for detailed troubleshooting.

Several other publications containing information pertinent to the operation and programming of the RC 70 Computer are also available. The titles and publication numbers of these manuals are listed at the front of this manual on the reverse side of the title page.

1.2 PHYSICAL DESCRIPTION

The RC 70 Computer standard configuration consists of two major assemblies: a processor cabinet assembly and a modified Teletype Model 33 Automatic Send/Receive (ASR) Set. The Teletype connects to either of two identical I/O connectors at the rear of the processor assembly.

Two standard versions of the processor assembly are available, one for rack mounting in a standard 19-inch rack and one for mounting in a desk-type enclosure. Both versions are essentially identical; the principal difference between the two versions is the way in which the control panel is mounted. In the rack-mounted version the control panel is installed in a hinged door attached to the front of the processor assembly. In the desk-mounted version the control panel is installed in an enclosure that attaches to the top of the desk. Since the differences between both versions are relatively minor, only the rack-mounted version of the processor assembly is illustrated and described in detail in this section.

1.3 PROCESSOR ASSEMBLY

The processor assembly consists of Front Panel Assembly 402114, Card Cage Assembly 400400-3, and Rear Cover Assembly 400117-1. The front panel assembly is hinge-mounted to the front of the card cage assembly as shown in figure 1-1. The front panel swings open for access to the wiring on the back of the control panel and to the backplane wiring of the card cage assembly. A cable connects the control panel of the front panel assembly to plug-in card assemblies located in slots J25 and J51 of the card cage assembly. The rear cover assembly normally attaches to the rear of the card cage assembly to protect the exposed portions of the memory card assemblies and the exposed terminals of the processor power supply. Figure 1-3 shows the rear of the card cage assembly with the rear cover assembly removed.

1.4 Front Panel Assembly

Located on the front panel assembly is the control panel which contains switches for controlling internal processor operations, and indicators and test jacks for displaying and monitoring internal processor operations. Figure 1-2 shows both the front and rear views of the control panel. As shown in figure 1-2, the panel is divided into two sections: an upper section containing most of the controls and indicators of interest to the operator or programmer and a lower section containing switches and test jacks that are used mainly for maintenance. The lower section is protected by a hinged cover that opens downward for access to the switches and test jacks.

The use and function of each control and indicator on the control panel is described in section II of this manual.

1.5 Card Cage Assembly

The card cage assembly houses the plug-in printed circuit card assemblies containing the memory and logic circuits of the processor; the power supply that provides operating power to the card assemblies and the control panel; and the blowers that provide cooling air for the memory card assemblies and the power supply.

The card assembly connectors (104 individual 40-pin connectors in all) are mounted on a connector plate at the front of the card cage assembly (figure 1-1). Access to the connector wiring is through the hinged front panel assembly. The 40-pin connectors are grouped in pairs, with each pair wired and designated as one connector. The lower row of connector pairs is designated J1 through J26; the upper row, J27 through J52. Each plug-in card assembly, except the memory card assemblies, inserts into one connector pair; the memory card assemblies require two connector pairs per card.

The plug-in card assemblies are inserted into the card cage assembly from the rear as shown in figure 1-3. The memory card assemblies occupy the four left-most card slots as viewed from the rear (J1 through J4, J27 through J30). Each memory card assembly provides 4096 (4K) words of core storage; hence, four memory

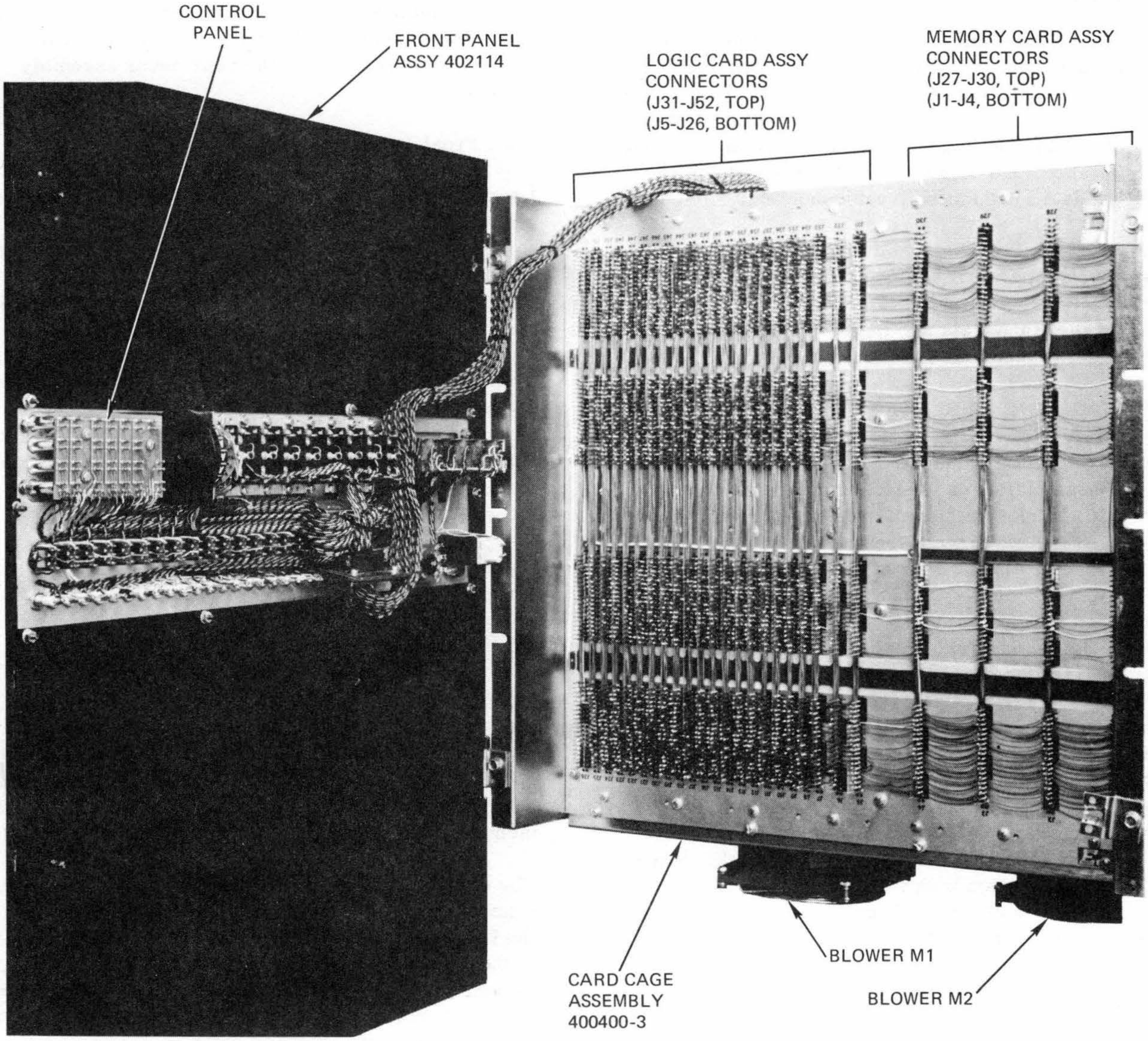
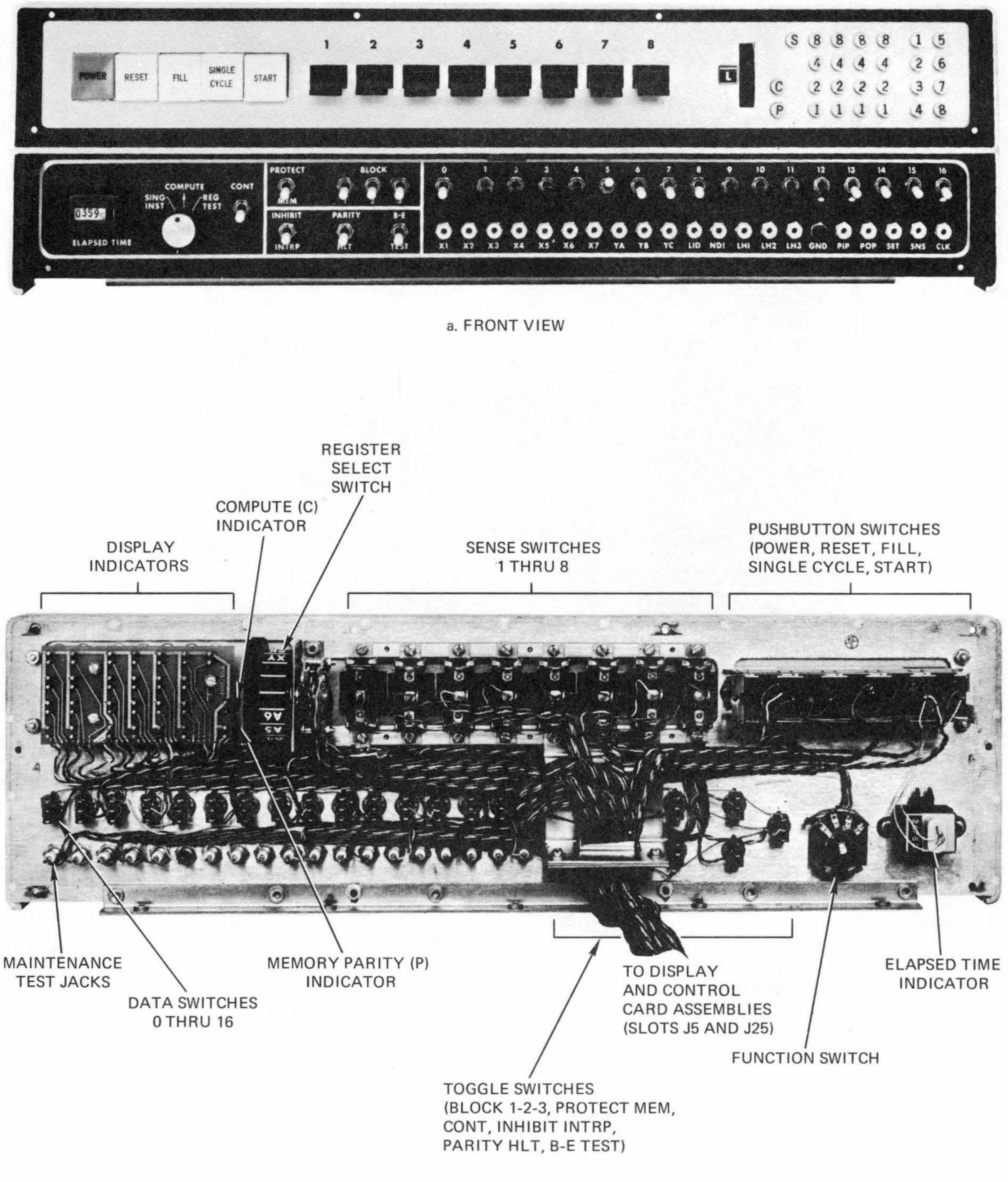

CONTROL PANEL
FRONT PANEL ASSY 402114
LOGIC CARD ASSY CONNECTORS (J31-J52, TOP) (J5-J26, BOTTOM)
MEMORY CARD ASSY CONNECTORS (J27-J30, TOP) (J1-J4, BOTTOM)
CARD CAGE ASSEMBLY 400400-3
BLOWER M1
BLOWER M2

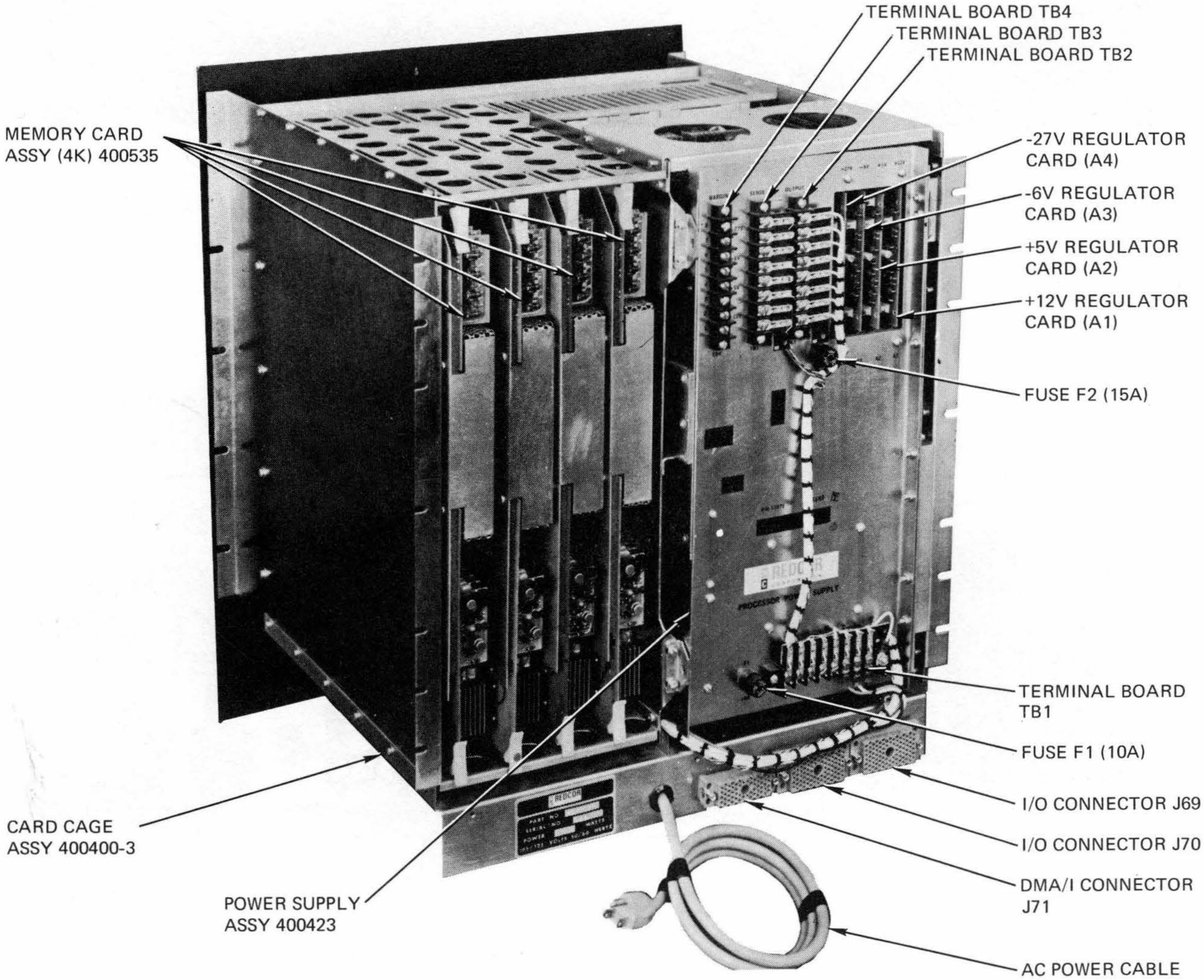
Figure 1-1. Processor Assembly, Front View with Front Panel Assembly Open



a. FRONT VIEW

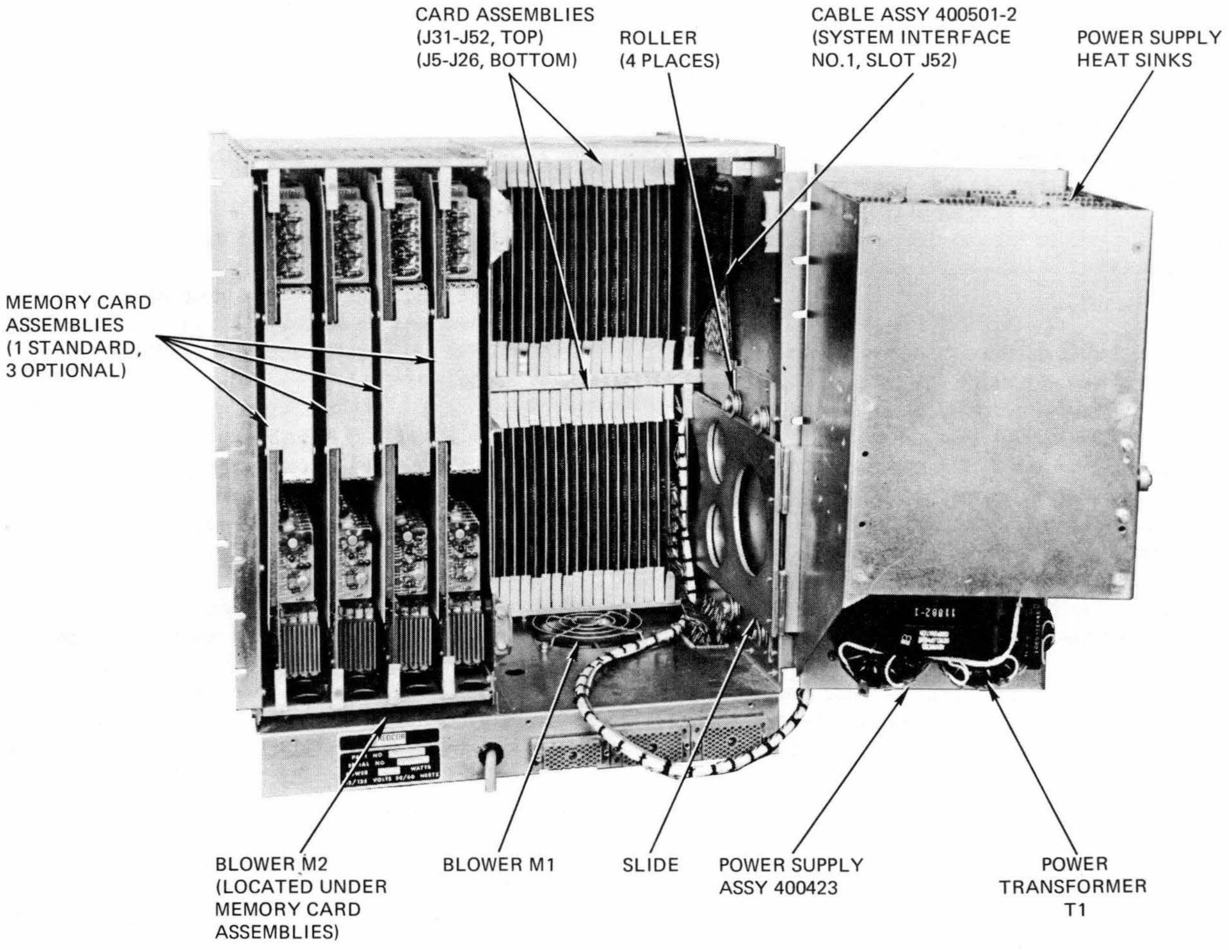
b. REAR VIEW

Figure 1-2. Control Panel



a. POWER SUPPLY SWUNG IN

Figure 1-3. Processor Assembly, Rear View (Sheet 1 of 2)



b. POWER SUPPLY SWUNG OUT

Figure 1-3. Processor Assembly, Rear View (Sheet 2 of 2)

card assemblies provide 16K words of core storage. The standard RC 70 processor contains two memory card assemblies for a memory capacity of 8K words. These card assemblies are normally installed in slots J4/J30 and J3/J29.

The logic card assemblies occupy two rows of card slots (J5 through J26 and J31 through J52) in front of the processor power supply. These card slots are accessible only when the power supply is swung out as shown in figure 1-3b. A list of the card assemblies and the locations they occupy in the card cage assembly is given in table 1-1.

The processor power supply occupies the right rear portion of the card cage assembly as shown in figure 1-3. It is hinged on the right side (as viewed from the rear) to a slide that moves between four rollers attached to the side of the card cage assembly. The power supply swings out on its hinge for access to the logic card assemblies. It is removed from the card cage assembly by sliding backwards until the slide disengages from the four rollers.

Power is distributed from the power supply to the rest of the processor through a cable that attaches to a card assembly in slot J26 of the card cage assembly. Primary ac power is brought into the supply by three wires connected to pins 1, 2, and 3 of terminal board TB1. The other ends of these wires connect to pins 1, 2, and 3 of terminal board TB2 located on the underside of the card cage assembly (figure 1-4).

The power supply itself contains rectifier and regulator circuits for developing the -27V, -6V, +5V, and +12V used in the processor. The regulator circuits are located on four small printed circuit cards that plug into slots A1 through A4 at the rear of the supply (figure 1-3a). Each card contains a potentiometer for adjusting the dc voltage regulated by that card. Also accessible at the rear of the power supply are fuses F1 and F2, and terminal boards TB1 through TB4.

Three connectors are provided at the rear of the card cage assembly (figure 1-3a) for connecting external equipment to the processor. Two of these (J69 and J70) are identical 90-pin connectors; the third (J71) is a 56-pin connector. The ASR-33 Teletype can be connected to either of the 90-pin connectors. The other is then available for connecting additional I/O devices. Connector J71 is used with the optional direct memory access and priority interrupt features of the computer.

Cooling air for the card assemblies and processor power supply is provided by blowers M1 and M2 located on the underside of the card cage assembly (figure 1-4). Blower M1 forces air up through the logic card assemblies and the power supply; blower M2 forces air up through the memory card assemblies. The blower motors receive ac power through pins 1 and 2 (M2) and pins 4 and 5 (M1) of terminal board TB2. The ac power cable that brings 115V ac power into the processor connects to pins 1, 2, and 3 of terminal board TB2.

1.6 ASR-33 TELETYPE

The ASR-33 Teletype (figure 4-6) is the standard I/O unit for the computer. It contains an input keyboard, an output printer, a punched paper tape reader, and a paper tape punch. It is modified for use with the RC 70 Computer by the addition of a controller card assembly mounted in the pedestal (figure 4-5). A ribbon cable from the controller card connects the Teletype to either I/O connector J69 or J70 at the rear of the processor.

This manual contains neither operating nor maintenance information for the Teletype unit. For information on the Teletype unit, consult the technical manuals provided by the manufacturer.

1.7 FUNCTIONAL DESCRIPTION

A simplified functional block diagram of the RC 70 Computer is shown in figure 1-5. The two main functional units of the computer are the ASR-33 Teletype and the processor.

Information is entered into the processor either through the Teletype keyboard or paper tape reader. Output data from the processor can be printed out on the Teletype printer or can be punched out on paper tape by the Teletype paper tape punch.

Other I/O devices can also be used with the processor. Slow-speed devices are typically connected to the basic I/O channel, which handles data transfers on a single word basis. High-speed devices are normally connected to the processor through optional direct memory access (DMA) channels, which handle single or multiple block transfers of data words. Up to four DMA channels are optionally available.

The basic I/O section of the processor features one external interrupt line. This basic interrupt capability can be expanded through the addition of priority interrupt options. Up to 32 levels of priority interrupts are optionally available in blocks of 8 levels each.

The processor can generally be thought of in terms of six functional sections: the control panel and control logic, the I/O and arithmetic/logical sections, core memory, and the power supply.

The control panel provides the facility for manually controlling internal operations of the processor and for displaying information of interest to the operator, programmer, or maintenance personnel.

The processor control logic contains circuits for stepping the processor through a stored program in an orderly

Table 1-1. Plug-In Card Assembly Complement

Card Type	Description	Assembly No.	Quantity	Location
453	Card Assembly (Power Supply Connection)	400453	1	J26
501	Cable Assembly (System Interface No. 1)	400501-2	1	J52
502	Cable Assembly (Display and Control)	400502	1	J25
503	Cable Assembly (System Interface No. 2)	402107	1	J24
504	Card Assembly (B1)	400504	1	J23
505	Card Assembly (B2)	400505	1	J22
506	Card Assembly (B3)	400506	1	J21
507	Card Assembly (C4)	400507	1	J45
508	Card Assembly (C5)	400508	1	J19
509	Card Assembly (Register)	400509	8	J7, J9, J10, J12, J13, J15, J16, J18
510	Card Assembly (Adder)	400510	2	J11, J17
511	Card Assembly (M-Register and Parity)	400511	1	J6
512	Card Assembly (Diode Memory)	400512	1	J5
514	Cable Assembly (Display and Control)	400514	1	J51
515	Card Assembly (C1)	400515	1	J50
516	Card Assembly (C2)	400516	1	J49
519	Card Assembly (C3)	400519	1	J46
520	Card Assembly (C6)	400520	1	J20
521	Card Assembly (C7)	400521	1	J44
523	Card Assembly (C8)	400523	1	J43
524	Card Assembly (C9)	400524	1	J41
525	Card Assembly (C10)	400525	1	J40
526	Card Assembly (C11)	400526	1	J39
527	Card Assembly (C12)	400527	1	J38
528	Card Assembly (C13)	400528	1	J37
530	Card Assembly (C14)	400530	1	J35
532	Card Assembly (C15)	400532	1	J33
533	Card Assembly (Computer Clock)	400533	1	J32
534	Card Assembly (Memory Timing)	400534	1	J31
535	Card Assembly (4K Memory)	400535	4	J1/J27*, J2/J28*, J3/J29, J4/J30
542	Card Assembly (Adder)	400542	2	J8, J14
545	Card Assembly (F1, G/J Register)	400545	2	J34, J36
546	Card Assembly (Multiply Control)	400546	1	J42
547	Card Assembly (F3, Hardware Index)	400547	1	J48
548	Card Assembly (F4, Logical Or)	400548	1	J47

*Optional

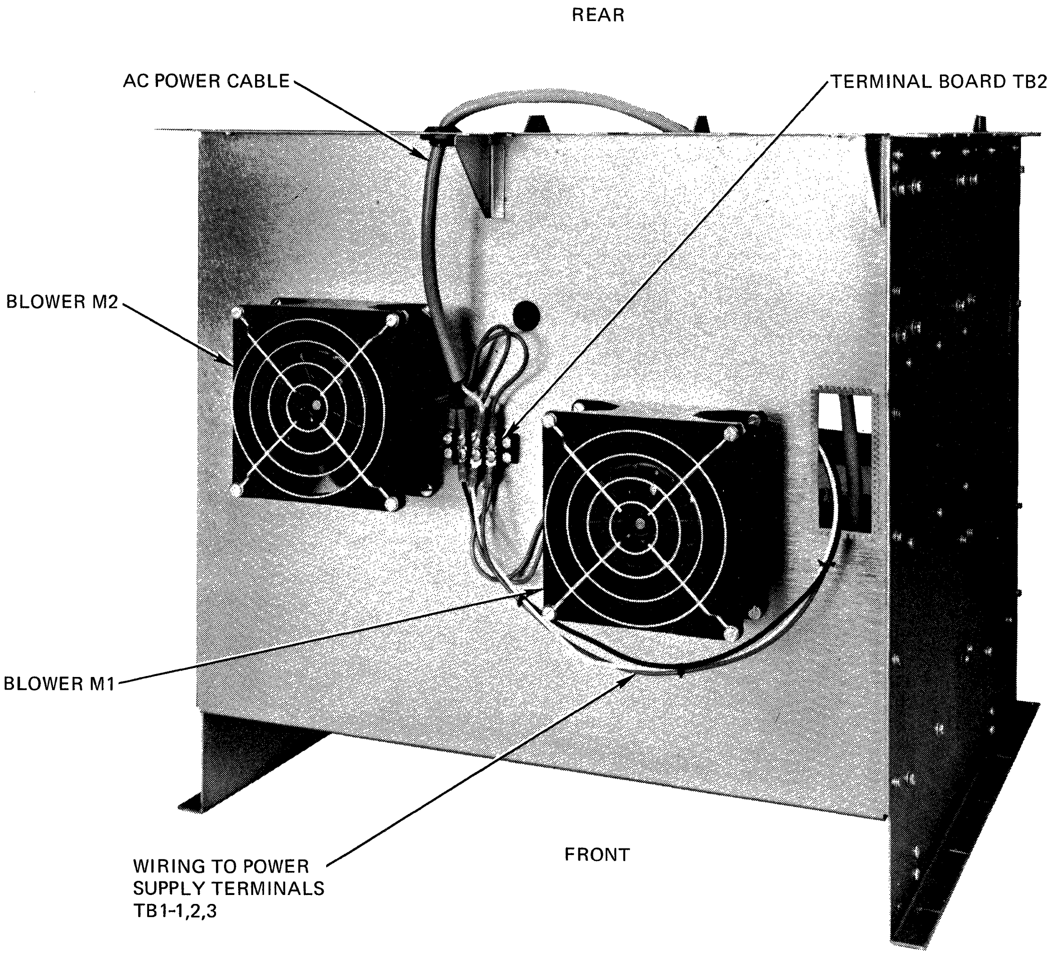


Figure 1-4. Card Cage Assembly, Bottom View

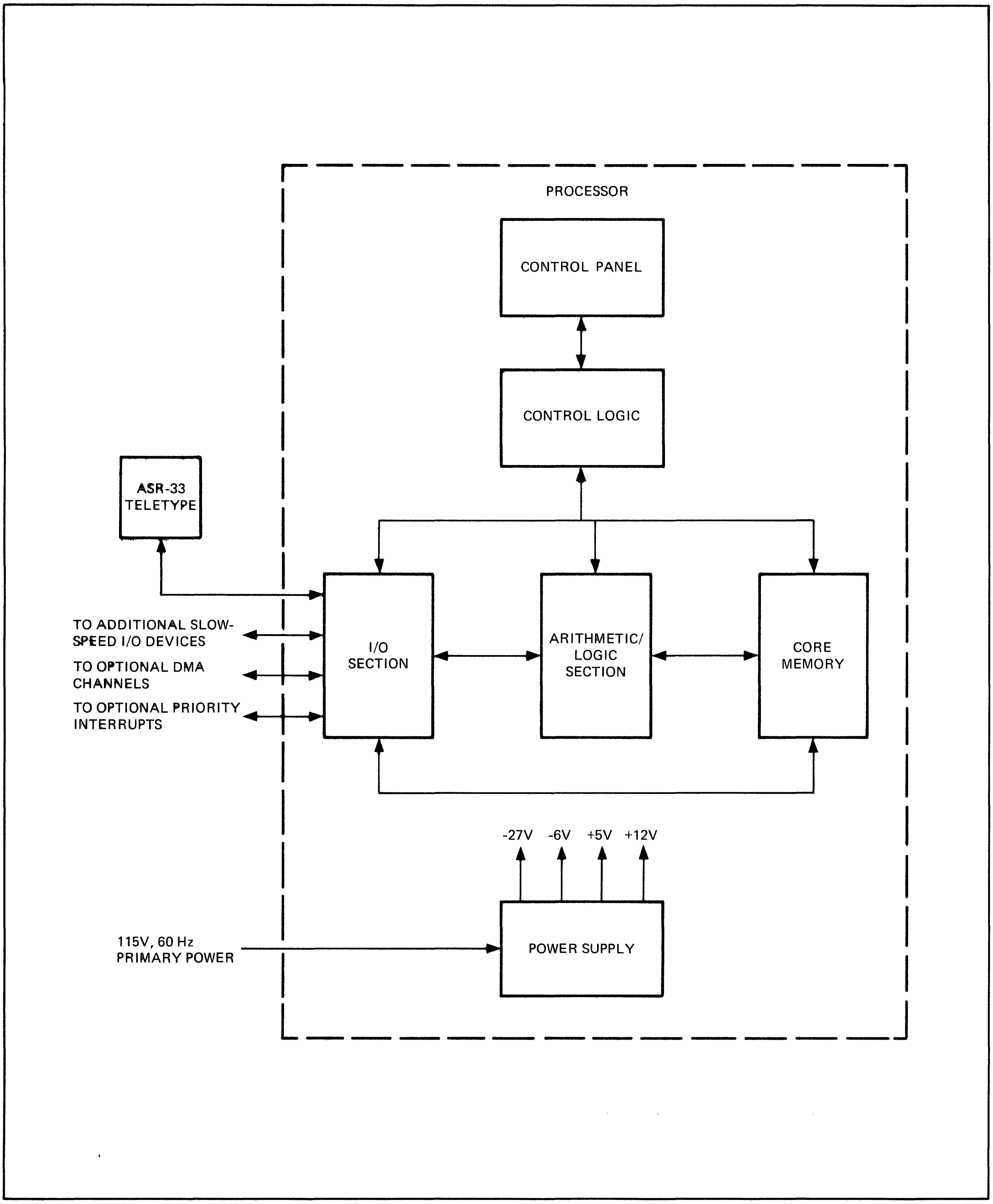


Figure 1-5. RC 70 Computer, Functional Diagram

sequence. It also contains circuits for decoding program instructions and generating the timing and control signals used to transfer data through the processor and to arithmetically or logically manipulate data according to the instructions of the stored program.

All data transfer operations between the processor and external I/O devices are performed through the I/O section. The basic I/O capability of the processor provides for discrete control and sensing of external devices, and for single word data transfers. Through the addition of DMA channel and priority interrupt options the basic I/O capability can be expanded to provide block transfers of data and true priority interrupts.

The arithmetic/logical section of the processor contains the adder circuits which perform all arithmetic (add, subtract, multiply, divide), logical (And, Inclusive Or, Exclusive Or, Compare), and shifting operations.

Core memory contains the magnetic core storage circuits for storing 16-bit (plus 1 parity bit) instruction, address, and data words. The standard core memory consists of 8K words of storage. Up to 8K words of additional storage, for a total of 16K words, can be accommodated internally in the processor. An additional 16K words of external storage is optionally available in 4K increments. Standard features of the memory include memory parity checking and memory write protection.

The processor power supply provides regulated -27V, -6V, +5V, and +12V power to the operating circuits of the processor.

1.8 SPECIFICATIONS AND MODEL DESIGNATIONS

Table 1-2 lists the general specifications for the RC 70 Computer. Table 1-3 lists the model designations for both the standard equipment configuration and optional equipment items.

Table 1-2. General Specifications

Characteristic	Specification
Memory	Random access magnetic core; 860 ns full cycle time; 16-bit word length plus parity bit; 4096 words minimum, expandable to 32,768 words in blocks of 4096 words; write protection standard
Memory addressing	Direct, indirect, indexed, relative to page register, relative to next instruction address
Arithmetic	Parallel, binary, fixed point, two's complement
Instructions	Single word and double word: 5 load/store, 8 arithmetic/logical, 12 branch, 5 shift, 5 input/output and control
Registers	Nine hardware, two memory: lower accumulator, 16-bit flip-flop; instruction register, 16-bit flip-flop; memory address register, 16-bit flip-flop; next-instruction address register, 16-bit flip-flop; page register, 6-bit flip-flop; index register, 16-bit flip-flop; D-, G-, and J-registers, 16-bit flip-flop; roll-arrow register, memory location X'0007'; upper accumulator, memory location X'0008'
Input/output	Discrete control and sensing; single word to and from lower accumulator; direct memory access (cycle steal); external interrupt
Logic signal levels	Binary or logical 1, +5V; binary or logical 0, 0V
Processor power supply voltages	-27V, 7.5A, regulated; -6V, 0.75A, regulated; +5V, 12.0A, regulated; +12V, 1.75A, regulated
Size (processor)	19 inches high, 19 inches wide, 19 inches deep
Weight (processor)	90 lb including power supply
Environment	0^{o} to 55^{o}C (32^{o} to 131^{o}F), 0 to 90% relative humidity
Input power	115 (±10)V, 47 to 63 Hz, 400W

Table 1-3. Equipment Model Descriptions

Model No.	Description
REDCOR 70	General purpose digital computer with 8192 words of memory, memory parity, hardware multiply/divide, hardware index register, hardware bootstrap loader, internal interrupt, memory write protection, eight sense switches, real-time clock, automatic power shutdown and restart, roll-arrow register, direct memory access, operator/maintenance panel, ASR-33 Teletype and controller, and 19-inch rack
RC 70-XX SERIES	
70-00	General purpose digital computer with 4096 words of memory, memory parity, hardware multiply/divide, hardware index register, hardware bootstrap loader, internal interrupt, memory write protection, eight sense switches, real-time clock, automatic power shutdown and restart, roll-arrow register, direct memory access, and operator/maintenance panel
70-03	4096-word memory module with parity
70-09	Memory expansion/peripheral controller chassis and power supply
70-17	Interval timer
70-70	Parallel input channel, 16 bits, contact closures or logic levels
70-71	Four parallel input channels, 16 bits each, contact closures or logic levels
70-72	Parallel input register, 16 bits
70-73	Four parallel input registers, 16 bits each
70-74	Sixteen output control pulses
70-75	Sixty-four output control pulses
70-76	Parallel output register, 16 bits
70-77	Four parallel output registers, 16 bits each
70-96	One DMA block transfer channel
70-97	Eight priority interrupt levels

SECTION II
OPERATING AND PROGRAMMING

2.1 GENERAL

This section contains operating and programming information considered necessary for normal servicing of the computer. In some instances, references are made to other publications for pertinent information instead of duplicating the information in this manual.

2.2 CONTROLS AND INDICATORS

The switches and indicators that are used for controlling and monitoring the internal operations of the processor are located on the processor control panel. The placement of each switch and indicator on the panel is shown in figure 1-2. A brief description of each control and indicator, along with a summary of its functions, is given in table 2-1. The controls and indicators located on the ASR-33 Teletype are not described in this manual. For a description of these controls and indicators consult the manufacturer's technical manuals.

2.3 OPERATING PROCEDURES

2.4 APPLYING POWER

The following procedure is used to apply power to the processor and set up initial operating conditions:

a. Verify that processor power cable is connected to 115V, 60 Hz electrical service.

b. Lower the cover of the lower portion of the processor control panel for access to the switches.

c. Set the function switch to COMPUTE.

d. Set the CONT and B-E TEST switches to the lower position.

e. If an area of memory is to be protected, set the PROTECT MEM and BLOCK switches as required (table 2-1); otherwise, set them to the lower position.

f. Set the INHIBIT INTRP and PARITY HLT switches for the required operation.

g. Set data switches 12, 13, and 16 to the upper position; set all other data switches to the lower position.

Note

> If a program is already stored in the computer, set the data switches to the starting address of the program.

h. Press the POWER switch. The POWER indicator should light to indicate that ac power has been applied.

i. Press the RESET switch. The address set into the data switches is loaded into the memory address register, and the page register is set to X'01'.

With the completion of step i the computer is in idle mode ready for operation. If a program is stored in memory, it can be executed by pressing the START switch. If there is no program in memory, one can be loaded as described in the following paragraph.

2.5 LOADING A PROGRAM

Loading a new program into the computer is initiated by a 19-instruction fill routine permanently stored in a read-only diode memory inside the processor. When the fill routine is executed, it loads a string of instructions, one byte at a time, from the input device into low-order memory (locations X'0028' through X'004F'). The storage location of the first word in the string is selected so that the last word loaded by the fill routine is always stored in location X'004F'. Since locations below X'0028' are reserved for special uses, the maximum number of words that can be loaded by the fill routine is 41 (82 bytes). The instruction string read into the computer by the fill routine is normally a bootstrap loader that is used to load the rest of the program into memory.

The input device from which the program is loaded must have X'12' as its I/O address. This address is fixed in the fill routine and cannot be changed. The string of instructions loaded by the fill routine must be headed by one byte containing the one's complement of the number of words to be loaded. This byte is used by the fill routine to determine the storage location of the first word. Normally, the last word stored (location X'004F') is a checksum used to verify correct program loading. When the last word has been loaded by the fill routine, program control is transferred to location X'004E'. This location normally contains a branch instruction to transfer control to the starting address of the bootstrap loader.

The procedure for loading the diagnostic program into the computer is given in paragraph 4.7m. For other program loading procedures consult REDCOR publication No. M-5001.

Table 2-1. Processor Controls and Indicators

Control or Indicator	Type	Function
POWER	Push-on, push-off switch; back-lighted indicator	Applies 115V ac power to power transformer T1 of processor power supply. Indicator lights when power is applied
RESET	Momentary pushbutton switch	Places processor in idle mode and sets up initial operating conditions: a. Processor enters idle mode at end of current memory cycle b. Contents of data switches 1 through 16 are loaded into L-register c. Contents of P-register are set to X'01' All processor controls and indicators are operative in idle mode. Processor remains in idle mode until START switch is pressed or an interrupt becomes active
FILL	Momentary pushbutton switch	Activates execution of 19-instruction fill routine from diode memory. Fill routine loads core memory from I/O device with address X'12'. Loading continues until program is loaded or until RESET or SINGLE CYCLE switch is pressed. FILL switch is operative only in idle mode
SINGLE CYCLE	Push-on, push-off switch; back-lighted indicator	Enables START switch to step processor through stored program one instruction at a time. Pressing START switch after pressing SINGLE CYCLE switch causes processor to leave idle mode, execute one complete instruction, and return to idle mode again. Pressing SINGLE CYCLE switch while processor is in compute mode returns processor to idle mode after completely executing instruction currently in progress. Indicator lights to indicate that single cycle operation is in effect
START	Momentary pushbutton switch	Controls instruction execution. Pressing START switch causes processor to leave idle mode, enter compute mode, and begin executing stored program. If single cycle operation is not in effect, processor continuously executes instructions until Halt instruction is executed, or RESET or SINGLE CYCLE switch is pressed. If single cycle operation is in effect, processor executes one complete instruction and returns to idle mode
Sense switches 1 through 8	Two-position toggle switches	Used for external control of stored program. Switches are on in upper position and off in lower position. Status of each switch can be tested and copied into KU indicator by Sense instruction with appropriate address. Either Branch Equal or Branch Unequal

Table 2-1. Processor Controls and Indicators (Cont.)

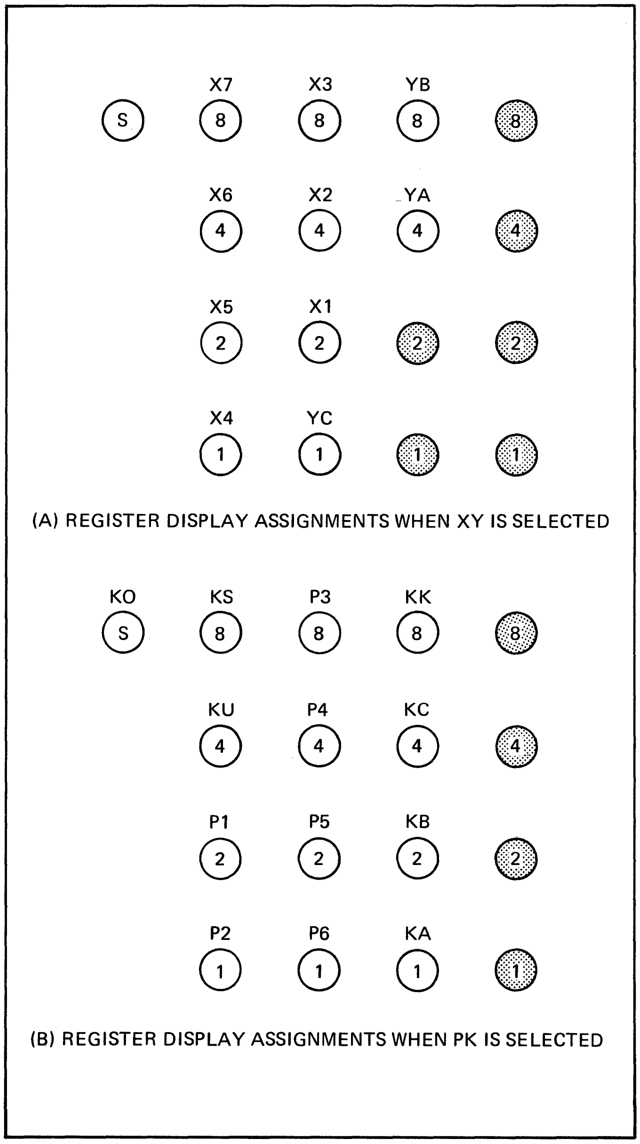
Control or Indicator	Type	Function
Sense switches 1 through 8 (Cont.)		instruction can then be used to test indicator to determine switch position. Hexadecimal addresses of sense switches are: Switch — Hex Address 1 — 01 2 — 02 3 — 03 4 — 04 5 — 05 6 — 06 7 — 07 8 — 00
Register select	15-position thumbwheel switch	Selects certain processor registers, indicators, or memory locations for display by register display indicators. It is also used with REG TEST position of function switch to enter data from 16 data switches into selected register. Information selected for display by this switch is meaningful only in idle mode. The switch positions and data selected for display by each position are summarized as follows: Position — Data Displayed XY — States of minor timing flip-flops X1 through X7 and major timing flip-flops YA through YC. Figure 2-1 shows display assignments for XY PK — Contents of P-register; states of flip-flops KA, KB, KC, KK, KO, KS, and KU. Figure 2-1 shows display assignments for PK D — Contents of D-register L — Contents of L-register I — Contents of I-register R — Contents of memory location X'0007' X — Contents of X-register B — Contents of B-register N — Contents of N-register A1-A6 — Contents of memory locations X'0001' through X'0006'

Table 2-1. Processor Controls and Indicators (Cont.)

Control or Indicator	Type	Function
C (Compute)	Indicator	Lights when processor is operating in compute mode
P (Memory parity)	Indicator	Lights when memory parity error is detected
Register display S-8-4-2-1	17 indicators	Display contents of register, flip-flops, or memory location selected by register select switch. Each column of indicators represents four binary digits (one hexadecimal digit). Information displayed is valid only in idle mode
Programmable display 1 through 8	8 indicators	Display eight most significant bits (B1 through B8) of B-register under program control. Display is turned on by Parallel Output instruction with I/O address X'00'. Each indicator lights if corresponding bit position of B-register contains a 1. Information displayed by indicators is lost if power shutdown occurs
Function switch SING INST COMPUTE REG TEST	Three-position rotary switch	Used mainly for maintenance and program debugging. Switch must be set to COMPUTE for normal processor operation. Setting switch to SING INST and then pressing START switch causes processor to execute instruction set into data switches 1 through 16 and then return to idle mode. Setting switch to REG TEST causes processor to load data from 16 data switches into register selected by register select switch
CONT (Continuous)	Two-position toggle switch	Used mainly for maintenance and troubleshooting. Setting switch to upper position causes processor to begin executing instructions starting with instruction at address set into 16 data switches. After 1 ms, processor halts and waits for 1 ms before resuming execution at starting address. This cycle continues until switch is set to lower position
PROTECT MEM	Two-position toggle switch	Used with BLOCK switches to prevent writing into protected areas of memory. Setting switch to upper position prevents writing into area of memory specified by setting of BLOCK switches. Setting switch to lower position disables memory protection
BLOCK 1 through 3	3 two-position toggle switches	Select area of memory that is protected when PROTECT MEM switch is set to upper position. Areas of memory selected for protection are as follows (0 represents switch down, 1 represents switch up): Switch Setting — Protected Area 000 — Entire memory 001 — X'0800' and up 010 — X'1000' and up

Table 2-1. Processor Controls and Indicators (Cont.)

Control or Indicator	Type	Function
BLOCK (Cont.)		Switch Setting — Protected Area 011 — X'1800' and up 100 — X'2000' and up 101 — X'2800' and up 110 — X'3000' and up 111 — X'3800' and up
INHIBIT INTRP	Two-position toggle switch	Inhibits or permits interrupts. Setting switch to upper position inhibits interrupts. Setting switch to lower position permits interrupts to occur provided level is armed and enabled
PARITY HLT	Two-position toggle switch	Determines whether processor halts when memory parity error is detected. If switch is set to lower position when parity error occurs, P indicator lights but processor continues executing instructions. If switch is in upper position, P indicator lights and processor halts after executing current instruction. RESET and START switches must then be pressed to restart program
B-E TEST	Two-position toggle switch	Tests processor power shutdown and restart circuits. Setting switch to upper position causes processor to continuously cycle through power shutdown and restart sequence consisting of 1 ms operation period followed by 7 ms shutdown period. Cycle continues until switch is set to lower position
Data switches 0 through 16	17 two-position toggle switches	Used to manually enter data, instructions, or address information into processor. Upper position of switches represents binary 1 and lower position, binary 0. Information is entered into processor from data switches in any of following ways: a. Pressing RESET switch loads address setting of data switches into L-register. b. Executing Parallel Input instruction with I/O address X'01' loads data from switches into B-register. c. Setting function switch to SING INST and pressing START switch causes processor to execute instruction set into switches and then return to idle mode. d. Setting function switch to REG TEST loads data from switches into register specified by setting of register select switch.
X1 through X7, YA, YB, YC, LID, NDI, LH1, LH2, LH3, GND, PIP, POP, SET, SNS, CLK	21 test jacks	Used for monitoring significant processor timing and control signals. Voltage levels at test jacks represent inverted logic levels: 0V for true level and +5V for false level



(A) REGISTER DISPLAY ASSIGNMENTS WHEN XY IS SELECTED

(B) REGISTER DISPLAY ASSIGNMENTS WHEN PK IS SELECTED

Figure 2-1. Register Display Indicator Assignments for XY and PK

2.6 CHANGING REGISTER CONTENTS

The following procedure is used to change the contents of any of the selectable registers:

a. Place the computer in idle mode by pressing the SINGLE CYCLE switch.

b. Set the register select switch to indicate the register whose contents are to be changed. The register display indicators display the current contents of the register.

c. Set the data that is to be entered into the register into data switches 1 through 16.

d. Set the function switch to REG TEST. The information set into the data switches is loaded into the selected register and is displayed by the register display indicators.

e. To restart the program, set the function switch to COMPUTE, press the SINGLE CYCLE switch, and then press the START switch.

2.7 DISPLAYING REGISTER CONTENTS

The following procedure is used to display the contents of any of the selectable registers:

a. Place the computer in idle mode by pressing the SINGLE CYCLE switch.

b. Set the register select switch to indicate the register whose contents are to be displayed. The register display indicators now display the contents of the selected register.

c. To restart the program, press the SINGLE CYCLE switch, and then press the START switch.

2.8 CHANGING MEMORY CONTENTS

The following procedure is used to change the contents of a location in memory:

a. Place the computer in idle mode by pressing the SINGLE CYCLE switch.

b. Set the register select switch to X.

c. Set the memory address into data switches 1 through 16.

d. Set the function switch to REG TEST and then back to COMPUTE. The memory address is now loaded into the index register.

e. Set the register select switch to B.

f. Set the data to be stored in memory into data switches 1 through 16.

g. Set the function switch to REG TEST and then back to COMPUTE. The data is now loaded into the B-register.

h. Set X'7800' (Store instruction) into data switches 1 through 16. Verify that the PROTECT MEM switch is off before proceeding with the next step.

i. Set the function switch to SING INST and then press the START switch. The computer now executes the Store instruction to write the data into the selected memory location.

j. To restart the program, reset the function switch to COMPUTE, press the SINGLE CYCLE switch, and then press the START switch.

2.9 DISPLAYING MEMORY CONTENTS

The following procedure is used to display the contents of a location in memory:

a. Place the computer in idle mode by pressing the SINGLE CYCLE switch.

b. Set the register select switch to X.

c. Set the memory address into data switches 1 through 16.

d. Set the function switch to REG TEST and then back to COMPUTE. The memory address is now loaded into the index register.

e. Set X'3800' (Load instruction) into data switches 1 through 16.

f. Set the function switch to SING INST and then press the START switch. The contents of the selected memory location are now loaded into the accumulator.

g. Set the register select switch to B to display the contents of the memory location.

h. To restart the program, reset the function switch to COMPUTE, press the SINGLE CYCLE switch, and then press the START switch.

2.10 STEPPING THROUGH A PROGRAM

The following procedure is used to step the processor through a stored program one instruction at a time:

a. Place the computer in idle mode by pressing the SINGLE CYCLE switch.

b. Set data switches 1 through 16 to the address of the instruction from which stepping is to start.

c. Press the RESET switch. The address set into the data switches is loaded into the memory address register, and the page register is set to X'01'.

d. Press the START switch. The processor executes one complete instruction and then returns to idle mode.

e. Repeat step d to step through consecutive instructions. Each time the START switch is pressed one complete instruction is executed.

f. To restart the program, set the starting address into the data switches, press the SINGLE CYCLE switch, press the RESET switch, and then press the START switch.

2.11 REMOVING POWER

To remove power from the computer, press the POWER switch. The POWER indicator should go out to indicate that ac power is off.

2.12 PROGRAMMING

Programming information for the RC 70 Computer is available in several REDCOR publications. Basic programming information including a description of data and instruction word formats, memory addressing and allocation, and detailed descriptions of all machine instructions is given in the RC 70 Computer Reference Manual, publication No. M-5000. Additional programming information is given in the RC 70 FORTRAN IV Reference Manual, publication No. M-5003, and the RC 70 Assembler Reference Manual, publication No. M-5002. Information pertaining to the operation of standard REDCOR program systems is contained in the RC 70 Operator's Manual, publication No. M-5001.

SECTION III
THEORY OF OPERATION

3.1 INTRODUCTION

This section contains the theory of operation of the RC 70 Computer. The theory is divided into two parts: a general theory of operation and a detailed theory of operation. In the general theory, subjects are covered in an introductory and simplified manner with emphasis on primary functions. In the detailed theory, emphasis is on logical operation.

3.2 GENERAL THEORY OF OPERATION

Figure 3-1 shows a block diagram of the computer. The processor contains the registers, arithmetic unit, and logic that performs the execution of instructions. The internal memory has a storage capacity of up to 16,384 words in multiples of 4096 words. Four external memories are optional; each has a storage capacity of 4096 words. The input/output system allows external devices to be controlled by the processor.

3.3 CENTRAL PROCESSOR

3.4 Registers

The processor contains seven standard and two optional registers used for addressing and instruction execution. The following paragraphs describe the function of each register.

3.5 Instruction Register (I-Register)

The I-register contains the instruction currently being executed. This register automatically receives the instruction word as it is fetched from core memory. The instruction is then decoded to provide the control signals required by the computer to execute the instruction and perform the specified operation.

3.6 Working Storage Register (D-Register)

The D-register is a working register used for temporary storage and as a working buffer.

3.7 Accumulator (B-Register)

The B-register is the primary arithmetic register of the computer. It provides one of the operands to the adder for all arithmetic and logical operations and also stores the result of the operation. The B-register contains the multiplicand at the beginning of a multiplication operation and contains the least significant half of the product at the end of the operation. It contains the divisor at the beginning of a division operation and contains the quotient at the end of the operation. It is the source register for a store operation and the destination register for a load operation. The contents of the B-register are affected by all shift operations specifying shifts of one or more places.

The B-register is used during single-word input/output operations. The contents of the B-register are transferred to the I/O device during an output operation. During an input operation, the accumulator receives the data transferred into the computer from the I/O device.

3.8 Index Register (X-Register)

The index register is used for address modification. Contents of the index register are affected by load index and branch index instructions.

3.9 Next-Instruction-Address Register (N-Register)

The N-register normally contains the address of the next instruction to be executed. The contents of this register are increased by one at the start of each instruction. At the end of the execution cycle the contents of the N-register are transferred to the memory address register.

3.10 Memory Address Register (L-Register)

The L-register contains the address of the memory location into which instruction or data words are stored, or from which they are fetched.

3.11 Page Register

The page register is used during page-relative addressing. During an instruction using page-relative addressing, the contents of the page register are copied into bit positions 3 through 8 of the memory address register and the reference address specified by the instruction is copied into bit positions 9 through 16. The effect is that of splitting memory up into 64 blocks or pages, each containing 256 memory locations.

3.12 G- and J-Registers

The G- and J-registers are part of high-speed multiply/divide and are used for temporary storage and as working buffer registers.

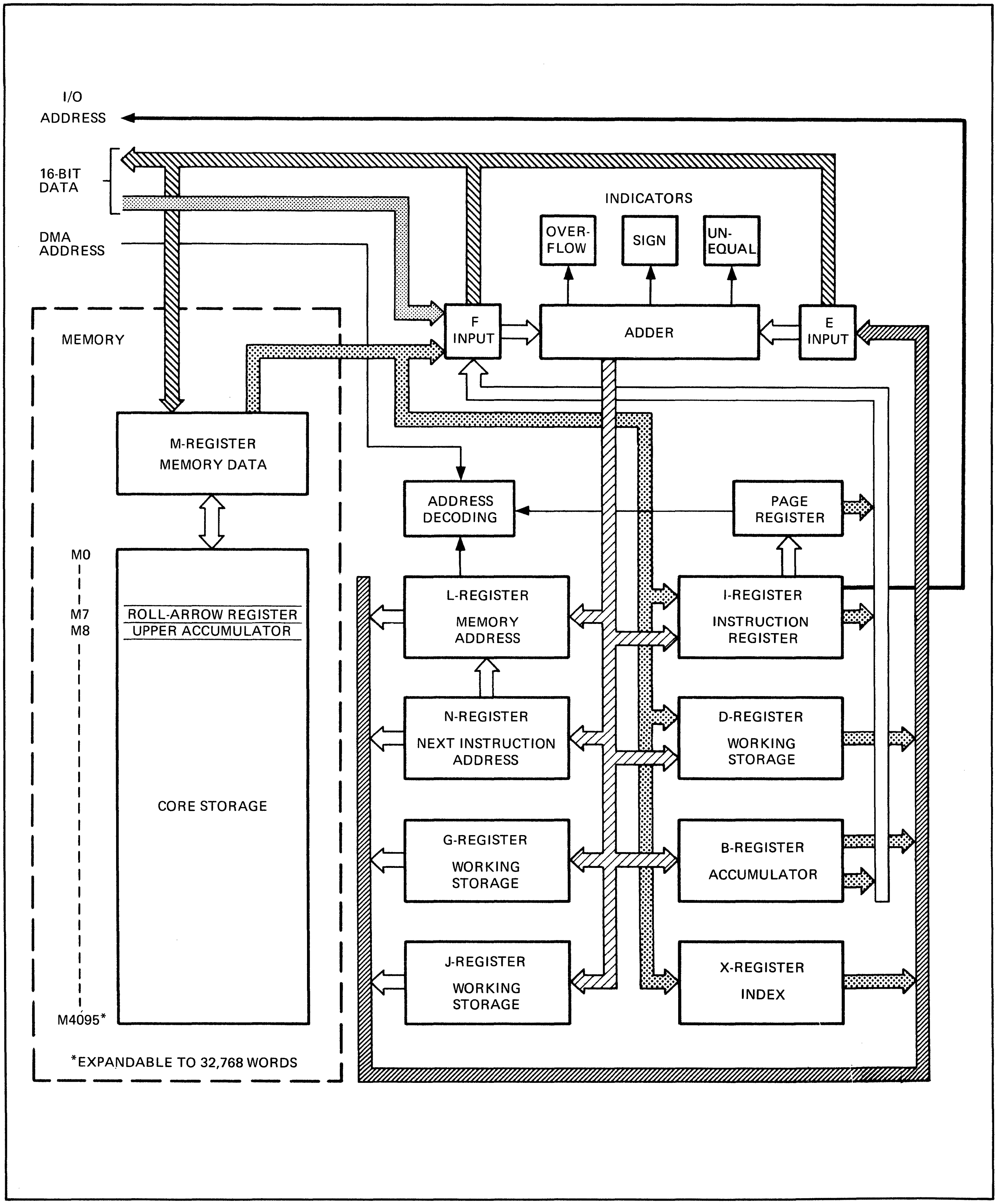


Figure 3-1. RC 70 Computer, Block Diagram

3.13 Adder

The adder consists of two input buses, a 16-bit full adder, three indicators for overflow, sign, and unequal indications, and an output data bus. These provide the control logic required to perform addition, subtraction, and other arithmetic functions in response to instructions in the program and gate the results onto the data bus. The following operations are performed between the adder and the data bus:

a. Register transfers

b. Addition

c. Subtraction (by adding the two's complement of the subtrahend)

d. Multiplication (by addition and shifting)

e. Division (by subtraction and shifting)

f. Increasing the contents of specific registers

g. Decreasing the contents of specific registers

h. Logical And and logical Or

Two additional registers are used for arithmetic operations: the lower accumulator (B-register) and upper accumulator (UBA-register). The upper accumulator resides in memory location X'0008'. It is used as an extension of the lower accumulator for multiplication, division, and double-length shift operations.

3.14 Core Memory

The basic memory of the computer consists of 4096 words of internal core storage. Additional core storage is available in blocks of 4096 words each up to a maximum of 32,768 words.

The computer memory word consists of 16 data bits plus a roll tag bit (bit position 0) and a parity bit. The roll tag bit is used only during a Branch Back instruction. Memory parity, however, is checked each time a word is read from memory.

All data enters and leaves core storage through the memory data register. The address of the memory location into which data is stored, or from which data is fetched, is provided to the address decoding logic by either the memory address register or the direct memory access (DMA) address lines.

3.15 Memory Protection Feature

The memory protection feature of the computer makes it possible to protect the contents of memory in 2048-word blocks by inhibiting writing into the protected areas. Memory protection is controlled from the operator's console by setting appropriate switches.

3.16 Diode Memory

The diode memory is a read-only memory containing a 19-instruction fill routine. It is activated by pressing the FILL switch on the processor control panel. Pressing the switch causes the computer to begin executing instructions beginning at location X'8000', the starting address of the diode memory.

3.17 Input/Output

The input/output section of the computer provides the capability for:

a. Discrete control and sensing of external I/O devices.

b. Single-word transfer (16-bit parallel) between the computer and external I/O devices under program control.

c. Single and multiple block transfers between external I/O devices and memory through the direct memory access feature under control of the optional block-transfer channels.

d. Control of input/output operations through the external interrupt with 32 levels of priority interrupt available as options.

3.18 DETAILED THEORY OF OPERATION

The detailed theory of operation describes the operation of each functional unit primarily in terms of logic elements (flip-flops, gates, registers) rather than in terms of circuit components (capacitors, resistors, transistors). Functional, logic, timing, and simplified diagrams are included where useful for clarification.

A glossary of logic terms is given in table 3-39. The glossary defines the logic terms used in this section.

3.19 CENTRAL PROCESSOR

The central processor contains seven standard registers for addressing and executing instructions, two registers for high-speed multiply/divide, address decoding logic, and an adder with indicators.

3.20 Registers

3.21 Instruction Register (I-Register)

The I-register consists of 16 flip-flops (I1 through I16). It normally contains the instruction presently being executed. During multiplication, division, and shifting operations, the I-register is used as a counter to indicate the number of iterations.

Logically, the I-register is divided into two parts, the lower part indicated by the term IL and the upper part indicated by the term IH. During counting operations, the lower part of the register is used.

During shift operations, the I-register counts one-bit shifts by gating bits I15 and I16 with gating signal CTIB. Four-bit position shifts are counted by gating bit positions I13 and I14 with signal CTID. Full word shifts can be counted by gating bit positions I10, I11, and I12 with signal CTIW.

In a double word instruction, the operation code is contained in bit positions 12 through 15 of the first word. These bits are shifted into bit positions I1 through I4 by signal ILTIH for decoding purposes.

Inputs are gated into the I-register from memory by gating signal MTI and from the adder sum bus by gating signal STI. Outputs from the I-register are gated onto the F-bus adder input and the page register. Bit positions I9 through I16 are also applied to the input/output address lines for use by peripheral equipment.

I1 = STI S1 + MTI M1 + ILTIH I12

$\overline{\text{I1}}$ = . . .

CI1 = CKIH

⋮ ⋮

I5 = STI S5 + MTI M5 + ILTIH I16

$\overline{\text{I5}}$ = . . .

CI5 = CKI

I6 = STI S6 + MTI M6

$\overline{\text{I6}}$ = . . .

CI6 = CKI

⋮ ⋮

I9 = STIL S9 + MTI M9

$\overline{\text{I9}}$ = . . .

CI9 = CKIW

I10 = CTIW I10 I11 + CTIW I10 I12
+ STIL S10 + MTI M10

$\overline{\text{I10}}$ = . . .

CI10 = CKIW

I11 = CTIW $\overline{\text{I11}}$ $\overline{\text{I12}}$ + CTIW I11 I12
+ STIL S11 + MTI M11

$\overline{\text{I11}}$ = . . .

CI11 = CKIW

I12 = CTIW $\overline{\text{I12}}$ + STIL S12 + MTI M12

$\overline{\text{I12}}$ = . . .

CI12 = CKIW

I13 = CTID I14 I13 + STIL S13 + MTI M13

$\overline{\text{I13}}$ = . . .

CI13 = CKID

I14 = CTID $\overline{\text{I14}}$ + STIL S14 + MTI M14

$\overline{\text{I14}}$ = . . .

CI14 = CKID

I15 = CTIB I16 I15 + STIL S15 + MTI M15
+ BZRO CTIB

$\overline{\text{I15}}$ = . . .

CI15 = CKIB

I16 = CTIB $\overline{\text{I16}}$ + STIL S16 + MTI M16

$\overline{\text{I16}}$ = . . .

CI16 = CKIB

Figure 3-2 illustrates the I-register gating logic.

3.22 Working Storage Register (D-Register)

The D-register consists of 16 flip-flops (D1 through D16). The register is used for temporary storage and as a working buffer.

Inputs to the D-register are from the sum bus and memory. During multiply, divide, branch and link, and branch and put operations, input is from the sum bus by gating signal STD. During branch index, exchange memory and accumulator, branch back, double-length shift, inclusive Or and exclusive Or operations, input is from memory by gating signal MTD. Output of the D-register is applied to the E-bus adder input and to the D-register lamp drivers.

Any register to be displayed on the operator's panel indicators must first be clocked into the D-register. The output of the D-register is applied to 16 lamp drivers consisting of D1L through D16L for gating to the indicator lamps.

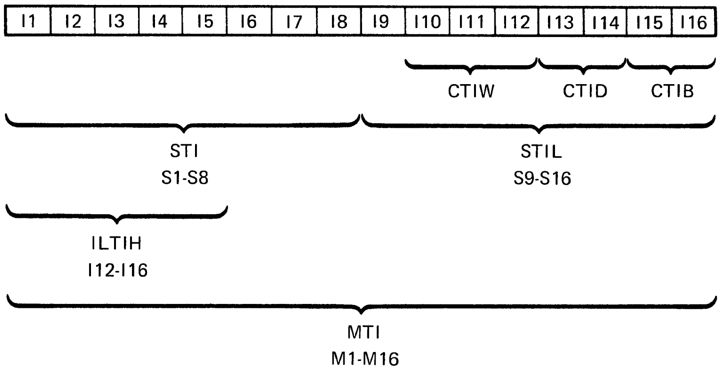


Figure 3-2. I-Register, Gating Logic

D1 = STD S1 + MTD M1

$\overline{\text{D1}}$ = . . .

CD1 = CKD

⋮ ⋮

D16 = STD S16 + MTD M16

$\overline{\text{D16}}$ = . . .

CD16 = CKD

Figure 3-3a illustrates the D-register gating logic.

3.23 Lower Accumulator (B-Register)

The B-register consists of 16 flip-flops (B1 through B16). Input to the B-register is from the data sum bus of the adder. The output is applied to the E-input of the adder by gating signal BTE or to the F-input by gating signal BTF.

B1 = S1

$\overline{\text{B1}}$ = . . .

CB1 = CKB

⋮ ⋮

B16 = S16

$\overline{\text{B16}}$ = . . .

CB16 = CKB

CKB = STB CLK

STB = PIP X30 + YBGN X31 + GRT1 BSW
+ XMB $\underline{\text{X31}}$ + IATH $\overline{\text{ICM}}$ X61
+ ISFT $\overline{\text{ELB}}$ X6 + ELB KA X6
+ STBD

Figure 3-3b illustrates the B-register gating logic.

3.24 Index Register (X-Register)

The X-register consists of 16 flip-flops (IX1 through IX16) used for address modification. The X-register is loaded by a Load Index instruction. During a Branch Index instruction the contents of the index register are checked for a zero condition. If the value is positive the contents are decreased by one; if the contents are negative, they are increased by one.

Input to the index register is from the M-register. The output is to the E-input bus of the adder.

Memory location X'0000' is a duplicate index register that contains the same information as the hardware index register. Information from the E- and F-buses is gated onto data lines C1 through C16. The outputs of C1 through C16 are applied to memory latches M1L through M16L. If the address on the memory address lines is X'0000', the data from M1L through M16L is stored in memory location X'0000' as well as clocked into the index register.

During a Branch Index instruction, if the index register is not zero, as indicated by bit position M1 of memory location X'0000', the contents of the index register are gated to the E-input bus of the adder by gating signal IXTE. One count is either added or subtracted to the contents of the index register as determined by bit position M1. The output of the adder is gated to the M-register and is then stored in memory location X'0000' and the index register.

IX1 = M1

$\overline{\text{IX1}}$ = . . .

CIX1 = CLKX

⋮ ⋮

IX16 = M16

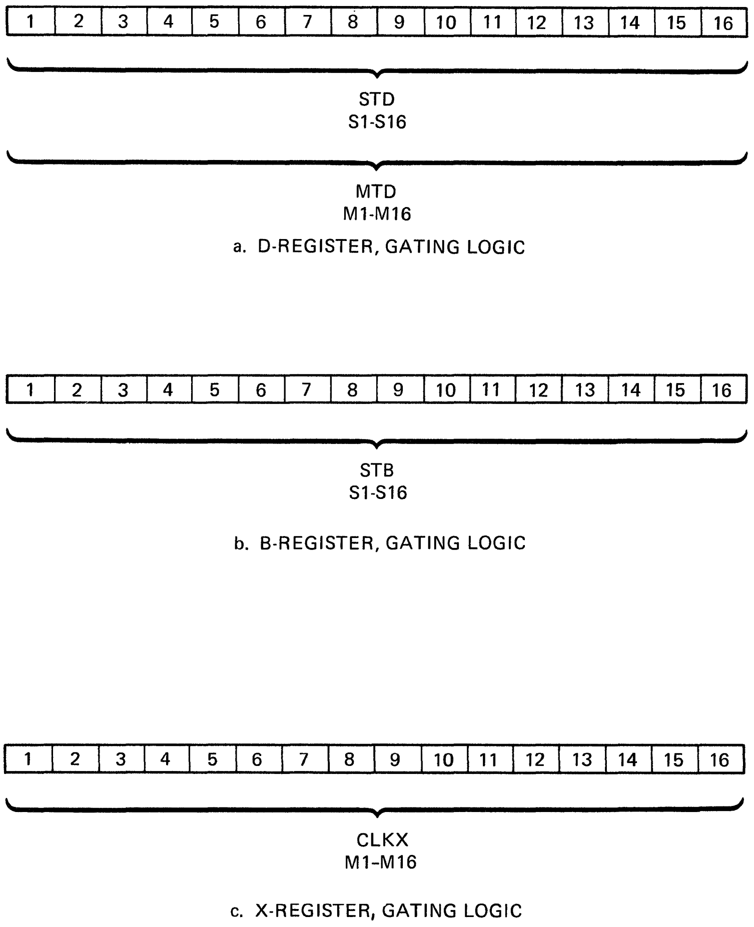


Figure 3-3. D-, B-, and X-Registers, Gating Logic

$$\overline{\mathrm{IX16}} = \ldots$$

$$\mathrm{CIX16} = \mathrm{CLKX}$$

$$\mathrm{CLKX} = \mathrm{MGOA}\ \overline{\mathrm{CLK}}$$

$$\mathrm{MGOA} = \overline{\mathrm{DMGOA}}\ \mathrm{CLKM}$$

$$\overline{\mathrm{DMGOA}} = \overline{\mathrm{XADSA}}\ \overline{\mathrm{MGO}}\ \overline{\mathrm{ADS13}}\ \overline{\mathrm{ADS14}}\ \overline{\mathrm{ADS15}}\ \overline{\mathrm{ADS16}}\ \overline{\mathrm{ADS2}}$$

Figure 3-3c illustrates the index register gating logic.

3.25 Next-Instruction-Address Register (N-Register)

The N-register consists of 16 flip-flops (N1 through N16). Input to the N-register is from the data sum bus. The transfer is performed by gating the clock with signal STN. The register output is gated to the E-input of the adder by gating signal NTE.

$$\mathrm{N1} = \mathrm{S1}$$

$$\overline{\mathrm{N1}} = \ldots$$

$$\mathrm{CN1} = \mathrm{CKN}$$

$$\vdots \qquad \vdots$$

$$\mathrm{N16} = \mathrm{S16}$$

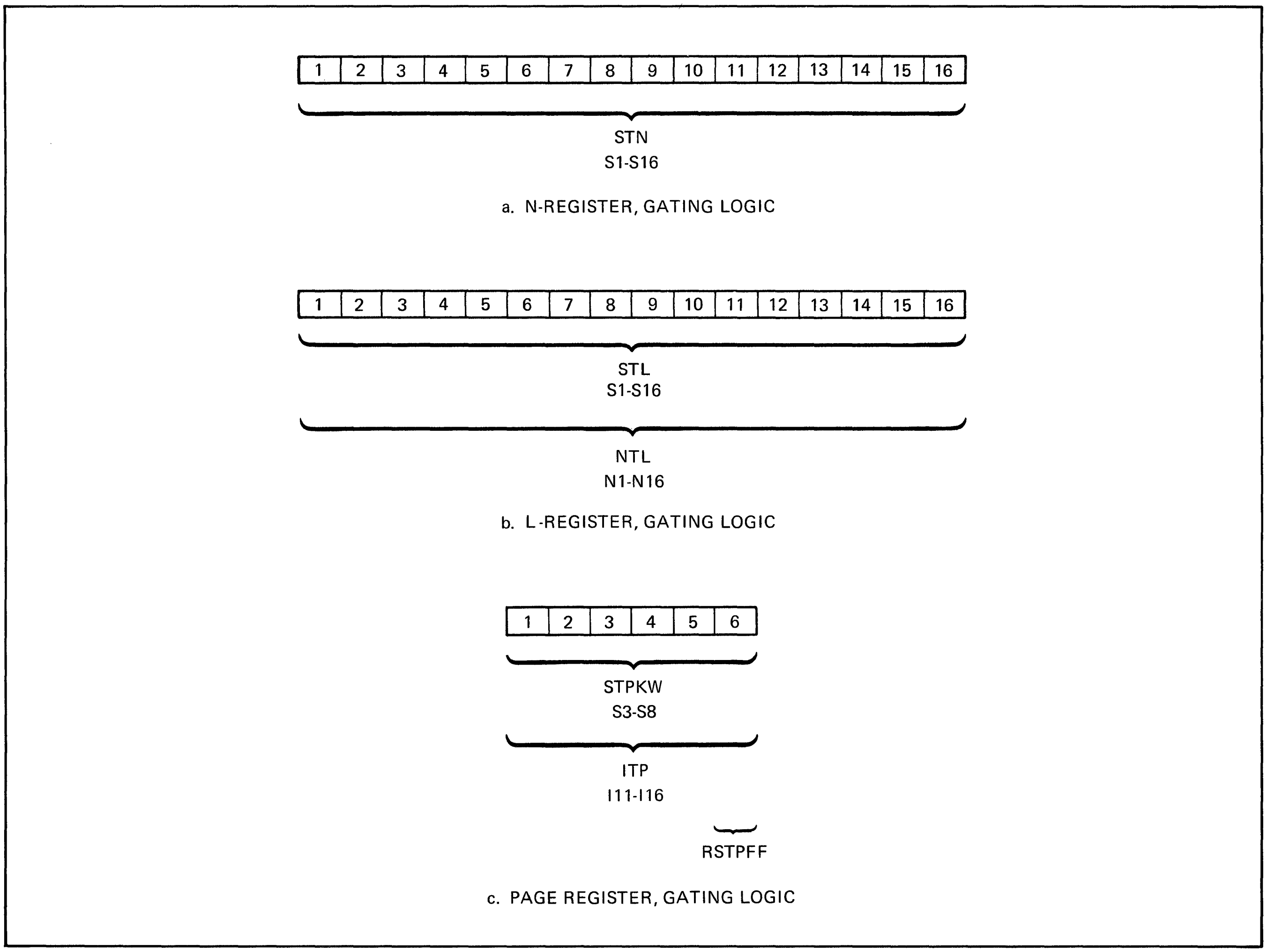
$$\overline{\mathrm{N16}} = \ldots$$

$$\mathrm{CN16} = \mathrm{CKN}$$

$$\mathrm{CKN} = \mathrm{STN}\ \mathrm{CLK}$$

$$\mathrm{STN} = \mathrm{YOAS}\ \mathrm{X10} + \mathrm{IBSV}\ \mathrm{X51} + \overline{\mathrm{BBK}}\ \mathrm{X31} + \mathrm{BBK}\ \mathrm{X61} + \mathrm{YBGN}\ \overline{\mathrm{VZRO}}\ \mathrm{X11} + \mathrm{GRT1}\ \mathrm{NSW} + \mathrm{YOAS}\ \mathrm{X20}$$

Figure 3-4a illustrates the N-register gating logic.



a. N-REGISTER, GATING LOGIC

b. L-REGISTER, GATING LOGIC

c. PAGE REGISTER, GATING LOGIC

Figure 3-4. N-, L-, and Page Registers, Gating Logic

3.26 Memory Address Register (L-Register)

The L-register consists of 16 flip-flops (L1 through L16). The L-register contains the address of the memory location from which the instruction word or data word is fetched or where it is stored. The starting address of a program can be entered into the L-register by setting the data switches on the processor control panel to the starting address and then pressing the RESET switch.

Inputs to the L-register are from the data sum bus and the N-register. Sum bus inputs are gated into the L-register by signal STL; N-register inputs by signal NTL. Output from the L-register is gated onto address lines ADS1 through ADS16 by gating signal LADS or into the E-input of the adder by gating signal LTE.

The address set into the data switches on the processor control panel is gated onto the sum bus. Signal LSW is generated resulting in signal STL gating the address from the data switches into the L-register.

$$
\begin{aligned}
\mathrm{L1} &= \mathrm{STL\ S1 + NTL\ N1} \\
\overline{\mathrm{L1}} &= \ldots \\
\mathrm{CL1} &= \mathrm{CKL} \\
\vdots\ \ & \quad \vdots \\
\mathrm{L16} &= \mathrm{STL\ S16 + NTL\ N16} \\
\overline{\mathrm{L16}} &= \ldots \\
\mathrm{CL16} &= \mathrm{CKL}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{STL} = {} & \mathrm{IBSV\ X71 + YBGN\ X41 + GRT1\ LSW} \\
& + \mathrm{LCS\ X11 + LTE}\ \overline{\mathrm{YIDL}} \\
& + \mathrm{CDBR}\ \underline{\mathrm{BRSAT\ XODD}} \\
& + \mathrm{YOAS}\ \overline{\mathrm{LNG}}\ \overline{\mathrm{CDBR}}\ \mathrm{XODD} \\
& + \mathrm{IBLPK\ X21 + YOAS\ X20}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{NTL} = {} & \mathrm{NY4}\ \overline{\mathrm{IBX}}\ \overline{\mathrm{IBSV}}\ \mathrm{ENDI} \\
& + \mathrm{IBX\ EZRO\ ENDI}
\end{aligned}
$$

Figure 3-4b illustrates the L-register gating logic.

3.27 Page Register

The page register consists of six flip-flops (P1 through P6). The register is used during page-relative addressing as determined by bits 7 and 8 of the single word instruction. The page register specifies the memory page being addressed and the reference address of the instruction specifies the address of the memory word within the page.

Inputs to the page register are from the data sum bus and the I-register. The contents of the page register are affected by a Load Page instruction and a Branch Back instruction performed after a Branch and Put instruction. During the Load Page instructions, bits 11 through 16 of the I-register are loaded into the page register by gating signal ITP. During the Branch Back instruction, the contents of the page register stored during the Branch and Put instruction are restored to the page register by gating signal STPKW. Output of the register is to the F-input bus of the adder.

$\mathrm{P1} = \mathrm{STPKW\ \ S3 + ITP\ \ I11}$

$\overline{\mathrm{P1}} = \ldots$

$\mathrm{CP1} = \mathrm{CKP}$

$\vdots \qquad \vdots$

$\mathrm{P5} = \mathrm{STPKW\ \ S7 + ITP\ \ I15}$

$\overline{\mathrm{P5}} = \ldots$

$\mathrm{CP5} = \mathrm{CKP}$

$\mathrm{P6} = \mathrm{STPKW\ \ S8 + ITP\ \ I16 + RSTPFF}$

$\overline{\mathrm{P6}} = \ldots$

$\mathrm{CP6} = \mathrm{CKP}$

$\mathrm{STPKW} = \mathrm{BBK\ \ X60 + YBGN\ \ X71}$
$\qquad\qquad + \mathrm{GRT1\ \ PKWSW}$

$\mathrm{GRT1} = \mathrm{RTFF\ \ YIDL\ \ X7}$

PKWSW = Register Select Thumbwheel switch set to PK position

RTFF = Function switch set to REG TEST position

Figure 3-4c illustrates the page register gating logic.

3.28 G- and J-Registers

Two additional registers are provided as part of the high-speed multiply/divide feature. These registers consist of 16 flip-flops each and are indicated by G1 through G16 and J1 through J16. Input to these registers is from the adder sum bus. Outputs are to the E-input of the adder.

Data in the G-register can be right-shifted two places by shift signal RSG2 or left-shifted one place by signal LSG1. Data in the J-register can be right-shifted one place by shift signal RSJ1.

$\mathrm{G1} = \mathrm{STG\ \ S1 + RSG2\ \ J15 + LSG1\ \ G2}$

$\overline{\mathrm{G1}} = \ldots$

$\mathrm{CG1} = \mathrm{CLKG}$

$\mathrm{G2} = \mathrm{STG\ \ S2 + RSG2\ \ J16 + LSG1\ \ G3}$

$\overline{\mathrm{G2}} = \ldots$

$\mathrm{CG2} = \mathrm{CLKG}$

$\mathrm{G3} = \mathrm{STG\ \ S3 + RSG2\ \ G1 + LSG1\ \ G4}$

$\overline{\mathrm{G3}} = \ldots$

$\mathrm{CG3} = \mathrm{CLKG}$

$\vdots \qquad \vdots$

$\mathrm{G16} = \mathrm{STG\ \ S16 + RSG2\ \ G14}$

$\overline{\mathrm{G16}} = \ldots$

$\mathrm{CG16} = \mathrm{CLKG}$

$\mathrm{STG} = \mathrm{MPA\ \ X11 + DVSF\ \ X31 + ELD\ \ X21}$

$\mathrm{RSG2} = \mathrm{SSPM + SPMS}$

$\mathrm{LSG1} = \mathrm{DVA\ \ X40 + DVA\ \ XX5}$
$\qquad\qquad + \mathrm{ELD}\ \ \overline{\mathrm{MDEND}}\ \ \mathrm{XX5}$

$\mathrm{J1} = \mathrm{SSPM\ \ S1 + SPMS\ \ BINSIN}$

$\overline{\mathrm{J1}} = \ldots$

$\mathrm{CJ1} = \mathrm{CLKJ}$

$\mathrm{J2} = \mathrm{SSPM\ \ S2 + SPMS\ \ S1}$

$\overline{\mathrm{J2}} = \ldots$

$\mathrm{CJ2} = \mathrm{CLKJ}$

$\vdots \qquad \vdots$

$\mathrm{J16} = \mathrm{SSPM\ \ S16 + SPMS\ \ S15}$

$\overline{\mathrm{J16}} = \ldots$

$\mathrm{CJ16} = \mathrm{CLKJ}$

$\mathrm{SSPM} = \overline{\mathrm{G16QKA}}\ \ \mathrm{MPA\ \ XX5}$

$\overline{\mathrm{G16QKA}} = \mathrm{G16}\ \ \overline{\mathrm{KA}} + \mathrm{G16\ \ KA}$

$\mathrm{SPMS} = \overline{\mathrm{SSPM}}\ \ \mathrm{MPA\ \ XX5}$

Figure 3-5 illustrates the G- and J-register gating logic.

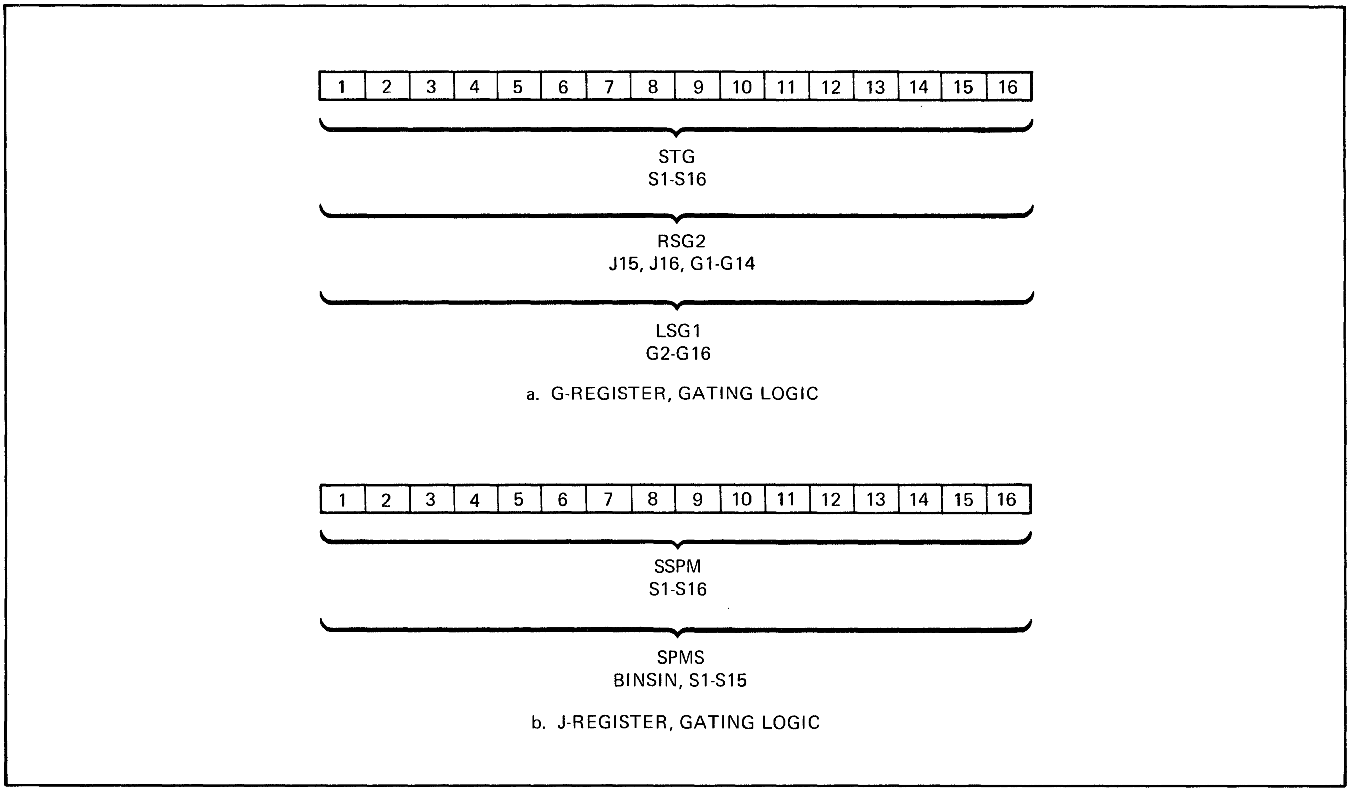


Figure 3-5. G- and J-Registers, Gating Logic

3.29 Adder

The adder consists of two 16-bit input buses, two 16-bit intermediate stages used for shifting and inverting, a 16-bit exclusive-Or stage, a 16-bit carry stage with an initial carry gate, 16 bits of adder control logic, a 16-bit output sum bus, and three indicators.

Inputs to the adder are through either a 16-bit E-input bus or a 16-bit F-input bus.

Figure 3-6 is a block diagram of the adder.

3.30 E-Input Bus

Outputs of all the registers except the instruction and page registers are individually gated to the E-input bus of the adder. (See figure 3-7a.)

E1 = LTE L1 + BTE B1 + DTE D1
+ NTE N1 + KTE KS + JTE J1
+ GTE G1 + RSJ1 J0 + IXTE IX1

E2 = LTE L2 + BTE B2 + DTE D2
+ NTE N2 + KTE KU + JTE J2
+ GTE G2 + RSJ1 J1 + IXTE IX2

E3 = LTE L3 + BTE B2 + DTE D3
+ NTE N3 + JTE J3 + GTE G3
+ RSJ1 J2 + IXTE IX3

⋮ ⋮

E9 = LTE L9 + BTE B9 + DTE D9
+ NTE N9 + KTE KK + JTE J9
+ GTE G9 + RSJ1 J8 + IXTE IX9

E10 = LTE L10 + BTE B10 + DTE D10
+ NTE N10 + KTE KC + JTE J10
+ GTE G10 + RSJ1 J9 + IXTE IX10

E11 = LTE L11 + BTE B11 + DTE D11
+ NTE N11 + KTE KB + JTE J11
+ GTE G11 + RSJ1 J10 + IXTE IX11

E12 = LTE L12 + BTE B12 + DTE D12
+ NTE N12 + KTE KA + JTE J12
+ GTE G12 + RSJ1 J11 + IXTE IX12

E13 = LTE L13 + BTE B13 + DTE D13
+ NTE N13 + TTE T1 + JTE J13
+ GTE G13 + RSJ1 J12 + IXTE IX13

⋮ ⋮

E16 = LTE L16 + BTE B16 + DTE D16
+ NTE N16 + ITE T4 + JTE J16
+ GTE G16 + RSJ1 J15 + IXTE IX16

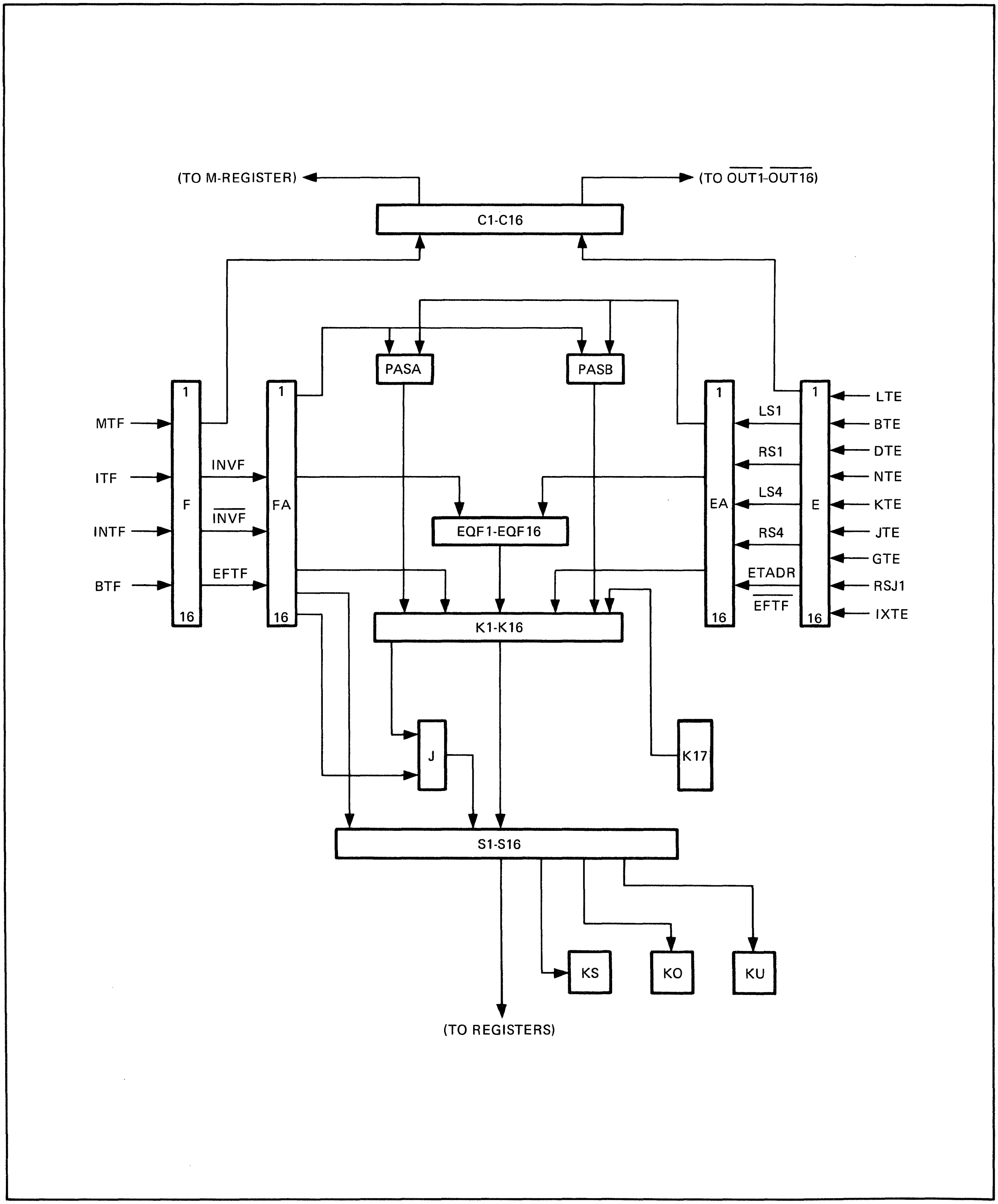
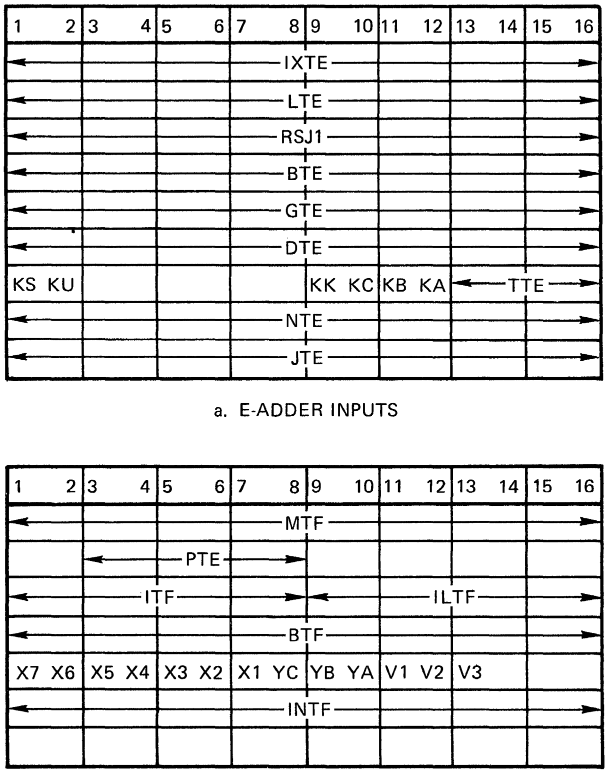


Figure 3-6. Adder, Block Diagram



a. E-ADDER INPUTS

b. F-ADDER INPUTS

Figure 3-7. Adder Inputs

Output of the E-input bus is applied to intermediate stages E1A through E16A where any required shifting is performed.

$$\begin{aligned}\text{E1A} &= \text{LS1 E2} + \text{RS1 E0} + \text{LS4 E5} + \text{RS4}\\ &\quad + \overline{\text{SHIFT}}\ \overline{\text{EFTF}}\ \text{E1}\end{aligned}$$

$$\begin{aligned}\text{E2A} &= \text{LS1 E3} + \text{RS1 E1} + \text{LS4 E6} + \text{RS4}\\ &\quad + \overline{\text{SHIFT}}\ \overline{\text{EFTF}}\ \text{E2}\end{aligned}$$

$$\begin{aligned}\text{E3A} &= \text{LS1 E4} + \text{RS1 E2} + \text{LS4 E7} + \text{RS4}\\ &\quad + \overline{\text{SHIFT}}\ \overline{\text{EFTF}}\ \text{E3}\end{aligned}$$

$$\begin{aligned}\text{E4A} &= \text{LS1 E5} + \text{RS1 E3} + \text{LS4 E8} + \text{RS4 E1}\\ &\quad + \overline{\text{SHIFT}}\ \overline{\text{EFTF}}\ \text{E4}\end{aligned}$$

$$\vdots \qquad \vdots$$

$$\begin{aligned}\text{E13A} &= \text{LS1 E14} + \text{RS1 E12} + \text{LS4 H1} + \text{RS4 E9}\\ &\quad + \overline{\text{SHIFT}}\ \overline{\text{EFTF}}\ \text{E13}\end{aligned}$$

$$\begin{aligned}\text{E14A} &= \text{LS1 E15} + \text{RS1 E13} + \text{LS4 H2} + \text{RS4 E10}\\ &\quad + \text{ETADR E14}\end{aligned}$$

$$\begin{aligned}\text{E15A} &= \text{LS1 E16} + \text{RS1 E14} + \text{LS4 H3} + \text{RS4 E11}\\ &\quad + \text{ETADR E15}\end{aligned}$$

$$\begin{aligned}\text{E16A} &= \text{LS1 H1} + \text{RS1 E15} + \text{LS4 H4} + \text{RS4 E12}\\ &\quad + \overline{\text{SHIFT}}\ \overline{\text{EFTF}}\ \text{E16}\end{aligned}$$

3.31 F-Input Bus

Outputs of the instruction and page registers are gated to the F-input bus of the adder by gating signals ITF, ILTF, and PTF. Data from memory and data from external devices are gated to the F-input bus by gating signals MTF and INTF. During a fast multiply or divide operation, the output of the B-register is transferred to the F-input bus by gating term BTF. (See figure 3-7b.)

$$\text{F1} = \text{MTF M1} + \text{ITF I1} + \text{INTF IN1} + \text{BTF B1}$$

$$\vdots \qquad \vdots$$

$$\begin{aligned}\text{F3} &= \text{MTF M3} + \text{ITF I3} + \text{INTF IN3} + \text{BTF B3}\\ &\quad + \text{PTE P1}\end{aligned}$$

$$\vdots \qquad \vdots$$

$$\begin{aligned}\text{F8} &= \text{MTF M8} + \text{ITF I8} + \text{INTF IN8} + \text{BTF B8}\\ &\quad + \text{PTE P6}\end{aligned}$$

$$\text{F9} = \text{MTE M9} + \text{ILTF I9} + \text{INTF IN9} + \text{BTF B9}$$

$$\vdots \qquad \vdots$$

$$\text{F16} = \text{MTE M16} + \text{ILTF I16} + \text{INTF IN16} + \text{BTF B16}$$

Output of the F-input bus is applied to intermediate stages F1A through F16A where any required negating or inverting is performed.

$$\mathrm{F1A} = \mathrm{INVF}\ \mathrm{F1} + \overline{\mathrm{INVF}}\ \overline{\mathrm{F1}} + \mathrm{EFTF}\ \overline{\mathrm{E1}}$$

$$\vdots \qquad \vdots$$

$$\mathrm{F16A} = \mathrm{INVF}\ \mathrm{F16} + \overline{\mathrm{INVF}}\ \overline{\mathrm{F16}} + \mathrm{EFTF}\ \overline{\mathrm{E16}}$$

3.32 Data Transfer

Outputs of E1 through E16 and F1 through F16 are also applied to gates C1 through C16 for output data. If the input to either Ex or Fx is true, where x represents 1 through 16, the output of Cx is true. The outputs of C1 through C16 are applied to the memory data register and to output drivers $\overline{\mathrm{OUT1}}$ through $\overline{\mathrm{OUT16}}$ for transfer to external equipment.

$$\mathrm{C1} = \overline{\mathrm{E1}} + \overline{\mathrm{F1}}$$

$$\vdots \qquad \vdots$$

$$\mathrm{C16} = \overline{\mathrm{E16}} + \overline{\mathrm{F16}}$$

$$\overline{\mathrm{OUT1}} = \mathrm{C1}$$

$$\vdots \qquad \vdots$$

$$\overline{\mathrm{OUT16}} = \mathrm{C16}$$

3.33 Arithmetic Operations

During an arithmetic operation, one operand is transferred from a storage register into the E-input bus of the adder. The other operand is gated into the F-input bus from core memory, the I-register, the B-register, or from external equipment. The operands are transferred to the ExA and FxA intermediate stages where any required shifting or inverting is performed. An exclusive-Or operation is then performed on the operands by exclusive-Or gates EQF1 through EQF16. Two look-ahead gates, PASA and PASB, determine any carries required for four bits ahead. The carries are stored in gates K1 through K17, and the result of the operation is transferred to the adder sum bus, S1 through S16.

Figure 3-8 illustrates the addition of two positive eight-bit numbers and the result that appears on the sum bus.

During a subtraction operation, the operand on the F-input bus is inverted to obtain the one's complement, a one is added to the result by gate K17 to obtain the two's complement, and the two operands are added. The result is transferred to the sum bus.

The contents of a register may be increased by gating the contents of the register to the E-input bus, forcing a one into gate K17 and adding the two. The result is transferred to the sum bus and returned to the source register.

PROBLEM: (+42) + (+24) = 66

00101010 + 00011000 = 01000010

	9	10	11	12	13	14	15	16	
Ex	0	0	1	0	1	0	1	0	+42
Fx	0	0	0	1	1	0	0	0	+24
ExA	0	0	1	0	1	0	1	0	
$\overline{\mathrm{ExA}}$	1	1	0	1	0	1	0	1	
FxA	0	0	0	1	1	0	0	0	
$\overline{\mathrm{FxA}}$	1	1	1	0	0	1	1	1	
EQFx	0	0	1	1	0	0	1	0	
$\overline{\mathrm{EQFx}}$	1	1	0	0	1	1	0	1	
PASA	1		1		1		1		
PASB	0		0		0		0		
Jx		1	0			0	0		
$\overline{\mathrm{Jx}}$		0	1			1	0		
Kx	0	0	1	1	1	0	0	0	
$\overline{\mathrm{Kx}}$	1	1	0	0	0	1	1	1	
Sx	0	1	0	0	0	0	1	0	+66

x = 1-16

Figure 3-8. Addition Example, Single Byte

The contents of a register are decreased by gating the contents of the register to the E-input bus, forcing all ones into the F-input bus and adding the two operands. The result is transferred to the sum bus and returned to the source register.

3.34 Indicators

Three indicator flip-flops are used to provide an indication of the result of an arithmetic operation. A Sign indicator (KS) indicates if the sign of the result is a positive or a negative quantity. An Unequal indicator (KU) indicates if the result is equal to zero. An Overflow indicator (KO) indicates whether the result of the arithmetic operation exceeded the capacity of the B-register or, during a division operation, whether the contents of the B-register are less than the magnitude of the divisor.

These indicators can be tested by applicable branch instructions.

$$\begin{aligned}
\mathrm{KS} &= \mathrm{IATH}\ \overline{\mathrm{IST}}\ \mathrm{BINSIN\ X61} \\
&\quad + \mathrm{(IMDB)\ D1\ [MPB]\ X30} + \mathrm{DVB\ D1\ X30} \\
&\quad + \mathrm{ISFT\ B1\ X61} + \mathrm{STPKW\ S1} \\
&\quad + \mathrm{PIP\ BINSIN\ X30} + \mathrm{IKSD} + \mathrm{KSTME} \\
&\quad + \mathrm{IXOR\ X21\ S1} + [\mathrm{MPA\ J1\ X60} + \mathrm{ILD\ G1\ X61}] \\
\overline{\mathrm{KS}} &= \mathrm{ILD\ X20} + \mathrm{NY4\ X10} + \mathrm{KUTME} + \mathrm{KSUSO} \\
\mathrm{CKS} &= \mathrm{CKKU}
\end{aligned}$$

$$\begin{aligned}
\mathrm{KO} &= \mathrm{DVB\ S1\ KA\ X41} + \mathrm{DVB}\ \overline{\mathrm{S1}}\ \overline{\mathrm{KA}}\ \mathrm{X41} \\
&\quad + \mathrm{IATH\ BINOVF\ X61} + \mathrm{ILD\ S0\ X61} \\
&\quad + \mathrm{PIP\ S0\ X30} + \mathrm{STPKW\ S0} \\
&\quad + \mathrm{DVB\ SZRO\ X41} \\
\overline{\mathrm{KO}} &= \mathrm{BNO\ ENDI} + \mathrm{BBK\ X50} + \mathrm{YBGN\ X60} \\
\mathrm{CKO} &= \mathrm{CKKU}
\end{aligned}$$

$$\begin{aligned}
\mathrm{KU} &= \mathrm{KUTME}\ \overline{\mathrm{SZRO}} + \mathrm{IATH}\ \overline{\mathrm{ILD}}\ \mathrm{IKO} \\
&\quad + (\mathrm{SMPB}\ \overline{\mathrm{EZRO}}\ \mathrm{X20})\ [\mathrm{GND}] \\
&\quad + \mathrm{SNS\ DSCRT\ X30} + \mathrm{STPKW\ S2} \\
\overline{\mathrm{KU}} &= \mathrm{RSUSO} \\
\mathrm{CKU} &= \mathrm{CKKU}
\end{aligned}$$

3.35 CORE MEMORY

The internal core memory system is described at three levels. First, the core memory is explained at a block diagram level. At this level, the general operation of the core memory system is defined and functional relationships between components are established.

The read and write cycles are also discussed at a block diagram level. The description of each cycle details how a single cell of an address location is affected by that cycle.

Second, each component of the internal core memory system is explained at either a logic level or at an electronics level, depending upon the operating characteristics of that component. Supplementing these descriptions are logic diagrams, schematics and, where applicable, logic equations.

Before discussing the internal core memory system functionally, it is described physically.

3.36 Physical Description

The basic internal core memory system of the computer physically consists of four printed circuit card assemblies:

a. A core storage card assembly

b. A timing and control card assembly

c. An M-register and odd parity card assembly

d. A diode memory card assembly

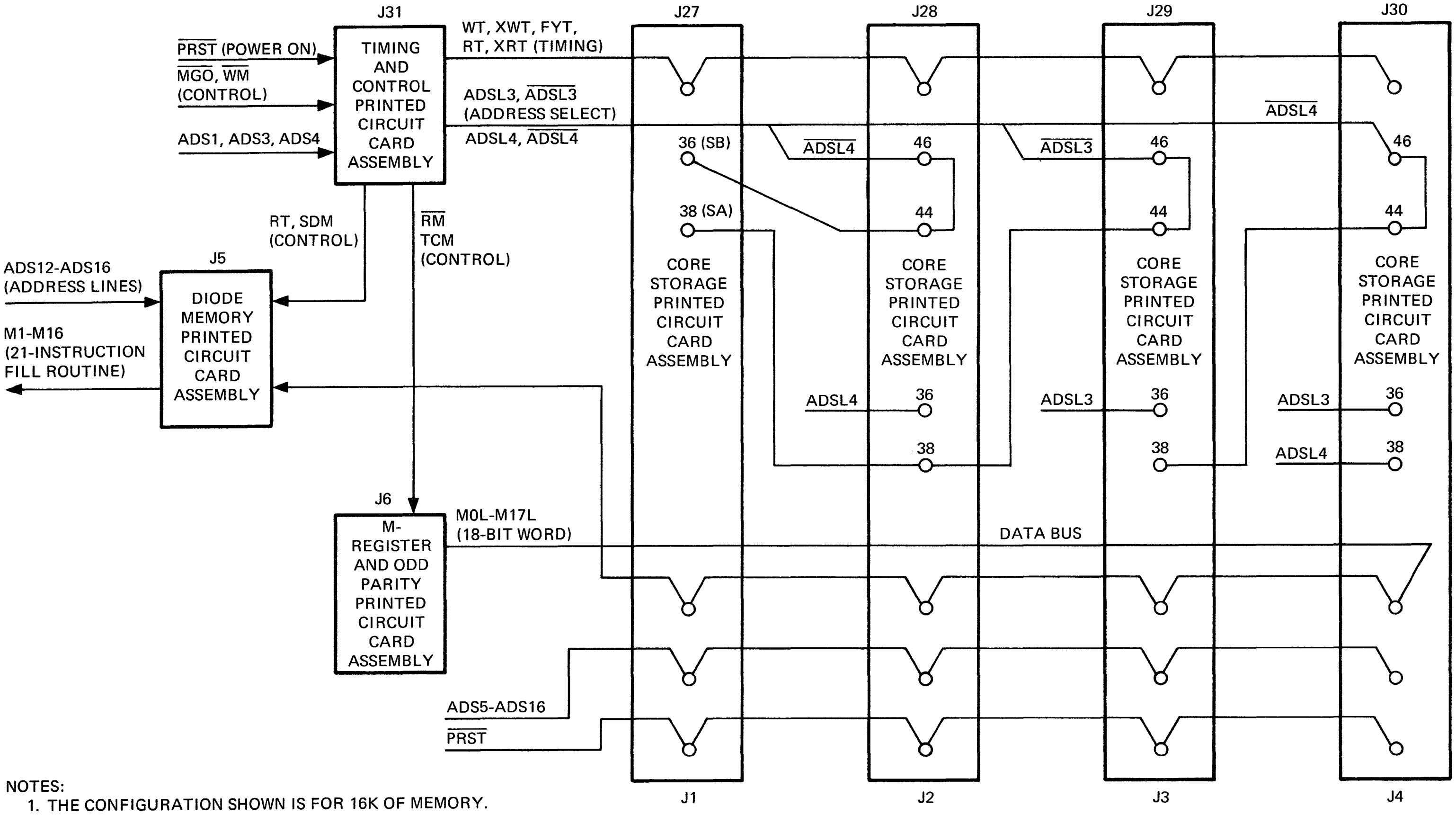
This configuration of assemblies provides 4096 words of internal storage. It can be expanded in blocks of 4096 words to a maximum of 16,384 words. All that is required is to add the desired number of core storage cards to the machine's present internal memory system configuration. The computer is prewired to accept 16,384 words of storage without the need for additional power supplies.

Every storage location in core memory is directly addressable by the central processor. The range of memory addresses extends from X'0000' (first location) to X'3FFF' (last location for a maximum size memory of 16,384 words). Regardless of size, the memory is wraparound, or circular; for example, with a 4096-word memory the next location after location X'0FFF' is location X'0000'.

The signal exchange between the card assemblies that constitute a 16,384-word memory is shown in figure 3-9. The timing and control assembly supplies control signals to each of the other assemblies in the system. The M-register and odd parity assembly accepts the word produced by a particular core storage assembly, transfers an 18-bit word to a selected core storage assembly, checks parity, and generates a parity bit. The diode memory assembly produces the 19-instruction fill routine or a word read from memory.

3.37 Core Storage Card Assembly

A core storage card assembly is shown in figure 3-10. It consists of a core storage array, 48 "stick" assemblies, drive resistors mounted on heat sinks, and integrated circuit (IC) chips.

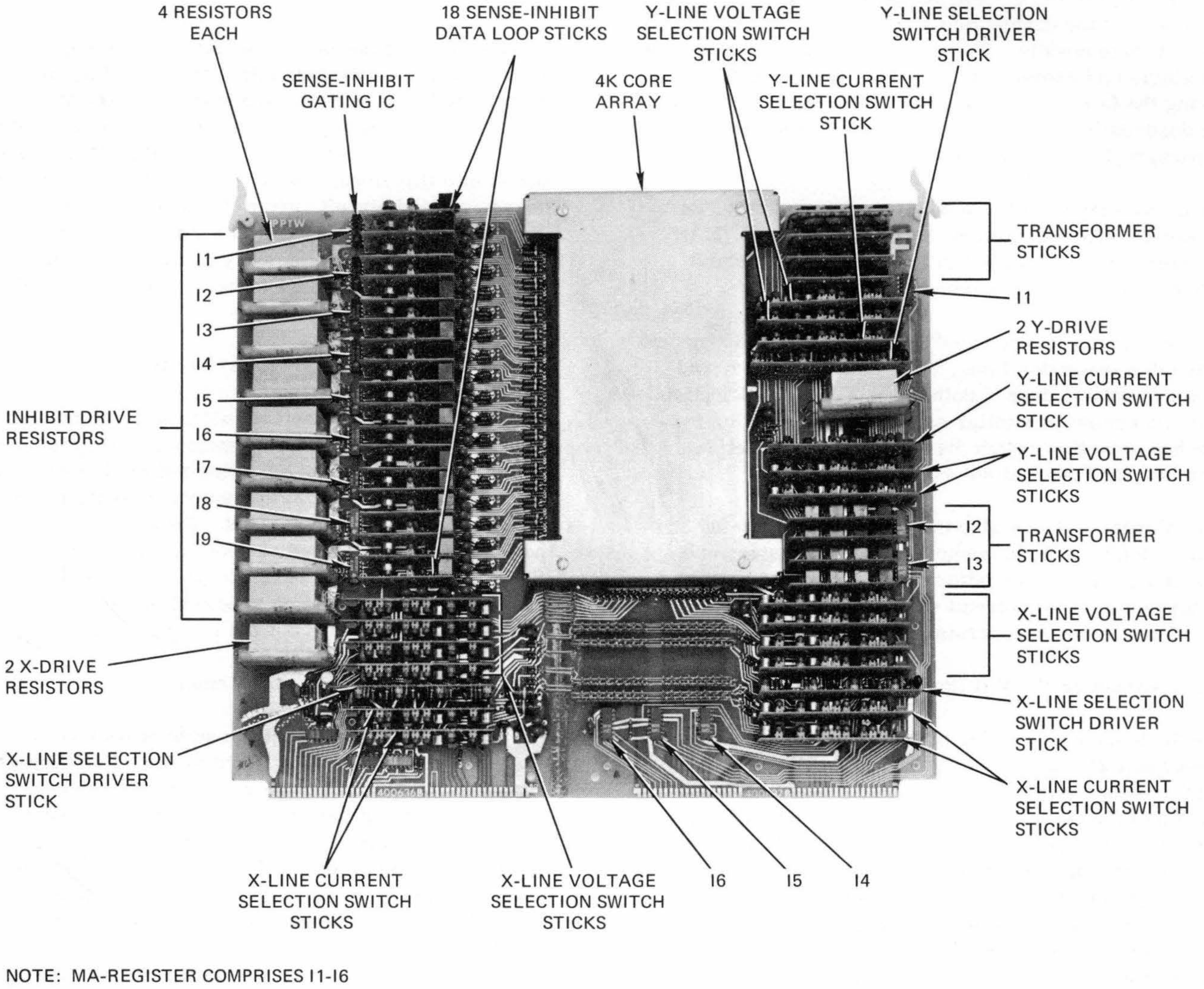

J31
TIMING AND CONTROL PRINTED CIRCUIT CARD ASSEMBLY
$\overline{PRST}$ (POWER ON)
$\overline{MGO}$, $\overline{WM}$ (CONTROL)
ADS1, ADS3, ADS4
WT, XWT, FYT, RT, XRT (TIMING)
ADSL3, $\overline{ADSL3}$ (ADDRESS SELECT)
ADSL4, $\overline{ADSL4}$
RT, SDM (CONTROL)
$\overline{RM}$ TCM (CONTROL)
J5
DIODE MEMORY PRINTED CIRCUIT CARD ASSEMBLY
ADS12-ADS16 (ADDRESS LINES)
M1-M16 (21-INSTRUCTION FILL ROUTINE)
J6
M-REGISTER AND ODD PARITY PRINTED CIRCUIT CARD ASSEMBLY
M0L-M17L (18-BIT WORD)
DATA BUS
ADS5-ADS16
$\overline{PRST}$
J27
J28
J29
J30
36 (SB)
38 (SA)
$\overline{ADSL4}$
$\overline{ADSL3}$
$\overline{ADSL4}$
46
44
46
44
46
44
CORE STORAGE PRINTED CIRCUIT CARD ASSEMBLY
CORE STORAGE PRINTED CIRCUIT CARD ASSEMBLY
CORE STORAGE PRINTED CIRCUIT CARD ASSEMBLY
CORE STORAGE PRINTED CIRCUIT CARD ASSEMBLY
ADSL4
36
38
ADSL3
36
38
ADSL3
36
ADSL4
38
J1
J2
J3
J4

NOTES:

1. THE CONFIGURATION SHOWN IS FOR 16K OF MEMORY.

2. IF THE COMPUTER HAS ONLY 4K OF MEMORY, THEN ONLY THE CORE STORAGE PRINTED CIRCUIT CARD ASSEMBLY IN SLOTS J1 AND J27 EXISTS. ADDITIONAL CORE STORAGE CAN BE ADDED AS SHOWN.

Figure 3-9. Signal Exchange Between Core Memory Card Assemblies


4 RESISTORS EACH
SENSE-INHIBIT GATING IC
18 SENSE-INHIBIT DATA LOOP STICKS
4K CORE ARRAY
Y-LINE VOLTAGE SELECTION SWITCH STICKS
Y-LINE CURRENT SELECTION SWITCH STICK
Y-LINE SELECTION SWITCH DRIVER STICK
TRANSFORMER STICKS
I1
2 Y-DRIVE RESISTORS
Y-LINE CURRENT SELECTION SWITCH STICK
Y-LINE VOLTAGE SELECTION SWITCH STICKS
I2
I3
TRANSFORMER STICKS
X-LINE VOLTAGE SELECTION SWITCH STICKS
X-LINE SELECTION SWITCH DRIVER STICK
X-LINE CURRENT SELECTION SWITCH STICKS
I1
I2
I3
I4
I5
I6
I7
I8
I9
INHIBIT DRIVE RESISTORS
2 X-DRIVE RESISTORS
X-LINE SELECTION SWITCH DRIVER STICK
X-LINE CURRENT SELECTION SWITCH STICKS
X-LINE VOLTAGE SELECTION SWITCH STICKS
I6
I5
I4

NOTE: MA-REGISTER COMPRISES I1-I6

Figure 3-10. Core Storage Card Assembly, Component Locations

The core array assembly is organized as a three-wire, three-D system. It has 18 data bit sections with 128 X-selection lines, 32 Y-selection lines, and 18 sense-inhibit pairs. The core array assembly is folded over such that even data bit sections are on the bottom or solder side of the core storage card assembly, while the odd data bit sections are on the top or component side of the assembly.

The stick assemblies comprise the X- and Y-selection systems, the sense system, and the inhibit system. The X-selection system has two selection switch driver sticks, eight voltage selection switch sticks, and four current selection switch sticks. The Y-selection system includes one selection switch driver stick, four voltage selection switch sticks, two current selection switch sticks, and eight transformer sticks. Mounted on each transformer stick are four transformers.

The sense and inhibit systems share 18 data-loop sticks. Mounted on each stick are a sense amplifier and an inhibit amplifier. An associated IC chip provides inhibit and sense gating. Integrated circuit chips I1 through I6 make up the memory address register.

3.38 Functional Description

The purpose of the internal memory is to store information. This information may represent data or instructions.

The internal memory organization of the computer is shown in figure 3-11. The components shown within the periphery of the dashed line are contained on one memory core storage module. The mnemonic letters, as well as the single cell shown in figure 3-11, are not referred to during the following description. They are used during the descriptions of the read and write cycles presented in paragraphs 3.46 and 3.47.

A core storage card has a 4096-word core array, an X- and Y-selection system, a memory address (MA) register, 18 sense amplifiers, sense gating, inhibit gating, and 18 inhibit drivers.

The core array is organized as a three-wire, 3D system with 128 X-selection lines, 32 Y-selection lines, and 18 sense-inhibit lines. Both of the X- and Y-selection systems comprise similar elements. A selection system has selection switch drivers, voltage selection switches, and current selection switches.

The internal memory also includes memory timing and control logic, a diode memory, the M-register, and odd parity logic. The latter two circuits are contained on the same printed circuit card, while each of the other two circuits are on separate cards.

3.39 Executing the Fill Routine

The diode memory, under program and manual control, produces a 21-instruction fill routine. Pressing the FILL switch on the processor control panel causes the machine to begin executing the fill routine at address X'8000'. The instruction is decoded by address decode, and the five least significant bits of the instruction are supplied to the diode memory. In addition, the diode memory receives from memory timing and control logic the select diode memory signal, derived from the most significant bit of the instruction, and the read time command. When read time occurs, the diode memory generates the first instruction of the fill routine.

This instruction is loaded into the I-register. At the same time, the contents of the L-register, representing address X'8000', are transferred to the adder via the E-input bus. The adder increases the address by a count of one, and the new address, X'8001', is transferred out of the adder and into the L-register. The instruction in the L-register is decoded by address decoding, and the five least significant bits of the instruction are sent to the diode memory. When read time occurs again, the diode memory issues the second instruction of the fill routine. The preceding sequence is repeated until the fill routine has been completed.

3.40 Initiating a Read Cycle

A read cycle is initiated when the central processor generates the memory go and the not memory write commands. These signals activate memory timing and control logic, which generates both the read and write commands during one complete memory cycle.

Two hundred nanoseconds after memory timing and control logic is activated, bits 5 through 16 of the instruction being executed are loaded into the MA-register. Instruction bits 1, 3, and 4 are also applied to memory timing and control logic. Bits 2 and 3 are loaded into flip-flops, whose outputs determine the core storage printed circuit card to be selected.* Bit 1 is also loaded into a flip-flop. The state of this flip-flop determines if the diode memory or the core memory is to be selected. When bit 1 is false, the core memory is selected.

The contents of the MA-register are transmitted to the X- and Y-line selection systems to enable X- and Y-line voltage and current selection switches. The read timing signals are generated first by memory timing and control logic. They activate the selection switch drivers, which in turn close one set of X-line selection switches and one set of Y-line selection switches. Consequently, one of the 32 Y-lines is selected, as well as one of the 128 X-lines. The read currents produced by this action select the memory location specified in the address of the instruction and read out the contents of that location.

3.41 Reading a Word From Memory

The sense amplifiers detect the state of each core via 18 sense lines. The 18-bit word is routed through the amplifiers, coupled through sense gating by a read strobe, and loaded into the M-register. The word is also applied to inhibit gating. When memory timing and control logic produces the write commands, the contents of the M-register are written back into the selected memory location, terminating the read cycle.

At the same time the word is transferred into the M-register, it is also supplied to odd parity logic. This circuit checks the 18 bits of the word for a parity error. If the word has an even number of one bits, odd parity logic generates the memory parity error signal and transmits it to MPTY logic.

3.42 Initiating a Write Cycle

A write cycle is initiated when the central processor generates the memory go and the memory write commands. These signals activate memory timing and control logic which produce the read and write commands.

Before the first read command is released, the 12 least significant bits of the instruction being executed by the

*It is assumed that the machine has more than one core storage card. If the computer has only one of these cards, it would automatically be selected because unit selection decoding would not be required.

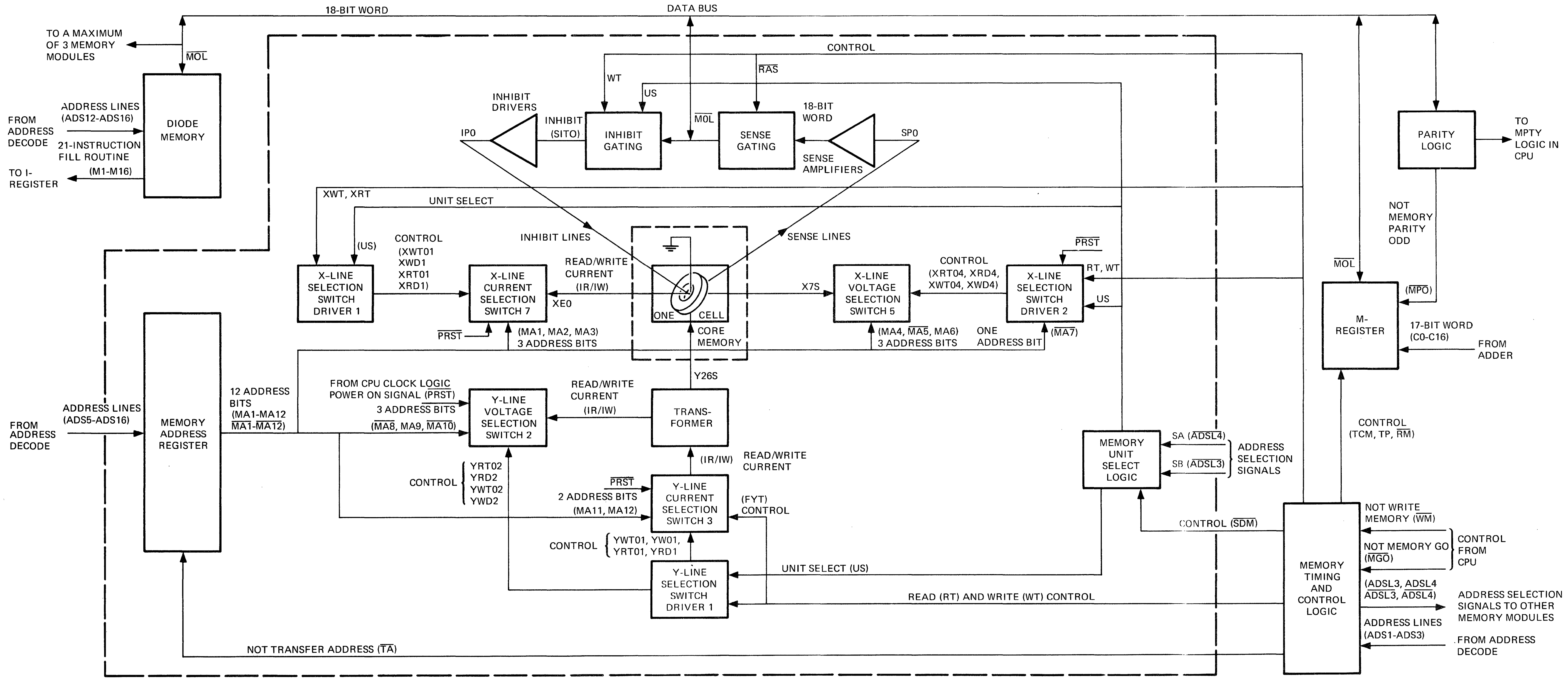

18-BIT WORD
DATA BUS
TO A MAXIMUM OF 3 MEMORY MODULES
MOL
CONTROL
WT
RAS
US
INHIBIT DRIVERS
18-BIT WORD
FROM ADDRESS DECODE
ADDRESS LINES (ADS12-ADS16)
21-INSTRUCTION FILL ROUTINE
DIODE MEMORY
TO I-REGISTER
(M1-M16)
IPO
INHIBIT (SITO)
INHIBIT GATING
MOL
SENSE GATING
SPO
SENSE AMPLIFIERS
PARITY LOGIC
TO MPTY LOGIC IN CPU
XWT, XRT
UNIT SELECT
NOT MEMORY PARITY ODD
(US)
CONTROL (XWT01 XWD1 XRT01 XRD1)
INHIBIT LINES
SENSE LINES
PRST
CONTROL (XRT04, XRD4, XWT04, XWD4)
MOL
X-LINE SELECTION SWITCH DRIVER 1
X-LINE CURRENT SELECTION SWITCH 7
READ/WRITE CURRENT (IR/IW)
XE0
ONE CELL
X7S
X-LINE VOLTAGE SELECTION SWITCH 5
X-LINE SELECTION SWITCH DRIVER 2
RT, WT
US
(MPO)
M-REGISTER
PRST
(MA1, MA2, MA3) 3 ADDRESS BITS
CORE MEMORY
(MA4, MA5, MA6) 3 ADDRESS BITS
ONE ADDRESS BIT
(MA7)
17-BIT WORD (C0-C16)
FROM ADDER
Y26S
12 ADDRESS BITS (MA1-MA12 MA1-MA12)
FROM CPU CLOCK LOGIC POWER ON SIGNAL (PRST)
3 ADDRESS BITS (MA8, MA9, MA10)
Y-LINE VOLTAGE SELECTION SWITCH 2
READ/WRITE CURRENT (IR/IW)
TRANS-FORMER
FROM ADDRESS DECODE
ADDRESS LINES (ADS5-ADS16)
MEMORY ADDRESS REGISTER
CONTROL (TCM, TP, RM)
MEMORY UNIT SELECT LOGIC
SA (ADSL4)
SB (ADSL3)
ADDRESS SELECTION SIGNALS
(IR/IW) READ/WRITE CURRENT
CONTROL YRT02 YRD2 YWT02 YWD2
PRST
2 ADDRESS BITS (MA11, MA12)
Y-LINE CURRENT SELECTION SWITCH 3
(FYT) CONTROL
CONTROL (SDM)
NOT WRITE MEMORY (WM)
NOT MEMORY GO (MGO)
CONTROL FROM CPU
CONTROL YWT01, YW01, YRT01, YRD1
Y-LINE SELECTION SWITCH DRIVER 1
UNIT SELECT (US)
READ (RT) AND WRITE (WT) CONTROL
MEMORY TIMING AND CONTROL LOGIC
(ADSL3, ADSL4 ADSL3, ADSL4)
ADDRESS SELECTION SIGNALS TO OTHER MEMORY MODULES
ADDRESS LINES (ADS1-ADS3)
FROM ADDRESS DECODE
NOT TRANSFER ADDRESS (TA)

Figure 3-11. Reading or Writing into a Single Cell of Location D2F

machine are loaded into the M-register, memory select logic is activated, and the 17-bit output of the adder is transferred to the M-register.

3.43 Producing a Parity Bit

Odd parity logic checks the number of one-bits in the M-register. If there is an even number, odd parity logic generates the not memory parity odd signal. It is sent to the M-register. When memory timing and control logic produces the transfer parity signal, a flip-flop is set in the M-register, adding a one-bit to the word stored in the register.

If there is an odd number of one-bits in the M-register, a zero-bit is added to the contents of the register. The 18-bit output of the M-register is routed to the inputs of inhibit gating.

3.44 Writing a Word into Memory

The output of memory select logic enables selection switch drivers in each of the X- and Y-selection systems. The read commands activate the selection switch drivers, closing one set of X-line selection switches and one set of Y-line selection switches. With these switches closed, one of the 32 Y-lines is selected, and one of the 128 X-lines is selected. At this time a one is written into each core of the selected memory location.

Inhibit gating is activated when memory timing and control logic produces the write time command. If a zero-bit is applied to an inhibit gating input from the M-register, a section of inhibit gating activates and turns on an associated inhibit driver. The output current produced by the driver cancels half the selection current provided by the X- and Y-selection lines. Thus, a zero is written into that particular cell of the selected memory location.

If a one-bit is applied to one of the inputs of inhibit gating, that section of inhibit gating generates a signal that disables an inhibit driver. Therefore, an inhibit current is not produced, and the one stored in the cell associated with the disabled inhibit driver is retained. With the contents of the M-register stored in memory, the write cycle terminates.

3.45 Memory Cycle

The duration of a memory cycle is 860 nanoseconds. During this time, an instruction or data word can be read from or stored into core memory upon command from the central processor.

An 18-bit word (16 data or instruction bits, a parity bit, and a roll tag bit) is fetched from memory during a read cycle. During a write cycle, an 18-bit word is transferred into memory. Subsequent paragraphs describe each of these operations at a block diagram level. The description of each cycle details how a single cell of an addressed memory location is affected by that cycle. A block diagram and a timing diagram are provided in figures 3-11 and 3-12, respectively. Refer to these figures during the following discussions.

3.46 Read Cycle

Consider that power has been applied to the computer. The computer clock generates $\overline{\text{PRST}}$, indicating that power is on. Signal $\overline{\text{PRST}}$ is routed to the X- and Y-line voltage and current selection switches. Also consider that the machine has executed a single-word instruction with an address mode of hexadecimal D and a reference address of hexadecimal 2F. The output of the L-register is decoded by address decoding, producing signals ADS5 through ADS16. These 12 address bits are transmitted to the memory address (MA) register.

MGO and $\overline{\text{WM}}$ Are Released. A read cycle is initiated when the central processor generates the memory go signal (MGO) and the not write memory command ($\overline{\text{WM}}$). Signal $\overline{\text{WM}}$, with the complement of MGO, is sent to memory timing and control logic to produce $\overline{\text{RM}}$ and $\overline{\text{TA}}$. The former signal is applied to the M-register and clears the register to zero. The latter signal, which is the negation of the transfer address signal, is routed to the MA-register, inverted, and loads the 12 least significant bits of the instruction word into the register.

Outputs MA1 through MA3 of the MA-register are applied to X-line current selection switch 7. The next three outputs of the register (MA4, $\overline{\text{MA5}}$, and MA6) are routed to X-line voltage selection switch 5. A single address bit, $\overline{\text{MA7}}$, is delivered to X-line selection switch driver 2. Outputs $\overline{\text{MA8}}$, MA9, and $\overline{\text{MA10}}$ are supplied to Y-line voltage selection switch 2. The last two outputs of the MA-register (MA11 and MA12) are transmitted to Y-line current selection switch 3. Thus, seven address bits are used to enable the X-selection line, while five address bits are used to enable the Y-selection line.

RT and FYT Are Produced. Approximately 60 nanoseconds after the MA-register is loaded, memory timing and control logic generates the read time (RT) and the fine Y-time (FYT) signals. The fine Y-time signal is sent to Y-line current selection switch 1. Signal RT, with the memory unit select (US) command,* is directly coupled to X-line selection switch driver 2, producing signals XRT04 and XRD4. These signals close X-line voltage selection switch 5.

At the same time, RT and US are supplied to Y-line selection switch driver 1. This circuit activates and

*When US is true (+5V), it indicates that a memory core storage module has been selected for use. A module is selected when the module decodes address lines ADS3 and ADS4.

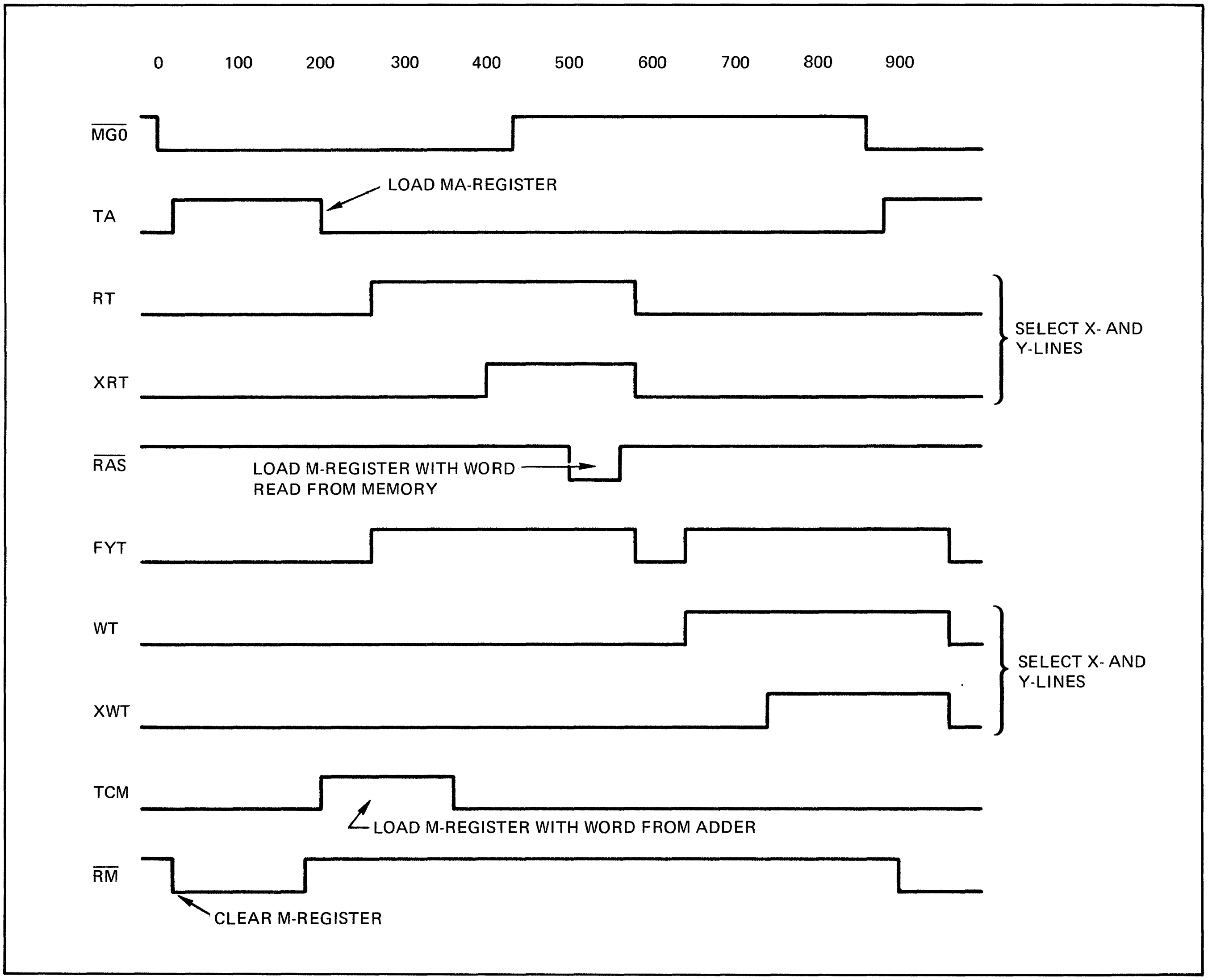


Figure 3-12. Memory Cycle, Timing Diagram

generates YRT01, YRD1, YRT02, and YRD2. Signals YRT01 and YRD1 close Y-line current selection switch 3; the other signals close Y-line voltage selection switch 2. With these switches closed, a read current, designated IR, flows through the primary of a transformer. An expanding flux field induces a voltage in the secondary of the transformer, causing a current to flow in one of the 32 Y-selection lines.

Selected Location Is Read. One hundred and forty nanoseconds later, memory timing and control logic generates signal XRT. Signals XRT and US activate X-line selection switch driver 1. The XRT01 and XRD1 outputs of that driver circuit close X-line current selection switch 7. A read current now flows in one of the 128 X-selection lines. The two read currents pass through each of the selected cells. If a core is in the one state, the currents change the magnetization of the core, setting the core to the zero state. The collapsing field of the core induces a voltage into the sense winding. This voltage, representing a binary one, is transmitted to the input of a sense amplifier.

Application of the read currents to a core in the zero state induces a negligible voltage pulse in the sense winding. Thus, the sense amplifier detects a binary zero. The readout of a core is destructive. When information is read from a core, it remains in the zero state.

$\overline{\text{RAS}}$ Is Generated. The output of the sense amplifier is applied to sense gating. Approximately 100 nanoseconds after signal XRT occurs, memory timing and control logic generates the sense amplifier strobe ($\overline{\text{RAS}}$). It is routed to sense gating, inverted, and strobes the output of the sense amplifier through sense gating. The output of the amplifier is loaded into the M-register and is applied to inhibit gating.

After the 18-bit word is stored in the M-register, memory timing and control logic completes generating the read time commands and automatically produces the write time commands. The first signals generated are write time (WT) and FYT.

The write time command activates Y-line selection switch driver 1 and X-line selection switch driver 2. The former driver produces signals that close Y-line current selection switch 3 (YWT01 and YWD1) and Y-line voltage selection switch 2 (YWT02 and YWD2). The latter driver generates XWT04 and XWD4 which close X-line voltage selection switch 5.

XWT Is Released. With the Y-line selection switches closed, the previously selected Y-line is again activated because the contents of the MA-register have not changed. One hundred nanoseconds later, memory timing and control logic generates XWT. Signal XWT activates X-line selection switch driver 1, producing outputs XWT01 and XWD1. These outputs close X-line current selection switch 7 and permit a current, called IW, to flow through the closed X-line selection switches.* The coincident current writes a one into the cell.

A zero is written into a cell when the associated bit in the M-register is a zero. If the bit is a zero the inhibit gating circuit associated with that cell is activated. The 0V output of the circuit turns on an inhibit driver, generating an inhibit current. The magnitude of the inhibit current is half of the combined X- and Y-write currents. Thus, half of the coincident current is canceled, and a zero is written into the cell. The read cycle now terminates with the original 18-bit word stored in location X'D2F'.

3.47 Write Cycle

The write cycle is similar to the read cycle because the sequence of control signals generated by memory timing and control does not change. Upon command from the central processor, 0V levels are generated for signals $\overline{\text{MGO}}$ and $\overline{\text{WM}}$, initiating the write cycle. These commands activate memory timing and control logic, which produces $\overline{\text{RM}}$, $\overline{\text{TA}}$, and TCM. The first signal clears the M-register to zero. The second signal loads the 12 most significant bits of the instruction being executed into the MA-register. The third signal transfers the 17-bit output of the adder (C0 through C16) into the M-register. The false output of each M-register flip-flop is applied to an input of inhibit gating via the data bus.

Location X'D2F' Is Selected. When the M-register is loaded, memory timing and control logic produces signal RT. Signal RT, with US, activates X-line selection switch driver 2 and Y-line selection switch driver 1. The outputs produced by the former switch driver close X-line voltage selection switch 5. The outputs generated by the latter switch driver close both the Y-line voltage and current selection switches. With both of these switches closed, a current flows through them which causes a transformer to produce a current in one of the 32 Y-selection lines.

One hundred and forty nanoseconds later, memory timing and control logic generates signal XRT. Signals XRT and US activate X-line selection switch driver 1, closing X-line current selection switch 7. A read current now flows in one of the 128 X-selection lines. Both the X- and Y-read currents pass through location X'D2F' and select that location. Thus, a zero is initially written into the core shown in figure 3-11, as well as the other 17 selected cores.

Inhibit Current Is Produced. If the bit applied to an input of inhibit gating is a zero-bit, that section of inhibit gating activates when memory timing and control logic generates signal WT. This output of inhibit gating turns on an inhibit driver. The inhibit driver produces a current that is delivered to the selected cell. At the same time, one of the X-selection lines is enabled, and one of the Y-selection lines is activated, producing a write current. The enabled X-selection line is activated when memory timing and control logic releases XWT.

Both write currents now flow through the core.* However, the inhibit current cancels out half of the selection currents. Therefore, the state of the core does not change; that is, it is still zero. If a one-bit is applied to inhibit gating, a one is written into the cell, because the inhibit current is not generated. Thus, at the end of the write cycle, the bit stored in the cell is the same as the associated bit in the M-register.

3.48 Diode Memory

The output of the diode memory represents either one of the fill routine instructions or the contents of the M-register, depending upon the instruction being executed by the computer.

Refer to figure 3-13. The diode memory consists of gating, a decoding circuit, inverters, and a memory circuit. The 21-instruction fill routine is hard wired into the memory circuit.

The diode memory is described in two parts. The first part explains how the 21-instruction fill routine is generated. The second part describes how the contents of the M-register are routed through the diode memory.

Consider that the operator has applied power to the machine and has pressed the FILL pushbutton switch

*Currents IW and IR flow in opposite directions.

*Actually the X- and Y-write currents flow through each of the cores that constitute location X'D2F'.

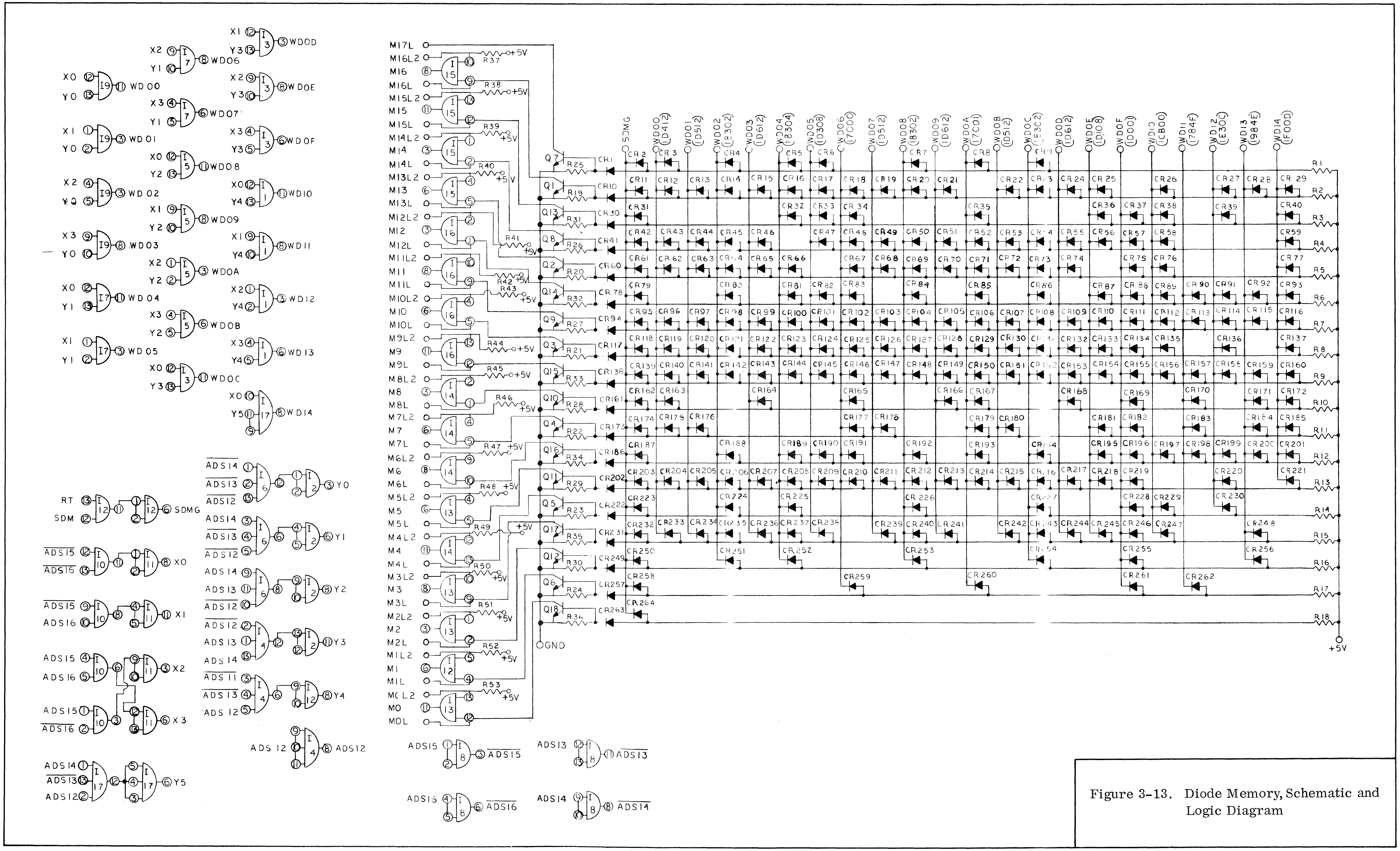


Figure 3-13. Diode Memory, Schematic and Logic Diagram

on the processor control panel. The computer begins executing the fill procedure at address X'8000'. The instruction is decoded by address decode. The most significant bit of the instruction (ADS1) is sent to memory timing and control logic and is applied to the input of the select diode memory (SDM) flip-flop, enabling that flip-flop.

The five least significant bits of the instruction (ADS12 through ADS16), which are all zero-bits, are supplied to the diode memory, inverted, and delivered to a decoding circuit. Nand gates $\overline{\overline{\text{ADS15}}\ \overline{\text{ADS16}}}$ and $\overline{\overline{\text{ADS12}}\ \overline{\text{ADS13}}\ \overline{\text{ADS15}}}$ of this circuit activate, generating signals X0 and Y0. These signals are directly coupled to Nand gate $\overline{\text{X0 Y0}}$. The gate activates and produces signal WD00. The output of the gate is routed to the memory circuit and is applied to the cathodes of the diodes connected to the WD00 line.

Approximately 200 nanoseconds after the read cycle has started, flip-flop SDM sets. The true output of the flip-flop (SDM) is transmitted to the diode memory and enables Nand gate $\overline{\text{RT SDM}}$. Sixty nanoseconds later, the gate is activated because memory timing and control logic releases signal RT. The output of the gate is inverted and is applied to input SDMG of the diode memory.

Signal SDMG, which is at a +5V level, cuts off the 18 diodes in the SDMG line. Thus, the diodes in the WD00 line conduct. A 0V signal is coupled through these diodes and cuts off other diodes, whose cathodes are common to the emitter and base circuits of transistors Q1 through Q4, Q8, Q9, Q11, Q15, and Q17. All of these transistors are cut off. Thus, the collector voltage of each transistor increases to +5V, which is equivalent to a binary one.

The other nine transistors are all conducting because the diodes associated with them are forward biased. The bit produced by each of these transistors represents a binary zero.

The outputs generated by eight of these transistors,* as well as the other nine transistors, are complemented (M0 through M16) and are transmitted to the I-register. This 17-bit output is the first instruction of the fill routine and has a hexadecimal bit configuration of 1D512. This configuration is a Sense instruction for testing the status of the I/O device having an address of X'12'.

The instruction is loaded into the I-register. At the same time, the contents of the L-register, representing address X'8000', are transferred to the adder via the E-input bus. The adder increases the address by a count of one, and the new address, X'8001', is transferred out of the adder and is loaded into the L-register. The instruction in the L-register is decoded by address decode, and the five least significant bits of the instruction are again supplied to the diode memory. The preceding sequence is repeated again and again until the 21-instruction fill routine has been generated.

When the most significant bit of an instruction is a zero-bit and the central processor releases signals MGO and WM or $\overline{\text{WM}}$, the core memory is selected for use. Two hundred nanoseconds after the memory cycle begins, a zero is loaded into the select diode memory flip-flop. Signal SDM, which is false, is applied to the diode memory and disables line SDMG. Thus, transistors Q1 through Q18 in the memory circuit are all cut off, inhibiting the WDXX inputs.

The false outputs of the M-register ($\overline{\text{M0L}}$ through $\overline{\text{M16L}}$) are also applied to the output inverters of the memory element. Thus, if a word is stored in the M-register, the complement of the word is routed to the diode memory via the data bus, inverted, and transmitted to either the I-register or to the adder. The destination of the word is a function of the instruction being executed by the computer.

3.49 Memory Timing and Control Logic

Memory timing and control logic performs the following functions:

a. Generates the read commands, then the write commands during each memory cycle

b. Produces the transfer parity (TP) signal during the write cycle

c. Releases the transfer-C-bits-to-M-register command (TCM) during the write cycle

d. Generates the M-register reset command ($\overline{\text{RM}}$) during each memory cycle

e. Produces the read sense strobe ($\overline{\text{RAS}}$) during the read cycle

f. Generates the select diode memory command (SMD) during the read cycle if the FILL pushbutton switch is pressed

g. Produces the unit select commands during each memory cycle if the diode memory has not been selected

Refer to figure 3-14. Memory timing and control logic comprises gating, three flip-flops, nine one-shots and a reference supply. Circuit timing is provided in figure 3-15.

*The output of Q7 (M17L) is supplied to the M-register flip-flop that produces the parity bit.

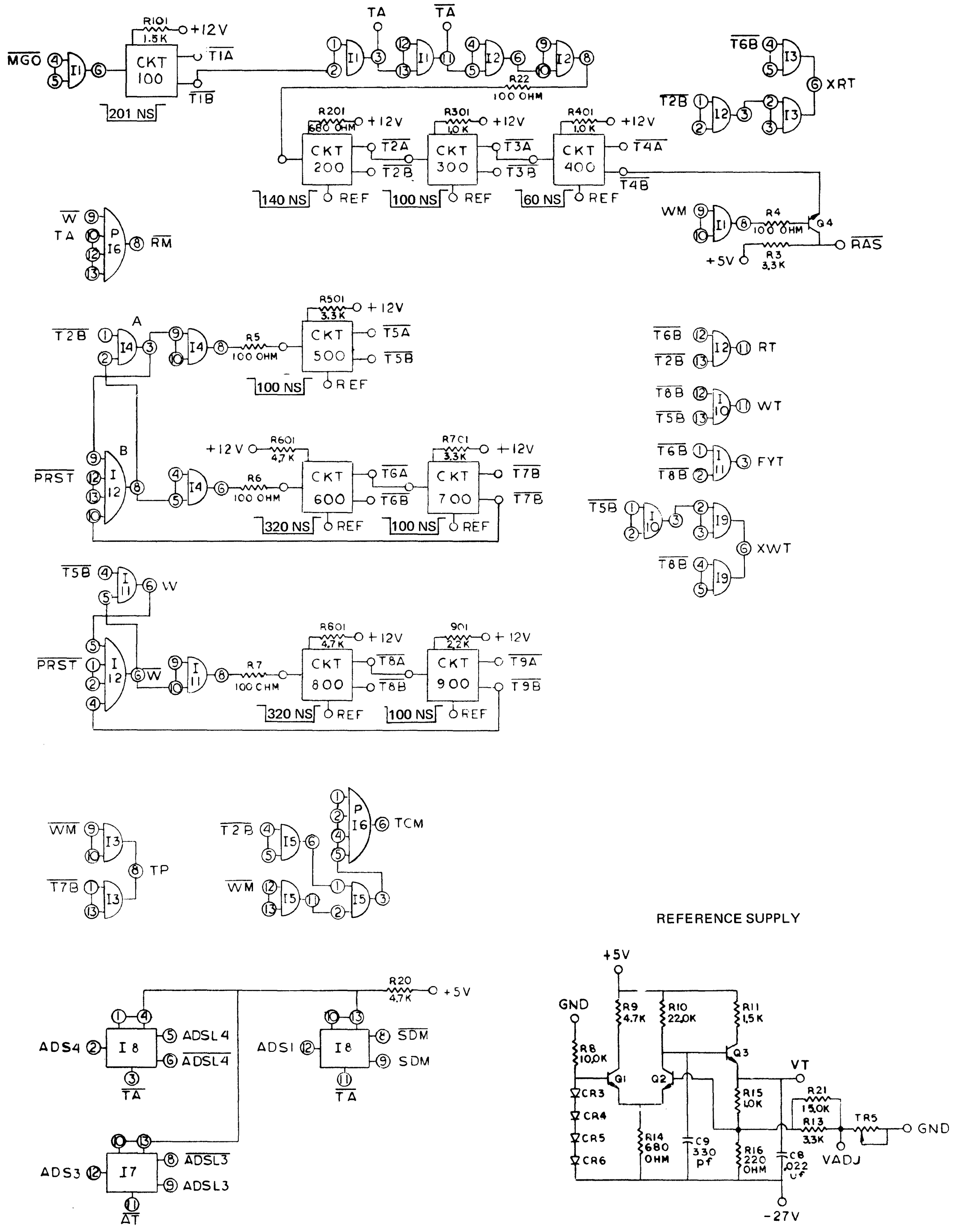


Figure 3-14. Memory Timing and Control Logic

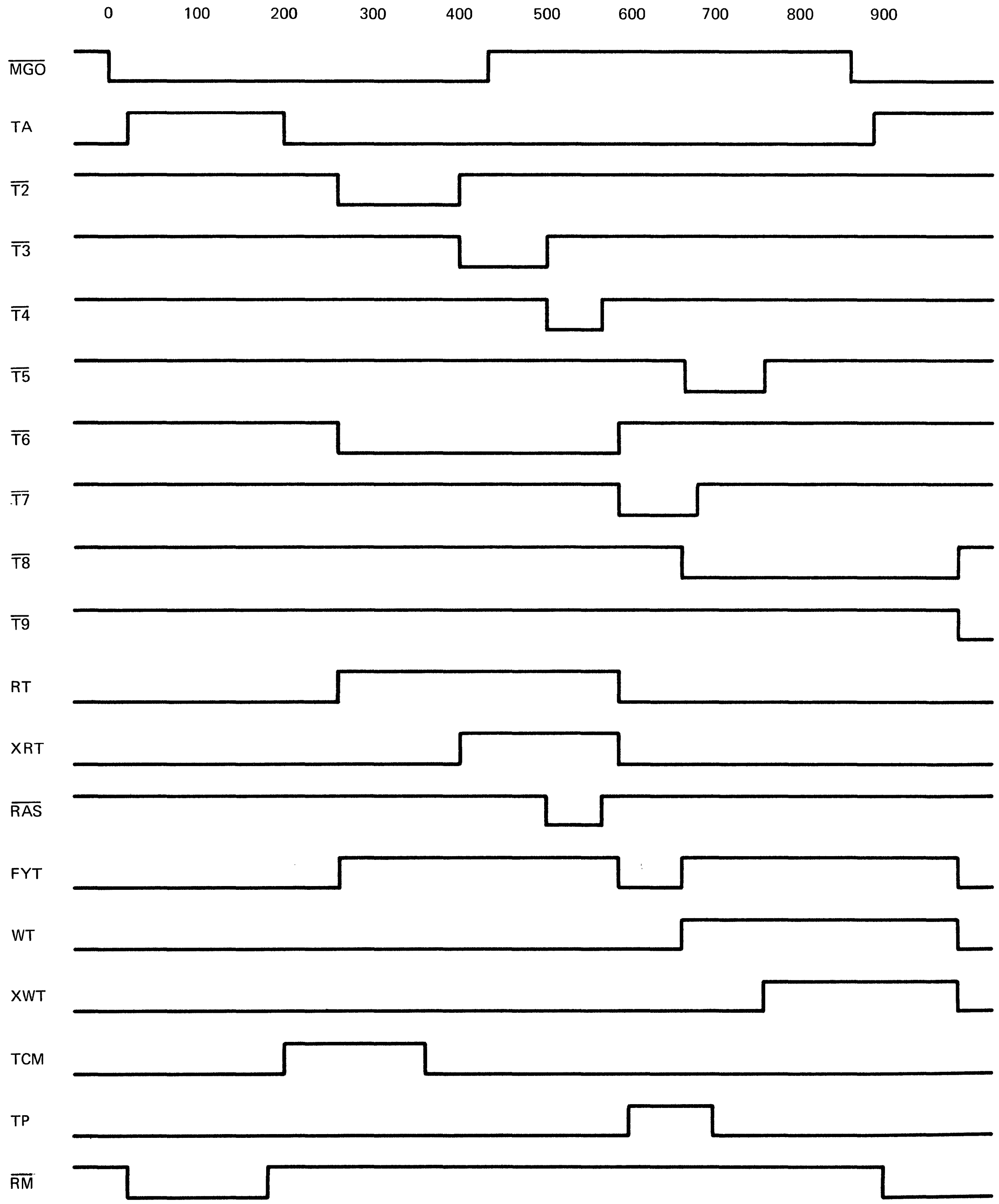


Figure 3-15. Memory Timing and Control Logic, Timing Diagram

3.50 Memory Cycle Is Initiated

Memory timing and control logic is activated when the central processor releases a low signal (0V) for not memory go ($\overline{\text{MGO}}$) or either a low or a high signal (+5V) for write memory (WM). If signal WM is high, a write cycle is executed. If WM is low, a read cycle is performed. Consider that signal WM is low.

When the central processor generates $\overline{\text{MGO}}$, it is supplied to memory timing and control logic, inverted, and applied to one-shot T1. On the negative excursion of $\overline{\text{MGO}}$, one-shot T1 fires for 200 nanoseconds. The output of the one-shot, which is a negative-going pulse, is coupled through four inverters and is sent to the input of one-shot T2.

After one-shot T1 is fired, bits 1, 3, and 4 of the instruction being executed are routed to the set inputs of flip-flops SDM, ADSL3, and ADSL4, respectively. If the diode memory has been programmed to generate the fill routine, bit 1 (ADS1) is true, and bits 3 (ADS3) and 4 (ADS4) are false. When the output of inverter $\overline{\text{TA}}$ goes true, these bits are loaded into the flip-flops. The true output of flip-flop SDM activates the diode memory; the false output of the flip-flop ($\overline{\text{SDM}}$) disables the core memory.

3.51 Core Memory Is Selected

Core memory is selected when signal SDM is false, or low, and any combination of ones and zeros exists for ADS3 and ADS4. (It is assumed that the machine has four core storage cards.) The contents of these flip-flops are delivered to the memory select logic circuit on each of the core storage cards. One of the four circuits decodes these signals and selects that circuit card for use.

Signal $\overline{\text{TA}}$ is also routed to the MA-register and is inverted (TAB). This signal loads the 12 least significant bits of the instruction being executed into the register. One-shot T2 is triggered on the trailing edge of $\overline{\text{TA}}$ and generates a 140-nanosecond, negative-going pulse. The output of the one-shot is applied to Nand gate $\overline{\text{T6}}$ $\overline{\text{T2}}$, producing signal RT. Signal RT is transmitted to the selection switch drivers in the X- and Y-selection systems.

The negative pulse produced by one-shot T2 also activates Nand gate A, which in turn activates Nand gate B. The output of Nand gate B is inverted and fires one-shot T6. The 320-nanosecond, negative-going output of the one-shot is supplied to Nand gate $\overline{\text{T6}}$ $\overline{\text{T2}}$ and to Nand gate $\overline{\text{T6}}$ $\overline{\text{T8}}$. The latter gate produces the fine Y-time signal (FYT), which is sent to the current selection switches in the Y-selection system.

When signal $\overline{\text{T2}}$ goes positive, one-shot T3 fires and And gate $\overline{\text{T2}}$ T6 activates, generating signal XRT. Signal XRT is applied to the X-line selection switch drivers. The read time signal (RT) is not affected by $\overline{\text{T2}}$ because $\overline{\text{T6}}$ is still at a 0V level. Thus, the X- and Y-selection systems select a location in memory.

3.52 $\overline{\text{RAS}}$ Goes Low

The 100-nanosecond, negative-going pulse produced by one-shot T3 is supplied to the input of one-shot T4. The one-shot fires for 60 nanoseconds. The 0V output of the one-shot is directly coupled to the emitter of Q4. Since signal WM is low, the transistor is forward biased, causing $\overline{\text{RAS}}$ to go low for 60 nanoseconds. During this time, the contents of the selected memory location are read out of memory and are stored in the M-register.

At the end of $\overline{\text{T6}}$ time, one-shot T7 fires and produces a 100-nanosecond, negative-going pulse. Signals RT, XRT, and FYT also go low.

The output of one-shot T7 disables Nand gate B, which in turn activates Nand gate A. The output of Nand gate A is inverted and triggers one-shot T5. The 100-nanosecond, negative-going output of the one-shot is sent to Nand gates $\overline{\text{T8}}$ $\overline{\text{T5}}$ and W. The former Nand gate generates signal WT, which activates inhibit gating and the Y-line selection switch drivers. The latter Nand gate activates Nand gate $\overline{\text{W}}$ because the power-on signal ($\overline{\text{PRST}}$) is high and one-shot T9 is disabled.

The 0V signal produced by Nand gate $\overline{\text{W}}$ is inverted and fires one-shot T8 for 320 nanoseconds. The negative-going output of the one-shot causes signal $\overline{\text{FYT}}$ to go true again. Signal FYT is routed to the Y-line current selection switches.

Approximately 100 nanoseconds after one-shot T5 fires, output $\overline{\text{T5}}$ goes high and activates And gate $\overline{\text{T5}}$ T8. This gate produces signal XWT which activates the X-line selection switches. At this time, the word stored in the M-register is written back into the location specified in the address field of the instruction being executed by the machine.

The read cycle terminates when signal $\overline{\text{T8}}$ goes high. Signal $\overline{\text{T8}}$ fires one-shot T9 and disables the Nand gates that generate signals WT and FYT and the And gate that produces signal XWT. The output of one-shot T9 activates a gate, permitting signal $\overline{\text{W}}$ to go true.

3.53 TP and TCM Are Generated

Memory timing and control logic functions in the same manner during a write cycle as during the read cycle, except that signals TP and TCM are generated, while signal $\overline{\text{RAS}}$ is not. At the beginning of the write cycle, the central processor releases a 0V level for signal $\overline{\text{WM}}$. Signal $\overline{\text{WM}}$ is transmitted to memory timing and control logic.

When one-shot T2 fires, its output is gated with $\overline{\text{WM}}$, generating signal TCM for the duration of $\overline{\text{T2}}$. Signal

TCM is sent to the M-register and loads the output of the adder into the register.

Approximately 240 nanoseconds later, one-shot T7 fires. The 0V output of the one-shot is gated with signal $\overline{\text{WM}}$ to produce signal TP. The transfer parity command is routed to the M-register and loads the parity bit, generated by odd parity logic, into latch circuit M17L of the M-register. The parity bit, as well as the other 17 bits stored in the M-register, is written into the selected memory location when memory timing and control logic produces the write commands.

3.54 Reference Supply

The reference supply provides a temperature-controlled voltage (VADJ) to the -27V power supply. Signal VADJ controls the output of the power supply so that the currents through the memory stacks are compensated for temperature variations. This control maintains the proper signal-to-noise ratio over the full temperature range of the memory stacks.

The reference supply is shown in figure 3-14. It consists of an amplifier (Q1 through Q3) and temperature sensitive diodes (CR3 through CR6).

A reference voltage is established at the base of Q1 by diodes CR3 through CR6. The voltage is tracked by the output of the amplifier (VADJ). When the temperature changes, the voltage at the amplifier's output changes. For example, if the temperature increases, the voltage at the base of Q1 decreases and VADJ decreases. This change in voltage is transmitted to the -27V power supply and controls the operation of that supply.

3.55 M-Register

The M-register stores the 18-bit word read from or written into memory.

The M-register, shown in figure 3-16, consists of 18 latch circuits, designated M0L through M17L. Latch circuits M1L through M16L store the data or instruction word. The roll tag bit and the parity bit are stored in latch circuits M0L and M17L, respectively.

Approximately 20 nanoseconds after the memory cycle starts, the M-register is cleared to zero. At this time, memory timing and control logic generates a 0V level for the not read memory ($\overline{\text{RM}}$) command. It is supplied to the reset input of the latch circuits. Since input TCM of each latch circuit is low, signal $\overline{\text{RM}}$ resets the 17 latch circuits.

When a word is going to be written into memory, the 17-bit output of the adder (C0 through C16) is transmitted to the M-register. One hundred and eighty nanoseconds after the M-register is cleared to zero, memory timing and control logic generates transfer command TCM and supplies it to the M-register. On the leading edge of TCM, the C bits are loaded into the M-register.

The true and false outputs of each latch circuit are sent to odd parity logic. If there are an even number of one-bits stored in the M-register, the $\overline{\text{MPO}}$ input of latch circuit M17L goes high. When memory timing and control logic produces a +5V level for signal TP, latch circuit M17L sets, thus adding an odd parity bit to the word stored in the M-register.

The false outputs of each latch circuit ($\overline{\text{M0L}}$ through $\overline{\text{M17L}}$) are also routed to the diode memory and to inhibit gating. When memory timing and control logic releases the write commands, the 18-bit word in the M-register is written into core memory.

During the read cycle, the contents of the selected memory location are strobed through sense gating and are forced into the M-register via the false outputs of the latch circuits. The contents of the M-register are delivered to odd parity logic and are checked for a parity error.

The following equations define the operation of each latch circuit:

$$M0L_s = (C0\ TCM) + M0L$$
$$M0L_r = RM\ (\overline{TCM} + \overline{C0})$$

$$M1L_s = (C1\ TCM) + M1L$$
$$M1L_r = RM\ (\overline{TCM} + \overline{C1})$$

$$\vdots \qquad\qquad \vdots$$

$$M16L_s = (C16\ TCM) + M16L$$
$$M16L_r = RM\ (\overline{TCM} + \overline{M16L})$$

$$M17L_s = (\overline{MPO}\ TP) + M17L$$
$$M17L_r = RM\ (MPO + \overline{TP})$$

3.56 Odd Parity Logic

Odd parity logic performs two functions:

a. During a write cycle, odd parity logic generates a one-bit if the word stored in the M-register has an even number of one-bits, or a zero-bit if the word stored in the M-register has an odd number of one-bits. The generated bit is sent to latch circuit M17L of the M-register.

b. During a read cycle, odd parity logic produces the parity error signal (MPE) if it detects an even number of one-bits in the M-register.

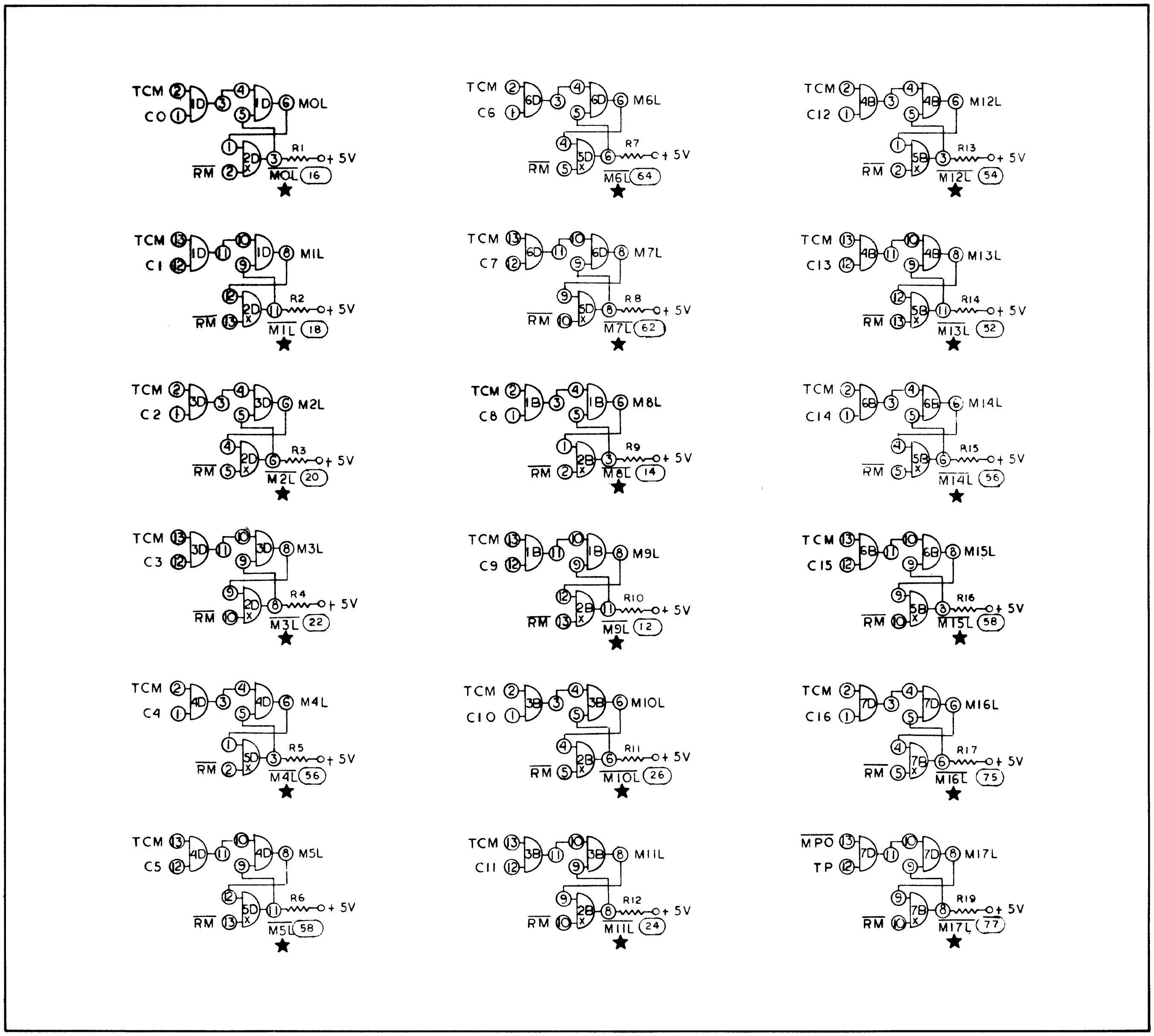


Figure 3-16. M-Register, Logic Diagram

A logic block diagram of odd parity logic is shown in figure 3-17. This circuit consists of 17 exclusive-Or gates. The output of each gate is inverted, except the gate that produces signal MPE.

The following description is divided into two parts. The first part explains how a parity error is detected during a read cycle. The second part describes how a parity bit is generated during a write cycle.

Consider, for example, that a word is read out of memory and that bit 3 of the word is lost during the transfer process (see figure 3-18). The word is loaded into the M-register during $\overline{\text{RAS}}$ time. The true and false outputs of the M-register latch circuits, except latch circuits M16L and M17L, are applied to circuits 1 through 8.

Circuit 1 generates a low output (0V) for $\overline{\text{MA}}$. The other seven circuits ($\overline{\text{MB}}$ through $\overline{\text{MJ}}$) produce high outputs (+5V). These outputs and their complements are transmitted to circuits 9 through 12. Circuit 9 produces a low output for 011, while circuits 10 through 12 generate high outputs for 012 through 014.

The outputs produced by these circuits and their complements are routed to circuits 13 and 14. The former circuit generates a 0V output for MK; the latter circuit

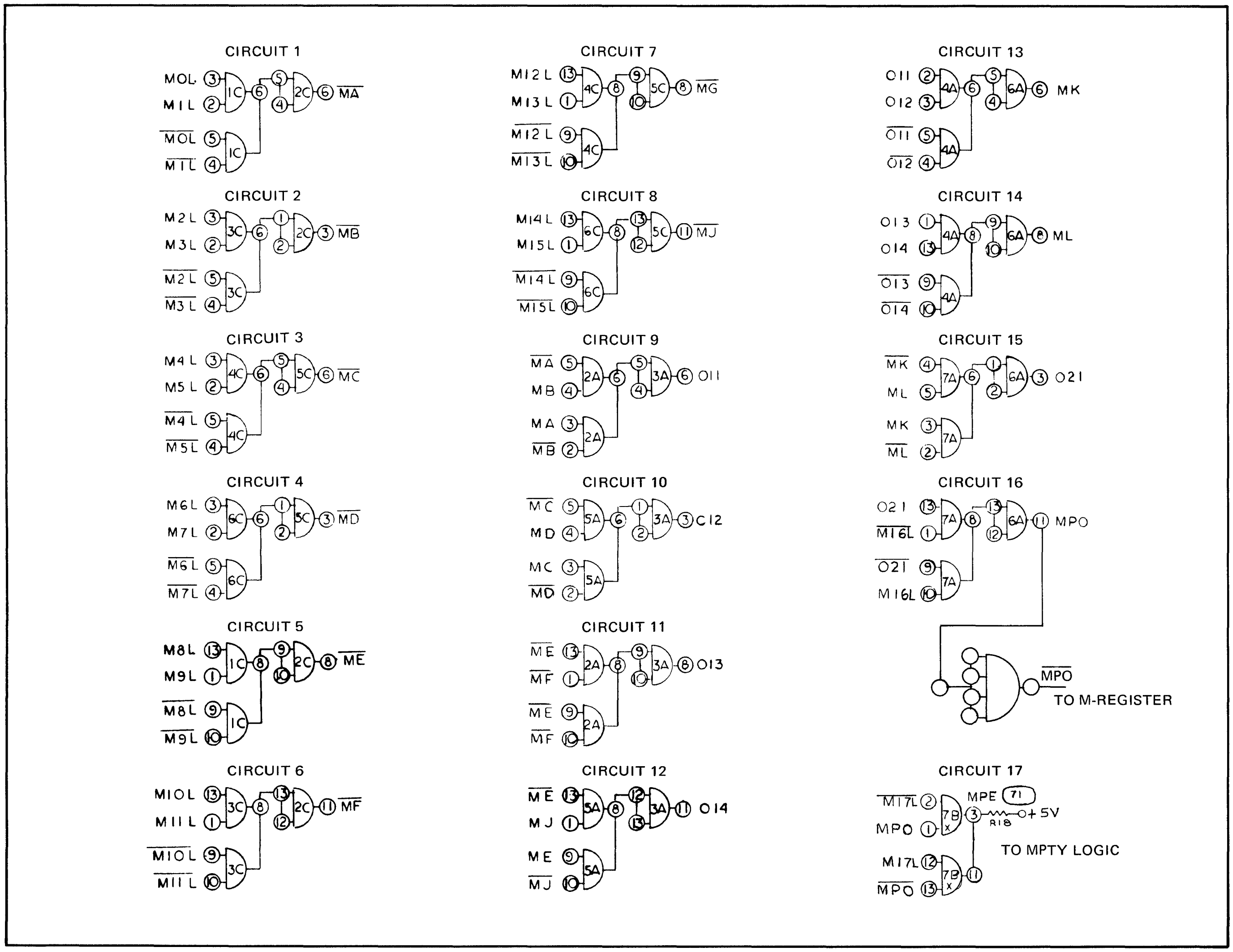


Figure 3-17. Odd Parity Logic

produces a +5V output for ML. Signals MK, $\overline{MK}$, ML, and $\overline{ML}$ activate circuit 15, generating a high output for 021. This output and its complement are applied to circuit 16.

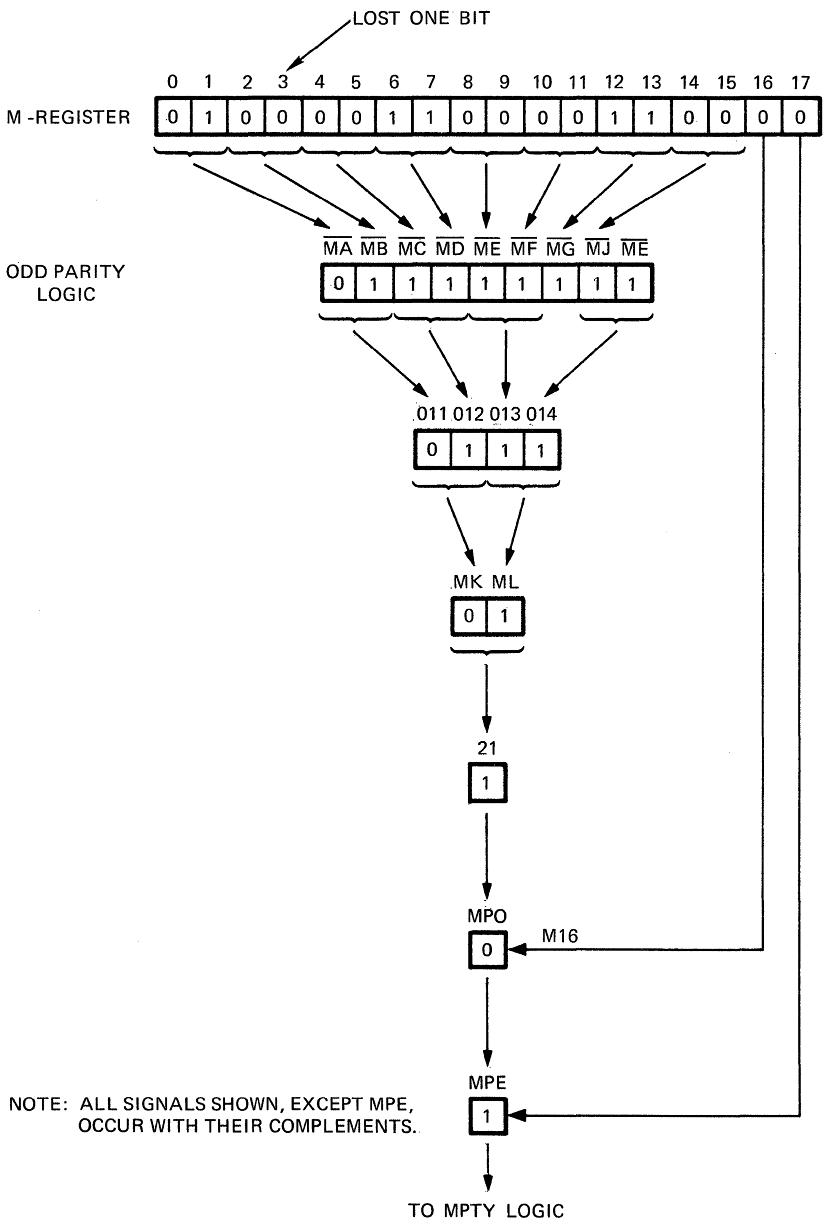
The least significant bit (M16L) of the word stored in the M-register is supplied, with its complement, to circuit 16. The inputs to circuit 16 disable that circuit and cause it to generate a 0V level for MPO. Signals MPO and $\overline{MPO}$ are delivered to circuit 17. These signals are exclusively Ored with the contents of latch circuit M17L of the M-register. Since M17L is normally reset during a read cycle, circuit 17 activates and produces the memory parity error signal. Signal MPE is transmitted to MPTY logic, indicating that a parity error exists.

During a write cycle, the 17-bit output of the adder is directly coupled to the M-register. At TCM time, the word is loaded into the register. If there are an even number of bits stored in the M-register, signal MPO goes low (figure 3-19) in the same fashion described in the preceding paragraphs. Signal MPO is inverted and is sent to latch circuit M17L of the M-register. When memory timing and control logic produces signal TP, latch circuit M17L is set, loading a one into that circuit. Thus, a one parity bit is transferred into memory with the other 17 bits stored in the M-register.

If an odd number of one-bits is transferred to the M-register, signal MPO goes high (see figure 3-20). The memory parity odd signal is complemented and is supplied to the M-register. Thus, when transfer parity time occurs, a zero-bit is added to the bits stored in the M-register and is written into memory at the proper time.

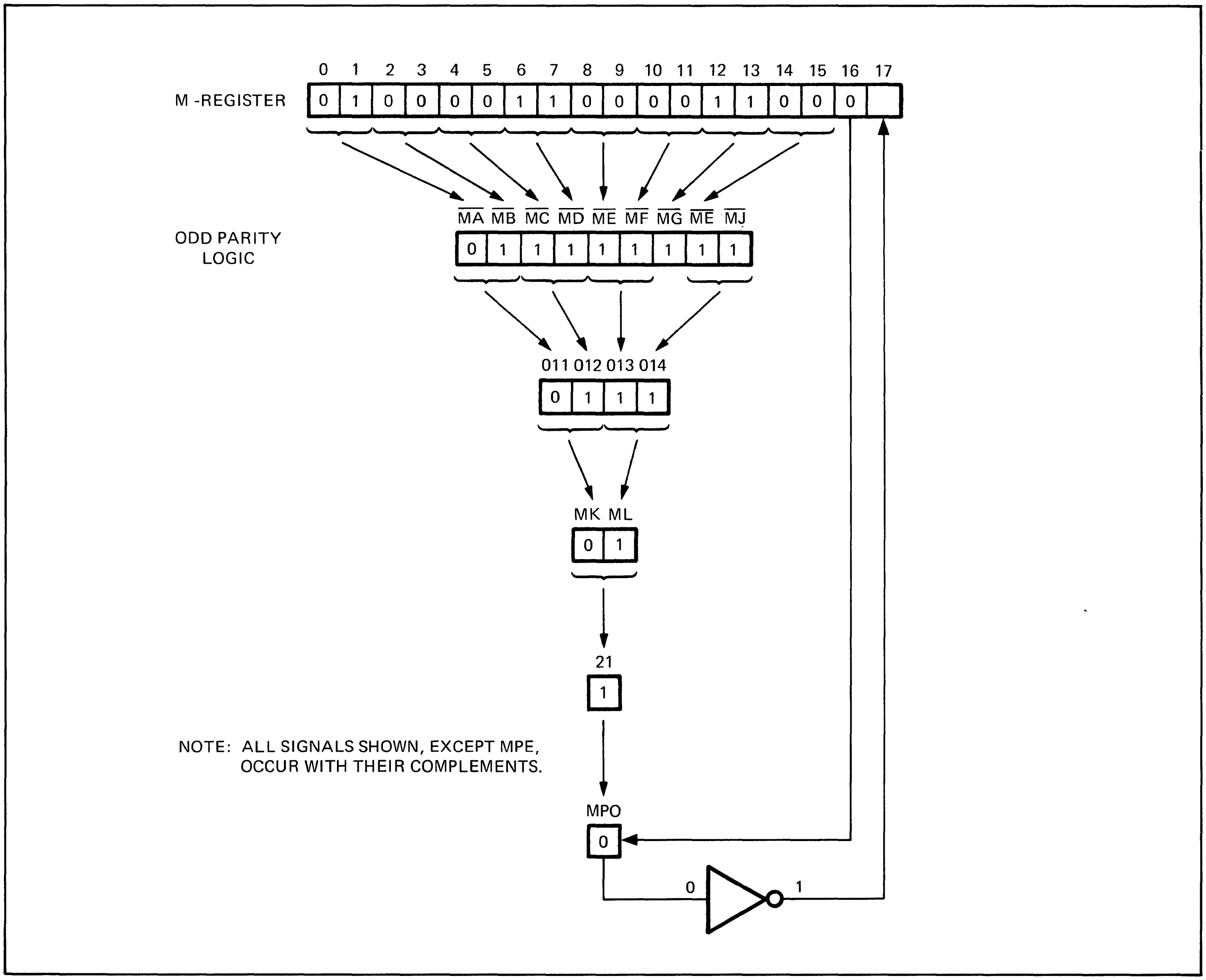
3.57 Memory Select Logic

Each core storage card assembly has a memory selection logic circuit. The purpose of this circuit is to produce a signal that selects the circuit card for use.



NOTE: ALL SIGNALS SHOWN, EXCEPT MPE, OCCUR WITH THEIR COMPLEMENTS.

Figure 3-18. Odd Parity Logic Generates a Parity Error



NOTE: ALL SIGNALS SHOWN, EXCEPT MPE, OCCUR WITH THEIR COMPLEMENTS.

Figure 3-19. Generating a One Parity Bit

Memory selection logic is shown in figure 3-21. It consists of a Nand gate and an inverter. Inputs SA and SB of the gate accept the address select bits. If the machine has a 4096-word memory, these inputs are open, and the circuit card assembly is automatically selected. A larger memory requires that the address select bits be supplied to the core storage card assemblies as shown in figure 3-9.

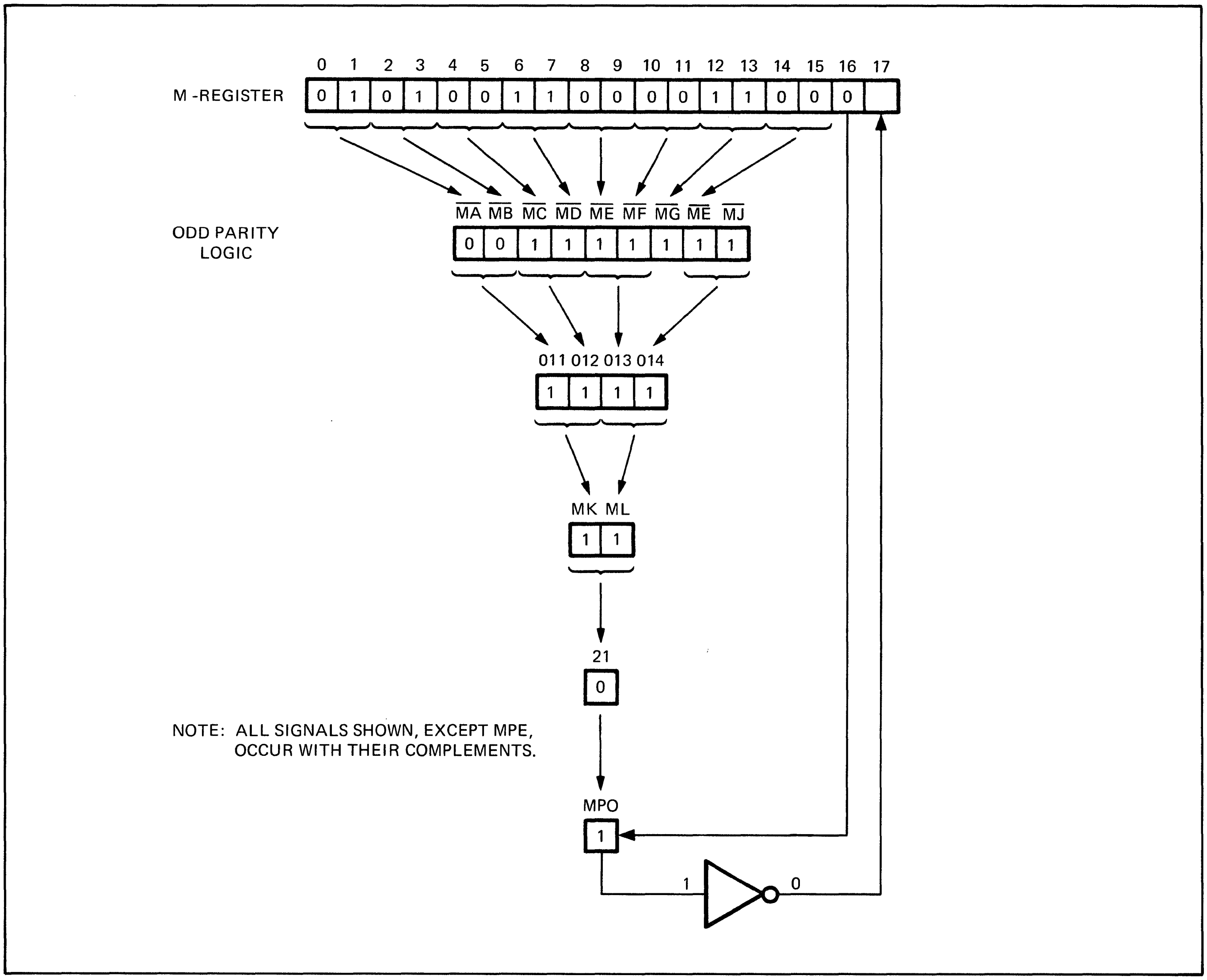
When the computer executes an instruction requesting memory access, bits 1,* 3, and 4 of the instruction word are loaded into timing and control logic flip-flops SDM, ADSL3, and ADSL4. The outputs of these flip-flops are transmitted to each memory selection logic circuit. One of these circuits recognizes the contents of the flip-flops and decodes them, activating the Nand gate. The output of the gate is inverted (US) and is routed to the X- and Y-selection switch drivers.

3.58 Memory Address Register

The memory address (MA) register stores the 12 least significant bits of the instruction being executed by the computer. The contents of the register represent a particular location in core memory.

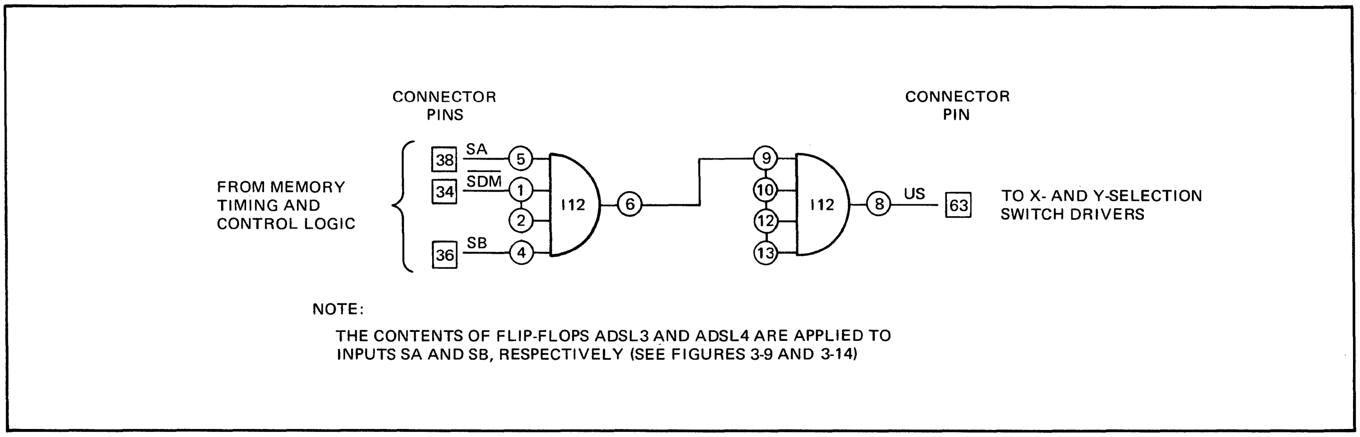
Twelve flip-flops, designated MA1 through MA12, constitute the MA-register (see figure 3-22). The function of each register flip-flop is shown in figure 3-23. The contents of flip-flops MA1 through MA7 control the selection of an X-line. The contents of flip-flops MA8 through MA12 control the selection of a Y-line.

*When bit 1 is false, the core memory is selected. When bit 1 is true, the diode memory is selected.



NOTE: ALL SIGNALS SHOWN, EXCEPT MPE, OCCUR WITH THEIR COMPLEMENTS.

Figure 3-20. Generating a Zero Parity Bit



NOTE:

THE CONTENTS OF FLIP-FLOPS ADSL3 AND ADSL4 ARE APPLIED TO INPUTS SA AND SB, RESPECTIVELY (SEE FIGURES 3-9 AND 3-14)

Figure 3-21. Memory Selection Logic

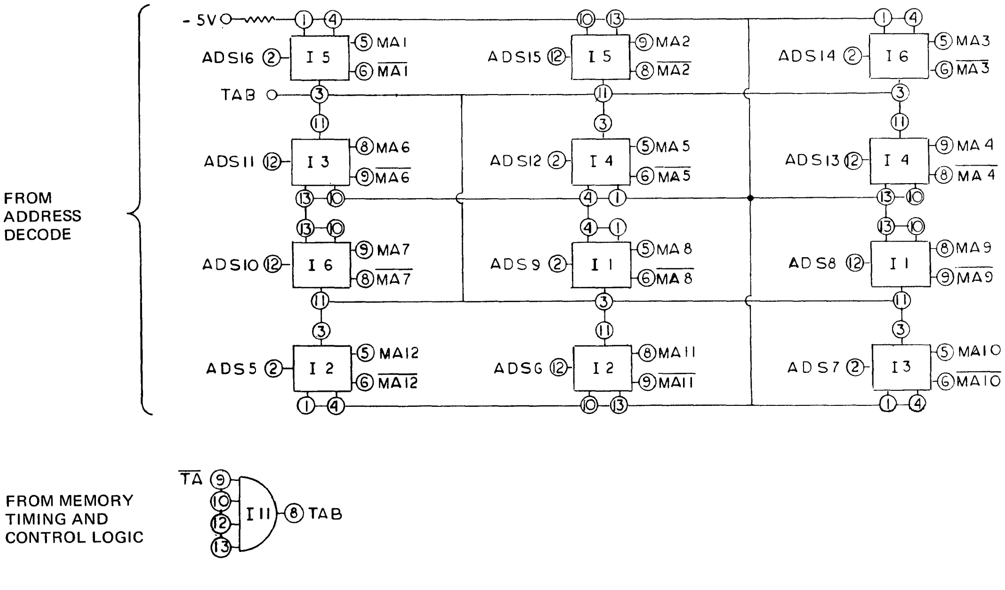
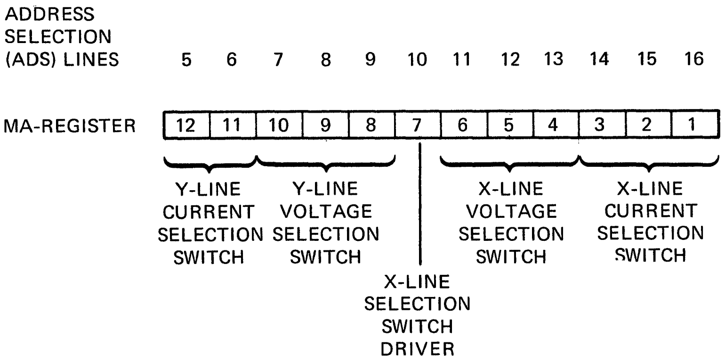


Figure 3-22. Memory Address Register



ADDRESS SELECTION (ADS) LINES	5	6	7	8	9	10	11	12	13	14	15	16
MA-REGISTER	12	11	10	9	8	7	6	5	4	3	2	1

Y-LINE CURRENT SELECTION SWITCH
Y-LINE VOLTAGE SELECTION SWITCH
X-LINE SELECTION SWITCH DRIVER
X-LINE VOLTAGE SELECTION SWITCH
X-LINE CURRENT SELECTION SWITCH

Figure 3-23. MA-Register Output Assignments

The MA-register is loaded at the beginning of the memory cycle. Approximately 220 nanoseconds after MGO occurs, memory timing and control logic generates signal $\overline{TA}$. Signal $\overline{TA}$ is sent to the MA-register and inverted (TAB). On the positive excursion of TAB, bits ADS5 through ADS16 are clocked into the MA-register.

The following equations define the operation of each MA-register flip-flop:

$MA1 = ADS16\ TAB$

$\overline{MA1} = \ldots$

$MA2 = ADS15\ TAB$

$\overline{MA2} = \ldots$

$\vdots \quad \vdots$

$MA12 = ADS5\ TAB$

$\overline{MA12} = \ldots$

3.59 X- and Y-Selection Systems

The X- and Y-selection systems provide currents that select the memory location specified in the reference address field of the instruction being executed by the computer.

A simplified block diagram of the X- and Y-selection systems is shown in figure 3-24. The Y-selection system includes selection switch drivers, current selection switches, voltage selection switches, and transformers. The X-selection system has the same type of circuits, except that it does not use transformers to drive the X-lines. Since the systems are practically identical, only the circuits that make up the X-selection system are described.

During these descriptions, it is assumed that the computer is executing a single-word instruction with an address mode of hexadecimal D and a reference address of hexadecimal 2F. It is also assumed that the machine is performing a read operation.

3.60 Selection Switch Driver

A selection switch driver generates signals that control the operation of an associated selection switch. When the driver is active, its XRD4 and XRT04 outputs are high and low, respectively, causing the selection switch to close. When the driver is disabled, the selection switch is open because XRT04 is high and XRD4 is low.

A combination logic and schematic diagram of a typical X-line selection switch driver is shown in figure 3-25. The driver circuit consists of a Nand gate, an inverter, and a turn-off circuit. The turn-off circuit comprises an input capacitor (C4), resistors, and transistors Q7 and Q8.

Approximately 220 nanoseconds after a read cycle begins, hexadecimal D2F is loaded into the MA-register. Bit 11 of the instruction is false; therefore, $\overline{MA7}$ is high. This output of the MA-register, with the unit select signal (US), is routed to Nand gate $\overline{MA7}$ RT US, enabling the gate. Sixty nanoseconds after the MA-register is loaded, memory timing and control logic generates signal RT, the read time command.

Signal RT is applied to the Nand gate and activates it. The 0V output of the gate is inverted (XRD4) and is transmitted to a voltage selection switch. Capacitor C4 detects the change in the gate's output and discharges through R16, developing a negative voltage across that component.

The emitter of Q7 therefore goes negative and permits the transistor to continue to conduct.* As long as Q7 conducts, the output of Q8 is low. Thus, signal XRT04 is low (+3V). Signal XRT04 is delivered to the same voltage selection switch as XRD4.

During read time, the word in location X'D2F' is read out of memory. At the end of read time, signal RT goes low and disables the selection switch driver. Output XRD4 of the driver goes low. When signal RT goes low, the output of the Nand gate goes high. The positive-going output of the gate is coupled through C4 and cuts off Q7. The base of Q8 therefore goes more positive and causes Q8 to conduct harder. Thus, XRT04 goes high (+6V) for approximately 100 nanoseconds. Signal XRT04 remains high until C4 is discharged by R16. At this time, Q7 again conducts. The change in state of these outputs is sensed by the voltage selection switch. The switch opens and terminates the read selection process.

3.61 Voltage Selection Switch

A voltage selection switch supplies a -27V signal to a current selection switch during the read cycle and a 0V signal during the write cycle.

Refer to figure 3-26. A typical voltage selection switch consists of a Nand gate and two switching circuits. Each switching circuit includes three transistors, three resistors, and a transformer. One circuit generates an output during a read operation, while the other circuit produces an output during a write operation.

Hexadecimal D2F is loaded into the MA-register when memory timing and control logic generates the transfer address (TA) signal. Output $\overline{MA7}$ of the MA-register and the unit select signal enable one of the X-line selection switch drivers. The power-on signal ($\overline{PRST}$),

*Transistor Q7 is normally conducting.

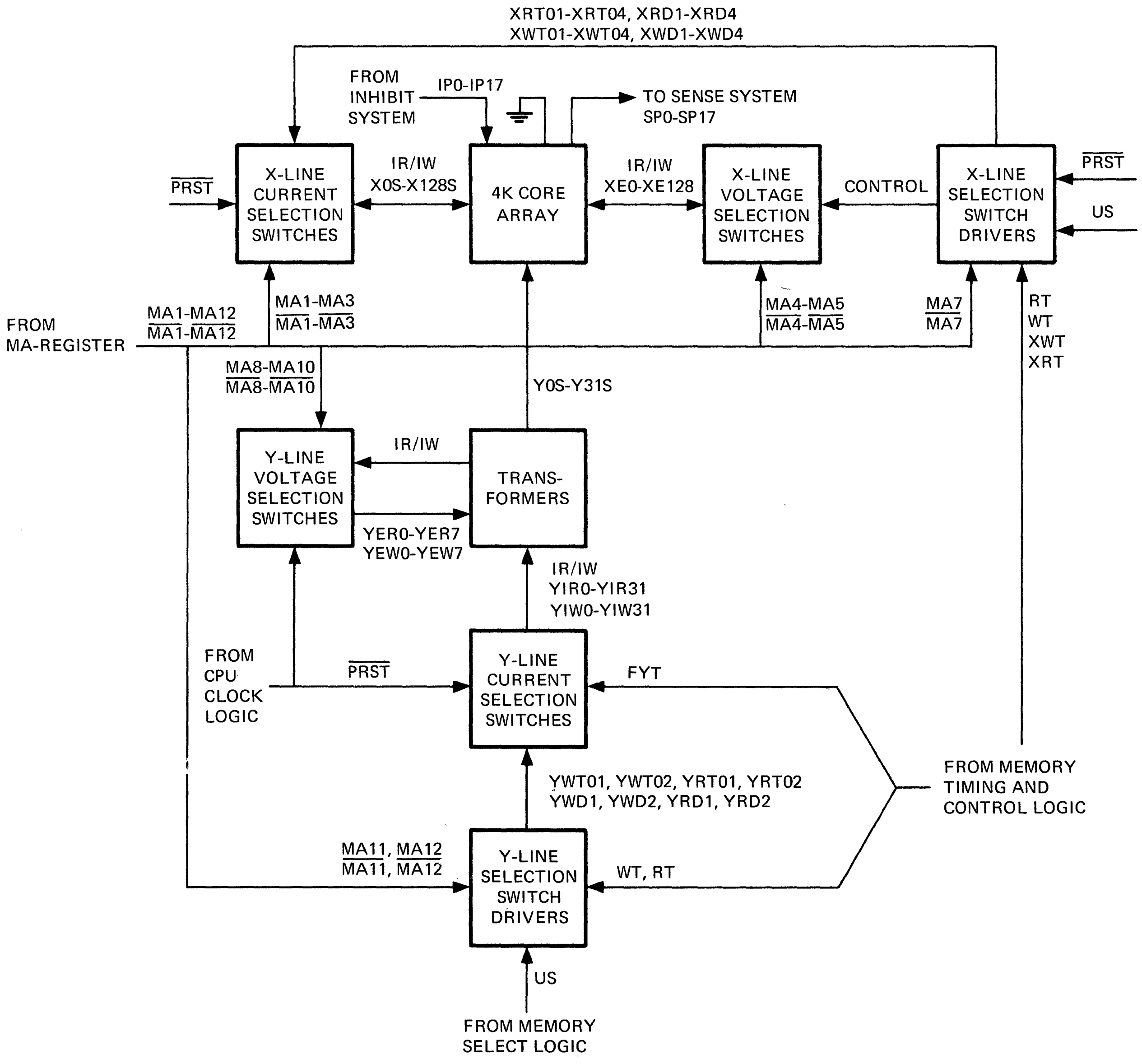

XRT01-XRT04, XRD1-XRD4
XWT01-XWT04, XWD1-XWD4
FROM INHIBIT SYSTEM
IP0-IP17
TO SENSE SYSTEM SP0-SP17
$\overline{PRST}$
X-LINE CURRENT SELECTION SWITCHES
IR/IW X0S-X128S
4K CORE ARRAY
IR/IW XE0-XE128
X-LINE VOLTAGE SELECTION SWITCHES
CONTROL
X-LINE SELECTION SWITCH DRIVERS
$\overline{PRST}$
US
FROM MA-REGISTER
MA1-MA12 $\overline{MA1}$-$\overline{MA12}$
MA1-MA3 $\overline{MA1}$-$\overline{MA3}$
MA4-MA5 $\overline{MA4}$-$\overline{MA5}$
MA7 $\overline{MA7}$
RT WT XWT XRT
MA8-MA10 $\overline{MA8}$-$\overline{MA10}$
Y0S-Y31S
Y-LINE VOLTAGE SELECTION SWITCHES
IR/IW
TRANS-FORMERS
YER0-YER7 YEW0-YEW7
IR/IW YIR0-YIR31 YIW0-YIW31
FROM CPU CLOCK LOGIC
$\overline{PRST}$
Y-LINE CURRENT SELECTION SWITCHES
FYT
FROM MEMORY TIMING AND CONTROL LOGIC
YWT01, YWT02, YRT01, YRT02
YWD1, YWD2, YRD1, YRD2
MA11, MA12 $\overline{MA11}$, $\overline{MA12}$
Y-LINE SELECTION SWITCH DRIVERS
WT, RT
US
FROM MEMORY SELECT LOGIC

Figure 3-24. X- and Y-Selection Systems, Simplified Block Diagram

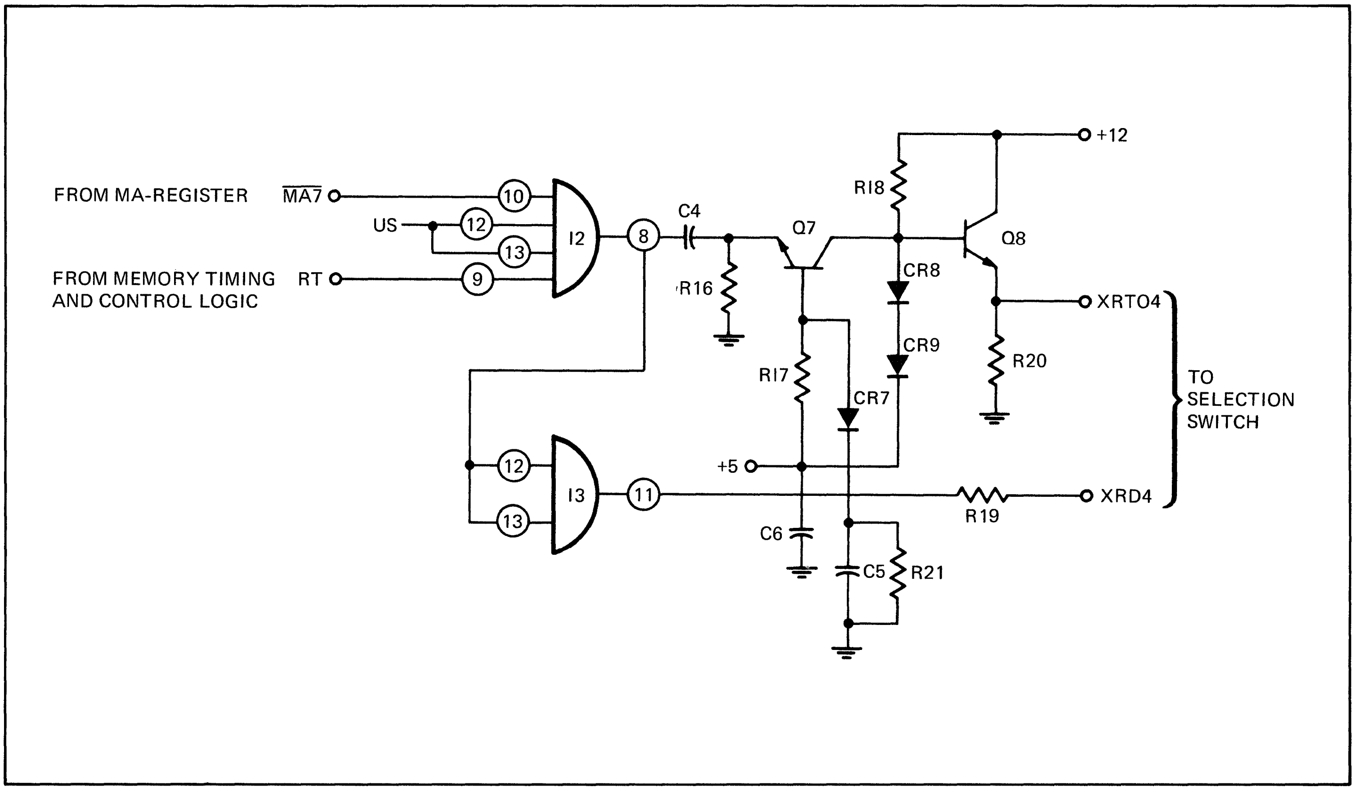


Figure 3-25. Typical Selection Switch Driver

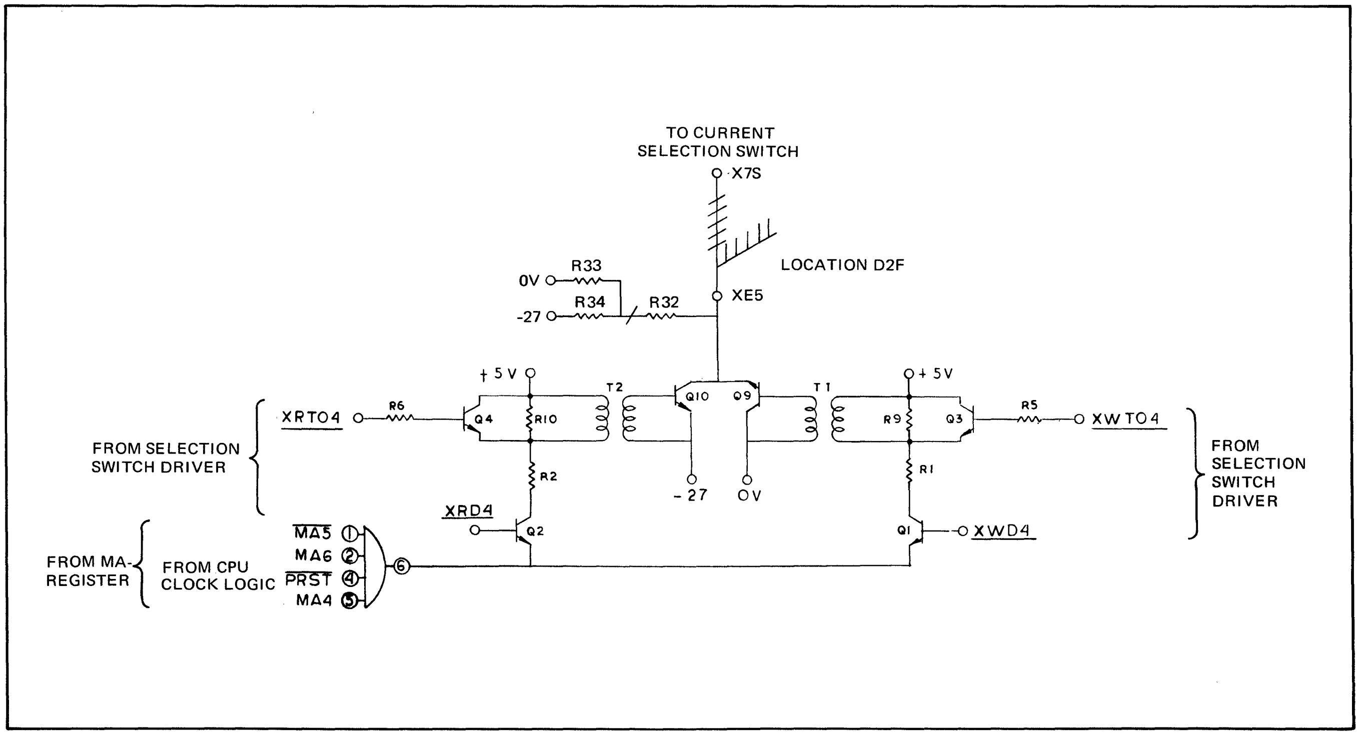


Figure 3-26. Typical Voltage Selection Switch

generated by computer clock logic, and MA-register outputs MA4, $\overline{\text{MA5}}$, and MA6 activate Nand gate $\overline{\text{PRST}}$ MA4 $\overline{\text{MA5}}$ MA6. The 0V output of the gate is applied to the emitters of Q1 and Q2.

Sixty nanoseconds after the MA-register is loaded, memory timing and control logic generates signal RT. This signal is sent to the enabled selection switch driver and activates it, producing a 0V level for XRT04 and a +5V level for XRD4. Signals XRT04 and XRD4 are applied to the bases of transistors Q4 and Q2, respectively. Transistor Q2 turns on; however, transistor Q4 remains off. The current produced by Q2 flows through R2 and T2. (A very small amount of Q2's current flows through R10.)

The expanding flux field of T2 induces a voltage into the secondary of T2. This voltage forward biases Q10, permitting it to conduct. The -27V applied to the emitter of Q10 is coupled through that transistor and is transmitted to a current selection switch.

During read time, the current selection switch closes, a Y-line is selected, and the contents of memory location X'D2F' are read. When signal RT goes low, XRT04 goes high and XRD4 goes low. Thus, Q2 cuts off. The flux field produced by T2 collapses, causing a heavy current to flow momentarily through Q4. The base of Q10 also shorts to the emitter of that transistor. The voltage switch opens, and current no longer flows in the selected X-line.

3.62 Current Selection Switch

Under program control, a current selection switch produces either a read or write current, depending upon which part of the memory cycle is being performed.*

A typical current selection switch is shown in figure 3-27. It consists of a Nand gate and two identical elements. One element is used during a read operation, while the other element is used during a write operation. An element consists of three transistors, five resistors, a transformer, and a diode.

Approximately 200 nanoseconds after the read cycle begins, hexadecimal D2F is loaded into the MA-register. Outputs MA1 through MA3 of the register are gated with signal $\overline{\text{PRST}}$, activating Nand gate $\overline{\text{PRST}}$ MA1 MA2 MA3. The 0V output of the gate is applied to the base of Q1' and to the base of Q2'.

When memory timing and control logic generates signal RT, the X-line voltage switch shown in figure 3-26 closes and supplies -27V to the cathode of CR2. One hundred and forty nanoseconds later, memory and control logic produces signal XRT. Signal XRT, with US, activates a selection switch driver, generating 0V and +5V for XRT01 and XRD1, respectively. Transistor Q2' is switched on by XRD1, and transistor Q4' is cut off by

*During every memory cycle, both read and write currents are produced (see paragraphs 3.45 through 3.47).

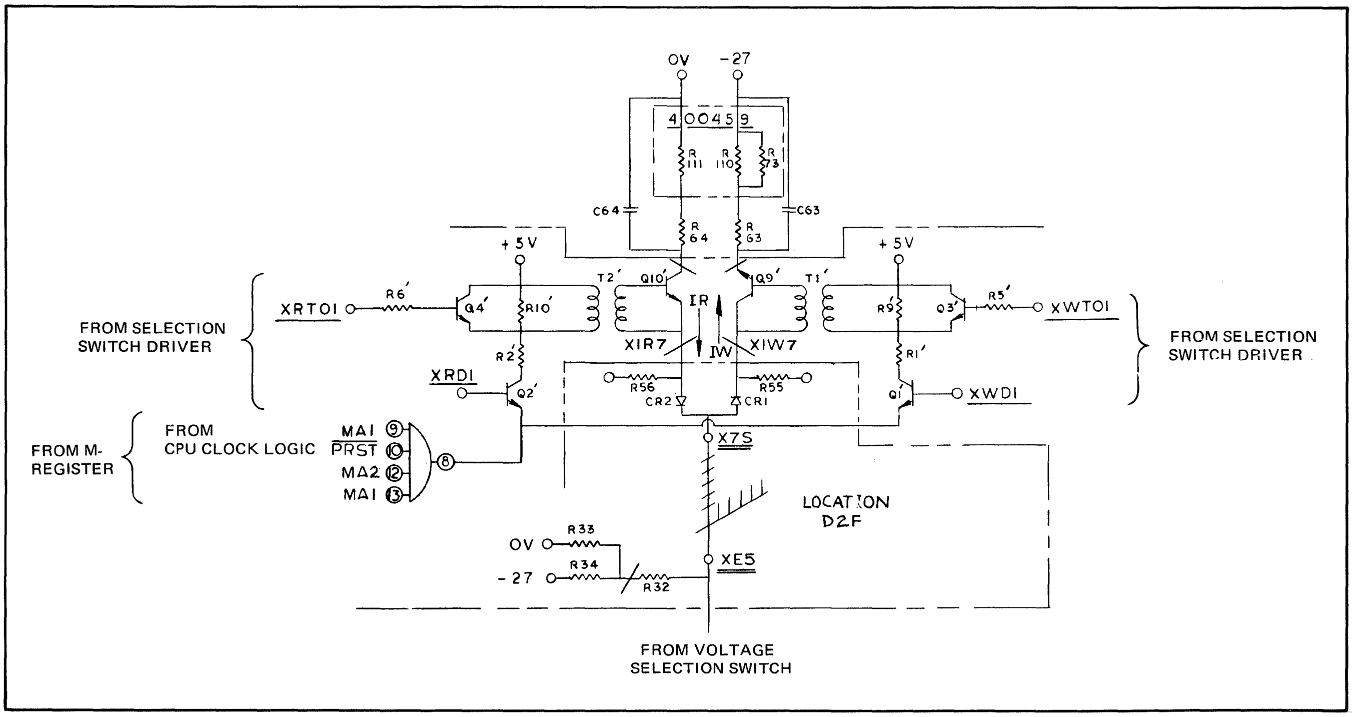


Figure 3-27. Typical Current Selection Switch

XRT01. Thus, a heavy current flows through Q2', R2, and the primary of T2'.

The expanding flux field produced by the primary of T2' induces a voltage into the secondary of T2'. The voltage is applied to the base of Q10', switching on the transistor. A current, designated IR, flows through the X-current selection switch, memory location X'D2F', and the X-voltage selection switch. Thus, one of the 128 X-selection lines is activated. At the same time, one of the 32 Y-lines is activated. The read current produced by the selected Y-line also flows through location X'D2F'. With both currents flowing through location X'D2F', the word stored in that location is read out of memory.

At the end of read time, signal XRT goes low and disables the selection switch driver. Therefore, XRT01 goes high, XRD1 goes low, and Q2' cuts off. The flux field produced by T2' collapses and causes a heavy current to flow momentarily through Q4'. The collapsing flux field of T2' also cuts off Q10'. Thus, the current switch opens, and current no longer flows in the selected X-line.

3.63 Sense System

The sense system supplies an 18-bit word to the M-register during the read cycle.

The sense system consists of 18 identical elements. Each element includes a Nand gate and a sense amplifier (see figure 3-28). The output of an element is applied to inhibit gating and to an M-register latch circuit.

A sense amplifier conducts when its input detects a 17-mV signal. A threshold voltage (VT) is established at the input of a sense amplifier by a voltage divider network. This voltage divider network consists of three resistors and the associated sense winding (see drawing 400838B in volume II of this manual). The threshold voltage determines the point at which the amplifier conducts.

When a bit is read out of memory, it is applied to the input of a sense amplifier via a sense line. Consider that a one bit is read from memory. The signal received from the selected core is large and a pulse, representing

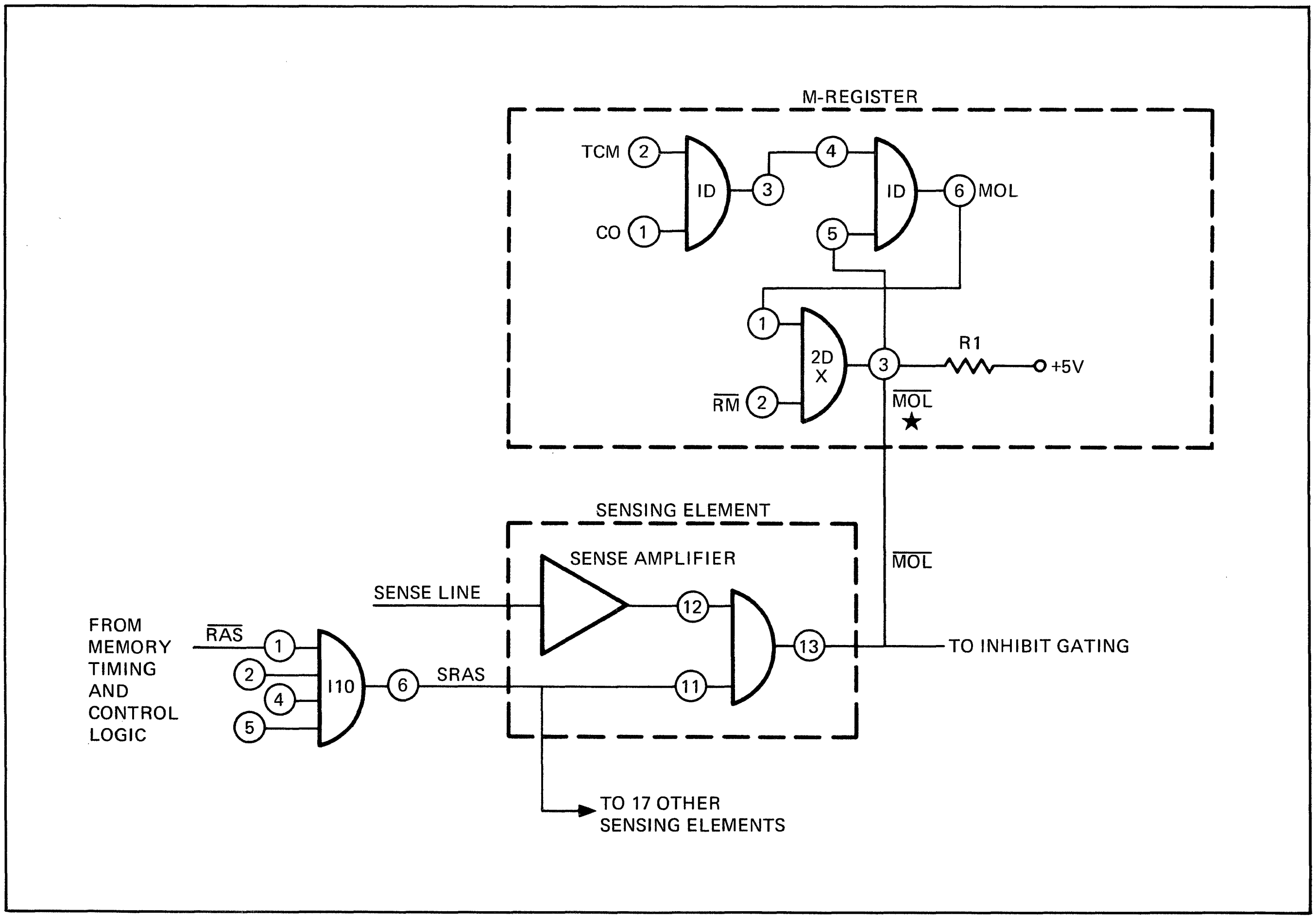


Figure 3-28. Sense System — Typical Sensing Element

a binary one, appears at the output of the sense amplifier. This output is sent to one of the inputs of the Nand gate.

Approximately 100 nanoseconds after a location has been selected in memory, memory timing and control logic generates signal $\overline{\text{RAS}}$. Signal $\overline{\text{RAS}}$, which occurs for 60 nanoseconds, is inverted (SRAS) and is routed to the other input of the Nand gate, activating the gate. The 0V output of the gate is routed to the false output of latch circuit M0L. The latch circuit is forced to set because TCM is low and $\overline{\text{RM}}$ is high (figure 3-13). Thus, the one-bit read out of the selected core is stored in latch circuit M0L of the M-register.

Now consider that a zero-bit is read from core memory. The signal received from the selected core is small and no pulse, representing a binary zero, appears at the output of the sense amplifier. This output is applied to the Nand gate and disables the gate, producing a +5V output signal. The gate remains disabled when memory timing and control logic releases signal $\overline{\text{RAS}}$. The output of the gate is routed to the latch circuit, but has no effect on the circuit because it is already reset. (The M-register is cleared to zero 20 nanoseconds after the read cycle begins.) Thus, the zero-bit read out of the selected core is stored in latch circuit M0L of the M-register.

3.64 Inhibit System

The inhibit system writes the contents of the M-register into a selected memory location during both the read and write cycles.*

Refer to figure 3-29. The inhibit system consists of an inhibit control circuit and 18 identical elements. The

*When a word is read out of memory, the readout of the location is destructive. Therefore, the word stored in the M-register is immediately written back into the selected location.

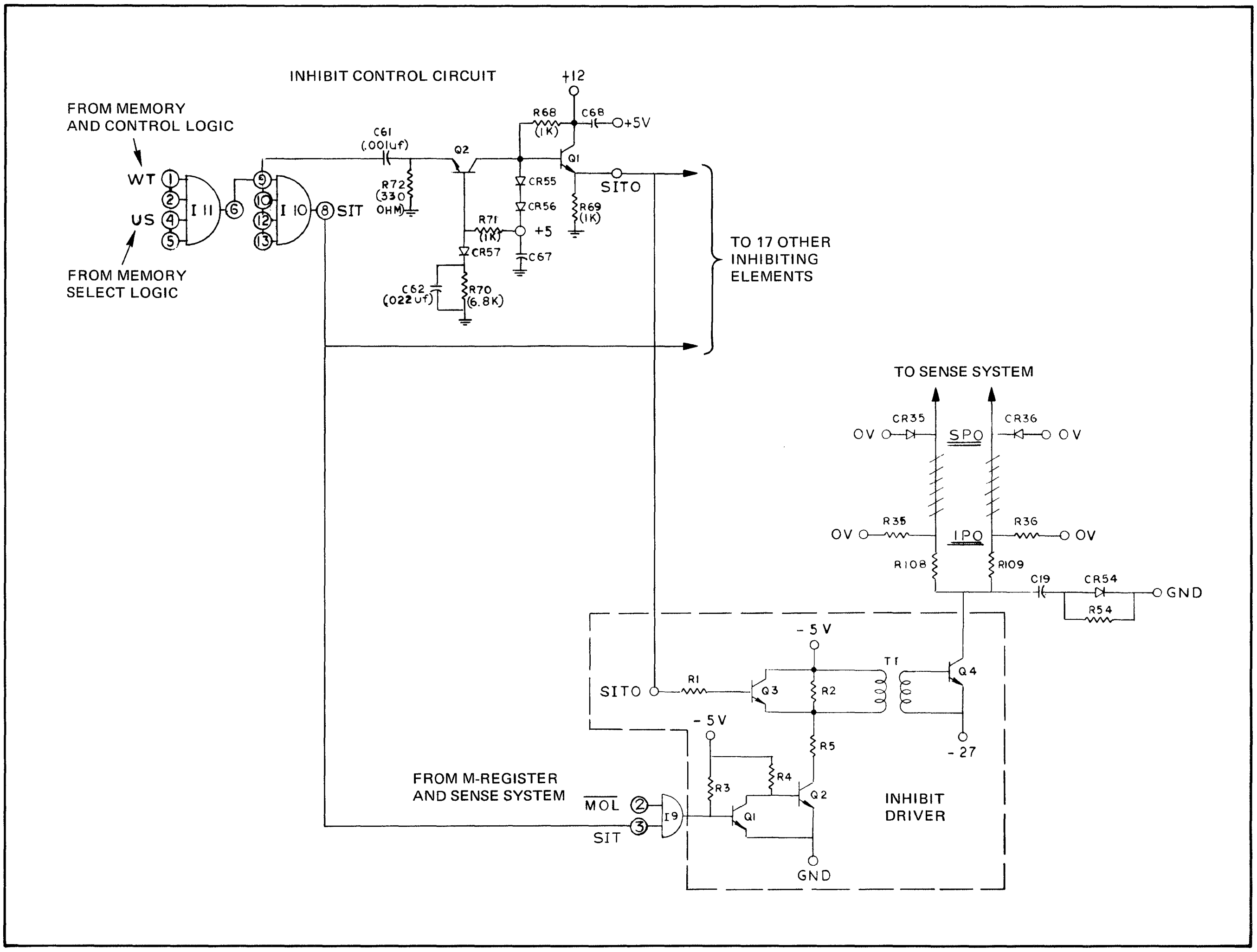


Figure 3-29. Inhibit System — Control Circuit and Typical Inhibiting Element

control circuit comprises a Nand gate, an inverter, and transistors Q1 and Q2. An inhibiting element includes a Nand gate and an inhibit driver. The inhibit driver has transistors Q1 through Q4, resistors, and a transformer.

Since the inhibit system functions in the same manner during both the read and write cycles, only the system's operation during a read cycle is discussed using the circuit shown in figure 3-29.

When a word is read out of memory, it is stored in the M-register and transmitted to the inhibit system. Consider that a zero-bit is loaded into latch circuit M0L of the M-register. Therefore, $\overline{\text{M0L}}$ is high and Nand gate $\overline{\overline{\text{M0L}}\ \text{SIT}}$ is enabled. Approximately 640 nanoseconds after the read cycle starts, memory timing and control logic generates WT, the write time command.

Signal WT is routed to Nand gate $\overline{\text{WT US}}$ and activates the gate. The 0V output of the gate is inverted (SIT) and is applied to Nand gate $\overline{\overline{\text{M0L}}\ \text{SIT}}$. The gate activates and delivers a 0V signal to Q1 of the inhibit driver. Transistor Q1 cuts off. The collector voltage of Q1 increases. This voltage is sensed at the base of Q2, switching on Q2.

At the same time, the 0V output of Nand gate $\overline{\text{WT US}}$ causes C61 to discharge through R72. A negative voltage is developed across R72, which permits Q2 of the inhibit control circuit to continue to conduct.* With Q2 conducting, the base of Q1 goes negative. Thus, the output of Q1 (SITO) remains low (3V). This voltage is sent to the base of Q3 of the inhibit driver. Transistor Q3 therefore remains cut off.

The current produced by Q2 of the inhibit driver flows through R5 and the primary of T1. The expanding flux field of T1's primary induces a voltage into the secondary of T1. This voltage forward biases Q4, producing an inhibiting current. The current flows through a selected core and cancels one half of the coincidence currents generated by the X- and Y-selection lines. Thus, a zero is written into the selected core.

At the end of write time, SITO goes high and SIT goes low. Signal SIT disables Nand gate $\overline{\overline{\text{M0L}}\ \text{SIT}}$, cutting off Q2 of the inhibit driver. The flux field produced by T1 collapses and causes a heavy current to flow momentarily through Q3. The collapsing flux field of T1 also cuts off Q4. Thus, the inhibit driver is disabled and current no longer flows in the inhibit line.

Now consider that a one-bit is loaded into latch circuit M0L of the M-register. Therefore, $\overline{\text{M0L}}$ is low and Nand gate $\overline{\overline{\text{M0L}}\ \text{SIT}}$ is disabled. The +5V output of the gate forward biases Q1 of the inhibit driver. The collector of Q1 and the base of Q2 drop to approximately 0V, cutting off Q2.

When memory timing and control logic generates WT, the inhibit circuit again generates SIT and SITO. Signal SIT does not affect the operation of Nand gate $\overline{\overline{\text{M0L}}\ \text{SIT}}$ because $\overline{\text{M0L}}$ is low; SITO back biases Q3 of the inhibit driver. A current, therefore, does not flow through T1, and the inhibit driver does not produce an inhibiting current. Thus, the X- and Y-selection currents write a one into the selected core.

3.65 CLOCK GENERATION AND TIMING

3.66 Basic Computer Clock

The basic computer clock (CLK) is a 430-nanosecond square wave that is used to clock most of the flip-flops in the computer. The clock is also applied to gating circuits to form an even and an odd clock designated as $\overline{\text{ODD}}$ and ODD. Figure 3-30 illustrates the timing relationships between CLK, $\overline{\text{ODD}}$ and ODD.

When used as a clock for the registers, basic clock signal CLK is usually combined with the register transfer term to allow the register flip-flops to change state only when the applicable register transfer term is true. Table 3-1 lists the clocks used throughout the computer, based on clock CLK.

*Transistor Q2 is normally conducting.

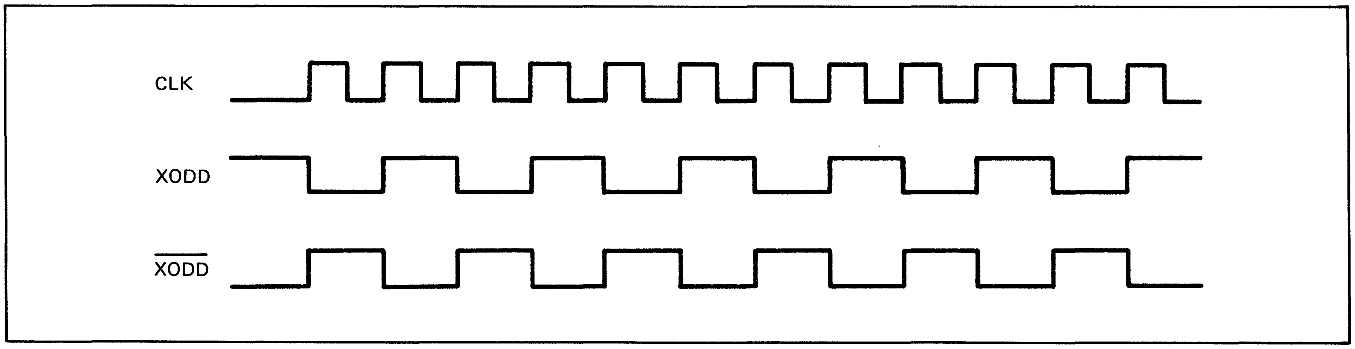


Figure 3-30. Basic Computer Clock

Table 3-1. Clock Signals

Clock Signal	Use	Logic
CKB	Clock for B-register	CKB = STB CLK
CKD	Clock for D-register	CKD = CLK (STD + MTD)
CKI	Clock for I-register flip-flops I6-I8	CKI = (STI + MTI)
CKIB	Clock for I-register flip-flops I15-I16	CKIB = CLK (CTIB + STIL + MTI)
CKID	Clock for I-register flip-flops I13-I14	CKID = CLK (CTID + STIL + MTI)
CKIH	Clock for I-register flip-flops I0-I5	CKIH = CLK (ILTIH + STI + MTI)
CKIW	Clock for I-register flip-flops I9-I12	CKIW = CLK (CTIW + STIL + MTI)
CKKA	Clock for KA-KC, KO, KS, KU, KK, Z0-Z2 flip-flops	CKKA = CLK
CKL	Clock for L-register	CKL = CLK (STL + NTL)
CKN	Clock for N-register	CKN = CLK STN
CKP	Clock for P-register	CKP = CLK (STPKW + ITP + RSTPFF)
CKSW	Clock for switch flip-flops	CKSW = CLK XODD
CKT	Clock for T1-T4 flip-flops	CKT = CLK
CKV	Clock for V1-V3 flip-flops	CKV = CLK (SIXV + STXYU + DECV)
CKX	Clock for X1-X7 flip-flops	CKX = CLK XODD Z1 Z2
CKXOD	Clock for ODD flip-flop	CKXOD = CLK
CKY	Clock for YA-YC flip-flops	CKY = CLK XODD
CLKB3	Clock for MPTY and MGODLY	CLKB3 = CLK
CLKG	Clock for G-register	CLKG = CLK (STG + RSG2 + RSG1 + LSG1)
CLKJ	Clock for J-register	CLKJ = CLK (SSPM + SPMS)
CLKM	Clock for flip-flop MGOA	CLKM = CLK
CLKX	Clock for index register	CLKX = CLK MGOA

During specific times a delay clock signal (DLYCLK) is generated. This signal delays generation of the clock for approximately 200 additional nanoseconds to allow time for the instruction or operand to be fetched and available to clock into the I- or D-register.

Basic clock CLK is developed from three single-shot circuits. Figure 3-31 is a block diagram of the clock generation circuits. The single-shots are indicated by CKT 100, 200, and 300 on this block diagram. Figure 3-32 is the schematic of the clock generation circuits.

When power is applied to the timing circuits, signal PSUF goes high enabling the $\overline{\text{CLK}}$ power gate. The enabling signal allows the $\overline{\text{CLK}}$ signal to be distributed to the computer circuits where it is used. Before passing through the power gate, the clock signal is identified

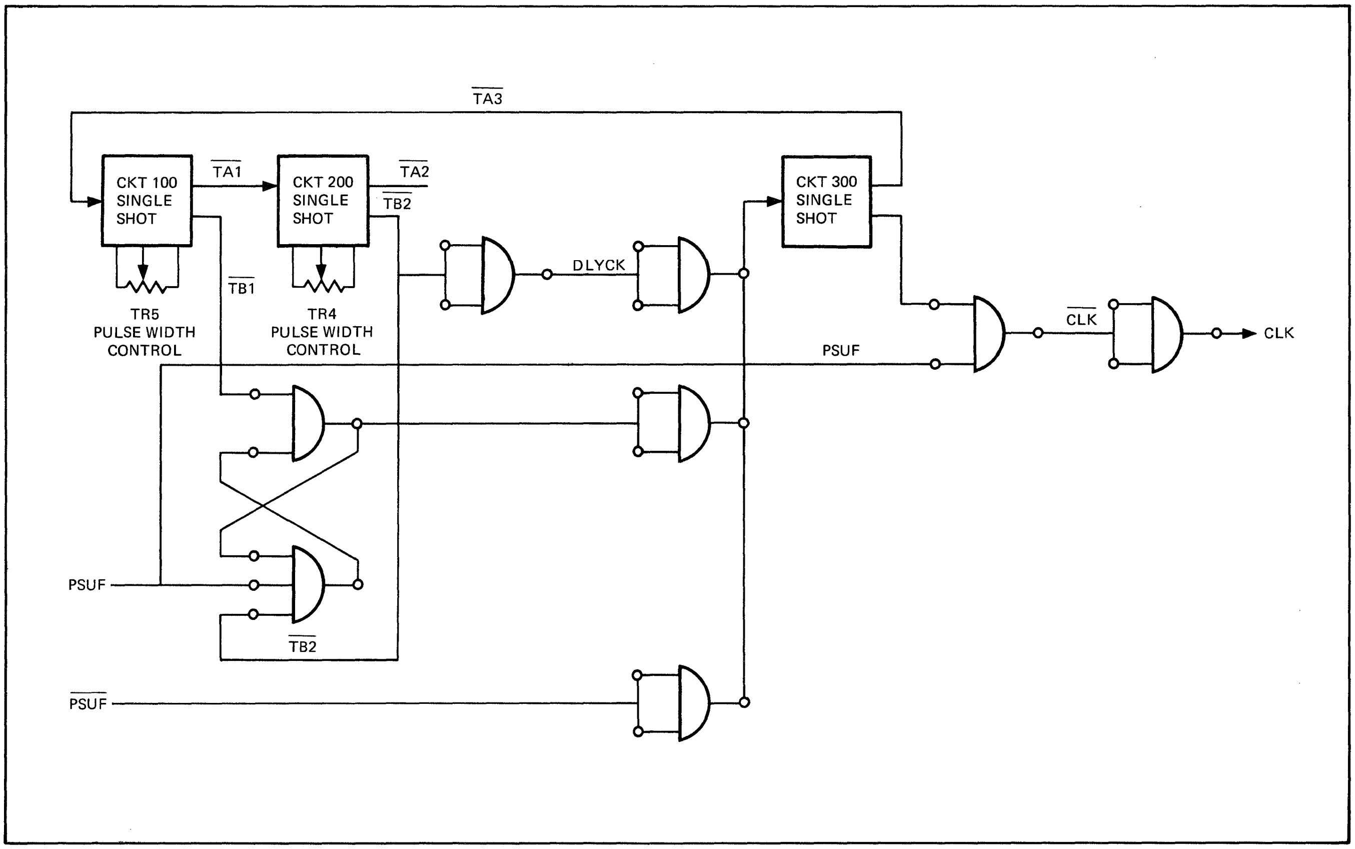


Figure 3-31. Basic Clock Generation, Block Diagram

as $\overline{TB3}$ and is generated by circuit 300. Signal $\overline{TA3}$ is another output from circuit 300. It is in phase with signal $\overline{TB3}$ but lacks the driving power of $\overline{TB3}$. Signal $\overline{TA3}$ is used as a feedback loop for the clock and triggers circuit 100. Circuit 300 is initially triggered by signal $\overline{PSUF}$ going low when power is applied. Circuit 100 triggers circuit 200 with signal $\overline{TA1}$. Circuit 200 is then used to trigger circuit 300 by applying signal $\overline{TB2}$ through the gating circuit that initially triggered circuit 300. Delay clock DLYCK is also inserted in this point to delay triggering circuit 300 when the clock must be delayed for memory accesses. Circuits 100 and 200 contain potentiometers for adjusting the pulse width of the clock. Figure 3-33 illustrates the timing for generation of the basic computer clock.

3.67 One- and Two-Millisecond Clocks

Both 1- and 2-millisecond clocks are generated for use in particular areas of the computer and input/output buffer. Figure 3-34 is the schematic diagram of the 1- and 2-millisecond clock circuits.

The circuits are equivalent to two series single-shot circuits with each single-shot triggering the other. The circuit is enabled when initial power is applied to the computer and signal PSUF goes high. The output at CK1M is a positive-going 1.5-microsecond pulse occurring approximately every millisecond. The output at CK2M is a square wave with a cycle time of 2 milliseconds. As the CK1M and CK2M clocks are not synchronized with the computer clock, the signals are not used for general computer clocking. Clock CK1M is used strictly for clocking the power startup and shutdown circuits. Clock CK2M is applied to a flip-flop to be synchronized with the computer clock and is then referred to as CK2MS. This signal is used on power startup flip-flop Z0, the interrupt logic, and the input/output buffer. Clock CK2MS is also applied to another flip-flop where it is delayed one clock and is referred to as CK2ML. This clock is used in the logic for go idle (GIDL) when used with the control panel CONT (continuous) switch.

3.68 Z-Timing

Three flip-flops (Z0 through Z2) are used for coarse timing during power startup, power shutdown and for inhibiting the Y-states during input/output operations when it is necessary to steal the next available memory cycle.

Flip-flop Z0 sets when power is applied to the computer (signal POK goes true), the 2-millisecond clock CK2MS goes true, and an even clock occurs ($\overline{XODD}$ true). It

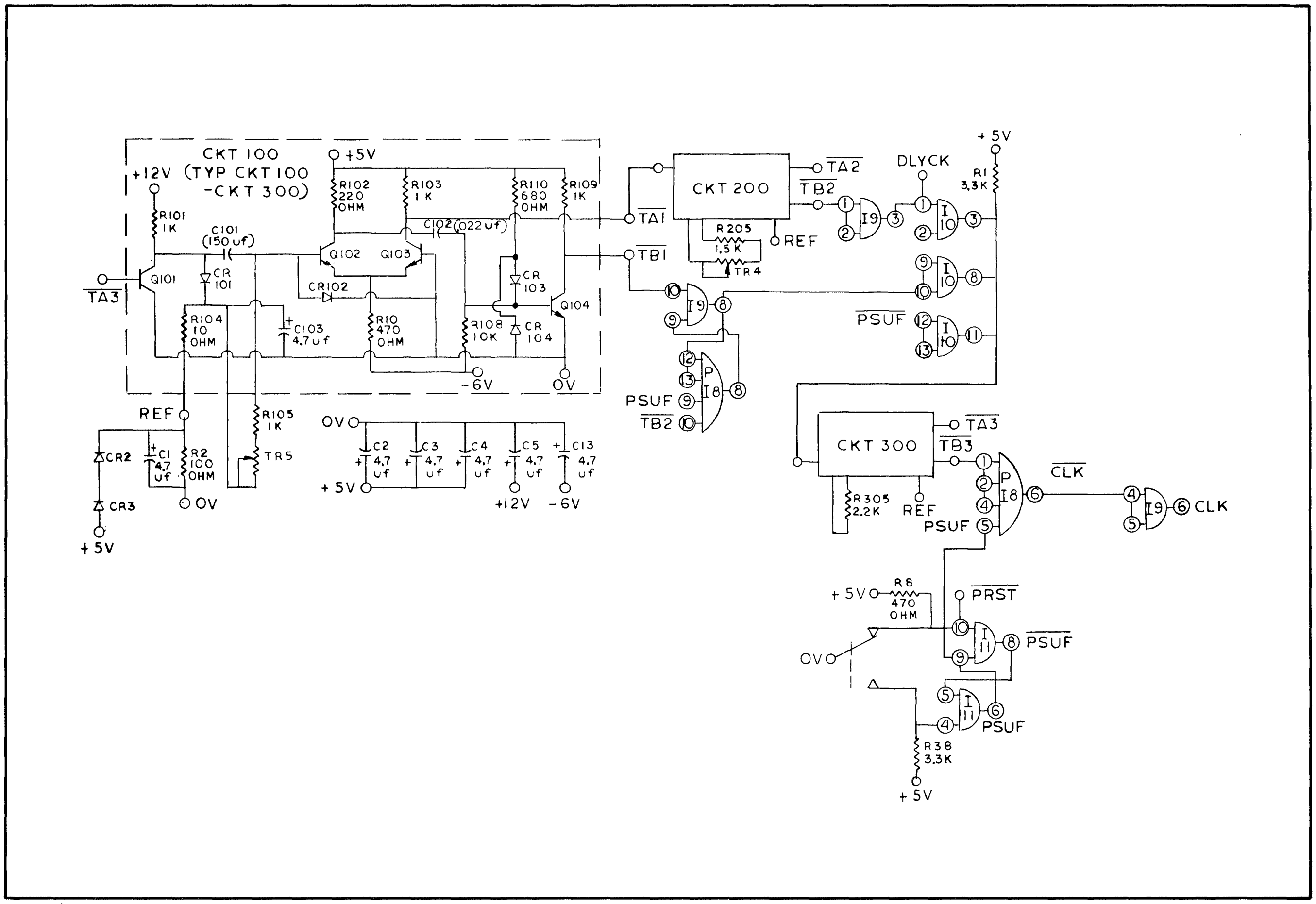


Figure 3-32. Basic Clock Generation, Schematic Diagram

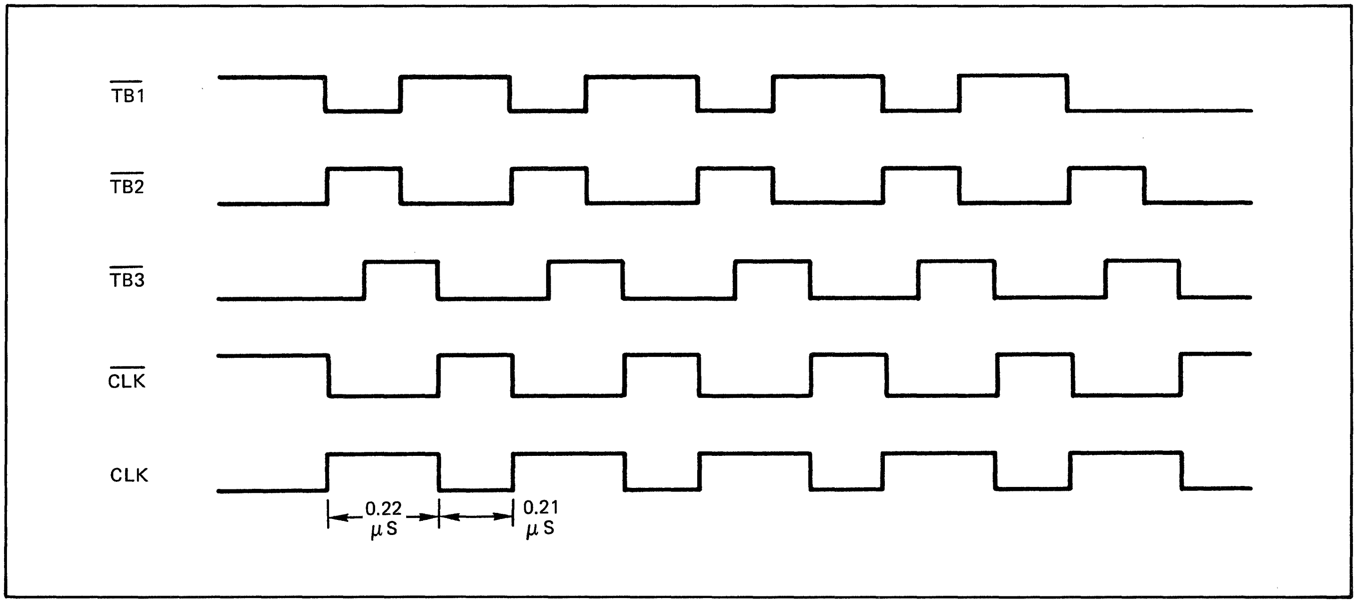


Figure 3-33. Basic Computer Clock Generation, Timing Diagram

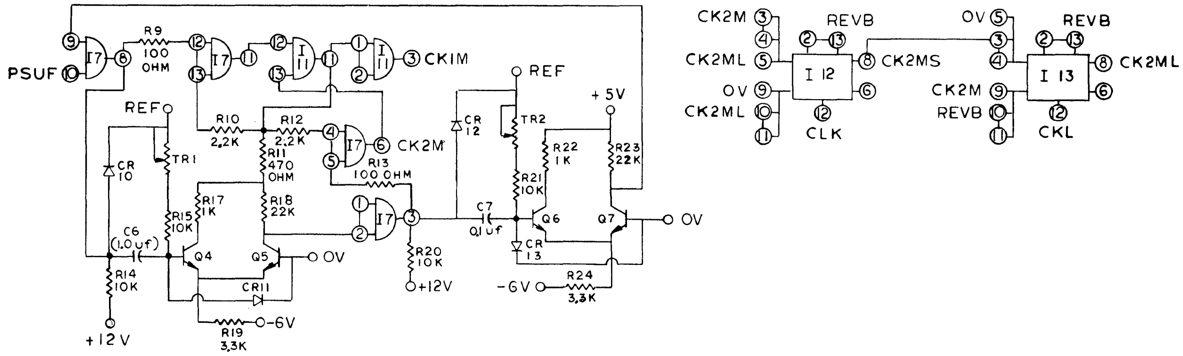


Figure 3-34. One- and Two-Millisecond Clocks, Schematic Diagram

then resets on the next computer clock. It also sets when power is removed ($\overline{POK}$ true), flip-flop Z1 is true indicating power has been on, and an even clock occurs. Flip-flop Z0 then resets at the next computer clock.

$$\begin{aligned} Z0 &= CK2MS\ POK\ \overline{Z1}\ \overline{XODD} \\ &\quad + \overline{POK}\ Z1\ \overline{IOIO}\ \overline{XODD} \\ \overline{Z0} &= \ldots \\ CZ0 &= CKKA \\ CKKA &= CLK \end{aligned}$$

Flip-flop Z1 is the power-on flip-flop. It sets one computer clock after flip-flop Z0 sets. Flip-flop Z1 then latches itself and stays set until flip-flop Z0 sets again, indicating power has been removed or has failed. Flip-flop Z1 then resets.

$$\begin{aligned} Z1 &= Z0\ \overline{Z1} + \overline{Z0}\ Z1 \\ \overline{Z1} &= \ldots \\ E/Z1 &= \overline{PRST} \\ CZ1 &= CKKA \\ CKKA &= CLK \end{aligned}$$

Flip-flop Z2 is the power shutdown flip-flop. When flip-flop Z0 goes true while flip-flop Z1 is true, a power failure or shutdown is indicated. Flip-flop Z2 sets on the next clock. Signal Z2 going true sets flip-flop YB and resets YA and YC to generate signal YEND. Signal YEND forces the power shutdown memory address onto the address lines to begin storing the volatile register contents. Flip-flop Z2 also sets if a cycle steal signal (CYSTL) is received from the input/output buffer indicating that the DMA channel desires the next available memory cycle. Signal Z2 inhibits the Y-states and allows the DMA channel to steal the next memory cycle.

$$\begin{aligned} Z2 &= Z0\ Z1 + Z2\ \overline{XODD} \\ &\quad + CYSTL\ XODD\ Z1\ \overline{YBGN}\ \overline{IOIO} \\ \overline{Z2} &= \ldots \\ CZ2 &= CKKA \\ CKKA &= CLK \end{aligned}$$

Figure 3-35 illustrates the timing of the Z-flip-flops.

3.69 Y-Timing States

Three flip-flops (YA, YB, and YC) are used for finer timing than the Z-states and indicate particular states of the computer. The states indicated by the Y-flip-flops are:

Power Startup	$YBGN = \overline{Z2}\ \overline{YC}\ YB\ YA$
Computer Idle	$YIDL = \overline{Z2}\ \overline{YC}\ \overline{YB}\ \overline{YA}$
Operand Address Select	$YOAS = \overline{Z2}\ \overline{YC}\ \overline{YB}\ YA$
Instruction Execution	$NY4 = \overline{Z2}\ YC\ \overline{YB}\ \overline{YA}$
Power Shutdown	$YEND = Z2\ \overline{YC}\ YB\ \overline{YA}$

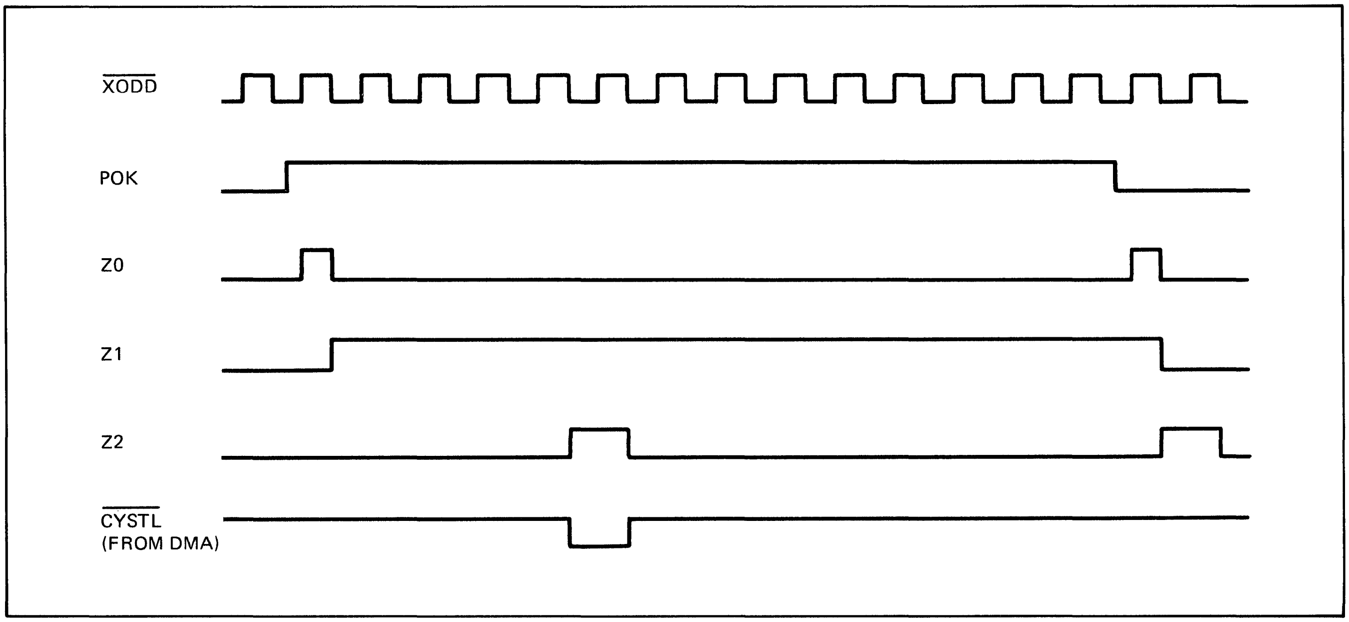


Figure 3-35. Z-Flip-Flops, Timing Diagram

The invalid Y-states are decoded to generate signals MY5, DY6, and MDY7. These signals indicate states not used at the present time.

$$\mathrm{YA} = (\mathrm{LIDL} + \mathrm{ENDI}\ \overline{\mathrm{GIDL}} + \mathrm{Z0}\ \overline{\mathrm{Z1}} + \mathrm{STY}\ \mathrm{S10})\ \overline{\mathrm{GIDL}}$$

$$\overline{\mathrm{YA}} = \mathrm{RSTY}$$

$$\mathrm{CYA} = \mathrm{CKY}$$

$$\mathrm{YB} = (\mathrm{Z2}\ \overline{\mathrm{Z1}} + \mathrm{Z0}\ \overline{\mathrm{Z1}} + \mathrm{STY}\ \mathrm{S9})\ \overline{\mathrm{GIDL}}$$

$$\overline{\mathrm{YB}} = \mathrm{RSTY}$$

$$\mathrm{CYB} = \mathrm{CKY}$$

$$\mathrm{YC} = (\mathrm{IGYC}\ \overline{\mathrm{I9}}\ \mathrm{X21} + \mathrm{IGYC}\ \overline{\mathrm{GX30}}\ \mathrm{X11} + \mathrm{IGYC}\ \mathrm{X31} + \mathrm{YOAS}\ \mathrm{X41} + \mathrm{STY}\ \mathrm{S8})\ \overline{\mathrm{GIDL}}$$

$$\overline{\mathrm{YC}} = \mathrm{RSTY}$$

$$\mathrm{CYC} = \mathrm{CKY}$$

$$\mathrm{RSTY} = \mathrm{YA} + \mathrm{YB} + \mathrm{YC} + \mathrm{RSTYA}$$

$$\mathrm{CKY} = \mathrm{XODD}\ \mathrm{CLK}$$

$$\mathrm{IGYC} = \mathrm{YOAS}\ \overline{\mathrm{CDBR}}\ \overline{\mathrm{DOX}}\ \overline{\mathrm{LNG}}$$

3.70 X-Timing

A ring counter consisting of flip-flops X1 through X7 provides the fine timing within the Y-states. All computer operations are performed within the various X-times. Without external gating, the flip-flops operate as a normal ring counter with each flip-flop gating the next. As most instructions do not require all of the X-times, external gating is used to skip from one X-time to another.

Each X-time consists of two clock periods, an even and an odd clock comprising a complete memory cycle. The even and odd clocks are gated with the X-times to form X10, X11, X20, X21, X30, X31, and so forth up to X71.

The even or odd X-time is gated with the appropriate logic signals to perform the operations desired for each instruction.

The outputs of the X-flip-flops can be monitored at test points on the front of the control panel. The signals at these test points are all in inverted form.

$$\mathrm{X1} = \mathrm{GX10} + \mathrm{X71}\ \mathrm{SKPX}$$

$$\overline{\mathrm{X1}} = \ldots$$

$$\mathrm{CN} = \mathrm{CKX}$$

$$\mathrm{X2} = \mathrm{GX20} + \mathrm{X11}\ \mathrm{SKPX}$$

$$\overline{\mathrm{X2}} = \ldots$$

$$\mathrm{CX2} = \mathrm{CKX}$$

$$\mathrm{X3} = \mathrm{GX30} + \mathrm{X21}\ \mathrm{SKPX}$$

$$\overline{\mathrm{X3}} = \ldots$$

$$\mathrm{CX3} = \mathrm{CKX}$$

$X4 = GX40 + X31\ SKPX$

$\overline{X4} = \ldots$

$CX4 = CKX$

$X5 = GX50 + X41\ SKPX$

$\overline{X5} = \ldots$

$CX5 = CKX$

$X6 = GX60 + X51\ SKPX$

$\overline{X6} = \ldots$

$CX6 = CKX$

$X7 = GX70 + X61\ SKPX$

$\overline{X7} = \ldots$

$CX7 = CKX$

$CKX = CLK + \overline{XOD} + Z1\ Z2$

$GX10 = GYC\ IGX10 + GIDL + STXYV\ S7 + \overline{YBGN}\ (\overline{GYA}\ \overline{GYB}\ \overline{ENDI})$

$GX20 = GYC\ IGX20 + STXYV\ S6$

$GX30 = STXYV\ S5 + SHRT\ I6\ X11$

$GX40 = STXYV\ S4 + DOX\ I7\ \overline{I6}\ X11 + DOX\ I10\ \overline{I9}\ X21$

$GX50 = IMDB\ \overline{IDZRO}\ X51 + STXYV\ S3$

$GX60 = ISFT\ \overline{NOSFT}\ X61 + GYC\ IGX60 + STXYV\ S2$

$GX70 = BBK\ MD\ X31 + BLK\ X31 + STXYV\ S1$

$SKPX = GX10 + GX20 + GX30 + GX40 + GX50 + GX60 + GX70$

$X10 = X1\ \overline{XODD}$
$X11 = X1\ XOD$

$X20 = X2\ \overline{XODD}$
$X21 = X2\ XOD$

$X30 = X3\ \overline{XODD}$
$X31 = X3\ XOD$

$X40 = X4\ \overline{XODD}$
$X41 = X4\ XOD$

$X50 = X5\ \overline{XODD}$
$X51 = X5\ XOD$

$X60 = X6\ \overline{XODD}$
$X61 = X6\ XOD$

$X70 = X7\ \overline{XODD}$
$X71 = X7\ XOD$

3.71 OPERATIONS

The following paragraphs describe the power startup circuits, power restart and power shutdown operations, memory requests and addressing, decoding of the instructions in the I-register and the preparation sequences for all instructions.

3.72 Power Startup

Figure 3-36 is a schematic diagram of the power startup circuits. The power startup consists of an initial condition circuit and a peak detector circuit with the logic elements to generate power OK signal POK and timing elements for various power functions such as power failure test with the B-E TEST switch on the control panel.

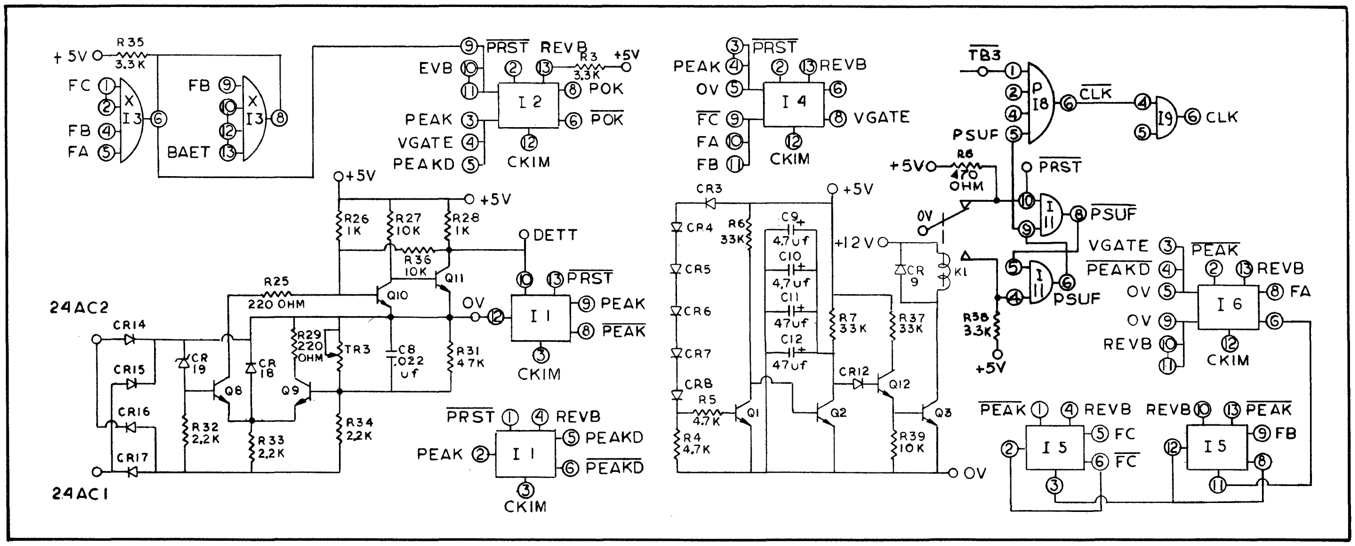


Figure 3-36. Power Detection, Schematic Diagram

The initial condition circuit consists of transistors Q1 through Q3, and Q12, relay K1, and their associated diodes, capacitors, and resistors. When power is first applied, relay K1 is deenergized and supplies signal PRST to circuits requiring an initial reset condition. Capacitors C9 through C12 begin charging to the positive supply voltage level. This charging current must go through 33k resistor R7 making the time constant approximately 0.6 second before the plate of CR12 is sufficiently positive to cause transistor Q12 to conduct.

A voltage divider network consisting of CR3 through CR8 and resistor R4 develops a positive voltage to keep transistor Q1 conducting, placing its collector at a 0V level and keeping transistor Q2 cut off. When the plate of diode CR12 becomes positive enough to conduct, transistor Q3 begins conducting and energizes relay K1 and removes the 0V $\overline{\text{PRST}}$ signal.

The peak detector circuit consists of transistors Q8 through Q11, diodes CR14 through CR18, flip-flop PEAK, and their associated resistors and capacitors. Alternating current voltage from the secondary winding of the power transformer is applied to 24AC1 and 24AC2. A diode bridge network consisting of CR14 through CR17 rectifies the ac voltage and applies the resultant voltage to the base of transistor Q8. Transistors Q8 and Q9 are connected in a modified Schmitt trigger configuration. The output of Q11 has a square wave pattern with a 0V time occurring at the peak of the ac signal approximately every 8.3 milliseconds. The output of the PEAK flip-flop is used to set the PEAKD and POK flip-flops.

Flip-flops FA, FB, and FC are used for timing during power startup and shutdown and also during the B-E test sequence to generate the 1 millisecond on and 7 milliseconds off power sequence.

3.73 Power Restart

When power is applied to the computer, signal POK is generated, beginning the power restart sequence. During power shutdown all information contained in the volatile registers was stored in preassigned memory locations. During the power restart operation the information stored in these memory locations is restored to their original registers and flip-flops. Table 3-2 gives the operation sequence during the power restart.

The following symbols are used in table 3-2 and in the sequence chart tables throughout this section:

Symbol	Meaning
$\longrightarrow$	Transfer to
$\not\longrightarrow$	Clock into

3.74 Power Shutdown

When input power to the computer falls below 100V, the power shutdown feature initiates a shutdown sequence to store the contents of all volatile registers in reserved memory locations. Table 3-3 gives the operation sequence during the power shutdown.

3.75 Memory Addressing and Access

Sixteen address lines (ADS1 through ADS16) provide the address to the memory. Inputs to the address lines are from the L-register, the V-register, the control panel, or the input/output buffer.

$$\begin{aligned}
\text{ADS16} &= \text{LADS L16} + \text{VADS V3} + \text{VPADS } \overline{\text{V3}} \\
&\quad + \text{UADS U16} + \text{GRT3 MRT16} + \text{CELL7} \\
\text{ADS15} &= \text{LADS L15} + \text{VADS V2} + \text{VPADS } \overline{\text{V2}} \\
&\quad + \text{UADS U15} + \text{GRT3 MRT15} + \text{CELL7} \\
\text{ADS14} &= \text{LADS L14} + \text{VADS V1} + \text{VPADS } \overline{\text{V1}} \\
&\quad + \text{UADS U14} + \text{GRT3 MRT14} + \text{CELL7} \\
\text{ADS13} &= \text{LADS L13} + \text{UADS U13} + \text{IMDB } \overline{\text{X1}} \\
&\quad + \text{ELD YC} \\
\text{ADS12} &= \text{LADS L12} + \text{UADS U12} + \overline{\text{Z1}}\text{ Z2} \\
&\quad + \text{YBGNF} + \text{YEND} \\
\text{ADS11} &= \text{LADS L11} + \text{UADS U11} \\
&\vdots \\
\text{ADS2} &= \text{LADS L2} + \text{UADS U2} \\
\text{ADS1} &= \text{YOAS X1 L1} + \text{LADS N1 L1}
\end{aligned}$$

The address of the next instruction to be executed is normally contained in the L-register. At the beginning of each instruction, the contents of the L-register are transferred to the adder where the address is increased by one. The new address is then returned to the L-register and also clocked into the N-register. During the Y-state configuration referred to as YOAS, gating signal LADS is true at all times except during time X4. Signal LADS gates the address from the L-register onto the address lines.

After the instruction has been read into the I-register, the lower half of the I-register containing the reference address is clocked into the L-register through the adder by gating signals ILTF and STL. In the adder, address modification is performed as designated by the XIRS bits in the instruction word. The resulting effective address is gated onto the address lines from the L-register as determined by the particular instruction logic.

During the last clock of each instruction, the contents of the N-register are returned to the L-register for fetching the next instruction.

When a power shutdown occurs, 0's are forced into flip-flops V1 through V3 and a 1 is forced onto address line ADS12. Gating signal VPADS comes true and forces address X'0010' onto the address lines for shutdown storage of the registers.

$$\text{VPADS} = \text{YEND}$$

Table 3-2. Power Restart, Sequence Chart

Time		Functions Performed	Signals Involved	Description
$\overline{\text{XODD}}$		Generate signal POK	POK = PEAK VGATE PEAKD	Indicates power level adequate for reliable computer operation
		Reset flip-flop $\overline{\text{Z0}}$	Z0 = POK $\overline{\text{Z1}}$ XODD CK2MS	Flip-flop $\overline{\text{Z0}}$ reversed. Reset for one clock period during restart and power failure conditions
		Generate signal $\overline{\text{GIDL}}$	$\overline{\text{GIDL}}$ = PSUF	

XODD		Set flip-flops YA and YB	YA = (Z0 $\overline{\text{Z1}}$ + . . .) $\overline{\text{GIDL}}$
			YB = (Z0 $\overline{\text{Z1}}$ + . . .) $\overline{\text{GIDL}}$
				Y-state flip-flops

		Generate SVNV	SVNV = Z0 $\overline{\text{Z1}}$	
		Generate SIXV	SIXV = SVNV	

		Reset flip-flops $\overline{\text{V1}}$, $\overline{\text{V2}}$, and $\overline{\text{V3}}$	V1 = SIXV + . . .
			V2 = SIXV + . . .
			V3 = SVNV + . . .
				Force X'7' into V-register

		Generate X10	X10 = (YA + YB) $\overline{\text{YBGN}}$	
X1	X10	Generate YBGN	YBGN = YA YB $\overline{\text{YC}}$ $\overline{\text{Z2}}$	Power startup Y-state

		Force X'17' onto address lines	ADS16 = VADS V3 + . . .
			ADS15 = VADS V2 + . . .
			ADS14 = VADS V1 + . . .
			ADS12 = YBGNF + . . .
			VADS = YBGN
				Force address X'17' onto address lines to read back contents stored during power shutdown

		$M \longrightarrow F$	MTF = YBGN + . . .	Contents of location X'17' gated into adder
		Reset flip-flop $\overline{\text{Z1}}$	Z1 = Z0 $\overline{\text{Z1}}$ + $\overline{\text{Z0}}$ Z1	Flip-flop $\overline{\text{Z1}}$ reversed. Power-on flip-flop
	X11	$S \not\longrightarrow N$	STN = YBGN $\overline{\text{VZRO}}$ X11 + . . .	Load next instruction address into N-register

		Set $\overline{\text{V3}}$; reset $\overline{\text{V2}}$ and $\overline{\text{V1}}$	$\overline{\text{V3}}$ = DECV V3 + . . .
			V2 = DECV V3 V2 + . . .
			V1 = DECV V2 V1 + . . .
			DECV = CTV $\overline{\text{SIXV}}$ $\overline{\text{VZRO}}$
				Count V-register down to 6

		$V \longrightarrow ADS$	VADS = YBGN	Force address X'16' onto address lines
		Generate X20	X20 = X11 $\overline{\text{SKPX}}$ + . . .	
X2	X20	$M \longrightarrow F$	MTF = YBGN + . . .	Gates contents of location X'16' into adder

Table 3-2. Power Restart, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X2	X21	$S \not\longrightarrow D$	STD = YBGN X21 + . . .	Load contents of memory location X'16' into D-register

		Reset flip-flops $\overline{V3}$ and $\overline{V1}$; set $\overline{V2}$	V3 = DECV V3 + . . .
			$\overline{V2}$ = DECV V2 $\overline{V3}$ + . . .
			V1 = DECV V2 V1 + . . .
			DECV = CTV $\overline{SIXV}$ $\overline{VZRO}$
				Count V-register to 5

		Reset flip-flop KC	$\overline{KC}$ = YBGN X21 + . . .	
		Generate X30	X30 = X21 $\overline{SKPX}$ + . . .	Follows X21 normally if no X-time is gated

X3
KC
	X30	$V \longrightarrow$ ADS inhibited	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Does not allow address from V-register onto address lines. Address on lines then is X'00'

	X31	Generate XADSA	XADSA = $\overline{ADS1}$ ($\overline{ADS3}$ thru $\overline{ADS12}$)	
		Generate MGO	MGO = ZMGO XODD + . . .	Begin memory cycle
		Reset flip-flop $\overline{MGOA}$	MGOA = MGO XADSA $\overline{ADS2}$ ($\overline{ADS13}$ thru $\overline{ADS16}$)	
		$M \not\longrightarrow IX$	CLKX = MGOA CLK	Clock contents of memory location X'00' into index register
		Set flip-flop KC	KC = YBGN $\overline{KC}$ X31 + . . .	

		Inhibit CTV	CTV = YBGNF $\overline{VZRO}$ XODD
			YBGNF = YBGN $\overline{X3}$ $\overline{KC}$
				Holds count of 5 in V-register

		Generate X30	X30 = YBGN $\overline{KC}$ X3 + . . .	Return to X30 time

	X30	$V \longrightarrow$ ADS	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Force address X'15' onto address lines

	X31	$M \longrightarrow F$	MTF = YBGN	Contents of memory location X'15' gated into adder
		$S \not\longrightarrow B$	STB = YBGN X31	Load contents of memory location X'15' into B-register
		Set flip-flops $\overline{V2}$ and $\overline{V3}$; reset $\overline{V1}$	$\overline{V3}$ = DECV V3 + . . . $\overline{V2}$ = DECV $\overline{V2}$ V3 + . . . V1 = DECV V1 V3 + . . .	Count V-register to 4
		Generate X40	X40 = X31 $\overline{SKPX}$ + . . .	Follows X31 normally if no X-time is gated

Table 3-2. Power Restart, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X4	X40	V ⟶ ADS	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Force address X'14' onto memory address lines

		Reset flip-flop KC	$\overline{KC}$ = YBGN X40 + . . .	
	X41	M ⟶ F	MTF = YBGN	Contents of memory location X'14' gated into adder
		$S \overset{/}{\longrightarrow} L$	STL = YBGN X41 + . . .	Load contents of memory location X'14' into L-register

		Reset flip-flops $\overline{V3}$ and $\overline{V2}$; set $\overline{V1}$	V3 = DECV $\overline{V3}$ + . . .
			V2 = DECV $\overline{V3}$ $\overline{V2}$ + . . .
			$\overline{V1}$ = DECV $\overline{V3}$ V1 + . . .
				Count V-register to 3

		Generate X50	X50 = X41 $\overline{SKPX}$ + . . .	Follows X41 normally if no X-time is gated

X5	X50	V ⟶ ADS	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Force address X'13' onto memory address lines

	X51	M ⟶ F	MTF = YBGN	Contents of memory location X'13' gated into adder
		$S \overset{/}{\longrightarrow} I$	STI = YBGN X51 + . . .	Load contents of memory location X'13' into I-register
		Set flip-flops $\overline{V3}$ and $\overline{V1}$; reset $\overline{V2}$	$\overline{V3}$ = DECV V3 + . . . V2 = DECV V3 $\overline{V2}$ + . . . $\overline{V1}$ = DECV V2 $\overline{V1}$ + . . .	Count V-register to 2
		Generate X60	X60 = X51 $\overline{SKPX}$ + . . .	X60 follows X51 normally if no X-time is gated

X6	X60	V ⟶ ADS	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Force address X'12' onto memory address lines

		Reset flip-flops KA, KB, KC, KK, KO, KS and KU	$\overline{KA}$, $\overline{KB}$, $\overline{KC}$, $\overline{KK}$ = RABCK
			RABCK = YBGN X40 + . . .
			$\overline{KO}$, $\overline{KS}$ and $\overline{KU}$ = RSUSO
			RSUSO = YBGN X60 + . . .	

	X61	M ⟶ F	MTF = YBGN	Contents of memory location X'12' gated into adder

		F ⟶ C	C13 = E13 + F13
			⋮ ⋮
			C16 = E16 + F16	

Table 3-2. Power Restart, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X6	X61	$C \nrightarrow T$	T1 = C13 RSHFT + . . .
			T2 = C14 RSHFT + . . .
			T3 = C15 RSHFT + . . .
			T4 = C16 RSHFT + . . .
			RSHFT = RS4
			RS4 = YBGNF X61 + . . .
				Load contents of memory location X'12' into T-flip-flops

		Reset flip-flop $\overline{V3}$; set flip-flops $\overline{V2}$ and $\overline{V1}$	V3 = DECV $\overline{V3}$ + . . . $\overline{V2}$ = DECV $\overline{V3}$ V2 + . . . $\overline{V1}$ = DECV V2 $\overline{V1}$ + . . .	Count V-register to 1
		Generate X70	X70 = X61 $\overline{SKPX}$ + . . .	X70 follows X61 normally if no X-time is gated

X7	X70	$V \longrightarrow ADS$	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Force address X'11' onto memory address lines

	X71	$M \longrightarrow F$	MTF = YBGN	Contents of memory location X'11' gated into adder
		$S \nrightarrow$ P, KA, KB, KC, KK, KO, KS and KU	STPKW = YBGN X71 + . . .	$S12 \nrightarrow KA$, $S11 \nrightarrow KB$, $S10 \nrightarrow KC$, $S9 \nrightarrow KK$, $S8\text{-}S3 \nrightarrow P1\text{-}P6$, $S2 \nrightarrow KU$, $S1 \nrightarrow KS$, $S0 \nrightarrow KO$
		Set flip-flops $\overline{V3}$, $\overline{V2}$, and $\overline{V1}$	$\overline{V3}$ = DECV V3 + . . . $\overline{V2}$ = DECV V3 $\overline{V2}$ + . . . $\overline{V1}$ = DECV V3 $\overline{V1}$ + . . .	Count V-register to 0
		Generate signal VZRO	VZRO = $\overline{V3}$ $\overline{V2}$ $\overline{V1}$	
		Generate X10	X10 = X71 $\overline{SKPX}$ + . . .	X10 normally follows X71 if no X-time is gated

X1	X10	$V \longrightarrow ADS$	VADS = YBGNF
			YBGNF = YBGN ($\overline{X3}$ + KC)
				Force address X'10' onto memory address lines

	X11	$M \longrightarrow F$	MTF = YBGN	Contents of memory location X'10' gated into adder

		$S \nrightarrow$ V, Y, X	STXYV = YBGN VZRO X11
			STY = STXYV $\overline{GIDL}$
				$S13 \nrightarrow V3$, $S12 \nrightarrow V2$, $S11 \nrightarrow V1$, $S10 \nrightarrow YA$, $S9 \nrightarrow YB$, $S8 \nrightarrow YC$, $S7\text{-}S1 \nrightarrow X1\text{-}X7$

Table 3-3. Power Shutdown, Sequence Chart

Time		Functions Performed	Signals Involved	Description
Xn	Xi	Generate $\overline{\text{POK}}$	$\overline{\text{POK}}$ = FA FB FC	Indicates power level has decreased below level for reliable computer operation
		Set flip-flop Z0	Z0 = $\overline{\text{POK}}$ Z1 $\overline{\text{IOIO}}$ $\overline{\text{XODD}}$	Flip-flop Z0 sets for one clock period during power restart or shutdown
	Xi+1	Set flip-flop Z2	Z2 = Z0 Z1 + . . .	Flip-flop indicates power failure
		Reset flip-flop Z1	$\overline{\text{Z1}}$ = Z0 Z1 + . . .	Flip-flop Z1 is the power-on flip-flop
	Xi+2	X, Y, V → Adder	XYVTF = $\overline{\text{Z1}}$ Z2	X1-X7 → F1-F7, YC → F8, YB → F9, YA → F10, V1-V3 → F11-F13

		F → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16	

		Reset flip-flop YA	$\overline{\text{YA}}$ = $\overline{\text{Z0}}$ $\overline{\text{Z1}}$ + . . .	
		Set flip-flop YB	YB = Z2 $\overline{\text{Z1}}$ $\overline{\text{GIDL}}$ + . . .	
		Reset flip-flop YC	$\overline{\text{YC}}$ = . . .	
		Inhibit L → ADS, $\overline{\text{V}}$ → ADS	ADS12 = $\overline{\text{Z1}}$ Z2 + . . .	Force address X'10' onto memory address lines
		Generate WM	WM = $\overline{\text{Z1}}$ Z2 + . . .	Write word from C-data gates into memory location X'10'

	Xi+3	6 → V	$\overline{\text{V3}}$ = DECV V3 + . . .
			V2 = SIXV + . . .
			V1 = SIXV + . . .
				Force a 6 into V-register

		Reset flip-flop Z2	$\overline{\text{Z2}}$ = Z2 XODD + . . .	
		Generate X10	X10 = YB $\overline{\text{YBGN}}$ + . . .	
X1	X10	Generate YEND	YEND = $\overline{\text{Z2}}$ $\overline{\text{YC}}$ YB $\overline{\text{YA}}$	

		$\overline{\text{V}}$ → ADS	ADS16 = VPADS $\overline{\text{V3}}$ + . . .
			ADS15 = VPADS $\overline{\text{V2}}$ + . . .
			ADS14 = VPADS $\overline{\text{V1}}$ + . . .
			ADS12 = $\overline{\text{YEND}}$ + . . .
			VPADS = YEND
				Force address X'11' onto memory address lines

Table 3-3. Power Shutdown, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X1	X10	P, K → Adder	DTE = YEND X10 + . . .
			PTF = KTE + . . .
				KO → E0, KS → E1,
				KU → E2,
				P1-P6 → F3-F8,
				KK → E9, KC → E10,
				KB → E11, KA → E12

		F, E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16	

	X11	Reset flip-flops $\overline{V3}$ and $\overline{V1}$; set $\overline{V2}$	V3 = DECV $\overline{V3}$ + . . .
			$\overline{V2}$ = DECV $\overline{V3}$ V2 + . . .
			V1 = DECV V2 V1 + . . .
			DECV = CTV $\overline{VZRO}$ $\overline{SIXV}$
			CTV = YEND $\overline{VZRO}$ XODD
				Count V-register to 5

		Generate WM	WM = YEND	Write word from C-data gates into memory location X'11'
		Generate X20	X20 = X11 $\overline{SKPX}$ + . . .	X20 follows X11 normally if no X-time is gated
X2	X20	$\overline{V}$ → ADS	VPADS = YEND	Force address X'12' onto memory address lines
		T → Adder	TTE = YEND X20	T1-T4 gated into E13-E16

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				E1-E16 gated into C1-C16

		Generate WM	WM = YEND	Write into memory signal
	X21	Reset flip-flop $\overline{MGO}$	MGO = YEND XODD + . . .	Write word from C-data gates into memory location X'12'
		Set $\overline{V3}$ and $\overline{V2}$; reset $\overline{V1}$	$\overline{V3}$ = DECV V3 + . . . $\overline{V2}$ = DECV V3 $\overline{V2}$ + . . . V1 = DECV V3 V1 + . . .	Count V-register to 4
		Generate X30	X30 = X21 $\overline{SKPX}$ + . . .	X30 follows X21 normally if no X-time is gated
X3	X30	$\overline{V}$ → ADS	VPADS = YEND	Force address X'13' onto memory address lines
		I → Adder	ITF = YEND X30 + . . .	Gate I-register into F-bus of adder

Table 3-3. Power Shutdown, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X3	X30	F ⟶ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E1 + F16
				F1-F16 gated into output data bus C1-C16

		Generate WM	WM = YEND + . . .	Write into memory signal
	X31	Reset flip-flop $\overline{\text{MGO}}$	MGO = YEND XODD + . . .	Write word from C-data bus into memory location X'13'

		Reset $\overline{\text{V3}}$ and $\overline{\text{V2}}$; set $\overline{\text{V1}}$	V3 = DECV $\overline{\text{V3}}$ + . . .
			V2 = DECV $\overline{\text{V3}}$ $\overline{\text{V2}}$ + . . .
			$\overline{\text{V1}}$ = DECV $\overline{\text{V3}}$ $\overline{\text{V1}}$ + . . .
				Count V-register to 3

		Generate X40	X40 = X31 $\overline{\text{SKPX}}$ + . . .	X40 follows X31 normally if no X-time is gated
X4	X40	$\overline{\text{V}}$ ⟶ ADS	VPADS = YEND	Force address X'14' onto memory address lines
		L ⟶ Adder	LTE = YEND X40 + . . .	Gate L-register contents into E-bus of adder

		E ⟶ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				E1-E16 gated into output data bus C1-C16

		Generate WM	WM = YEND + . . .	Write into memory signal
	X41	Reset flip-flop $\overline{\text{MGO}}$	MGO = YEND XODD + . . .	Write word from C-data bus into memory location X'14'
		Set $\overline{\text{V3}}$ and $\overline{\text{V1}}$; reset $\overline{\text{V2}}$	$\overline{\text{V3}}$ = DECV V3 + . . . V2 = DECV V3 V2 + . . . $\overline{\text{V1}}$ = DECV V3 $\overline{\text{V1}}$ + . . .	Count V-register to 2
		Generate X50	X50 = X41 $\overline{\text{SKPX}}$ + . . .	X50 follows X41 normally if no X-time is gated
X5	X50	$\overline{\text{V}}$ ⟶ ADS	VPADS = YEND	Force address X'15' onto memory address lines
		B ⟶ Adder	BTE = YEND X50 + . . .	B-register contents gated into E-bus of adder

		E ⟶ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				E1-E16 gated into output data bus C1-C16

Table 3-3. Power Shutdown, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X5	X50	Generate WM	WM = YEND + . . .	Write into memory signal
	X51	Reset flip-flop $\overline{MGO}$	MGO = YEND XODD + . . .	Write word from C-data bus into memory location X'15'
		Reset $\overline{V3}$; set $\overline{V2}$ and $\overline{V1}$	V3 = DECV $\overline{V3}$ + . . . $\overline{V2}$ = DECV V3 V2 + . . . $\overline{V1}$ = DECV V2 $\overline{V1}$ + . . .	Count V-register to 1
		Generate X60	X60 = X51 $\overline{SKPX}$ + . . .	X60 follows X51 normally if no X-time is gated
X6	X60	$\overline{V}$ → ADS	VPADS = YEND	Force address X'16' onto memory address lines
		$\overline{D}$ → Adder	DTE = YEND X60 + . . .	D-register contents gated into E-bus of adder

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				E1-E16 gated into output data bus C1-C16

		Generate WM	WM = YEND + . . .	Write into memory signal
	X61	Reset flip-flop $\overline{MGO}$	MGO = YEND XODD + . . .	Write word from C-data bus into memory location X'16'
		Set $\overline{V3}$, $\overline{V2}$ and $\overline{V1}$	$\overline{V3}$ = DECV V3 + . . . $\overline{V2}$ = DECV $\overline{V3}$ V2 + . . . $\overline{V1}$ = DECV V3 $\overline{V1}$ + . . .	Count V-register to 0
		Generate X70	X70 = X61 $\overline{SKPX}$ + . . .	X70 follows X61 normally if no X-time is gated
X7	X70	$\overline{V}$ → ADS	VPADS = YEND	Force address X'17' onto memory address lines
		N → Adder	NTE = YEND X70 + . . .	N-register contents gated into E-bus of adder

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				E1-E16 gated into output data bus C1-C16

		Generate VZRO	VZRO = $\overline{V3}$ $\overline{V2}$ $\overline{V1}$	
		Generate WM	WM = YEND + . . .	Write into memory signal

Table 3-3. Power Shutdown, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X7	X71	Reset flip-flop MGO	MGO = YEND XODD + . . .	Write word from C-data lines into memory location X'17'
		Generate GIDL	GIDL = YEND X7 + . . .	
		Reset flip-flop YB	$\overline{YB}$ = GIDL + . . .	Computer enters idle mode YIDL = $\overline{Z2}\ \overline{YC}\ \overline{YB}\ \overline{YA}$

When power startup occurs, 1's are forced in flip-flops V1 through V3 and a 1 is forced onto address line ADS12. Gating signal VADS comes true and forces address X'0017' onto the address lines for readout of the registers stored during power shutdown.

VADS = YBGN

YBGNF = YBGN + . . .

Memory locations X'0001' through X'0006' can be addressed from the control panel by means of the register select switch. The selected address is decoded by MRT16, MRT15, and MRT14 and forced onto address lines ADS16, ADS15, and ADS14 by gating signal GRT3.

When flip-flops Z1 and Z2 are true, indicating the input/output is requesting memory, signal UADS is generated, forcing the address from the input/output buffer lines U1 through U16 onto the memory address lines. Flip-flops Z1 and Z2 inhibit all Y-states, thereby inhibiting memory access by the computer.

UADS = Z1 Z2

After the address desired is on the address lines, information cannot be transferred to or from memory until memory gating signal MGO is generated. Signal MGO is generated by a flip-flop connected in a reverse configuration so its normal set state is $\overline{MGO}$. When memory access is desired, the flip-flop is reset, causing MGO to go high and $\overline{MGO}$ to go low. Signal $\overline{MGO}$ being low gates the memory timing cycle.

If data is to be written into memory, write memory signal WM must be generated. When the particular instruction requires writing into memory, the logic for the instruction forces signal $\overline{WM}$ to go low if the particular address is not in a locked-out area of memory. Signal $\overline{WM}$ is sent to memory to designate that a write operation is desired. When signal $\overline{WM}$ is high, a read operation is indicated.

$$\begin{aligned} MGO &= [(\overline{YIDL}\ Z1 + ZMGO + MRT\ X31\ Z1 \\ &\quad + MRT\ X61\ Z1 + LIDL\ Z1) + YEND]\ XODD \end{aligned}$$

CMGO = CLK

$$\begin{aligned} WM = {} & (IBLPK\ X4 + IBSV\ X6 + IBLPK\ X7 + YEND \\ & + GRT1\ MRT + Z2\ \overline{Z1} + Z1\ Z2\ CYSTW \\ & + IST\ X6 + IMDB\ X2 + IMDB\ X7 + LDX\ X2 \\ & + IMSC\ X3 + IBSV\ X3)\ \overline{INHWM} \end{aligned}$$

3.76 Instruction Word Decoding

After the instruction word is clocked into the I-register, the bits of the instruction word are decoded to provide the instruction decode signal, double or single word format, and effective address.

Bits 1 through 5 are first decoded to determine if the instruction is a single or double word instruction. If bits I1, I2, I4 and I5 are 1's and I3 is a 0, signal LNG is generated, indicating a double word instruction. Otherwise signal SHRT is generated.

LNG = YOAS IH3 IM1 I5

IH3 = I1 I2

IM1 = $\overline{I3}$ I4

SHRT = YOAS $\overline{IH3}$ + YOAS I3 + YOAS $\overline{I4}$ $\overline{I5}$

If the instruction is a double word type, the operation code is contained in bit positions I12 through I15 and the index bit is in position I16. Signal ILTIH is generated during time X2 to shift bit positions I12 through I16 into bit positions I1 through I5. Bit positions I1 through I4 are then decoded to generate the instruction gating signals as listed in table 3-4.

If the instruction is a double word instruction, memory access is made for the second word containing the reference address. The X, I, and R bits (bits I5, I9, and I10 in a double word, bits I5, I6, I7 and I8 in a single word) are decoded to determine address modification. Any address modification is performed in the adder and the resulting effective address is clocked into the L-register for operand fetching.

3.77 Preparation Sequences

All instructions cause one of three different preparation sequences to be performed as determined by the

Table 3-4. Instruction Decode Signals

Signal	Logic Equation	Instruction
AND	AND = IATH IH0 IM2	And
BBK	BBK = IRST IM3 $\overline{\text{I5}}$	Branch Back
BLK	BLK = IRST IM2 I5	Branch and Link
BNO	BNO = CDBR IM2 $\overline{\text{I5}}$	Branch No Overflow
BPT	BPT = IRST IM3 I5	Branch and Put
BXB	BXB = IRST IM2 $\overline{\text{I5}}$	Branch Index
CDBR	CDBR = YOAS IH2	Branch Equal, Branch Less Than or Equal, Branch Greater Than or Equal, Branch Unconditional, Branch No Overflow, Branch Less Than, Branch Greater Than, Branch Unequal
DVA DVB	DVA, DVB = NY4 IH1 IM1	Divide
ELB	ELB = ISFT ILL0	End Left
ELD	ELD = IXLH $\overline{\text{I6}}$ I7	Double Length Shift
HLT	HLT = IXLH $\overline{\text{I6}}$ ILL1	Halt
IADSB	IADSB = IATH IMD	Add, Subtract
IASF	IASF = ISFT I7	Arithmetic Left, Arithmetic Right, End Left, Logical Right
IATH	IATH = NY4 $\overline{\text{I1}}$ $\overline{\text{IM1}}$	Add, Subtract, Compare, And, Load, Store
IATMD	IATMD = NY4 $\overline{\text{I1}}$	Add, Subtract, Compare, Load, Store, And, Multiply, Divide
IBLPK	IBLPK = BLK + BPT + BBK	Branch and Link, Branch and Put, Branch Back
IBSV	IBSV = BLK + BPT	Branch and Link, Branch and Put
IBX	IBX = IRST IM2 $\overline{\text{I5}}$	Branch Index
ICM	ICM = IATH IH1 IM2	Compare
ILD	ILD = IATH IH0 IM3	Load
ILPW	ILPW = IXLH I6	Inclusive Or, Exclusive Or, Load Page
ILSF	ILSF = ISFT $\overline{\text{I7}}$	End Left, Logical Right
IMPY	IMPY = IH0 IM1 NY4	Multiply

Table 3-4. Instruction Decode Signals (Cont.)

Signal	Logic Equation	Instruction
IMSC	IMSC = IRST IM0 + IBX	Inclusive Or, Exclusive Or, Exchange Memory and Accumulator, Load Page, Halt, Double Length Shift, Branch Index
IOIO	IOIO = IRST IM1 $\overline{\text{I5}}$ I6	Set, Sense, Parallel Input, Parallel Output
ISFR	ISFR = ISFT I8	Logical Right, Arithmetic Right
ISFL	ISFL = ISFT $\overline{\text{I8}}$	End Left, Arithmetic Left
ISFT	ISFT = IRST IM1 $\overline{\text{I5}}$ $\overline{\text{I6}}$	End Left, Logical Right, Arithmetic Left, Arithmetic Right
ISHMSC	ISHMSC = IRST IM0 $\overline{\text{I5}}$ + ILPW + HLT	Load Index, Inclusive Or, Exclusive Or, Load Page, Halt
IST	IST = IATH IH1 IM3	Store
ISUB	ISUB = IATH IH1 $\overline{\text{I4}}$	Subtract
IXLH	IXLH = IRST IM0 I5	Inclusive Or, Exclusive Or, Exchange Memory and Accumulator, Load Page, Halt, Double Length Shift
LDX	LDX = IRST IMD $\overline{\text{I5}}$	Load Index
MPA MPB	MPA, MPB = IMPY	Multiply
OSFR	OSFR = IH3 $\overline{\text{I5}}$ ILL1 IM1	Logical Right, Sense
PIP	PIP = IOIO I7 $\overline{\text{I8}}$	Parallel Input
POP	POP = IOIO I7 I8	Parallel Output
SET	SET = IOIO ILL0	Set
SNS	SNS = IOIO ILL1	Sense
XMB	XMB = IXLH $\overline{\text{I6}}$ ILL0	Exchange Memory and Accumulator

instruction configuration: a single word instruction, a double word instruction that is not a conditional branch, or a double word instruction that is a conditional branch. Tables 3-5 through 3-7 give the preparation sequences for the three types.

3.78 INSTRUCTION OPERATION

The following paragraphs describe the computer instructions. Table 3-8 lists the instructions, their mnemonics, and their operation codes.

The instruction descriptions are arranged alphabetically by mnemonic. A sequence chart is given for each instruction detailing the functions performed and the applicable logic equations.

3.79 Add Instruction (ADB) (See table 3-9)

The add instruction adds the contents of the effective address to the contents of the B-register and loads the sum into the B-register. If overflow occurs, flip-flop KO is set; if the sign of the sum is negative, flip-flop KS is set, and if the sum is not equal to zero, flip-flop KU is set.

During odd clock X61, the contents of the effective address are transferred to the F-input bus of the adder

Table 3-5. Preparation Sequence, Single Word Instruction

Time		Functions Performed	Signals Involved	Description

		YOAS SHRT	YOAS = $\overline{\text{Z2}}$ $\overline{\text{YC}}$ $\overline{\text{YB}}$ YA
			SHRT = YOAS $\overline{\text{IH3}}$
				Power is applied and operand address state is true

		Generate X1	X1 = GX10
			GX10 = $\overline{\text{YBGN}}$ ($\overline{\text{GYA}}$ $\overline{\text{GYB}}$ $\overline{\text{ENDI}}$)	

X1 YOAS	X10	L+1 $\nrightarrow$ N	LTE = YOAS X10 + . . .	Transfer contents of L-register to adder

			K17 = YOAS X10 KK + . . .
			KK = ENDI + YIDL
				Force a 1 into K17 and add to L-register contents

			STN = YOAS X10 + . . .	Transfer address of next instruction into N-register
		L → ADS	LADS = YOAS $\overline{\text{X4}}$ + . . .	Gate instruction address onto memory address lines
		M $\nrightarrow$ I	MTI = YOAS X10 KK	Clock instruction into I-register
		Generate DLYCLK	DLYCLK = MTI + . . .	Delay clock to allow instruction to be fetched
		Generate X11	X11 = X1 XOD	
	X11	Decode instruction		Generate instruction signals as listed in table 3-4

		If CDBR and $\overline{\text{I6}}$, return to X10	GX10 = ENDI $\overline{\text{YBGN}}$ + . . .
			ENDI = CDBR X11 $\overline{\text{I6}}$ + . . .
				Instruction is a conditional branch and is not indirectly addressed

		If $\overline{\text{I5}}$-$\overline{\text{I8}}$: I9-I16 $\nrightarrow$ L	ILTF = YOAS X16 + . . .
			STL = YOAS LNG CDBR X11 + . . .
				Transfer reference address from I9-I16 to adder and clock address into L-register

		If $\overline{\text{I5}}$ $\overline{\text{I6}}$ I7 $\overline{\text{I8}}$:
		N+IL $\nrightarrow$ L
			ILTF = YOAS X11 + . . .
			NTE = YOAS I7 X11 + . . .
			STL = YOAS $\overline{\text{LNG}}$ $\overline{\text{CDBR}}$ X11 + . . .
				Only R-bit is a 1; add next instruction address from N-register to reference address from I-register and transfer result to L-register

		If $\overline{\text{I5}}$ $\overline{\text{I6}}$ I7 I8:
		N-IL $\nrightarrow$ L
			ILTF = YOAS X11 + . . .
			INVF = YOAS I7 I8 X11 + . . .
			NTE = YOAS I7 X11 + . . .
				R- and S-bits are both 1's indicating backward addressing; subtract reference address from next instruction address and transfer result to L-register

		If $\overline{\text{I5}}$ $\overline{\text{I6}}$ $\overline{\text{I7}}$ $\overline{\text{I8}}$:
		P+IL $\nrightarrow$ L
			STL = YOAS $\overline{\text{LNG}}$ $\overline{\text{CDBR}}$ X11 + . . .
			PTF = YOAS $\overline{\text{I7}}$ I8 X11 + . . .
			STL = YOAS $\overline{\text{LNG}}$ CDBR X11 + . . .
				Indicates page-relative addressing; contents of P-register transferred to F3-F8 and added to reference address in E9-E16. Result transferred to L-register

Table 3-5. Preparation Sequence, Single Word Instruction (Cont.)

Time		Functions Performed	Signals Involved	Description
X1 YOAS	X11	Fetch operand	LADS = YOAS $\overline{\text{X4}}$ + . . .	Fetch operand if required by instruction

		If $\overline{\text{I5}}$ $\overline{\text{I6}}$, reset flip-flop YA, set YC, and generate NY4	YC = IGYC $\overline{\text{GX30}}$ X11 + . . .
			IGYC = YOAS $\overline{\text{CDBR}}$ $\overline{\text{DOX}}$ $\overline{\text{LNG}}$
			NY4 = $\overline{\text{Z2}}$ YC $\overline{\text{YB}}$ $\overline{\text{YA}}$
				Begin execution sequence of instruction if not indexed or indirectly addressed

		If I6, generate X30	X30 = SHRT I6 X11 + . . .	If instruction is indirectly addressed, go to X30

		If $\overline{\text{I6}}$ I7, generate X40 and DOX	X40 = DOX I7 $\overline{\text{I6}}$ X11 + . . .
			DOX = YOAS $\overline{\text{I1}}$ I3
				If instruction is indexed, go to X40

X3 YOAS	X30	No Operation		
	X31	$M \longrightarrow F$	MTF = YOAS X31 + . . .	Fetch operand address from location specified in L-register
		$S \not\longrightarrow L$	STL = YOAS X31 + . . .	Clock new operand address into L-register
		Fetch operand	LADS = YOAS $\overline{\text{X4}}$ + . . .	Fetch operand from effective address in L-register
		Delay clock	DLYCLK = MTD + . . .	Delay clock to allow operand to be fetched
		If I5, generate X40	X40 = X31 $\overline{\text{SKPX}}$ + . . .	X40 follows X31 if no other X-time is gated

		If $\overline{\text{I5}}$, generate NY4	YC = YOAS $\overline{\text{CDBR}}$ X31 $\overline{\text{GIDL}}$
			NY4 = $\overline{\text{Z2}}$ YC $\overline{\text{YB}}$ $\overline{\text{YA}}$
				Begin execution sequence of instruction

X4 YOAS	X40	No operation		
	X41		DOX = YOAS $\overline{\text{I1}}$ I5	Indexing required
		Memory location X'0000' $\longrightarrow$ adder	MTF = YOAS X41 + . . .	LADS inhibited during X4 thereby forcing all 0's on address lines. Memory location X'0000' transferred to adder
		$L \longrightarrow E$	LTE = YOAS X41 + . . .	Direct address from L-register transferred to adder
		$S \not\longrightarrow L$	STL = LTE $\overline{\text{YIDL}}$ + . . .	Effective address transferred to L-register for fetching operand

		Generate NY4	YC = YOAS X41 $\overline{\text{GIDL}}$ + . . .
			NY4 = $\overline{\text{Z2}}$ YC $\overline{\text{YB}}$ $\overline{\text{YA}}$
				Begin execution sequence of instruction

Table 3-6. Preparation Sequence, Double Word Instruction, No Conditional Branch

Time		Functions Performed	Signals Involved	Description

		YOAS LNG	YOAS = $\overline{Z2}$ $\overline{YC}$ $\overline{YB}$ YA
			LNG = YOAS IH3 IM1 I5
				Power is applied and operand address select state is true

		Generate X1	X1 = GX10
			GX10 = $\overline{YBGN}$ ($\overline{GYA}$ $\overline{GYB}$ $\overline{INDI}$)
				Double word format indicated

X1 YOAS	X10	$L+1 \nrightarrow N$	LTE = YOAS X10 + . . .	Transfer contents of L-register to adder

			K17 = YOAS X10 KK + . . .
			KK = ENDI + YIDL
				Force a 1 into K17 and add to L-register contents

			STN = YOAS X10 + . . .	Transfer address of next instruction into N-register
		$L \rightarrow ADS$	LADS = YOAS $\overline{X4}$ + . . .	Gate instruction address onto address lines
		$M \nrightarrow I$	MTI = YOAS X10 KK	Clock instruction into I-register
		Generate DLYCLK	DLYCLK = MTI + . . .	Delay clock to allow instruction to be fetched
		Generate X11	X11 = X1 XOD	
	X11	No functions		
		Generate X20	X20 = X11 $\overline{SKPX}$ + . . .	X20 follows X11 if no other X-time is gated
X2 YOAS	X20	$I12\text{-}I16 \nrightarrow I1\text{-}I5$	ILTIH = YOAS X20	Shifts operand and index bit from I12-I16 into I1-I5 for decoding
			Decode instruction	Generate instruction signals as listed in table 3-4

		$N+1 \nrightarrow N$	NTE = YOAS X20 + . . .
			K17 = YOAS X20 + . . .
			STN = YOAS X20 + . . .
				Gate next instruction address into adder. Force a 1 into K17 and add to N-register contents. Next address skipped because of double word. New address returned to N-register

		$L \rightarrow ADS$	LADS = YOAS $\overline{X4}$	Address to fetch next half of double word

Table 3-6. Preparation Sequence, Double Word Instruction, No Conditional Branch (Cont.)

Time		Functions Performed	Signals Involved	Description

X2
YOAS
	X21	If I10:
		N+M $\not\longrightarrow$ L and IL
			MTF = YOAS X21 + . . .
			NTE = YOAS I10 X21 + . . .
			STL = YOAS $\overline{CDBR}$ X21 + . . .
			STIL = YOAS X21 + . . .
				Second half of double word gated into adder from memory. Next instruction address gated into adder. The resulting address is clocked into the L-register and lower half of I-register

		L $\longrightarrow$ ADS	LADS = YOAS $\overline{X4}$ + . . .	Gate operand address onto address lines

		If $\overline{I5}$ $\overline{I9}$, generate NY4	YC = $\overline{I9}$ X21 YOAS $\overline{CDBR}$ $\overline{DOX}$ $\overline{LNG}$ $\overline{GIDL}$
			NY4 = $\overline{Z2}$ YC $\overline{YB}$ $\overline{YA}$
				Begin execution sequence of instruction if not indexed and not indirectly addressed. $\overline{LNG}$ went true when I12-I16 was shifted into I1-I5

		If I9, generate X30	X30 = X21 $\overline{SKPX}$ + . . .	X30 follows X21 if no other X-time is gated
		If I5 $\overline{I9}$, generate X40	X40 = DOX I10 $\overline{I9}$ X21 + . . .	
X3 YOAS	X30	No functions		

	X31	If I9: M $\not\longrightarrow$ L and IL	MTF = YOAS X31 + . . .
			STL = YOAS X31 + . . .
			STIL = YOAS X31 + . . .
				Indirect addressed memory location gated into adder and then clocked into L-register and lower half of I-register

		L $\longrightarrow$ ADS	LADS = YOAS $\overline{X4}$ + . . .	Address to fetch operand

		If $\overline{I5}$, generate NY4	YC = X31 YOAS $\overline{CDBR}$ $\overline{LNG}$ $\overline{DOX}$ $\overline{GIDL}$
			NY4 = $\overline{Z2}$ YC $\overline{YB}$ $\overline{YA}$
				Begin execution sequence of instruction if not indexed

		If I5, generate X40	X40 = X31 $\overline{SKPX}$ + . . .	X40 follows X31 if no other X-time is gated
X4 YOAS	X40	No functions		

	X41	If I5:
		M X'0000' $\longrightarrow$ Adder
			DOX = YOAS $\overline{I1}$ I5
			MTF = YOAS X41 + . . .
				Indexing required. LADS inhibited during X4 thereby forcing all 0's onto address lines. Memory location X'0000' transferred to adder

		L $\longrightarrow$ E	LTE = YOAS X41 + . . .	Direct address from L-register transferred to adder
		S $\not\longrightarrow$ L	STL = LTE $\overline{YIDL}$ + . . .	Effective address transferred to L-register for fetching operand

		Generate NY4	YC = YOAS X41 $\overline{GIDL}$ + . . .
			NY4 = $\overline{Z2}$ YC $\overline{YB}$ $\overline{YA}$
				Begin execution sequence of instruction

Table 3-7. Preparation Sequence, Double Word Instruction, Conditional Branch

Time		Functions Performed	Signals Involved	Description

		YOAS LNG	YOAS = $\overline{\text{Z2}}$ $\overline{\text{YC}}$ $\overline{\text{YB}}$ YA
			LNG = YOAS IH3 IM1 I5
				Power is applied and operand address select state is true

		Generate X1	X1 = GX10
			GX10 = YBGN ($\overline{\text{GYA}}$ $\overline{\text{GYB}}$ $\overline{\text{GYC}}$)
				Double word format indicated

X1
YOAS
	X10	L+1 —/→ N	LTE = YOAS X10 + . . .
			K17 = YOAS KK X10 + . . .
			KK = ENDI + . . .
			STN = YOAS X10 + . . .
				Address from N-register gated to adder and increased by one for next instruction. Resulting address clocked into N-register

		L ——→ ADS	LADS = YOAS $\overline{\text{X4}}$	Address gated onto address lines to fetch instruction
		M —/→ I	MTI = YOAS X10 KK	Instruction clocked into I-register
		Generate DLYCLK	DLYCLK = MTI + . . .	Delay next clock to allow time to fetch instruction
	X11	Generate X20	X20 = X11 $\overline{\text{SKPX}}$ + . . .	X20 generated normally after X11 if no X-time is gated
X2 YOAS	X20	I12-I16 —/→ I1-I5	ILTIH = YOAS X20	Opcode and index bit shifted into I1-I5 for decoding

		N+1 —/→ N	NTE = YOAS X20 + . . .
			K17 = YOAS X20 + . . .
			STN = YOAS X20 + . . .
				Transfer contents of N-register to adder. Force a 1 into K17 and add to contents of N-register for next instruction address. Transfer new address to N-register

		L ——→ ADS	LADS = YOAS $\overline{\text{X4}}$ + . . .	Gate address for second half of double word onto address lines

	X21	If I10 is a 1 and BRSAT is true: N+M —/→ L	MTF = YOAS X21 + . . .
			NTE = YOAS I10 X21 + . . .
			STL = CDBR BRSAT XODD + . . .
				If the branch conditions are satisfactory and the R-bit is a 1, add contents of N-register to reference address and clock into L-register

		If I10 is a 0 and BRSAT is true: M —/→ L	MTF = YOAS X21 + . . .
			STL = CDBR BRSAT XODD + . . .
				If the branch conditions are satisfactory and the R-bit is a 0, transfer the reference address from memory to the L-register

		If $\overline{\text{I9}}$, generate ENDI	ENDI = CDBR $\overline{\text{I9}}$ X21 + . . .	
		If $\overline{\text{I9}}$ and CDBR conditions were met: L ——→ ADS	LADS = YOAS $\overline{\text{X4}}$ + . . .	If branch conditions were met, branch to effective address contained in L-register

Table 3-7. Preparation Sequence, Double Word Instruction, Conditional Branch (Cont.)

Time		Functions Performed	Signals Involved	Description
X2 YOAS	X21	If $\overline{I9}$ and CDBR conditions were not met: N $\nrightarrow$ L	NTL = IBX X21 EZRO + . . .	Clock next instruction address into L-register to read next instruction
		If I9, generate X30	X30 = X21 $\overline{SKPX}$ + . . .	X30 follows X21 if no other X-state has been gated
		If $\overline{I9}$, generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	Begin preparation sequence of next instruction
X3 YOAS	X30	No functions		
	X31	M → Adder	MTF = YOAS X31 + . . .	Gate resulting indirect address from memory into adder

		S $\nrightarrow$ L and IL	STL = YOAS X31 + . . .
			STIL = YOAS X31 + . . .
				Gate effective address from adder into L-register and lower half of I-register

		Generate ENDI	ENDI = CDBR X31 + . . .	
		Generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	Begin preparation sequence of next instruction

Table 3-8. RC 70 Instructions

MNEMONIC	NAME	OPERATION CODE	
		Single Word	Double Word
ADB	Add	00	D820
ALB	Arithmetic Left	D2	DA3A
ANB	And	20	D824
ARB	Arithmetic Right	D3	DB3A
BBK	Branch Back	F0	D83E
BEQ	Branch Equal	80	D830
BGE	Branch Greater Than or Equal	90	D832
BGT	Branch Greater Than	B0	D836
BLE	Branch Less Than or Equal	88	D831
BLK	Branch and Link	E8	D83D
BLT	Branch Less Than	A8	D835

Table 3-8. RC 70 Instructions (Cont.)

MNEMONIC	NAME	OPERATION CODE	
		Single Word	Double Word
BNE	Branch Unequal	B8	D837
BNO	Branch No Overflow	A0	D834
BPT	Branch and Put	F8	D83F
BUC	Branch Unconditional	98	D833
BXB	Branch Index	E0	D83C
CMB	Compare	60	D82C
DVB	Divide	50	D82A
ELB	End Left	D0	D83A
ELD	Double Length Shift	CA	DA39
HLT	Halt	C9	D939
IOR	Inclusive Or	CC	DC39
LDB	Load	30	D826
LDP	Load Page	CE	DE39
LDX	Load Index	C0	D838
LRB	Logical Right	D1	D93A
MPB	Multiply	10	D822
PIP	Parallel Input	D6	DE3A
POP	Parallel Output	D7	DF3A
SBB	Subtract	40	D828
SET	Set	D4	DC3A
SNS	Sense	D5	DD3A
STB	Store	70	D82E
XMB	Exchange Memory and Accumulator	C8	D839
XOR	Exclusive Or	CD	DD39

Table 3-9. Add Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = $\overline{I1}$ $\overline{IM1}$ YOAS + . . .
			IM1 = $\overline{I3}$ I4	

X6 NY4	X61	$L \longrightarrow ADS$	LADS = IATH X6 + . . .	Operand address gated onto address lines
		$M \longrightarrow Adder$	MTF = IATH $\overline{IST}$ X61 + . . .	Operand gated into adder

		$B \longrightarrow Adder$	BTE = IATH $\overline{ILD}$ X61 + . . .
			IATH = NY4 $\overline{I1}$ $\overline{IM1}$
				Operand from B-register gated into adder

		$M+B \not\longrightarrow B$	STB = IATH $\overline{ICM}$ X61 + . . .	Sum transferred to B-register
		Generate ENDI	ENDI = IATH X61 + . . .	End instruction execution
		$N \not\longrightarrow L$	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	

and the contents of the B-register are transferred to the E-input bus of the adder. The sum of the data is transferred to the B-register.

3.80 Arithmetic Left Instruction (ALB) (See table 3-10)

The arithmetic left instruction shifts the contents of the B-register to the left the number of bit positions specified by bit positions 13 through 16 of the instruction. As the contents are shifted, bits shifted out of bit position 1 are lost and zeros are loaded into bit position 16. Sign flip-flop KS is set if the sign of the result is negative. Unequal flip-flop KU is set if the contents are not equal to zero after the shift.

During even clock X60, the contents of the B-register are transferred to the E-input bus of the adder. Bits I13 through I16 are decoded to determine if the shift should be a 1-bit or a 4-bit shift. The shift is performed and the I-register is then counted down by either 1 count or 4 counts as determined by the shift. The shifted data is then returned to the B-register.

During odd clock X61, the process is repeated and the B-register contents are again shifted either 1 or 4 bit places. The I-register count is again decreased by the amount of the shift, and the shifted data is returned to the B-register.

If the shift count is not zero, the instruction cycles through X60 and X61 until the count reaches zero. Signal NOSFT is generated, and shifting is inhibited.

3.81 And Instruction (ANB) (See table 3-11)

The And instruction forms the logical product of the contents of the effective address and the contents of the B-register and loads the result into the B-register. If the sign of the result is negative, flip-flop KS is set; flip-flop KU is set if the result is not equal to zero.

During odd clock X61, the contents of the effective address are transferred to the F-input bus of the adder and the contents of the B-register are transferred to the E-input bus of the adder. Signal EFTF causes a logical And operation to be performed, and the result is loaded into the B-register. Flip-flops KS and KU are set as determined by the contents of the sum bus.

3.82 Arithmetic Right Instruction (ARB) (See table 3-12)

The Arithmetic Right instruction shifts the contents of the B-register to the right the number of bit positions specified by bits 13 through 16 of the instruction word. As the contents are shifted, the sign bit is extended to the right and the bits shifted out of bit position 16 are lost. Sign flip-flop KS is set if the contents are negative and Unequal flip-flop KU is set if the shifted result is not equal to zero.

During even clock X60, the contents of the B-register are transferred to the E-input bus of the adder. Bits I13 through I16 are decoded to determine if the shift should be a 1-bit or a 4-bit shift. The shift is performed, and the data is returned to the B-register. The I-register

Table 3-10. Arithmetic Left Shift Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS I0 I1 IM1 $\overline{\text{I6}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X6
NY4
	X60	Generate IASF	IASF = ISFT I7
			ISFT = IRST IM1 $\overline{\text{I5}}$ $\overline{\text{I6}}$
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{\text{I3}}$ I4	

		Generate ISFL	ISFL = ISFT $\overline{\text{I8}}$	
		If $\overline{\text{I15}}$ $\overline{\text{I16}}$, generate DSFT	DSFT = $\overline{\text{I15}}$ $\overline{\text{I16}}$ (I13 + I14)	Indicates 4-bit shifts
		If $\overline{\text{I13}}$ $\overline{\text{I14}}$, generate BSFT	BSFT = $\overline{\text{I13}}$ $\overline{\text{I14}}$ (I15 + I16)	Indicates 1-bit shifts
		If DSFT, generate LS4	LS4 = ISFL DSFT SFTTME	
		If I15 or I16, generate LS1	LS1 = ISFL $\overline{\text{BZRO}}$ SFTTME $\overline{\text{BZRO}}$ = I15 + I16	
		Generate LSHFT	LSHFT = LS1 + LS4	
		Generate SHIFT	SHIFT = LS1 + LS4 + . . .	
		B → Adder	BTE = ISFT X6 + . . .	Contents of B-register gated to E-input of adder
		If LS1: E2 → E1A : : E16 → E15A		Left shift E-data 1 bit place into EA bus

		If LS4: E5 → E1A
		: :
		H4 → E16A
			H1-H4 = HET HRST
			HET = DVB + ELB
				Left shift E-data 4 bit places into EA bus. H1-H4 are 0's

		Count I-register	CTI = ISFT $\overline{\text{ELB}}$ X6 + . . .	
		If I15 or I16, generate CTIB	CTIB = CTI $\overline{\text{BZRO}}$ + . . . $\overline{\text{BZRO}}$ = I15 + I16	Decrease I-register count by 1

		If I13 or I14, generate CTID	CTID = CTI DSFT
			DSFT = $\overline{\text{DZRO}}$ BZRO
			$\overline{\text{DZRO}}$ = I13 + I14
				Decrease I-register count by 4

		S —/→ B	STB = ISFT $\overline{\text{ELB}}$ X6 + . . .	Return shifted data to B-register
	X61	B → Adder	BTE = ISFT X6 + . . .	Contents of B-register gated to E-input of adder
		If LS1: E2 → E1A : : E16 → E15A		Left shift E-data 1 bit place into EA bus

Table 3-10. Arithmetic Left Shift Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X6
NY4
	X61	If LS4: E5 → E1A
		⋮ ⋮
		H4 → E16A
			H1-H4 = HET + HRST
			HET = DVB + ELB
				Left shift E-data 4 bit places into EA bus. H1-H4 are 0's

		Count I-register	CTI = ISFT $\overline{\text{ELB}}$ X6 + . . .	
		If I15 or I16, generate CTIB	CTIB = CTI $\overline{\text{BZRO}}$ + . . . $\overline{\text{BZRO}}$ = I15 + I16	Decrease I-register count by 1

		If I13 or I14, generate CTID	CTID = CTI DSFT
			DSFT = $\overline{\text{DZRO}}$ BZRO
			$\overline{\text{DZRO}}$ = I13 + I14
				Decrease I-register count by 4

		S $\nrightarrow$ B	STB = ISFT $\overline{\text{ELB}}$ X6 + . . .	Return shifted data to B-register
		If shift count is not 0, return to X60	X60 = ISFT $\overline{\text{NOSFT}}$ X61 + . . .	

		When shift count is 0, generate NOSFT	NOSFT = ISFT IDZRO WZRO
			IDZRO = $\overline{\text{I13}}$ $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$
			WZRO = $\overline{\text{I10}}$ $\overline{\text{I11}}$ $\overline{\text{I12}}$
				Shift count has been counted down to 0

		Generate ENDI	ENDI = NOSFT X61 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$ + . . .	
		Set flip-flop KS if sign of result is negative	KS = ISFT B1 X61 + . . .	

		If contents are not all 0's, set flip-flop KU	KU = KUTME $\overline{\text{SZRO}}$ + . . .
			KUTME = ISFT X61 + . . .
			$\overline{\text{SZRO}}$ = S1 + S2 + S3 + S4 + S5 + S6 + S7 + S8 + S9 + S10 + S11 + S12 + S13 + S14 + S15 + S16	

Table 3-11. And Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS $\overline{\mathrm{I1}}$ $\overline{\mathrm{IM1}}$ + . . .
			IM1 = $\overline{\mathrm{I3}}$ I4	

X6 NY4	X60	No functions		
	X61	L ⟶ ADS	LADS = IATH X6 + . . .	Effective address gated onto address lines

		M ⟶ Adder	MTF = IATH $\overline{\mathrm{IST}}$ X61 + . . .
			IST = IATH IH1 IM3
			IH1 = $\overline{\mathrm{I1}}$ I2
			IM3 = I3 I4
				Contents of effective address memory location gated into adder

		B ⟶ Adder	BTE = IATH $\overline{\mathrm{ILD}}$ X61 + . . .
			ILD = IATH IH0 IM3
				Contents of B-register gated into adder

		$E \cap F \longrightarrow F$	EFTF = AND X6	Perform logical And operation
		$S \not\longrightarrow B$	STB = IATH $\overline{\mathrm{ICM}}$ X61 + . . .	Store result in B-register

		Set flip-flop KS if sign of result is negative	KS = IATH BINSIN $\overline{\mathrm{IST}}$ X61 + . . .
			BINSIN = S1 ⊕ BINOVF
			$\overline{\mathrm{KS}}$ = IATH $\overline{\mathrm{IST}}$ X60 + . . .
				If both operands were negative, flip-flop KS is set

		If the result is not all 0's, set flip-flop KU	KU = KUTME $\overline{\mathrm{SZRO}}$ + . . . $\overline{\mathrm{KUTME}}$ = IATH $\overline{\mathrm{IST}}$ X61 + . . . $\overline{\mathrm{SZRO}}$ = S1 + S2 + S3 + S4 + S5 + S6 + S7 + S8 + S9 + S10 + S11 + S12 + S13 + S14 + S15 + S16 $\overline{\mathrm{KU}}$ = IATH $\overline{\mathrm{IST}}$ X60 + . . .	Result on sum bus is not all 0's
		Generate ENDI	ENDI = IATH X61 + . . .	End execution sequence
		$N \not\longrightarrow L$	NTL = NY4 $\overline{\mathrm{IBX}}$ $\overline{\mathrm{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\mathrm{YBGN}}$ + . . .	

Table 3-12. Arithmetic Right Shift Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS I0 I1 IM1 $\overline{\text{I6}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X6
NY4
	X60	Generate IASF	IASF = ISFT I7
			ISFT = IRST IM1 $\overline{\text{I5}}$ $\overline{\text{I6}}$
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{\text{I3}}$ I4	

		Generate ISFR	ISFR = ISFT I8	
		If $\overline{\text{I15}}$ $\overline{\text{I16}}$, generate DSFT	DSFT = $\overline{\text{I15}}$ $\overline{\text{I16}}$ (I13 + I14)	Indicates 4-bit shifts
		If $\overline{\text{I13}}$ $\overline{\text{I14}}$, generate BSFT	BSFT = $\overline{\text{I13}}$ $\overline{\text{I14}}$ (I15 + I16)	Indicates 1-bit shifts
		If DSFT, generate RS4	RS4 = ISFR DSFT SFTTME	
		If I15 or I16, generate RS1	RS1 = ISFR $\overline{\text{BZRO}}$ SFTTME	
		Generate RSHFT	RSHFT = RS1 + RS4	
		Generate SHIFT	SHIFT = RS1 + RS4 + . . .	
		B → Adder	BTE = ISFT X6 + . . .	Contents of B-register gated into E-input of adder

		B1 → H1-H4	H1-H4 = HET + HRST
			HRST = IASF ISFR B1 X6
				Sign bit copied into H1-H4

		If RS1: H4 → E1A ⋮ ⋮ E15 → E16A		Right shift E-data 1 bit place into EA bus. Extend sign bit from H4
		If RS4: H1 → E1A ⋮ ⋮ E12 → E16A		Right shift E-data 4 bit places into EA bus. Extend sign bit from H1-H4
		Count I-register	CTI = ISFT $\overline{\text{ELB}}$ X6 + . . .	
		If I15 or I16, generate CTIB	CTIB = CTI $\overline{\text{BZRO}}$ + . . . $\overline{\text{BZRO}}$ = I15 + I16	Decrease I-register count by 1
		If I13 or I14, generate CTID	CTID = CTI DSFT	Decrease I-register count by 4
		S ↛ B	STB = ISFT $\overline{\text{ELB}}$ X6 + . . .	Return shifted data to B-register
	X61	B → Adder	BTE = ISFT X6 + . . .	Contents of B-register gated into E-input of adder

Table 3-12. Arithmetic Right Shift Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X6
NY4
	X61	B1 $\rightarrow$ H1-H4	H1-H4 = HET + HRST
			HRST = IASF ISFR B1 X6
				Sign bit copied into H1-H4

		If RS1: H4 $\rightarrow$ E1A ⋮ ⋮ E15 $\rightarrow$ E16A		Right shift E-data 1 bit place into EA bus. Extend sign bit from H4
		If RS4: H1 $\rightarrow$ E1A ⋮ ⋮ E12 $\rightarrow$ E16A		Right shift E-data 4 bit places into EA bus. Extend sign bit from H1-H4
		Count I-register	CTI = ISFT $\overline{\text{ELB}}$ X6 + . . .	
		If I15 or I16, generate CTIB	CTIB = CTI $\overline{\text{BZRO}}$ + . . . $\overline{\text{BZRO}}$ = I15 or I16	Decrease I-register count by 1
		If I13 or I14, generate CTID	CTID = CTI DSFT	Decrease I-register count by 4
		S $\nrightarrow$ B	STB = ISFT $\overline{\text{ELB}}$ X6 + . . .	Return shifted data to B-register
		If shift count is not 0, return to X60	X60 = ISFT $\overline{\text{NOSFT}}$ X61 + . . .	

		When shift count is 0, generate NOSFT	NOSFT = ISFT IDZRO WZRO
			IDZRO = $\overline{\text{I13}}$ $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$
			WZRO = $\overline{\text{I10}}$ $\overline{\text{I11}}$ $\overline{\text{I12}}$
				Shift count has been counted down to 0

		Set flip-flop KS if sign of result is negative	KS = ISFT B1 X61 + . . .	

		If contents are not all 0's, set KU	KU = KUTME $\overline{\text{SZRO}}$ + . . .
			KUTME = ISFT X61 + . . .
			$\overline{\text{SZRO}}$ = S1 + S2 + . . . S16	

		Generate ENDI	ENDI = NOSFT X61 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

is then counted down by either 1 count or 4 counts as determined by the shift performed.

During odd clock X61, the process is repeated, and the B-register contents are again shifted either 1 or 4 bit positions. The I-register count is decreased by the amount of the shift, and the shifted data is returned to the B-register.

If the shift count is not zero, the instruction cycles through X60 and X61 until the count reaches zero. Signal NOSFT is generated, and shifting is inhibited.

3.83 Branch Back Instruction (BBK) (See table 3-13)

The Branch Back instruction returns the program to the instruction immediately following the last executed

Table 3-13. Branch Back Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate BBK	BBK = IRST IM3 $\overline{\text{I5}}$
			IRST = NY4 IH3
			IH3 = I1 I2
			IM3 = I3 I4	

		Generate IBLPK	IBLPK = BBK + . . .	
		Generate CELL7	CELL7 = IBLPK X2 + . . .	
		1 ⟶ ADS16, 15, 14	ADS16, ADS15, ADS14 = CELL7	Force address X'0007' onto address lines to select roll-arrow register
		M —/→ D	MTD = IBLPK X2 + . . .	Load contents of roll-arrow register into D-register
	X21	D —/→ Adder	DTE = BBK X21 + . . .	Transfer contents of D-register to adder
		Generate INVF	INVF = BBK X21 + . . .	Force all 1's onto F-input bus and add to contents of D-register
		S —/→ L	STL = IBLPK X21 + . . .	Transfer new address to L-register
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	Follows X21 normally if no X-time is gated
X3 NY4	X30	L ⟶ ADS	LADS = IBLPK X3 + . . .	Address of last entry stored in roll table
	X31	M ⟶ Adder	MTF = BBK X31 + . . .	Contents of last entry in roll table transferred to adder
		If M0 is a 1, generate X70	X70 = BBK M0 X31 + . . .	If M0 is a 1, entry was stored by a Branch and Link instruction
		S —/→ N	STN = BBK X31 + . . .	Last entry from roll table is loaded into next instruction register
		If M0 is a 0, generate X40	X40 = X31 $\overline{\text{SKPX}}$ + . . .	X40 normally follows X31 if no X-time is gated
X4 NY4	X40	N ⟶ Adder	NTE = BBK X40 + . . .	Last entry from roll table transferred to adder

Table 3-13. Branch Back Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X4 NY4	X40	Inhibit L → ADS	No gating term true	Force address X'0000' onto address lines (index register)

		E → C	C1 = E1 + F1
			⋮
			C16 = E16 + F16
				Data transferred to output bus

		Generate MGO	MGO = $\overline{\text{YIDL}}$ XODD + . . .	Begin memory cycle
		Generate WM	WM = IBLPK X4 + . . .	Write data into memory location X'0000' and into index register IX0-IX16
	X41	L → Adder	LTE = IBLPK X41 + . . .	Transfer address from L-register to adder
		Generate INVF	INVF = BBK X41 + . . .	Force all 1's into F-input bus and add to address on E-input bus to decrease address by one
		S $\not\rightarrow$ L	STL = LTE $\overline{\text{YIDL}}$ + . . .	Transfer new address to L-register
		Generate X50	X50 = X41 $\overline{\text{SKPX}}$ + . . .	X50 follows X41 normally if no X-time is gated
X5 NY4	X50	No functions		
	X51	S $\not\rightarrow$ L	STL = LTE $\overline{\text{YIDL}}$ + . . .	Transfer new address to L-register
		L → ADS	LADS = BBK X5 + . . .	Transfer next entry address onto address lines
		M $\not\rightarrow$ D	MTD = BBK X51 + . . .	Contents of addressed memory location transferred to D-register
		L → Adder	LTE = BBK X51 + . . .	Transfer address from L-register to adder
		Generate INVF	INVF = BBK X51 + . . .	Force all 1's into F-input bus and add to address on E-input bus to decrease address by one
		Generate X60	X60 = X51 $\overline{\text{SKPX}}$ + . . .	X60 follows X51 normally if no X-time is gated
X6 NY4	X60	D → Adder	DTE = BBK X60 + . . .	Contents of D-register transferred to adder

Table 3-13. Branch Back Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X60	Generate STPKW	STPKW = BBK X60 + . . .	
		S0 $\nrightarrow$ KO	KO = STPKW S0 + . . .	Bit S0 clocked into Overflow flip-flop
		S1 $\nrightarrow$ KS	KS = STPKW S1 + . . .	Bit S1 clocked into Sign flip-flop
		S2 $\nrightarrow$ KU	KU = STPKW S2 + . . .	Bit S2 clocked into Unequal flip-flop
		S3-S8 $\nrightarrow$ P	P1-P6 = (STPKW) S3-S8	Bits S3-S8 clocked into page register
	X61	L → ADS	LADS = IBLPK X6 + . . .	Address of next entry in roll table gated onto address lines
		M → Adder	MTF = BBK X61 + . . .	Contents of addressed location transferred to adder
		S $\nrightarrow$ N	STN = BBK X61 + . . .	Link address returned to N-register
		Generate X70	X70 = X61 $\overline{\text{SKPX}}$ + . . .	X70 follows X61 normally if no X-time is gated
X7 NY4	X70	Generate CELL7	CELL7 = IBLPK X7 + . . .	
		1's → ADS16, 15, 14	ADS16, ADS15, ADS14 = CELL7	Force address X'0008' onto address lines
		L → Adder	LTE = IBLPK X70 + . . .	Last address of roll table transferred to adder

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Data transferred to output data bus

		Generate MGO	MGO = $\overline{\text{YIDL}}$ Z1 + . . .	Start memory cycle
		Generate WM	WM = IBLPK X7 + . . .	Write roll table address into location X'0007'
	X71	Generate ENDI	ENDI = NY4 X71 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

Branch and Link or Branch and Put instruction. If the last executed instruction was Branch and Link, the Branch Back instruction fetches the return link address from the roll table in memory and returns program control to the instruction stored at the link address. If the last executed instruction was Branch and Put, the Branch Back instruction fetches program status information from the roll table and restores the contents of the index register, page register, and Overflow, Sign, and Unequal indicators to the conditions that existed at the time the Branch and Put instruction was executed. Then the Branch Back instruction fetches the return link address from the roll table and returns program control to the instruction stored at the link address.

3.84 Conditional Branch Instructions (CDBR)

The conditional branch instructions are: Branch Equal (BEQ), Branch Greater Than or Equal (BGE), Branch Greater Than (BGT), Branch Less Than or Equal (BLE), Branch Less Than (BLT), Branch Unequal (BNE), Branch No Overflow (BNO), and Branch Unconditional (BUC).

There are no execution sequences to these instructions as the complete operation is performed during the preparation sequences as given in table 3-7.

The Branch Equal instruction causes the program to branch to the effective address if the result of the last instruction tested equals zero (Unequal indicator KU reset). If the tested result does not equal zero (KU set), program execution continues with the next instruction in sequence.

The Branch Greater Than or Equal instruction causes the program to branch to the effective address if the result of the last instruction tested is either greater than or equal to zero according to the states of Sign indicator KS and Unequal indicator KU. If the tested result is less than zero, program execution continues with the next instruction in sequence.

The Branch Greater Than instruction causes the program to branch to the effective address if the result of the last instruction tested is greater than zero according to the states of Sign indicator KS and Unequal indicator KU. If the tested result is equal to or less than zero, program execution continues with the next instruction in sequence.

The Branch Less Than or Equal instruction causes the program to branch to the effective address if the result of the last instruction tested is less than or equal to zero according to the states of Sign indicator KS and Unequal indicator KU. If the tested result is greater than zero, program execution continues with the next instruction in sequence.

The Branch Less Than instruction causes the program to branch to the effective address if the result of the last instruction tested is less than zero according to the states of Sign indicator KS and Unequal indicator KU. If the tested result is equal to or greater than zero, program execution continues with the next instruction in sequence.

The Branch Unequal instruction causes the program to branch to the effective address if the result of the last instruction tested is not equal to zero according to the states of Sign indicator KS and Unequal indicator KU. If the tested result is equal to zero, program execution continues with the next instruction in sequence.

The Branch No Overflow instruction causes the program to branch to the effective address if the last instruction tested does not cause an overflow condition (Overflow indicator KO reset). If the tested result caused an overflow condition (KO set), program execution continues with the next instruction in sequence.

The Branch Unconditional instruction causes the program to branch to the effective address.

3.85 Branch and Link Instruction (BLK) (See table 3-14)

The Branch and Link instruction enters the next instruction address into the memory roll table at the location specified by the address in the roll-arrow register. The address in the roll-arrow register is increased by one, and the program branches to the effective address. Bit 0 of the roll table location in which the return link address is stored is set to 1 to indicate that a Branch and Link instruction has been performed. The link address stored by the Branch and Link instruction permits the program to return to the next instruction when a Branch Back instruction is executed.

3.86 Branch and Put Instruction (BPT) (See table 3-15)

The Branch and Put instruction stores current program status information and a return link address in the memory roll table at the locations specified by roll-arrow register, and then branches to the effective address. The program status stored by this instruction includes the states of indicators KO, KS, and KU; the contents of the page register, and the contents of the index register. The return link address is the address of the next instruction in sequence following the Branch and Put instruction.

3.87 Branch Index Instruction (BXB) (See table 3-16)

The Branch Index instruction causes the program to branch to the effective address if the contents of the index register are not equal to zero. If the contents are equal to zero, program execution continues with the next instruction in sequence. If the value of the index register contents is negative, the value is increased by one; if the value is positive or equal to zero, it is decreased by one.

Table 3-14. Branch and Link Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate BLK	BLK = IRST IM2 I5
			IRST = NY4 IH3
			IH3 = I1 I2
			IM2 = I3 I4	

		Generate IBSV	IBSV = BLK + . . .	
		Generate IBLPK	IBLPK = BLK + . . .	
		L → Adder	LTE = IBSV X20 + . . .	Transfer effective address to adder
		$S \nrightarrow D$	STD = IBSV X20 + . . .	Load effective address into D-register
		Generate CELL7	CELL7 = IBLPK X2 + . . .	
		1's → ADS16, 15, 14	ADS16, ADS15, ADS14 = CELL7	Force address X'0007' onto address lines
	X21	M → Adder	MTF = IBSV X21 + . . .	Contents of roll-arrow register
		$S \nrightarrow L$	STL = IBLPK X21 + . . .	Load contents of roll-arrow register into L-register
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 follows X21 normally if no X-time is gated
X3 NY4	X30	N → Adder	NTE = IBSV X30 + . . .	Next instruction address transferred to adder
		L → ADS	LADS = IBLPK X3 + . . .	Address from roll-arrow register gated onto address lines
		1 → C0	C0 = BLK + . . .	Indicates a Branch and Link instruction was performed

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Data transferred to output data bus

		Generate MGO	MGO = Z1 $\overline{\text{YIDL}}$ + . . .	Start memory cycle
		Generate WM	WM = IBSV X3 + . . .	Next instruction address stored in memory location specified by roll-arrow register

Table 3-14. Branch and Link Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X3 NY4	X31	L ——→ Adder	LTE = IBSV X31 + . . .	Address from roll-arrow register transferred to adder
		1 —/→ K17	K17 = IBSV X31 + . . .	Force a 1 into K17 to add to address
		S —/→ L	STL = LTE $\overline{\text{YIDL}}$ + . . .	Return new address to L-register
		Generate X70	X70 = BLK X31 + . . .	
X7 NY4	X70	Generate CELL7	CELL7 = IBLPK X7 + . . .	
		1's ——→ ADS16, 15, 14	ADS16, ADS15, ADS14 = CELL7	Force address X'0007' onto address lines
		L ——→ Adder	LTE = IBLPK X70 + . . .	Address where next instruction address is stored transferred to adder

		E ——→ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Data transferred to output data bus

		Generate MGO	MGO = Z1 $\overline{\text{YIDL}}$ + . . .	Start memory cycle
		Generate WM	WM = IBLPK X7 + . . .	Write address into memory location X'0007'
	X71	D ——→ Adder	DTE = IBSV X71 + . . .	Effective address transferred to adder
		S —/→ L	STL = IBSV X71 + . . .	Address transferred to L-register for branch
		Generate ENDI	ENDI = NY4 X71 + . . .	End instruction execution
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$ + . . .	

Table 3-15. Branch and Put Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate BPT	BPT = IRST IM3 I5
			IRST = NY4 IH3
			IH3 = I1 I2
			IM3 = I3 I4	

		Generate IBSV	IBSV = BPT + . . .	
		Generate IBLPK	IBLPK = BPT + . . .	
		L → Adder	LTE = IBSV X20 + . . .	Transfer effective address to adder
		S $\not\rightarrow$ D	STD = IBSV X20 + . . .	Load effective address into D-register
		Generate CELL7	CELL7 = IBLPK X2 + . . .	
		1's → ADS16, 15, 14	ADS16, ADS15, ADS14 = CELL7	Force address X'0007' onto address lines
	X21	M → Adder	MTF = IBSV X21 + . . .	Contents of roll-arrow register transferred to adder
		S $\not\rightarrow$ L	STL = IBLPK X21 + . . .	Address transferred to L-register
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 follows X21 normally when no X-time is gated
X3 NY4	X30	L → ADS	LADS = IBLPK X3 + . . .	Address of roll table gated onto address lines
		N → Adder	NTE = IBSV X30 + . . .	Next instruction address

		E → C	C1 = E1 + F1
			: :
			C16 = E16 + F16
				Address transferred to output data bus

		Generate MGO	MGO = X1 $\overline{\text{YIDL}}$ + . . .	Start memory cycle
		Generate WM	WM = IBSV X3 + . . .	Write address into location specified by roll-arrow register
	X31	L → Adder	LTE = IBSV X31 + . . .	Increase roll table address by 1
		1 $\not\rightarrow$ K17	K17 = IBSV X31 + . . .	

Table 3-15. Branch and Put Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X3 NY4	X31	Generate X40	X40 = X31 $\overline{SKPX}$ + . . .	X40 follows X31 normally if no X-time is gated
X4 NY4	X40	KO, KS, KU → Adder	KTE = IBSV X40 + . . .	KO → E0, KS → E1, KU → E2
		P → Adder	PTF = KTE + . . .	P1-P5 → F3-F8

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16	

		Generate MGO	MGO = Z1 $\overline{YIDL}$ + . . .	Start memory cycle
		Generate WM	WM = IBLPK X4 + . . .	Write data into roll table
	X41	L → Adder	LTE = IBLPK X41 + . . .	Increase roll table address by 1
		1 → K17	K17 = IBSV X41 + . . .	
		S $\nrightarrow$ L	STL = LTE $\overline{YIDL}$ + . . .	Load new address into L-register
		Generate X50	X50 = X41 $\overline{SKPX}$ + . . .	X50 follows X41 normally if no X-time is gated
X5 NY4	X50	Inhibit L → ADS	No gating terms true	Force address X'0000' onto address lines
	X51	M → Adder	MTF = IBSV X51 + . . .	Contents of index register
		S $\nrightarrow$ N	STN = IBSV X51 + . . .	Load contents of index register into N-register
		Generate X60	X60 = X51 $\overline{SKPX}$ + . . .	X60 follows X51 normally if no X-time is gated
X6 NY4	X60	N → Adder	NTE = IBSV X60 + . . .	Contents of index register gated to adder and transferred to output data bus

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16	

		L → ADS	LADS = IBLPK X6 + . . .	Next roll table address
		Generate MGO	MGO = X1 $\overline{YIDL}$ + . . .	Start memory cycle
		Generate WM	WM = IBSV X6 + . . .	Write data into roll table

Table 3-15. Branch and Put Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X61	L → Adder	LTE = IBSV X61 + . . .	Roll table address increased by 1
		1 → K17	K17 = IBSV X61 + . . .	
		S –/→ L	STL = LTE $\overline{\text{YIDL}}$ + . . .	New address loaded into L-register
		Generate X70	X70 = X61 $\overline{\text{SKPX}}$ + . . .	X70 follows X61 normally if no X-time is gated
X7 NY4	X70	Generate CELL7	CELL7 = IBLPK X7 + . . .	
		1 → ADS16, 15, 14	ADS16, ADS15, ADS14 = CELL7	Force address X'0007' onto address lines
		L → Adder	LTE = IBLPK X70 + . . .	Address of last location in roll table used transferred through adder onto output data bus

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16	

		Generate MGO	MGO = X1 $\overline{\text{YIDL}}$ + . . .	Start memory cycle
		Generate WM	WM = IBLPK X7 + . . .	Store last roll table address in roll-arrow register
	X71	D → Adder	DTE = IBSV X71 + . . .	Effective address transferred through adder to L-register
		S –/→ L	STL = IBSV X71 + . . .	
		Generate ENDI	ENDI = NY4 X71 + . . .	End instruction execution
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$ + . . .	

Table 3-16. Branch Index Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate BXB	BXB = IRST IM2 $\overline{\text{I5}}$
			IRST = NY4 IH3
			IH3 = I1 I2
			IM2 = I3 $\overline{\text{I4}}$	

		Generate IMSC	IMSC = IBX + . . .
			IBX = IRST IM2 $\overline{\text{I5}}$	

		Inhibit L ——→ ADS	No gating term true	Force location X'0000' on memory address lines
		M $\not\longrightarrow$ D	MTD = IMSC X20 + . . .	Contents of memory location X'0000' (index register) clocked into D-register
	X21	D ——→ Adder	DTE = IBX + . . .	Contents of D-register gated to adder
		If M1 is a 1, set K17	K17 = BXB M1 X21 + . . .	If M1 is a 1 indicating value is negative, increase value by 1 by forcing K17 true and adding to contents
		If M1 is a 0, generate INVF	INVF = BXB $\overline{\text{M1}}$ X21 + . . .	If M1 is a 0 indicating value is positive force all 1's into F-input bus by inverting F and adding to E. This decreases index value by 1
		S $\not\longrightarrow$ D	STD = IBX X21 + . . .	Load new value into D-register
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 follows X21 normally if no X-time is gated
X3 NY4	X30	Inhibit L ——→ ADS	No gating term true	Forces memory address X'0000' onto address lines
		D ——→ Adder	DTE = IBX	New index value gated into adder

		E ——→ C	C1 = E1 + F1
			: :
			C16 = E16 + F16
				Index value transferred to output data bus

		Generate MGO	MGO = $\overline{\text{YIDL}}$ Z1 $\overline{\text{YEND}}$ + . . .	Start memory cycle
		Generate WM	WM = IMSC X31 $\overline{\text{INHWM}}$	Write index value into memory location X'0000'

Table 3-16. Branch Index Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X3 NY4	X30	Index value —/→ IX	CLKX = ($\overline{ADS1}$ thru $\overline{ADS16}$) MGO	Load new index value into index register
	X31	D ——→ Adder	DTE = IBX + . . .	Contents of D-register gated into adder
		If S1-S16 is not all 0's: inhibit N —/→ L	NTL = IBX X31 SZRO + . . .	If index value is not all 0's, inhibit next instruction address from being used, and next instruction will be from effective address location contained in L-register
		If S1-S16 is all 0's: N —/→ L		If index value is all 0's, transfer next instruction address to L-register
		Generate ENDI	ENDI = IBX X31 + . . .	End instruction execution
		Generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	

As memory location X'0000' contains the same data as the index register, during even clock X20, memory location X'0000' is addressed by inhibiting signal LADS from gating the effective address from the L-register onto the address lines. The data from location X'0000' is then clocked into the D-register and during odd clock X21 is transferred to the E-input bus of the adder. If the value is negative, a one is forced into flip-flop K17 and is added to the index register contents. If the value is positive, signal INVF inverts the contents of the F-input bus. As there is no data on the F-input bus, it becomes all ones and is added to the index register contents, thereby decreasing its contents by one. The result is loaded into the D-register.

The new index register value is written into memory location X'0000' by inhibiting LADS and forcing the address to all zeros and generating write memory signal WM. The new value is also clocked into the index register.

The contents of the D-register are transferred through the adder to the sum bus for decoding. If S1 through S16 are not all zeros, signal NTL is inhibited and the program branches to the effective address contained in the L-register. If bits S1 through S16 are all zeros, signal NTL is generated and the next instruction address is transferred to the L-register.

3.88 Compare Instruction (CMB) (See table 3-17)

The compare instruction subtracts the contents of the effective address from the contents of the B-register without affecting the contents of the B-register. Overflow indicator KO is set if the difference exceeds the capacity of the B-register. Sign indicator KS and Unequal indicator KU are set or reset to indicate the result of the comparison.

During odd clock X61, the contents of the effective address are transferred from memory onto the F-input bus of the adder, and the contents of the B-register are transferred to the E-input bus. Signal INVF is generated to invert the data on the F-input bus, and a one is forced into flip-flop K17 to form the 2's complement of the F-input data. The two operands are added, and the result is tested on the sum bus. If an overflow condition exists as indicated by flip-flops K1 or K2 being true, Overflow indicator flip-flop KO is set. If flip-flop KO is true, flip-flop KU is set indicating that the number in the B-register is larger than the contents of the effective address. If bit S1 is true and there was no overflow or S1 is a zero but there was overflow, flip-flop KS is set. If flip-flops KS and KU are both set, the magnitude of the number in the B-register was smaller than the contents of the effective address.

3.89 Divide Instruction (DVB) (See tables 3-18 and 3-19)

Two divide instructions are available for use on the RC 70 Computer: a slow divide instruction which is the standard instruction on the basic computer and a fast divide instruction which is optional. Both instructions use nonrestoring division.

Table 3-17. Compare Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description
		Generate X60	X60 = YOAS $\overline{\text{I1}}$ I3 $\overline{\text{I4}}$ + . . .	

X6
NY4
	X60	Generate ICM	ICM = IATH IH1 IM2
			IATH = NY4 $\overline{\text{I1}}$ $\overline{\text{IM1}}$
			IM1 = $\overline{\text{I3}}$ I4
			IH1 = $\overline{\text{I1}}$ I2
			IM2 = I3 $\overline{\text{I4}}$	

	X61	L → ADS	LADS = IATH X6 + . . .	Effective address
		M → Adder	MTF = IATH $\overline{\text{IST}}$ X61 + . . .	Contents of effective address transferred to adder
		B → Adder	BTE = IATH ILD X61 + . . .	Contents of B-register transferred to adder
		Generate ISUB	ISUB = IATH IH1 $\overline{\text{I4}}$	

		Generate INVF;
		1 $\nrightarrow$ K17
			INVF = ISUB X61 + . . .
			K17 = NEGF + . . .
			NEGF = INVF + . . .
				Invert contents on F-input bus and add K17 to form the 2's complement. Add data from E- and F-input buses

		If overflow, set KO	KO = IATH BINOVF X61 + . . .
			BINOVF = K1 ⊕ K2
				Difference exceeds capacity of B-register

		If B > M, set KU	KU = IATH $\overline{\text{ILD}}$ KO + . . .	Number in B-register is larger than contents of effective address

		If B < M, set KS and KU	KS = IATH BINSIN $\overline{\text{IST}}$ X61
			BINSIN = S1 $\overline{\text{BINOVF}}$
			+ $\overline{\text{S1}}$ BINOVF
			KU = IATH $\overline{\text{IST}}$ X61 $\overline{\text{SZRO}}$
				Magnitude of number in B-register is smaller than contents of effective address

		Generate ENDI	ENDI = IATH X61 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

Table 3-18. Divide Instruction (Slow), Sequence Chart

Time		Functions Performed	Signals Involved	Description
		Generate X10	X10 = YOAS $\overline{GX20}$ $\overline{GX60}$	

X1
NY4
	X10	Generate DVA	DVA = NY4 IH1 IM1
			IH1 = $\overline{I1}$ I2
			IM1 = $\overline{I3}$ I4	

		Generate IATMD	IATMD = NY4 $\overline{I1}$	
		Generate DVB	DVB = NY4 IH1 IM1	
		Generate IMDB	IMDB = DVB	

		B → Adder;
		S ⇸ D
			BTE = IMDB X10 + . . .
			STD = IMDB X10 + . . .
				Transfer contents of B-register through the adder and load into D-register

	X11	L → ADS	LADS = IMDB X1 + . . .	Transfer effective address to address lines
		M → Adder	MTF = IMDB X11 + . . .	Read divisor from effective address

		If M1 is a 1, generate NEGF; 1 ⇸ K17	NEGF = IMDB M1 X11 + . . .
			K17 = NEGF + . . .
				If divisor is a negative quantity, invert the contents of F-input bus and add K17 to form 2's complement

		S ⇸ B	STB = IMDB X11 + . . .	Load divisor into B-register
		If M1 is a 1, set KB	KB = IMDB M1 X11 + . . .	Remember sign of original divisor
		Generate X20	X20 = X11 $\overline{SKPX}$ + . . .	X20 follows X11 normally if no X-time is gated
X2 NY4	X20	B → Adder	BTE = DVB X20 + . . .	Transfer divisor to adder

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Transfer data onto output data lines

		Inhibit LADS	No gating term true	

		1 → ADS16, ADS13	ADS16 = IMDB $\overline{X1}$ $\overline{X3}$ $\overline{X7}$ + . . .
			ADS13 = IMDB $\overline{X1}$ + . . .
				Force address X'0009' onto address lines

		Set MGO	MGO = $\overline{YIDL}$ Z1 XODD + . . .	Start memory cycle
		Generate WM	WM = IMDB X2 + . . .	Write divisor in memory location X'0009'

Table 3-18. Divide Instruction (Slow), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X2 NY4	X21	Generate INVF	INVF = IMDB X21 + . . .	Force all 1's onto F-input bus of adder
		S $\not\rightarrow$ IL	STIL = IMDB X21 + . . .	Load 1's into lower half of I-register for a count of 15
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 follows X21 normally if no X-time is gated
X3 NY4	X30	If D1 is a 1, set KS	KS = IMDB D1 X30 + . . .	Dividend is a negative number
	X31	Inhibit LADS	No gating term true	
		1 ⟶ ADS13	ADS13 = IMDB $\overline{\text{X1}}$ + . . .	Force address X'0008' onto address lines (upper accumulator)

		M ⟶ Adder;
		S $\not\rightarrow$ B
			MTF = DVB X31 + . . .
			STB = DVB X31 + . . .
				Read most significant half of dividend from upper accumulator and load into B-register

		Generate X40	X40 = X31 $\overline{\text{SKPX}}$ + . . .	X40 follows X31 normally if no X-time is gated
X4 NY4	X40	B ⟶ Adder	BTE = DVB X40 + . . .	MSH of dividend transferred to adder

		Generate LS1	LS1 = DVB X50 + . . .
			E1A = E2 LS1 + . . .
			: :
			E16A = H1 LS1 + . . .
			H1 = 0
				Shift MSH left 1 bit position into EA-bus

		S $\not\rightarrow$ B	STB DVB BTE + . . .	Shifted MSH of dividend loaded into B-register
		If KB was set during X11, generate KSTME	KSTME = IMDB KB X40 + . . .	
		If KS was not set during X30 and KB is true, set KS. If KS was set during X30 and KB is true, reset KS	KS = KSTME $\overline{\text{KS}}$ + . . . $\overline{\text{KS}}$ = KSTME KS + . . .	Perform KS ⊕ KB to determine sign of quotient

		KS $\not\rightarrow$ KA, KC	KA = DVB KS X40 + . . .
			KC = DVB KS X40 + . . .
				KA and KC remember state of dividend

	X41	Inhibit LADS	No gating terms true	

Table 3-18. Divide Instruction (Slow), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X4
NY4
	X41	1 ⟶ ADS16, ADS13	ADS16 = IMDB $\overline{X1}$ $\overline{X3}$ $\overline{X7}$ + . . .
			ADS13 = IMDB $\overline{X1}$ + . . .
				Force address X'0009' onto address lines

		M ⟶ Adder	MTF = DVB X41 + . . .	Read divisor from location X'0009'
		D ⟶ Adder	DTE = DVB X41 + . . .	Transfer dividend to adder

		If KA is true, add M and D. If $\overline{KA}$ is true, generate NEGF and set K17	NEGF = DVB $\overline{KA}$ X41 + . . .
			K17 = NEGF + . . .
				If KA is set, bit in dividend was negative and a simple addition is performed; if $\overline{KA}$ is true, both dividend and divisor were negative or both were positive and a subtraction is performed (2's complement addition)

		S $\nrightarrow$ D	STD = DVB DTE + . . .	Load dividend and partial quotient into D-register
		If KA and S1 are both true, or $\overline{KA}$ and $\overline{S1}$ are true, set KO	KO = DVB S1 KA X41 + DVB $\overline{S1}$ $\overline{KA}$ X41 + . . .	Divisor was equal to or smaller than dividend
		Set X50	X50 = X41 $\overline{SKPX}$ + . . .	X50 follows X41 normally if no X-time is gated
X5 NY4	X50	B ⟶ Adder	BTE = IMDB X50 + . . .	MSH of dividend transferred to adder
		Generate LS1	LS1 = DVB X50 + . . .	Left shift dividend 1 bit position with H1 ⟶ E16A

		KA ⊕ KB ⇒ 1 ⟶ H1-H4	H1-H4 = HRST + . . .
			HRST = DVB $\overline{KA}$ KB X50
			+ DVB KA $\overline{KB}$ X50 + . . .
				If sign of dividend and sign of divisor are not both negative or both positive, force 1's into H1-H4. This is sign bit of quotient

		S $\nrightarrow$ B	STB = DVB BTE + . . .	Return shifted dividend and partial quotient to B-register
		B1 $\nrightarrow$ T1	T1 = LSHFT C1	Load sign bit of residue into T1
	X51	Inhibit LADS	No gating terms true	

		1 ⟶ ADS16, ADS13	ADS16 = IMDB $\overline{X1}$ $\overline{X3}$ $\overline{X7}$ + . . .
			ADS13 = IMDB $\overline{X1}$ + . . .
				Force address X'0009' onto address lines

		M ⟶ Adder	MTF = DVB X51 + . . .	Read divisor from location X'0009'

Table 3-18. Divide Instruction (Slow), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X5 NY4	X51	D → Adder	DTE = IMDB X51 + . . .	Transfer least significant half of dividend to adder

		If KA is true, generate NEGF and set K17	NEGF = DVB KA X51 + . . .
			K17 = NEGF + . . .
				If sign of residue was positive, subtract divisor from dividend (2's complement addition); if sign was negative, perform simple addition

		Generate CTI	CTI = IMDB X51 + . . .
			CTIB = CTI (I15 + I16)
			CTID = CTI $\overline{I15}\ \overline{I16}$ (I13 + I14)
				Count I-register down by 1 count

		If I-register is counted down to zero, generate IDZRO	IDZRO = $\overline{I13}\ \overline{I14}\ \overline{I15}\ \overline{I16}$	

		If IDZRO, generate LS1 and SHIFT	LS1 = $\overline{IDZRO}$ DVB X51 + . . .
			SHIFT = LS1 + . . .
				Left shift residue 1 bit place with H1 → E16A

		T1 → H1	H1 = T1 + . . .
			E1A = LS1 E2 + . . .
			: :
			E16A = LS1 H1 + . . .
				Shift next bit of quotient into T1

		S $\nrightarrow$ D	STD = IMDB $\overline{IDZRO}$ X51 + IMDB $\overline{KAQKC}$ X51 + . . .	Return result to D-register if shift count does not equal zero or KA and KB are both ones or both zeros
		If $\overline{S1}$, set KA	KA = DVB $\overline{S1}$ X51 + . . .	Sign bit of residue is positive
		If $\overline{IDZRO}$, generate X50	X50 = IMDB $\overline{IDZRO}$ X51 + . . .	If shift count is not equal to zero, return to X50 and cycle through X50 and X51 until count equals zero
		Generate X60	X60 = X51 $\overline{SKPX}$ + . . .	After shift count reaches zero, generate X60
X6 NY4	X60	B → Adder	BTE = IMDB X60 + . . .	Transfer quotient to adder

		Generate DVBIT if either KS or KC is a one and set K17	DVBIT = KS $\overline{KC}$ + $\overline{KS}$ KC
			K17 = DVB DVBIT X60 + . . .
				Signs of divisor and dividend are not the same. If signs of dividend and divisor are not the same, the quotient must be corrected by adding one to it

		S $\nrightarrow$ B	STB = DVB BTE + . . .	Return corrected quotient to B-register

Table 3-18. Divide Instruction (Slow), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X60	If sum bus is not equal to zero, set KU	KU = $\overline{\text{SZRO}}$ KUTME + . . . $\overline{\text{KUTME}}$ = IMDB X60 + . . . $\overline{\text{SZRO}}$ = S1 + S2 + . . . S16	
	X61	Generate X70	X70 = X61 $\overline{\text{SKPX}}$ + . . .	X70 follows X61 normally if no X-time is gated
X7 NY4	X70	D → Adder	DTE = DVB $\overline{\text{KC}}$ X70 + . . .	Transfer remainder to adder if dividend was positive

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Transfer remainder data word to output data bus

		Inhibit LADS	No gating terms true	
		1 → ADS13	ADS13 = IMDB $\overline{\text{X1}}$ + . . .	Force address X'0008' onto address lines
		Set MGO	MGO = $\overline{\text{YIDL}}$ Z1 XODD + . . .	Start memory cycle
		Generate WM	WM = IMDB X7 + . . .	Write remainder into memory location X'0008'
	X71	Generate ENDI	ENDI = NY4 X71 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = $\overline{\text{YBGN}}$ ENDI + . . .	

Table 3-19. Divide Instruction (Fast), Sequence Chart

Time		Functions Performed	Signals Involved	Description
		Generate X10	X10 = YOAS $\overline{GX20}$ $\overline{GX60}$	

X1
NY4
	X10	Generate DVA	DVA = NY4 IH1 IM1
			IH1 = $\overline{I1}$ I2
			IM1 = $\overline{I3}$ I4	

		Generate DVB	DVB = NY4 IH1 IM1	
		Generate IATMD	IATMD = NY4 $\overline{I1}$	
		Generate IMDB	IMDB = DVB	
		Generate DVSF	DVSF = IMDB + DVA + . . .	
		Generate IMDS	IMDS = IMDB + . . .	

		B $\longrightarrow$ Adder;
		S $\nrightarrow$ D
			BTF = DVA X10 + . . .
			STD = DVB X10 + . . .
				LSH of dividend transferred from B-register through the adder to the D-register

		Reset KO, KS, and KU	$\overline{KO}$, $\overline{KS}$, $\overline{KU}$ = RSUSO RSUSO = DVSF + . . .	
	X11	L $\longrightarrow$ ADS	LADS = IMDB X1 + . . .	Effective address transferred to address lines
		M $\longrightarrow$ Adder	MTF = IMDS X11 + . . .	Contents of effective address read

		If $\overline{M1}$, S $\nrightarrow$ B; if M1, generate NEGF and K17	NEGF = DVB M1 X11 + . . .
			K17 = NEGF + . . .
			STB = DVB X11 + . . .
				If divisor is negative, form 2's complement and load into B-register

		M1 $\nrightarrow$ KB	KB = DVB M1 X11 + . . .	Store original sign of divisor
		Generate X30	X30 = DVA X11 + . . .	
X3 NY4	X30	Generate INVF	INVF = DVA X30 + . . .	Force all 1's into F-adder input bus
		S $\nrightarrow$ I13-I16	STIL = DVA X30 + . . .	Load count of 15 into I-register
		If D1, set KS	KS = DVB D1 X30 + . . .	Dividend is negative quantity
	X31	Inhibit LADS	No gating terms true	
		1 $\longrightarrow$ ADS13	ADS13 = IMDB $\overline{X1}$ + . . .	Force address X'0008' onto address lines

		M $\longrightarrow$ F;
		S $\nrightarrow$ G
			MTF = DVA X31 + . . .
			STG = DVSF X31 + . . .
				Read MSH of dividend from UBA (X'0008') and load into G-register

Table 3-19. Divide Instruction (Fast), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X3 NY4	X31	Generate X40	X40 = X31 $\overline{\text{SKPX}}$ + . . .	X40 follows X31 normally if no X-time is gated

X4
NY4
	X40	KS $\not\longrightarrow$ KA, KC	KA = DVB KS X40 + . . .
			KC = DVB KS X40 + . . .
				Store sign of dividend in KA and KC

		If KS was not set previously and KB is a one, set KS	KS = DVA KB X40 + . . . $\overline{\text{KS}}$ = DVA KB X40 KS + . . .	Perform KS ⊕ KB to determine sign of quotient
		Generate LSG1	LSG1 = DVA X40 + . . .	Shift MSH of dividend in G-register left one bit position

	X41	If $\overline{\text{KA}}$, subtract B from D and load into D; if KA, add D and B and load into D; S $\not\longrightarrow$ D	DTE = DVA X41 + . . .
			BTF = DVA X41 + . . .
			NEGF = DVB $\overline{\text{KA}}$ X41 + . . .
			K17 = NEGF + . . .
			STD = DVB X41 + . . .
				LSH of dividend transferred to adder; divisor transferred to adder. If dividend is positive, form 2's complement and add, otherwise perform simple addition and transfer result to D-register

		If S1 KA or $\overline{\text{S1}}$ $\overline{\text{KA}}$, set KO	KO = DVB S1 KA X41 + DVB $\overline{\text{S1}}$ $\overline{\text{KA}}$ X41 + DVB SZRO X41 + . . .	Set KO if KA and SL are the same or if the sum bus is equal to zero
		Generate X50	X50 = X41 $\overline{\text{SKPX}}$ + . . .	X50 follows X41 normally if no X-time is gated

X5
NY4
	X50	D ⟶ Adder	DTE = DVSF XX5 + . . .
			XX5 = IMDS X5
				Transfer LSH of dividend to adder

		B ⟶ Adder	BTF = DVA XX5 + . . .	Transfer divisor to adder

		Generate LS1, LSG1, SHIFT, and LSHFT; G1 ⟶ H1	LS1 = DVSF X50 + . . .
			LSG1 = DVA XX5 + . . .
			SHIFT = LS1 + . . .
			LSHFT = LS1 + . . .
			H1 = DVSF XX5 G1 + . . .
				Left shift LSH of dividend 1 bit place into EA bus with H1 ⟶ E16A. Left shift MSH of dividend 1 bit place with G1 ⟶ H1

		If KA is true and shift count does not equal zero, generate NEGF and set K17; S $\not\longrightarrow$ D	NEGF = DVA KA XX5 + . . .
			STD = DVB $\overline{\text{MDEND}}$ XX5
			+ DVB $\overline{\text{KAQKC}}$ XX5 + . . .
				If sign bit of residue is positive, subtract divisor from residue (add 2's complement); if sign bit is negative, perform a simple addition. Load result into D-register if $\overline{\text{MDEND}}$ is true. When MDEND is true, load result into D-register if KA KC + $\overline{\text{KA}}$ $\overline{\text{KC}}$

		If S1 is true, set KA	KA = DVB $\overline{\text{S1}}$ X5 + . . .	Sign of residue is positive

Table 3-19. Divide Instruction (Fast), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X5
NY4
	X50	Generate CTI	CTI = IMDS X5 + . . .
			CTIB = CTI (I15 + I16)
			CTID = CTI $\overline{\text{I15}}$ $\overline{\text{I16}}$ (I13 + I14)
				Count I-register down by one count

	X51	D → Adder	DTE = DVSF XX5 + . . .	Transfer LSH of dividend to adder
		B → Adder	BTF = DVA XX5 + . . .	Transfer divisor to adder

		If $\overline{\text{MDEND}}$, generate LS1, LSG1, SHIFT, and LSHFT; G1 → H1	LS1 = DVSF $\overline{\text{MDEND}}$ X51 + . . .
			LSG1 = DVA XX5 + . . .
			SHIFT = LS1 + . . .
			LSHFT = LS1 + . . .
			H1 = DVSF XX5 G1 + . . .
				If shift count does not equal zero, left shift LSH of dividend one bit place into EA-bus with H1 → E16A. Left shift MSH of dividend one bit place with G1 → H1

		If KA is true and shift count does not equal zero, generate NEGF and set K17	NEGF = DVA KA XX5 + . . .
			K17 = NEGF + . . .
				If sign bit of residue is positive, subtract divisor from dividend (add 2's complement); if sign bit is negative, add divisor and dividend

		If $\overline{\text{S1}}$ is true, set KA	KA = DVB $\overline{\text{S1}}$ X5 + . . .	Sign bit of residue is positive

		Generate CTI	CTI = IMDS X5 + . . .
			CTIB = CTI (I15 + I16)
			CTID = CTI $\overline{\text{I15}}$ $\overline{\text{I16}}$ (I13 + I14)
				Count I-register down by one count

		S ↛ D	STD = DVB $\overline{\text{MDEND}}$ XX5 + . . . + DVB $\overline{\text{KAQKC}}$ XX5 + . . .	Load remainder into D-register if shift count does not equal zero ($\overline{\text{MDEND}}$). When shift count does equal zero, load result into D-register if KA KC or $\overline{\text{KA}}$ $\overline{\text{KC}}$
		If shift count does not equal zero, generate X50	X50 = IMDB $\overline{\text{MDEND}}$ X51 + . . .	If shift count does not equal zero, return to X50 and cycle through X50 and X51 until count reaches zero
		When shift count reaches zero, generate MDEND	MDEND = $\overline{\text{I13}}$ $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$	
		Generate X60	X60 = X51 $\overline{\text{SKPX}}$ + . . .	X60 follows X51 normally if no X-time is gated
X6 NY4	X60	D → Adder	DTE = DVSF X60 + . . .	Remainder transferred to adder

		E → C	C1 = E1 + F1
			. .
			. .
			C16 = E16 + F16
				Transfer remainder word to output data lines

Table 3-19. Divide Instruction (Fast), Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X60	Inhibit LADS	No gating terms true	
		1 → ADS13	ADS13 = IMDB $\overline{X1}$ + . . .	Force address X'0008' onto address lines
		Set MGO	MGO = $\overline{YIDL}$ Z1 XODD + . . .	Start memory cycle
		Generate WM	WM = IMDS X6 + . . .	Write remainder into memory location X'0008'
	X61	G → Adder	GTE = IMDS X61 + . . .	Quotient transferred to adder

		If KS KC or $\overline{KS}$ $\overline{KC}$, set K17	K17 = DVA DVBIT X61 + . . .
			DVBIT = KS KC + $\overline{KS}$ $\overline{KC}$
				Correct quotient by adding one to it

		$S \not\rightarrow B$	STB = IMDS X61 + . . .	Load quotient into B-register
		If S-bus does not equal zero, set KU	KU = $\overline{SZRO}$ IMDS X61 + . . . $\overline{SZRO}$ = S1 + S2 + . . . S16	Contents of sum bus do not equal zero
		Generate ENDI	ENDI = IMDS X61 + . . .	End instruction execution
		$N \not\rightarrow L$	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = $\overline{YBGN}$ ENDI + . . .	

The two types of division are restoring and nonrestoring. Restoring division is the most basic type and is the type used in a normal long division. Restoring division uses repeated subtractions to obtain a quotient.

Multiples of the divisor are subtracted from the dividend or partial dividend (called the residue). Each subtraction results in either a 1 or a 0 in the quotient, depending upon whether the residue is positive or negative. In restoring division a negative residue signifies that the divisor multiple is larger than the previous residue and cannot be contained in it. The previous positive residue is then restored.

Nonrestoring division is similar to restoring division in that repeated subtractions of multiples of the divisor are used but it differs from the restoring type in that the residue can be negative.

Successive subtractions of multiples of the divisor are performed. If the residue is negative, a 0 is placed in the quotient for that order and the next divisor multiple is added rather than subtracted from the residue. Each time the residue is positive, a 1 is added to the appropriate order of the quotient and the next divisor multiple is subtracted. Every time the residue is negative, a 0 is added to the appropriate order of the quotient and the next divisor multiple is added.

The RC 70 Computer uses the 2's complement method for division. The divisor multiple to be subtracted can then be added to the residue. The division process using 2's complement addition is the same as given above. Each time a 1 appears in the sign bit of the residue, a 0 is added to the appropriate order of the quotient and the next divisor multiple is added (remaining in uncomplemented form). Each time a 0 appears in the sign bit of the residue a 0 is added to the appropriate order of the quotient and the next divisor multiple is subtracted (2's complement form is added).

Nonrestoring division and complementing are used in all division cases. There are four different divisions in regard to sign – a positive divided by a positive, a positive divided by a negative, a negative divided by a positive, and a negative divided by a negative. If the signs of the dividend and divisor are the same (both negative or both positive), the sign of the quotient is positive; if the signs of the dividend and divisor are different, the sign of the quotient is negative.

If the dividend and divisor do not both have the same signs, the correct quotient must be obtained by adding one to the quotient obtained as a result of the division operation. When the signs of the dividend and the divisor are the same, the quotient is correct as obtained.

The following is an example of nonrestoring division using 2's complement addition:

Example: $434_{10} \div 11_{10} = 39_{10}$ R5_{10}

$$\begin{array}{lrrl}
\text{Sign bit} \longrightarrow & & \underline{010011} & \\
\text{Divisor} & 01011\,/ & 0110110010 & \\
 & & \underline{1010100000} & \\
\text{Residue} & & 0001010010 & = 1 \times 2^{5} \\
 & & \underline{1101010000} & \\
\text{Residue} & & 1110100010 & = 0 \times 2^{4} \\
 & & \underline{01011000} & \\
\text{Residue} & & 1111111010 & = 0 \times 2^{3} \\
 & & \underline{0101100} & \\
\text{Residue} & & 0000100110 & = 1 \times 2^{2} \\
 & & \underline{1111101010} & \\
\text{Residue} & & 0000010000 & = 1 \times 2^{1} \\
 & & \underline{1111110101} & \\
\text{Sign bit} \longrightarrow & & 0000000101 & = 1 \times 2^{0} \\
 & & \text{(Remainder)} &
\end{array}$$

Divide Execution. The Divide instructions use the technique of adding or subtracting and shifting to generate the final quotient and remainder. The remainder is transferred to memory location X'0008' during time X60 and the quotient is transferred to the B-register after correction if correction is necessary. Whether the divisor is to be added or subtracted to the residue during the iterations of time X50 depends on the result of the previous addition or subtraction. After each addition or subtraction the residue is left shifted one bit position. One quotient bit is generated for each operation and is transferred to B16. Whether the quotient bit is a 1 or a 0 for each operation depends on the relationship of the signs of the divisor and the original dividend. As the least significant half of the dividend is shifted out of the high end of the B-register, the quotient is transferred into the low end of the register.

Table 3-18 is the sequence chart for slow division and table 3-19 is the sequence chart for fast division.

The difference in the fast and slow operations consists of an additional register being available for fast division, thereby eliminating the requirement for a memory fetch operation each time the divisor is required.

3.90 End Left Instruction (ELB) (See table 3-20)

The End Left instruction shifts the contents of the B-register to the left the number of bit positions specified by the number-of-shifts field of the instruction (I13 through I16). As the contents are shifted, the bits shifted out of the sign bit position are recirculated and loaded back into the least significant bit position. Sign indicator KS is set if the sign of the shifted result is negative; Unequal indicator KU is set if the contents of the B-register are not equal to zero.

During even clock X60, bit positions I13 through I16 are decoded to determine if the first shift required is to be a one-bit position or a four-bit position shift. The contents of the B-register are transferred to the E-input bus of the adder, and the appropriate shift is performed to the EA intermediate bus. The contents of the E- or F-input buses are always present on output data bus C1 through C16. During the left shift operation, the contents of C1 through C4 (B1 through B4) are loaded into flip-flops T1 through T4. The remaining shifted data on the EA intermediate bus is not used during this clock. A one is forced into flip-flop KA for use during the next clock.

During odd clock X61, the contents of the B-register are transferred to the E-input bus of the adder and the same shift is performed as decoded during X60. However, the outputs of flip-flops T1 through T4 are transferred into H1 through H4, and H1 through H4 are shifted into the least significant bit positions of the EA intermediate bus to recirculate the bits shifted out of the sign bit position. The shifted data is then returned to the B-register as flip-flop KA being true allows a sum bus to B-register transfer. Flip-flop KA is then reset.

The count in the I-register is then counted down by the number of bit positions shifted. If the count in the I-register is not all zeros the computer cycles through X60 and X61 performing the required shifts until the count reaches zero and signal NOSFT is generated. When signal NOSFT is generated, shifting ceases. If the sign of the shifted result is negative, Sign indicator KS is set and if the contents of the B-register are not all zeros, Unequal indicator KU is set.

3.91 Double Length Shift Instruction (ELD) (See table 3-21)

The Double Length Shift instruction requires the use of the high speed multiply/divide feature. This instruction shifts the contents of the upper and lower accumulators to the left the number of bit positions specified by the number-of-shifts field of the instruction (bits I13 through I16). As the contents are shifted, the bits shifted out of the sign bit position of the upper accumulator are recirculated and loaded back into the least significant bit position of the lower accumulator. Sign indicator KS is set if the sign of the result left in the lower accumulator is negative and Unequal indicator KU is set if the result left in the lower accumulator is not equal to zero.

During even clock X20, the effective address from the L-register is inhibited from being gated onto the address lines. Instead, a one is forced onto address line ADS13 to generate address X'0008', the address of the upper

Table 3-20. End Left Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS I1 IM1 $\overline{I6}$ + . . .
			IM1 = $\overline{I3}$ I4	

X6
NY4
	X60	Generate ELB	ELB = ISFT ILL0
			ISFT = IRST IM1 $\overline{I5}$ $\overline{I6}$
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{I3}$ $\overline{I4}$
			ILL0 = $\overline{I7}$ $\overline{I8}$	

		Generate ISFL	ISFL = ISFT $\overline{I8}$	
		If $\overline{I13}$ $\overline{I14}$, generate DZRO	DZRO = $\overline{I13}$ $\overline{I14}$	
		If $\overline{I15}$ $\overline{I16}$, generate BZRO	BZRO = $\overline{I15}$ $\overline{I16}$	
		If BZRO, generate DSFT	DSFT = BZRO (I13 + I14)	Indicates 4-bit shift
		If DZRO, generate BSFT	BSFT = DZRO (I15 + I16)	Indicates 1-bit shift
		If DSFT, generate LS4	LS4 = ISFL DSFT SFTTME	
		If BSFT, generate LS1	LS1 = ISFL BSFT SFTTME	
		B → Adder	BTE = ISFT X6 + . . .	

		C1 $\nrightarrow$ T1,
		C2 $\nrightarrow$ T2,
		C3 $\nrightarrow$ T3,
		C4 $\nrightarrow$ T4
			T1 = LSHFT C1 + . . .
			T2 = LSHFT C2 + . . .
			T3 = LSHFT C3 + . . .
			T4 = LSHFT C4 + . . .
			LSHFT = LS1 + LS4	

		If DSFT, shift E four places left into EA		Insignificant as shift result is not used
		If BSFT, shift E one place left into EA		Insignificant as shift result is not used
		1 → KA	KA = ISFT $\overline{OKA}$ + . . .	Inhibits S $\nrightarrow$ B when $\overline{KA}$ is true
	X61	B → Adder	BTE = ISFT X6 + . . .	

		If DSFT, shift E four places left into EA with H → EA	E1A = E5 LS4 + . . .
			⋮ ⋮
			E13A = H1 LS4 + . . .
			E14A = H2 LS4 + . . .
			E15A = H3 LS4 + . . .
			E16A = H4 LS4 + . . .
			H1 = HRST T1
			⋮ ⋮
			H4 = HRST T4
			HRST = ISFR + . . .
				E shifted four places left with E1 shifted into E13A by means of H1

Table 3-20. End Left Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X6
NY4
	X61	If BSFT, shift E one place left into EA with $H \longrightarrow EA$	E1A = E2A LS1 + . . .
			: :
			E16A = H1 LS1 + . . .
				E shifted one place left with E1 shifted into E16A by means of H1

		$S \not\longrightarrow B$	STB = ELB KA X6 + . . .	Shifted data loaded into B-register
		Reset KA	$\overline{KA}$ = ELB XODD + . . .	Inhibits $S \not\longrightarrow B$ during next clock period
		Count I-register	CTI = ELB KA X6 + . . .	
		If DSFT, generate CTID	CTID = CTI DSFT + . . .	Count I-register down by four counts
		If BSFT, generate CTIB	CTIB = CTI BSFT + . . .	Count I-register down by one count
		If I-register is not all 0's, return to X60	X60 = ISFT $\overline{NOSFT}$ X61 + . . .	Cycle through X60 and X61 until shift count reaches 0

		If I-register is all 0's, generate NOSFT	NOSFT = ISFT IDZRO WZRO
			IDZRO = $\overline{I13}$ $\overline{I14}$ $\overline{I15}$ $\overline{I16}$
			WZRO = $\overline{I10}$ $\overline{I11}$ $\overline{I12}$
				Shift count is all 0's

		If NOSFT, generate ENDI	ENDI = NOSFT X61 + . . .	End instruction execution
		$N \not\longrightarrow L$	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{YBGN}$	

Table 3-21. Double Length Shift Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate ELD	ELD = IXLH $\overline{\text{I6}}$ I7
			IXLH = IXLH IM0 I5
			IRST = NY4 IH3
			IH3 = I1 I2
			IM0 = $\overline{\text{I3}}$ $\overline{\text{I4}}$	

		Generate DVSF	DVSF = ELD + . . .	
		Generate IMSC	IMSC = IRST IM0	
		Generate IMDS	IMDS = ELD + . . .	
		Inhibit LADS	No gating term true	
		Force 1 → ADS13	ADS13 = ELD YC + . . .	Force address X'0008' onto address lines (upper accumulator)
		M ⇸ D	MTD = IMSC X20 + . . .	Contents of upper accumulator transferred to D-register

	X21	B → Adder;
		S ⇸ G
			BTF = ELD X21 + . . .
			STG = ELD X21 + . . .
				Contents of B-register transferred through adder to G-register

		Generate X50	X50 = ELD X21 + . . .	

X5
NY4
	X50	Generate LSG1	LSG1 = ELD $\overline{\text{MDEND}}$ XX5 + . . .
			MDEND = IDZRO
			IDZRO = $\overline{\text{I13}}$ $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$
			XX5 = X5 IMDS
			H1 = HRST + . . .
			HRST = DVSF G1 XX5 + . . .
				Shift count is in I13-I16. Shift G-register contents left one bit position with G1 → H1 and D1 ⇸ G17

		D → Adder	DTE = DVSF XX5 + . . .	Transfer contents of D-register to adder
		Generate LS1	LS1 = DVSF X50 + . . .	

		Left shift D one bit position	E16A = LS1 H1 + . . .
			: :
			E1A = LS1 E2 + . . .
				Left shift D one bit position

		S ⇸ D	STD = ELD $\overline{\text{MDEND}}$ XX5 + . . .	Transfer shifted data to D-register

Table 3-21. Double Length Shift Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X5
NY4
	X50	Count I-register	CTI = CTID + . . .
			CTID = IMDS X5
			CTIB = $\overline{\text{ISFT}}$ CTI (I13 + I14 + I15 + I16)
			CTID = CTI (I13 + I14) $\overline{\text{I15}}$ $\overline{\text{I16}}$
				Count I-register down by one count

	X51	Generate LSG1	LSG1 = ELD $\overline{\text{MDEND}}$ XX5 + . . .
			H1 = HRST + . . .
			HRST = DVSF G1 XX5 + . . .
				Shift G-register contents left one bit position with G1 → H1 and D1 $\not\rightarrow$ G17

		D → Adder	DTE = DVSF XX5 + . . .	Transfer contents of D-register to adder
		Generate LS1	LS1 = DVSF $\overline{\text{IDZRO}}$ X51 + . . .	

		Left shift D one bit position	E16A = LS1 H1 + . . .
			: :
			E1A = LS1 E2 + . . .
				Left shift contents of E-bus one bit position onto EA-bus

		S $\not\rightarrow$ D	STD = ELD $\overline{\text{MDEND}}$ XX5 + . . .	Transfer shifted data to D-register

		Count I-register	CTI = CTID + . . .
			CTID = IMDS X5
			CTIB = $\overline{\text{ISFT}}$ CTI (I13 + I14 + I15 + I16)
			CTID = CTI (I13 + I14) $\overline{\text{I15}}$ $\overline{\text{I16}}$
				Count I-register down by one count

		If I-count is not zero, return to X50	X50 = ELD $\overline{\text{MDEND}}$ + . . .	If I-count is not zero, cycle through X50 and X51 until shift count reaches zero
		If shift count is zero, generate MDEND	MDEND = $\overline{\text{I13}}$ $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$	
		If MDEND, inhibit X50 and generate X60	X60 = X51 $\overline{\text{SKPX}}$ + . . .	X60 follows X51 normally if no X-time is gated
X6 NY4	X60	D → Adder	DTE = DVSF X60 + . . .	Transfer contents of D-register to adder

		E → C	C1 = E1 + F1
			: :
			C16 = E16 + F16
				Transfer data to data output bus

		Inhibit LADS	No gating term true	
		Force 1 → ADS13	ADS13 = ELD YC + . . .	Force address X'0008' onto address lines (upper accumulator)

Table 3-21. Double Length Shift Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X60	Generate MGO	MGO = $\overline{\text{YIDL}}$ Z1 + . . .	Start memory cycle

		Generate WM	WM = WMD + . . .
			WMD = IMDS X6
				Write word into upper accumulator

	X61	G ——► Adder	GTE = IMDS X61 + . . .	Transfer contents of G-register to adder
		S $\nrightarrow$ B	STB = IMDS X61 + . . .	Load contents of adder into B-register
		If G1 was a 1, set KS	KS = ELD G1 X6 + . . .	Sign of shifted result is negative

		If S does not equal 0, set KU	KU = KUTME $\overline{\text{SZRO}}$ + . . .
			KUTME = IMDS X61 + . . .
			$\overline{\text{SZRO}}$ = S1 + S2 + . . . S16
				Contents of S-bus are not equal to 0

		Generate ENDI	ENDI = ENDID + . . .
			ENDID = IMDS X61 + . . .
				End instruction execution

		N $\nrightarrow$ L	NTL = ENDI NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ + . . .	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

accumulator. The contents of the upper accumulator are then loaded into the D-register. The contents of the lower accumulator are then transferred through the adder and loaded into the G-register.

During even clock X50, the contents of the G-register are left-shifted one bit position with G1 gated into H1 and bit D1 shifted into G17. The contents of the D-register are then transferred to the adder, left-shifted one bit position with H1 being transferred into E16A, and returned to the D-register. The shift count in the I-register is counted down by one count. During odd clock X51, the same shift procedure is performed and the I-register is counted down by one more count. If the count in the I-register is not zero, the computer returns to even clock X50 and cycles through X50 and X51 until the shift count reaches zero. When the shift count reaches zero, signal MDEND is generated and the shifting ceases.

During even clock X60, the contents of the D-register are transferred through the adder onto data output bus C1 through C16. A one is forced onto address line ADS13 to select the upper accumulator (X'0008'). Flip-flop MGO is set to start a memory cycle, and signal WM is generated to write the data word into the upper accumulator.

During odd clock X61, the contents of the G-register are transferred to the adder and loaded into the B-register. If bit G1 was a one, Sign indicator KS is set to indicate the shifted result is a negative number. If the contents of the sum bus are not equal to zero, Unequal indicator KU is set.

3.92 Halt Instruction (HLT) (See table 3-22)

The Halt instruction stops instruction execution and places the computer in the idle mode. Further instructions cannot be executed until the START switch on the control panel is pressed or until an interrupt becomes active.

During odd clock X21, signal ENDI is generated. Signal ENDI and signal HLT cause GIDL to be generated resetting flip-flops YA, YB, and YC. Y-state $\overline{\text{YA}}$ $\overline{\text{YB}}$ $\overline{\text{YC}}$ causes signal YIDL to be generated placing the computer in the idle mode.

3.93 Inclusive Or Instruction (IOR) (See table 3-23)

The Inclusive Or instruction forms the logical sum of the contents of the effective address and the B-register and loads the result into the B-register. Sign indicator KS is set if the sign of the result is negative and

Table 3-22. Halt Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate HLT	HLT = IXLH $\overline{\text{I6}}$ ILL1
			IXLH = IRST IM0 I5
			IRST = NY4 IH3
			IH3 = I1 I2
			IM0 = $\overline{\text{I3}}$ $\overline{\text{I4}}$
			ILL1 = $\overline{\text{I7}}$ $\overline{\text{I8}}$	

		Generate ISHMSC	ISHMSC = HLT + . . .	
	X21	Generate ENDI	ENDI = ISHMSC X21 + . . .	End instruction execution
		Generate GIDL	GIDL = HLT ENDI + . . .	
		Reset flip-flops YA, YB, and YC	$\overline{\text{YA}}$, $\overline{\text{YB}}$, $\overline{\text{YC}}$ = GIDL + . . .	
		Generate YIDL	YIDL = $\overline{\text{YA}}$ $\overline{\text{YB}}$ $\overline{\text{YC}}$ $\overline{\text{Z2}}$	Places computer in idle mode
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$ + . . .	

Table 3-23. Inclusive Or Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate IMSC	IMSC = IRST IM0 + . . .
			IRST = NY4 IH3
			IH3 = I1 I2
			IM0 = $\overline{I3}$ $\overline{I4}$	

		Generate X20	X20 = YOAS I1 I6 + . . .	
X2 NY4	X20	M —/→ D	MTD = IMSC X20 + . . .	Clock contents of effective address memory location into D-register

	X21	Generate ILPW	ILPW = IXLH I6
			IXLH = NY4 I1 I2 $\overline{I3}$ $\overline{I4}$	

		Generate IXOR	IXOR = ILPW $\overline{I7}$	
		B ——→ Adder	BTE = ILPW $\overline{I7}$ + . . .	Transfer contents of B-register to adder
		D ——→ Adder	DTE = ILPW $\overline{I7}$ + . . .	Transfer contents of D-register to same adder bus as B-register contents and merge
		If sign of result is negative, set flip-flop KS	KS = IXOR S1 X21 + . . . $\overline{KS}$ = IXOR + . . .	Set if sign of result is negative; otherwise reset

		If the result is not all 0's, set flip-flop KU	KU = KUTME $\overline{SZRO}$ + . . .
			KUTME = IXOR X21 + . . .
			$\overline{SZRO}$ = S1 + S2 + . . . S16
			$\overline{KU}$ = KUTME SZRO + . . .	

		Generate ENDI	ENDI = ISHMSC X21 + . . .
			ISHMSC = ILPW + . . .
				End execution sequence

		N —/→ L	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	

Unequal indicator KU is set if the result is not equal to zero.

The contents of the effective address are transferred to the D-register during even clock X20. During odd clock X21, the contents of the B-register are transferred to the E-input bus of the adder and the contents of the D-register are transferred to the E-input bus of the adder causing a merge to occur. If the sign of the result is negative, flip-flop KS is set; if the result is not all zeros, flip-flop KU is set.

3.94 Load Instruction (LDB) (See table 3-24)

The Load instruction loads the contents of the effective address into the B-register. Sign indicator KS is set if the sign of the effective word is negative; Unequal indicator KU is set if the effective word is not equal to zero.

During odd clock X61, the contents of the effective address are transferred through the adder into the B-register. If the sign is negative, Sign indicator KS is set; if the effective word is not all zeros, Unequal indicator KU is set.

3.95 Load Page Instruction (LDP) (See table 3-25)

The Load Page instruction copies the contents of its page field into the page register.

During even clock X20, bits I11 through I16 (page field) are transferred into page register P1 through P5.

Table 3-24. Load Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description
		Generate X60	X60 = YOAS $\overline{I1}$ I3 I4 + . . .	
X6 NY4	X60	Generate IATH	IATH = NY4 $\overline{I1}$ $\overline{IM1}$	
		Generate IATMD	IATMD = NY4 $\overline{I1}$	

		Generate ILD	ILD = IATH IH0 IM3
			IM3 = I3 I4	

	X61	$L \rightarrow ADS$	LADS = IATH X6 + . . .	Effective address
		$M \rightarrow$ Adder	MTF = IATH $\overline{IST}$ X61 + . . .	Contents of effective address transferred to adder
		$S \nrightarrow B$	STB = IATH $\overline{IST}$ X61 + . . .	Transfer contents of effective address to B-register
		Generate ENDI	ENDI = IATH X61 + . . .	End instruction execution
		$N \nrightarrow L$	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{YBGN}$	

Table 3-25. Load Page Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{IM1}$ + . . .
			IM1 = $\overline{I3}$ I4	

X2
NY4
	X20	Generate ILPW	ILPW = IXLH I6
			IXLH = IRST IM0 I5
			IRST = NY4 IH3
			IH3 = I1 I2
			IM0 = $\overline{I3}$ $\overline{I4}$	

		Generate IMSC	IMSC = IRST IM0	
		Generate ISHMSC	ISHMSC = ILPW + . . .	
		I11-I16 $\nrightarrow$ P	ITP = ILPW I7	Transfer bits I11-I16 (page field) into P1-P6
	X21	Generate ENDI	ENDI = ISHMSC X21 + . . .	End instruction execution
		$N \nrightarrow L$	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	

3.96 Load Index Instruction (LDX) (See table 3-26)

The Load Index instruction loads the contents of the effective address into the index register.

During even clock X20, address X'0000' is forced on the address lines by inhibiting the gating of the L-register onto the address lines. The contents of the L-register are then transferred to the E-input bus of the adder and onto output data bus C1 through C16. A memory cycle is then started, and write memory signal WM is generated to store the contents of the L-register into memory location X'0000'. The data from C1 through C16 is also transferred into index register bits IX1 through IX16.

3.97 Logical Right Instruction (LRB) (See table 3-27)

The Logical Right instruction shifts the contents of the B-register to the right the number of bit positions specified by the number-of-shifts field of the instruction (bits I13 through I16). As the contents are shifted, zeros are loaded into the sign bit position and the bits shifted out of the least significant bit position are lost. Sign indicator KS is set if the sign of the shifted result is negative and Unequal indicator KU is set if the shifted result is not all zeros.

During even clock X60, bits I13 through I16 are decoded to determine if a one-bit position or a four-bit position shift is required. The contents of the B-register are transferred to the E-input bus of the adder, and the appropriate shift is performed to the EA intermediate bus. Zeros are forced into gates H1 through H4 and are transferred into the most significant end of the EA intermediate bus. The shifted result is loaded into the B-register. The I-register is then counted down by the number of bit places shifted.

During odd clock X61, the new count in the I-register is decoded to determine the shift required. The contents

Table 3-26. Load Index Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate LDX	LDX = IRST IM0 $\overline{\text{I5}}$
			IRST = NY4 IH3
			IH3 = I1 I2
			IM0 = $\overline{\text{I3}}$ $\overline{\text{I4}}$	

		Inhibit L ——► ADS	No gating terms true	Force address X'0000' onto address lines
		Generate ISHMC	ISHMC = LDX + . . .	
		L ——► Adder	LTE = ISHMC X20 + . . .	Effective address gated to adder

		E ——► C	C1 = E1 + F1
			: :
			C16 = E16 + F16
				Address gated onto data output bus

		Generate MGO	MGO = Z1 $\overline{\text{YIDL}}$ XODD + . . .	Start memory cycle
		Generate WM	WM = LDX X2 + . . .	Write data word from C-bus into memory location X'0000'
	X21	Generate ENDI	ENDI = ISHMC X21 + . . .	End instruction execution
		N —/► L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$ + . . .	

Table 3-27. Logical Right Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS I1 IM1 $\overline{\text{I6}}$
			IM1 = $\overline{\text{I3}}$ I4	

X6
NY4
	X60	Generate OSFR	OSFR = IH3 $\overline{\text{I5}}$ ILL1 IM1
			IH3 = I1 I2
			ILL1 = $\overline{\text{I7}}$ I8
			IM1 = $\overline{\text{I3}}$ I4	

		Generate ILSF	ILSF = ISFT $\overline{\text{I7}}$
			ISFT = IRST IM1 $\overline{\text{I5}}$ $\overline{\text{I6}}$
			IRST = NY4 IH3	

		Generate ISFR	ISFR = ISFT I8	
		If I15 or I16, generate BSFT	BSFT = $\overline{\text{I13}}$ $\overline{\text{I14}}$ (I15 + I16)	Indicates 1-bit shift
		If I13 or I14, generate DSFT	DSFT = $\overline{\text{I15}}$ $\overline{\text{I16}}$ (I13 + I14)	Indicates 4-bit shift
		If BSFT, generate RS1	RS1 = ISFR BSFT SFTTME	
		If DSFT, generate RS4	RS4 = ISFR DSFT SFTTME	
		Generate RSHFT	RSHFT = RS1 + RS4	
		B → Adder	BTE = ISFT X6 + . . .	Transfer contents of B-register to adder

		0 → H1-H4	H1-H4 = HET + HRST + . . .
			HET = ELB X6
			HRST = IASF ISFR B1 X6 + . . .
				Force 0's into H1-H4

		If RS1, shift E right one place onto EA with H4 → E1A	E1A = RS1 H4 + . . .
			⋮ ⋮
			E16A = RS1 E15 + . . .
				Shift data right one bit position from E-input bus to EA intermediate bus. Shift 0's into most significant end

		If RS4, shift E right four places onto EA with H4 → E4A	E1A = RS4 H1 + . . .
			⋮ ⋮
			E16A = RS4 E12 + . . .
				Shift data right four bit positions from E-input bus to EA intermediate bus. Shift 0's into most significant end

		S $\not\longrightarrow$ B	STB = ISFT $\overline{\text{ELB}}$ X6 + . . .	Load shifted data into B-register
		Count I-register	CTI = ISFT $\overline{\text{ELB}}$ X6 + . . .	
		If DSFT, generate CTID	CTID = CTI DSFT + . . .	Count I-register down by 4
		If BSFT, generate CTIB	CTIB = CTI BSFT + . . .	Count I-register down by 1

Table 3-27. Logical Right Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X61	B → Adder	BTE = ISFT X6 + . . .	Transfer contents of B-register to adder
		0 → H1-H4	H1-H4 = HET + HRST + . . .	

		If RS1, shift E right one place onto EA	E1A = RS1 H4 + . . .
			⋮ ⋮
			E16A = RS1 E15 + . . .
				Shift data right one bit position onto EA-bus. Shift 0's into most significant end

		If RS4, shift E right four places onto EA	E1A = RS4 H1 + . . .
			⋮ ⋮
			E16A = RS4 E12 + . . .
				Shift data right four bit places onto EA-bus. Shift 0's into most significant end

		$S \xrightarrow{/} B$	STB = ISFT $\overline{\text{ELB}}$ X6 + . . .	Load shifted data into B-register
		Count I-register	CTI = ISFT $\overline{\text{ELB}}$ X6 + . . .	
		If DSFT, generate CTID	CTID = CTI DSFT + . . .	Count I-register down by 4
		If BSFT, generate CTIB	CTIB = CTI BSFT + . . .	Count I-register down by 1
		If I-count is not 0, return to X60	X60 = ISFT $\overline{\text{NOSFT}}$ X61 + . . .	Return to X60 and cycle through X60 and X61 until shift count reaches 0

		If I-count is 0, generate NOSFT	NOSFT = ISFT IDZRO WZRO
			IDZRO = $\overline{\text{I13}}$ $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$
			WZRO = $\overline{\text{I10}}$ $\overline{\text{I11}}$ $\overline{\text{I12}}$
				Shift count has been counted down to 0

		If B1 is a 1, set KS	KS = ISFT B1 X61 + . . .	Sign bit of shifted result is negative
		If B-register is not all 0's, set KU	KU = KUTME $\overline{\text{SZRO}}$ + . . . $\underline{\text{KUTME}}$ = ISFT X61 + . . . $\overline{\text{SZRO}}$ = S1 + S2 + . . . S16	B-register contents are not all 0's
		If NOSFT, generate ENDI	ENDI = NOSFT X61 + . . .	End instruction execution
		$N \xrightarrow{/} L$	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

of the B-register are transferred to the E-input bus of the adder, and the appropriate shift is performed to the EA intermediate bus with H1 through H4 transferred into the most significant end. The shifted result is loaded into the B-register. The I-register is counted down by the number of bit places shifted. If the count in the I-register is not zero, the computer returns to X60 and cycles through X60 and X61 until the count reaches zero. When the count reaches zero, signal NOSFT is generated and shifting ceases.

If the sign of the shifted result is negative, Sign indicator KS is set. Unequal indicator KU is set if the shifted result is not equal to zero.

3.98 Multiply Instructions (MPA, MPB) (See tables 3-28 and 3-29)

Two multiply instructions are available for use on the computer: a slow multiply instruction which is the standard multiply instruction and an optional fast multiply instruction. If the fast multiply option is used, the slow multiply instruction is disabled.

Slow Multiply Instruction (MPB). The slow multiply instruction multiplies the contents of the effective address by the contents of the B-register. The most significant half of the product is loaded into the upper accumulator and the least significant half of the product is loaded into the B-register. Overflow indicator KO is reset. Sign indicator KS is set if the sign of the product is negative, and Unequal indicator KU is set if the product is not equal to zero. Table 3-28 is the sequence chart for the slow multiply instruction.

The multiply instruction uses conventional techniques of shifting and adding, or of shifting but not adding, depending on the status of the current multiplier bit. Sixteen iterations of clock times X50 and X51 are required to generate the doubleword product.

During clock times X10 and X11, the multiplicand is transferred from the B-register through the adder and loaded into the D-register. The multiplier is fetched from the effective address memory location and loaded into the B-register. Flip-flop KB is set if the multiplier is negative. An address of X'0009' is forced onto the memory address lines during X20 and the multiplicand is written into memory. The lower half of the I-register is loaded with 1's for a count of 16.

During clock times X50 and X51, the actual multiplication is performed. The first bit of the multiplier to be used is transferred to flip-flop KA and the multiplier is shifted right one bit position. If the multiplier bit is a 1 and the multiplier was negative, the multiplicand is added to the partial product contained in the D-register and then shifted right one bit position with D16 transferred into H1 through H4. If the multiplier bit is a 1 and the multiplier was negative, the multiplicand is subtracted from the partial product (2's complement added) and shifted right one bit position. If the multiplier bit is a 0, the partial product is simply shifted right one bit position. The count in the I-register is counted down by one count. If the count is not equal to 0 ($\overline{\text{IDZRO}}$), clock times X50 and X51 are repeated until IDZRO comes true. The partial product is then shifted right one bit position and the least significant half of the product is returned to the B-register. Sign flip-flop KS is set if the sign of the product is negative. The most significant half of the product is stored in the upper accumulator during clock time X70, and the address of the next instruction is transferred to the L-register.

Fast Multiply Instruction (MPA). The fast multiply feature adds two additional registers to eliminate the need to fetch the multiplicand for each iteration. Table 3-29 is the sequence chart for fast multiplication.

The fast multiply instruction uses bit-pair multiplication to reduce the number of iterations required by one-half. During bit-pair multiplication, two bits of the multiplier are examined at one time and one addition or subtraction is performed for that bit pair.

Four different bit pairs are possible: 00, 01, 10, and 11. Multiplying by 00, 01, and 10 is done by normal shifting and addition. Multiplying by bit pair 00 is performed by adding 0's to the partial product; multiplying by bit pairs 01 and 10 is done by adding the multiplicand or two times the multiplicand, respectively. Multiplying by bit pair 11 requires that four times the multiplicand be added and then one times the multiplicand be subtracted.

During clock times X10 and X11, the lower half of the I-register is loaded with 1's, the Overflow, Sign, and Unequal flip-flops (KO, KS, KU) are reset and the multiplier is fetched from the effective address memory location and loaded into the G-register. The J-register is cleared to hold the most significant half of the product.

The actual multiplication is performed during clock times X50 and X51. Bit pair G15 and G16 are examined to determine if they are both 1's or both 0's. If G15 and G16 are both 0's, the partial product in the J-register is shifted right two bit positions; if bit G16 is a 1 and G15 is a 0, the multiplicand is added to the partial product and the product is shifted right two bit positions. If bit G16 is a 0 and bit position G15 is a 1, the partial product is shifted right one bit position, the multiplicand is added and the product is shifted. If bits G15 and G16 are both 1's, the multiplicand is first inverted (1's complement) and subtracted from the partial product. The count in the I-register is counted down by one count. If bits I14, I15, and I16 are not all 0's ($\overline{\text{MDEND}}$), clock times X50 and X51 are repeated until MDEND is true.

The most significant half of the product is transferred from the J-register to the adder, address X'0008' is

Table 3-28. Slow Multiply Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description
			X10 = YOAS $\overline{\text{X20}}$ $\overline{\text{X60}}$ YC + . . .	
X1 NY4	X10	Generate signal IATMD	IATMD = NY4 $\overline{\text{I1}}$	

		Generate signal IMPY	IMPY = IH0 IM1 NY4
			IH0 = $\overline{\text{I1}}$ $\overline{\text{I2}}$
			IM1 = $\overline{\text{I3}}$ I4	

		Generate signals MPA, MPB	MPA, MPB = IMPY	
		Generate signal IMDB	IMDB = MPB + . . .	
		B $\rightarrow$ Adder	BTE = IMDB X10 + . . .	Transfer contents of B-register (multiplicand) to adder
		S $\nrightarrow$ D	STD = IMDB X10 + . . .	Transfer multiplicand from adder sum bus to D-register
		Reset KU, KS, KO	$\overline{\text{KU}}$, $\overline{\text{KS}}$, $\overline{\text{KO}}$ = RSUSO RSUSO = NY4 $\overline{\text{NOSFT}}$ $\overline{\text{KA}}$ X10 + . . .	Reset status flip-flops KU, KS, and KO
	X11	L $\rightarrow$ ADS	LADS = IMDB X1 + . . .	Transfer effective address to address lines
		Generate signal MGO	MGO = $\overline{\text{YIDL}}$ Z1 XODD + . . .	Generate memory cycle to fetch multiplier
		M $\rightarrow$ Adder	MTF = IMDB X11 + . . .	Transfer multiplier from effective address memory location to adder
		If $\overline{\text{M1}}$, M $\nrightarrow$ B	STB = IMDB X11 + . . .	If bit M1 is a 0 (multiplier positive), transfer multiplier to B-register

		If M1, generate NEGF and INVF; set K17	NEGF = IMDB M1 X11 + . . .
			INVF = NEGF + . . .
			K17 = NEGF + . . .
				If bit M1 is a 1 (multiplier negative), invert multiplier and add K17 to obtain 2's complement

		S $\nrightarrow$ B	STB = IMDB X11 + . . .	Transfer multiplier to B-register
		If M1, set KB	KB = IMDB M1 X11 + . . .	Remember sign of multiplier
		Generate X20	X20 = X11 $\overline{\text{SKPX}}$ + . . .	X20 follows X11 normally if no other X-time is gated
X2 NY4	X20	Inhibit L $\rightarrow$ ADS	No gating term true	

Table 3-28. Slow Multiply Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X2
NY4
	X20	Force 1's into ADS16 and ADS13	ADS16 = IMDB $\overline{\text{X1}}$ $\overline{\text{X3}}$ $\overline{\text{X7}}$ + . . .
			ADS13 = IMDB $\overline{\text{X1}}$ + . . .
				Force address X'0009' onto address lines

		D ⟶ Adder	DTE = MPB X20 + . . .	Transfer multiplicand to adder

		Adder ⟶ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Transfer multiplicand onto output data bus

		Generate MGO	MGO = $\overline{\text{YIDL}}$ Z1 XODD + . . .	Begin memory cycle
		Generate WM	WM = IMDB X2 + . . .	Write multiplicand into memory location X'0009'
		If E-bus is not all 0's, set KU	KU = MPB $\overline{\text{EZRO}}$ X20 + . . . $\overline{\text{EZRO}}$ = E1A + E2A + . . . E16A	If multiplicand is not all 0's, set flip-flop KU
	X21	Force 1's onto F-bus	INVF = IMDB X21 + . . .	Force 1's onto F-bus by inverting all 0's
		S —/→ IL	STIL = IMDB X21 + . . .	Force a count of 15 into lower half of I-register for iteration counter
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 follows X21 normally if no other X-time is gated
X3 NY4	X30	B ⟶ Adder	BTE = IMDB X30 + . . .	Transfer multiplier to adder
		If the E-bus is all 0's, reset KU	$\overline{\text{KU}}$ = MPB EZRO X30 + . . .	If the multiplier is all 0's, reset Unequal flip-flop KU
		If D1 is a 1, set KS	KS = IMDB D1 X30 + . . .	If multiplicand was negative, set Sign flip-flop KS
	X31	Generate X40	X40 = X31 $\overline{\text{SKPX}}$ + . . .	X40 follows X31 normally if no other X-time is gated
X4 NY4	X40	S —/→ D	STD = MPB X40 + . . .	
		If either KS or KB are set, set KS	KS = IMDB KB X40 + . . . $\overline{\text{KS}}$ = IMDB KB X40 + . . .	If either the sign of the multiplicand or multiplier is negative, but not both, set Sign flip-flop KS
	X41	If flip-flop KU was reset during X30, reset KS	$\overline{\text{KS}}$ = MPB $\overline{\text{KU}}$ X41 + . . .	Multiplier was all 0's
		Generate X50	X50 = X41 $\overline{\text{SKPX}}$ + . . .	X50 follows X41 normally if no other X-time is gated

Table 3-28. Slow Multiply Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X5 NY4	X50	B → Adder	BTE = IMDB X50 + . . .	Multiplier transferred to adder
		Generate RS1	RS1 = MPB X5 + . . .	

		E1-E16 → E2A-E16A	E2A = E1 RS1 + . . .
			⋮ ⋮
			E16A = E15 RS1 + . . .
				Right shift E-bus one bit place onto EA-bus with D16 shifted into H1-H4

		D16 → H1-H4	H1-H4 = MPB D16 X50 + . . .	
		B16 $\not\rightarrow$ KA	KA = MPB B16 X50 + . . .	Least significant bit of multiplier transferred to flip-flop KA
	X51	D → Adder	DTE = IMDB X51 + . . .	Transfer partial product to adder
		Inhibit L → ADS	No gating term true	

		Force 1's into ADS16 and ADS13	ADS16 = IMDB $\overline{X1}$ $\overline{X3}$ $\overline{X7}$ + . . .
			ADS13 = IMDB $\overline{X1}$ + . . .
				Force address X'0009' onto address lines

		If KA is true: M → Adder	MTF = MPB KA X51 + . . .	If least significant bit of multiplier is a 1 (KA), fetch multiplicand and transfer to adder
		If $\overline{KB}$ KA: D+M $\not\rightarrow$ D		If sign of multiplicand is positive and multiplier bit is a 1, add and transfer to D-register

		If KB KA: D-M $\not\rightarrow$ D	NEGF = MPB KB X51 + . . .
			K17 = NEGF + . . .
				If sign of multiplicand is negative and multiplier bit is a 1, form 2's complement and subtract (add 2's complement)

		If $\overline{KA}$, return multiplier to D-register	STD = IMDB $\overline{IDZRO}$ X51 + . . .	If multiplier bit is a 0, transfer shifted partial product to D-register unless iteration count is 0
		Generate RS1	RS1 = MPB X5 + . . .	Shift partial product right one bit position from E-bus onto EA-bus
		If KC KS: 1's → H1-H4	H1-H4 = MPB KC KS X51 + . . .	Sign of multiplier or multiplicand was negative and multiplier bit is a 0
		If KA, set KC	KC = MPB KA X51 + . . .	Multiplier bit was a 1

Table 3-28. Slow Multiply Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X5 NY4	X51	Generate CTI	CTI = IMDB X51 + . . .	Count I-register down by one count
		If not IDZRO, X50	X50 = IMDB $\overline{\text{IDZRO}}$ X51 + . . .	If I-register is not counted down to 0, return to X50 and cycle through X50 and X51 until IDZRO
		If IDZRO, generate X60	X60 = X51 $\overline{\text{SKPX}}$ + . . .	
X6 NY4	X60	B → Adder	BTE = IMDB X60 + . . .	Transfer least significant half of product to adder
		Generate RS1	RS1 = MPB X60 + . . .	

		Shift E right one bit position onto EA-bus	E16A = E15 RS1 + . . .
			⋮ ⋮
			E1A = H4 RS1 + . . .
				Shift product right one bit position

		S ⇸ B	STB = IMDB BTE + . . .	Transfer shifted least significant half of product to B-register
		If SZRO, set KU	KU = IMDB X60 $\overline{\text{SZRO}}$ + . . .	If S-bus is not all 0's, set Unequal flip-flop KU
	X61	Generate X70	X70 = X61 $\overline{\text{SKPX}}$ + . . .	X70 follows X61 normally if no other X-time is gated
X7 NY4	X70	D → Adder	DTE = MPB X70 + . . .	Transfer contents of D-register to adder

		E → C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Most significant half of product transferred to output data bus

		Inhibit L → ADS	No gating term is true	
		Force 1 → ADS13	ADS13 = IMDB $\overline{\text{X1}}$ + . . .	Force address X'0008' onto memory address lines
		Generate MGO	MGO = $\overline{\text{YIDL}}$ Z1 XODD + . . .	Start memory cycle
		Generate WM	WM = IMDB X7 + . . .	Write most significant half of product in memory location X'0008'
	X71	Generate ENDI	ENDI = NY4 X71 + . . .	End instruction execution
		N ⇸ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI + . . .	Transfer next instruction address to L-register

Table 3-29. Fast Multiply Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description
			X10 = YOAS $\overline{X20}$ $\overline{X60}$ YC + . . .	
X1 NY4	X10	Generate signal IATMD	IATMD = NY4 $\overline{I1}$	

		Generate signal IMPY	IMPY = IH0 IM1 NY4
			IH0 = $\overline{I1}$ $\overline{I2}$
			IM1 = $\overline{I3}$ I4	

		Generate signal MPA	MPA = IMPY	
		Generate signal IMDB	IMDB = MPB + . . .	
		Generate signal MPB	MPB = IMPY	
		Generate signal IMDS	IMDS = IMDB + . . .	

		Generate signal NEGF	NEGF = INVFD + . . .
			INVFD = MPA X10 + . . .
				Force all 1's onto F-bus of adder

		S $\not\longrightarrow$ IL	STIL = MPA X10 + . . .	Force count of 15 into lower half of I-register
	X11	L $\longrightarrow$ ADS	LADS = IMDB X1 + . . .	Transfer effective address onto address lines
		Set MGO	MGO = $\overline{YIDL}$ Z1 XODD + . . .	Start memory cycle
		M $\longrightarrow$ Adder	MTF = IMDS X11 + . . .	Read multiplier from effective address memory location
		S $\not\longrightarrow$ G	STG = MPA X11 + . . .	Transfer multiplier to G-register
		Reset J1-J16	$\overline{J1}$-$\overline{J16}$ = RSTJ RSTJ = MPA X11	Clear J-register
		Generate X50	X50 = MPA X11 + . . .	
X5 NY4	X50	Check G15 and G16	G16QG15 = G16 $\overline{G15}$ + $\overline{G16}$ G15	Perform an exclusive-Or operation to determine if first two multiplier bits are both 1's or both 0's
		If G15 is a 1, set KA; otherwise reset KA	KA = MPA G15 XX5 $\overline{KA}$ = MPA $\overline{G15}$	

		If G15 and G16 are not both 1's or both 0's, generate BTF	BTF = MPA XX5 G16QG15
			XX5 = X5 IMDS
				Transfer multiplicand to adder

		If either G15 or G16 is a 1: J $\longrightarrow$ Adder	JTE = SPMS + . . .
			SPMS = G16 $\overline{KA}$ + $\overline{G16}$ KA
				Transfer partial product to adder (all 0's for first iteration)

Table 3-29. Fast Multiply Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X5 NY4	X50	If G15 is a 1, generate NEGF	NEGF = MPA G15 XX5	Invert multiplicand to form 1's complement and subtract. (If both G15 and G16 are 1's, one multiplicand must be subtracted from a multiplication by 4.)

		If G16 and G15 are both 1's or both 0's: S $\not\longrightarrow$ J	J1 = S1 SSPM + . . .
			⋮ ⋮
			J16 = S16 SSPM + . . .
			SSPM = (G16 KA + $\overline{\text{G16}}$ $\overline{\text{KA}}$) MPA XX5
				Return partial product to J-register

		If G16 and G15 are not both 1's, S $\not\longrightarrow$ J shifted right one bit position	J1 = BINSIN SPMS + . . .
			⋮ ⋮
			J16 = J15 SPMS + . . .
			SPMS = (G16 $\overline{\text{KA}}$ + $\overline{\text{G16}}$ KA) MPA XX5
				Shift partial product right one bit position during return to J-register

		Generate RSG2	RSG2 = SSPM + SPMS	Right shift multiplier two bit positions to place next two multiplier bits in positions G15 and G16
		If G16 and G15 are both 1's or both 0's, generate RSJ1	RSJ1 = SSPM	Right shift partial product one bit position to E-bus
		If J0 is a 1, force 1's into H1-H4	H1-H4 = MPA X5 J0	
		G15 $\not\longrightarrow$ KA	KA = MPA G15 XX5	Set KA if G15 is a 1
		Generate CTI	CTI = IMDS X5	Count I-register down by one count
	X51	Check G15 and G16	G16QG15 = G16 $\overline{\text{G15}}$ + $\overline{\text{G16}}$ G15	Perform an exclusive-Or operation to determine if first two multiplier bits are both 1's or both 0's
		If G15 is a 1, set KA; otherwise reset KA	KA = MPA G15 XX5 $\overline{\text{KA}}$ = MPA $\overline{\text{G15}}$	
		If G15 and G16 are not both 1's or both 0's, generate BTF	BTF = MPA XX5 G16QG15	Transfer multiplicand to adder
		If either G15 or G16 is a 1: J ⟶ Adder	JTE = SPMS + . . .	Transfer partial product to adder

Table 3-29. Fast Multiply Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X5 NY4	X51	If G15 is a 1, generate NEGF	NEGF = MPA G15 XX5	Invert multiplicand to form 1's complement and subtract. (If both G15 and G16 are 1's, one multiplicand must be subtracted from a multiplication by 4.)
		If G16 and G15 are both 1's or both 0's: S $\nrightarrow$ J	J1 = S1 SSPM + . . .	Return partial product to J-register
		If G16 and G15 are not both 1's, S $\nrightarrow$ J shifted right one bit position	J1 = BINSIN SPMS + . . .	Shift partial product right one bit position during return to J-register
		Generate RSG2	RSG2 = SSPM + SPMS	Right shift multiplier two bit positions to place next two multiplier bits in positions G15 and G16
		If G16 and G15 are both 1's or both 0's, generate RSJ1	RSJ1 = SSPM	Right shift partial product one bit position to E-bus
		If J0 is a 1, force 1's into H1-H4	H1-H4 = MPA X5 J0	
		G15 $\nrightarrow$ KA	KA = MPA G15 XX5	Set KA if G15 is a 1
		Generate CTI	CTI = IMDS X5	Count I-register down by one count

		If I-register has not been counted down four iterations, generate X50	X50 = IMDB $\overline{\text{MDEND}}$ X51
			MDEND = MPA $\overline{\text{I14}}$ $\overline{\text{I15}}$ $\overline{\text{I16}}$
				If I-register is not counted down four iterations, return to X50 and cycle through X50 and X51 until MDEND is true

		When MDEND is true, generate X60	X60 = X51 $\overline{\text{SKPX}}$ + . . .	X60 follows X51 normally if no X-time is gated
X6 NY4	X60	J ⟶ Adder	JTE = MPA X60 + . . .	Transfer upper half of product to adder

		E ⟶ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Upper half of product transferred to output data bus

		Inhibit L ⟶ ADS	No gating term true	
		Force 1 ⟶ ADS13	ADS13 = IMDB $\overline{\text{X1}}$ + . . .	Force address X'0008' onto address lines
		Set MGO	MGO = $\overline{\text{YIDL}}$ Z1 XODD + . . .	Begin memory cycle

Table 3-29. Fast Multiply Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X6 NY4	X60	Set WM	WM = IMDS X6 + . . .	Write upper half of product in memory location X'0008'
		If J1 is a 1, set KS	KS = MPA X60 J1	Sign of product is negative
	X61	G $\longrightarrow$ Adder	GTE = IMDS X61	Lower half of product transferred to adder
		S $\not\longrightarrow$ B	STB = IMDS X61	Transfer lower half of product to B-register
		If the sum bus is not all 0's, set KU	KU = IMDS X61 $\overline{\text{SZRO}}$ + . . . $\overline{\text{SZRO}}$ = S1 + S2 + . . . S16	If contents of sum bus are not all 0's, set Unequal flip-flop KU
		Generate ENDI	ENDI = IMDS X61 + . . .	End instruction execution
		N $\not\longrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register

forced on the address lines, and the most significant half of the product is stored in memory. Sign indicator KS is set if the sign of the product is negative. The least significant half of the product is transferred from the G-register through the adder and loaded into the B-register. The next instruction address is then transferred to the L-register.

3.99 Subtract Instruction (SBB) (See table 3-30)

The Subtract instruction subtracts the contents of the effective address from the contents of the B-register and loads the difference into the B-register. Overflow indicator KO is set if the difference exceeds the capacity of the B-register. Sign indicator KS is set if the sign of the difference is negative; Unequal indicator KU is set if the difference is not equal to zero.

During clock X61, the contents of the effective address are transferred to the F-input bus of the adder, and the contents of the B-register are transferred to the E-input bus of the adder. The data on the F-input bus is inverted by signal INVF; a one is forced into carry flip-flop K17 and is added to the F-input bus to form the two's complement. The contents of the E- and F-buses are added together and the result is transferred to the B-register. If either K1 or K2 is true, indicating overflow occurred, flip-flop KO is set indicating the difference exceeded the capacity of the B-register. If sign bit S1 is a one, indicating a negative difference, Sign indicator KS is set; if the contents of the sum bus are not all zeros, Unequal indicator KU is set.

3.100 Store Instruction (STB) (See table 3-31)

The Store instruction stores the contents of the B-register into the effective address without altering the contents of the B-register.

During even clock X60, the contents of the B-register are transferred through the adder onto output data bus C1 through C16. A memory cycle is started and signal WM is generated to write the contents of the B-register into the effective memory location.

Table 3-30. Subtract Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS $\overline{\text{I1}}$ $\overline{\text{IM1}}$
			IM1 = $\overline{\text{I3}}$ $\overline{\text{I4}}$	

X6
NY4
	X60	Generate IATH	IATH = NY4 $\overline{\text{I1}}$ $\overline{\text{IM1}}$
			IM1 = $\overline{\text{I3}}$ I4	

		Generate IATMD	IATMD = NY4 $\overline{\text{I1}}$	
		Generate IADSB	IADSB = IATH IMD	

		Generate ISUB	ISUB = IATH IH1 $\overline{\text{I4}}$
			IH1 = $\overline{\text{I1}}$ I2	

	X61	L → ADS	LADS = IATH X6 + . . .	Effective address
		M → Adder	MTF = IATH $\overline{\text{IST}}$ X61 + . . .	Contents of effective address
		B → Adder	BTE = IATH $\overline{\text{ILD}}$ X61 + . . .	Contents of B-register

		Generate INVF	INVF = NEGF + . . .
			NEGF = ISUB X61 + . . .
				Invert data on F-input bus

		1 $\nrightarrow$ K17	K17 = NEGF + . . .	Add to F-input bus to form 2's complement. Perform subtraction by adding contents of E- and F-input buses
		S $\nrightarrow$ B	STB = IATH $\overline{\text{ICM}}$ X61 + . . .	Load result into B-register

		If overflow, set KO	KO = IATH BINOVF X61 + . . .
			BINOVF = K1 ⊕ K2
				Difference exceeds capacity of B-register

		If S1 is a 1, set KS	KS = IATH BINSIN $\overline{\text{IST}}$ X61 + . . .
			BINSIN = S1 $\overline{\text{BINOVF}}$ + . . .
				Sign of difference is negative

		If S is not all 0's, set KU	KU = KUTME $\overline{\text{SZRO}}$ + . . .
			KUTME = IATH $\overline{\text{IST}}$ X61 + . . .
			$\overline{\text{SZRO}}$ = S1 + S2 + . . . S16
				Contents of S-bus are not all 0's

		Generate ENDI	ENDI = IATH X61 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

Table 3-31. Store Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X60	X60 = YOAS $\overline{\text{I1}}$ $\overline{\text{IM1}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X6 NY4	X60	B ⟶ Adder	BTE = IATH X60 + . . .	Transfer contents of B-register to adder

		E ⟶ C	C1 = E1 + F1
			: :
			C16 = E16 + F16	

		L ⟶ ADS	LADS = IATH X6 + . . .	Effective address gated onto address lines
		Generate signal MGO	MGO = Z1 $\overline{\text{YIDL}}$ + . . .	Start memory cycle

		Generate WM	WM = IST X6 + . . .
			IST = IATH IH1 IM3
				Write data signal to memory

	X61	Generate ENDI	ENDI = IATH X61 + . . .	End instruction execution
		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$ + . . .	
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register

3.101 Exchange Memory and Accumulator Instruction (XMB) (See table 3-32)

The Exchange Memory and Accumulator instruction exchanges the contents of the effective address with the contents of the B-register.

During even clock X20, the contents of the effective address are loaded into the D-register. During even clock X30, the contents of the B-register are transferred through the adder onto output data bus C1 through C16. Signal MGO is generated to start a memory cycle, and write memory signal WM is generated to write the data from the output bus into the effective address memory location.

The contents of the D-register are transferred to the adder and loaded into the B-register during odd clock X31.

3.102 Exclusive Or Instruction (XOR) (See table 3-33)

The Exclusive Or instruction forms the logical difference of the contents of the effective address and the B-register and loads the result into the B-register. If the sign of the result is negative, Sign indicator KS is set; if the result is not equal to zero, Unequal indicator KU is set.

The contents of the effective address are transferred into the D-register during even clock X20. During odd clock X21, the contents of both the B- and D-registers are transferred to the adder where an exclusive-Or operation is performed and inverted on the F-input bus. A one is forced into carry flip-flop K17 and is added to the contents of the F-bus to form the two's complement. The contents of both the B- and D-registers are transferred to the E-input bus of the adder causing a merge to be performed. A subtraction operation is performed by adding the contents of the E- and F-input buses. The result is transferred to the B-register. If bit S1 is a 1, indicating the result is negative, Sign indicator KS is set; if the contents of the sum bus are not all zeros, Unequal indicator KU is set.

3.103 OPERATOR'S/MAINTENANCE PANEL

The operator's/maintenance panel (processor control panel) acts as an intermediary between the CPU and the operator. The control panel contains indicators to display internal operating conditions to the operator and provides controls for the operator to modify internal operations in the CPU. The control panel, with its controls and indicators, is illustrated in figure 3-37.

Table 3-32. Exchange Memory and Accumulator Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

		Generate X20	X20 = YOAS I1 $\overline{\text{IM1}}$ + . . .
			IM1 = $\overline{\text{I3}}$ I4	

X2
NY4
	X20	Generate IMSC	IMSC = IRST IM0 + . . .
			IRST = NY4 IH3
			IH3 = I1 I2	

		Generate IXLH	IXLH = IRST IM0 I5
			IM0 = $\overline{\text{I3}}$ $\overline{\text{I4}}$	

		Generate XMB	XMB = IXLH $\overline{\text{I6}}$ ILL0
			ILL0 = $\overline{\text{I7}}$ $\overline{\text{I8}}$	

		L $\longrightarrow$ ADS	LADS = XMB + . . .	Effective address transferred to address lines
		M $\not\longrightarrow$ D	MTD = IMSC X20 + . . .	Contents of effective address transferred to D-register
	X21	No functions performed		
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$	X30 follows X21 normally when no X-time is gated
X3 NY4	X30	L $\longrightarrow$ ADS	LADS = XMB + . . .	Effective address transferred to address lines
		B $\longrightarrow$ Adder	BTE = XMB X30 + . . .	Contents of B-register transferred to adder

		E $\longrightarrow$ C	C1 = E1 + F1
			⋮ ⋮
			C16 = E16 + F16
				Transfer B-register contents to output data bus

		Generate MGO	MGO = Z1 $\overline{\text{YIDL}}$ + . . .	Start memory cycle
		Generate WM	WM = IMSC X3 + . . .	Write data word into effective address memory location
	X31	Generate ENDI	ENDI = XMB X31 + . . .	End instruction execution
		N $\not\longrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Next instruction address transferred to L-register

		D $\longrightarrow$ Adder,
		S $\not\longrightarrow$ B
			DTE = XMB X31 + . . .
			STB = XMB X31 + . . .
				Transfer contents of D-register to B-register

		Generate X10	X10 = ENDI $\overline{\text{YBGN}}$	

Table 3-33. Exclusive Or Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description
		Generate X20	X20 = YOAS I1 I6 + . . .	

X2
NY4
	X20	Generate IMSC	IMSC = IRST IM0 + . . .
			IRST = NY4 IH3
			IH3 = I1 I2
			IM0 = $\overline{I3}$ $\overline{I4}$	

		Generate ILPW	ILPW = IXLH I6
			IXLH = NY4 I1 I2 $\overline{I3}$ $\overline{I4}$	

		Generate IXOR	IXOR = ILPW $\overline{I7}$	
		Generate XOR	XOR = IXOR I8	
		M $\not\longrightarrow$ D	MTD = IMSC X20 + . . .	Clock contents of effective address memory location into D-register
	X21	B, D ⟶ Adder	F1-F16 = (B1 ⊕ D1) XOR thru (B16 ⊕ D16) XOR	Exclusive-Or operation performed on contents of B- and D-registers and inverted on F-bus
		1 $\not\longrightarrow$ K17	K17 = IXOR $\overline{I7}$ + . . .	A 1 is forced into K17 to form 2's complement
		B ⟶ Adder	BTE = ILPW $\overline{I7}$ + . . .	Contents of B- and D-registers merged
		D ⟶ Adder	DTE = ILPW $\overline{I7}$ + . . .	Subtraction operation performed

		S $\not\longrightarrow$ B	STB = STBD + . . .
			STBD = IXOR X21
				Result clocked into B-register

		If sign of result is negative, set flip-flop KS; otherwise reset it	KS = IXOR S1 X21 + . . .
			$\overline{KS}$ = RSUSO
			RSUSO = IXOR + . . .
				Indicates sign of result

		If result is not all 0's, set flip-flop KU	KU = KUTME $\overline{SZRO}$ + . . . $\overline{KU}$ = RSUSO $\overline{SZRO}$ = S1 + S2 + . . . S16	
		Generate ENDI	ENDI = ISHMSC X21 + . . .	End instruction execution
		N $\not\longrightarrow$ L	NTL = NY4 $\overline{IBX}$ $\overline{IBSV}$ ENDI	Transfer next instruction address to L-register
		Generate X10	X10 = ENDI $\overline{YBGN}$ + . . .	

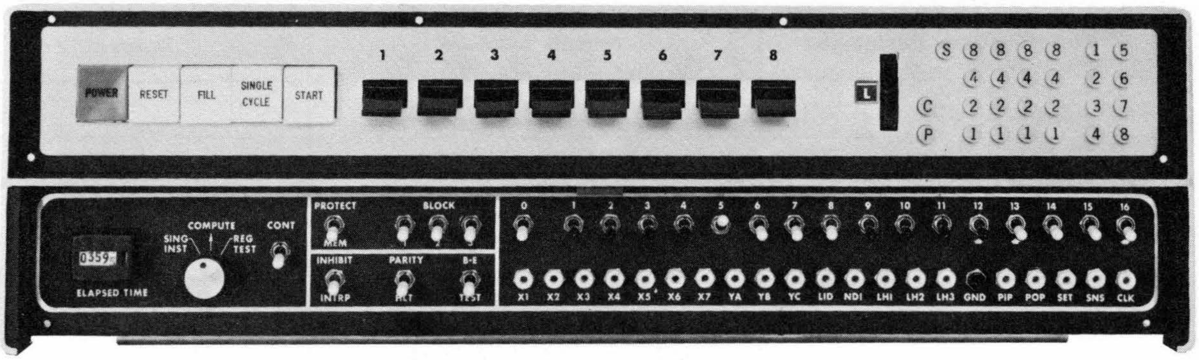


Figure 3-37. Processor Control Panel

3.104 Processor Control Panel Logic

The processor control panel switches and indicators with their logic signals are illustrated in figure 3-38 and their functions are described in table 2-1.

3.105 POWER Switch

The POWER switch is a push-on, push-off latching switch that allows the operator to apply power to the system if power is off, or to remove power from the system if power is on. The switch is backlighted by an incandescent lamp when power is on.

If the automatic power shutdown and restart option is installed, interrupt level one is triggered when power is applied.

3.106 RESET Switch

The RESET switch places the computer in the idle mode and sets up initial operating conditions.

Pressing the switch generates signal RSTPSW which sets flip-flop RSTPFF at the next XODD clock. Signal RSTPFF generates signal GIDL causing flip-flops YA, YB, and YC to reset at the next clock and generating signal YIDL. Start flip-flop STRT is reset at the next clock causing the CPU to enter the idle mode at the end of the current memory cycle.

RSTPFF = RSTPSW
$\overline{\text{RSTPFF}}$ = . . .
CRSTPFF = CKSW
    CKSW = CLK XODD

GIDL = RSTPFF

$\overline{\text{YA}}$ = RSTY GIDL
CYA = CKY

$\overline{\text{YB}}$ = RSTY GIDL
CYB = CKY

$\overline{\text{YC}}$ = RSTY GIDL
CYC = CKY
    CKY = CLK XODD

YIDL = $\overline{\text{YA}}$ $\overline{\text{YB}}$ $\overline{\text{YC}}$ $\overline{\text{Z2}}$

$\overline{\text{STRT}}$ = YIDL + . . .
CSTRT = CKST
    CKST = CLK

Signal YIDL also generates signal DOPIP at X1 time causing signals PIPPA, PIPP and LCS to be generated. Signal PIPP generates transfer signal INTF and gates data set into the 16 data switches (PIN1 through PIN16) onto input lines IN1 through IN16 to the F-input bus of the adder. Signal LCS transfers the data from the adder to the L-register at time X11.

DOPIP = X1 YIDL

PIPPA = RSTPFF DOPIP

PIPP = PIPPA

LCS = RSTPFF DOPIP

INTF = PIPP
IN1 = PIPP PIN1
: :
IN16 = PIPP PIN16

L1 = S1 STL
$\overline{\text{L1}}$ = . . .
CL1 = CKL
: :
L16 = S16 STL
$\overline{\text{L16}}$ = . . .
CL16 = CKL
    STL = LCS X11
    CKL = CKL (STL + . . .)

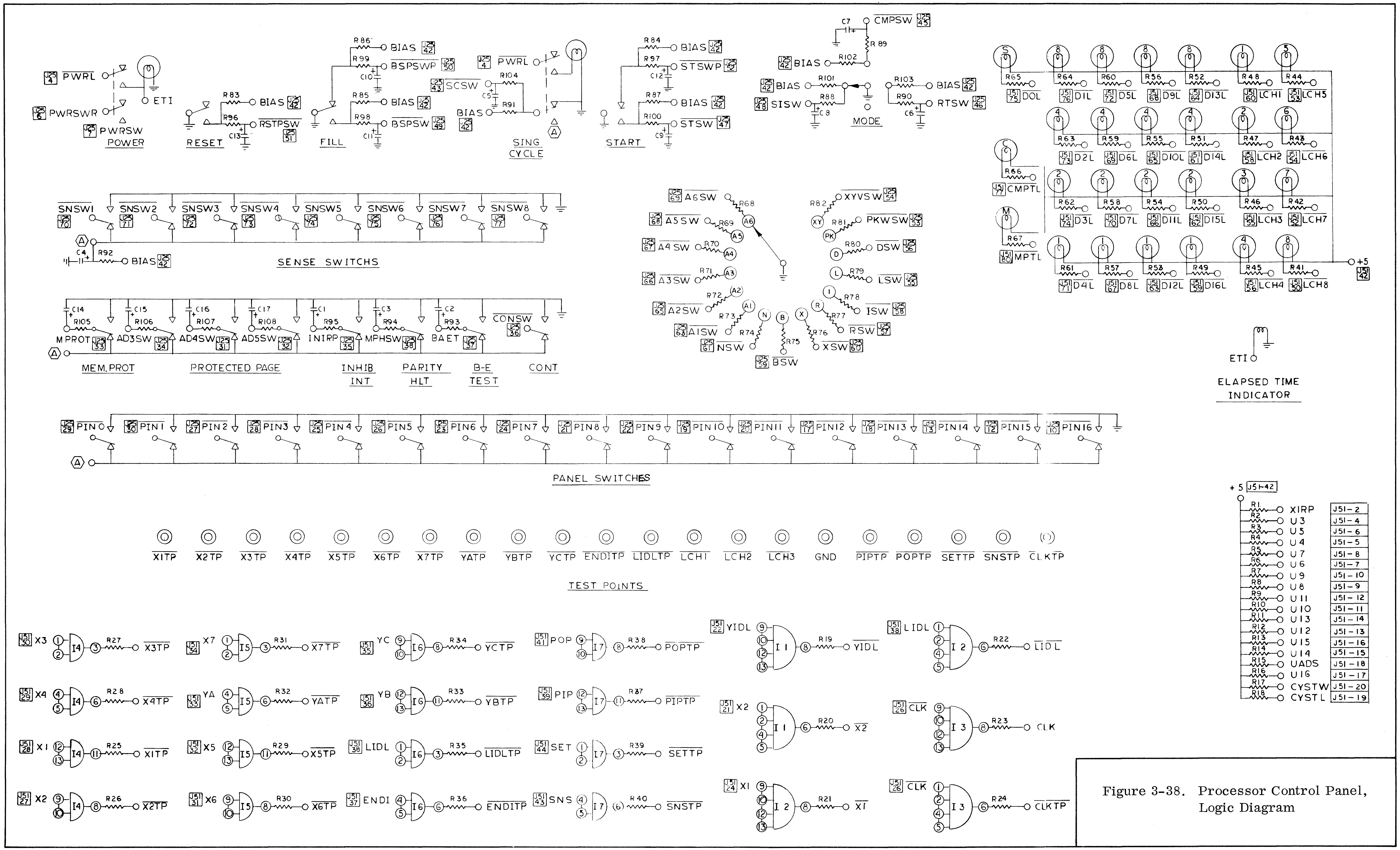


Figure 3-38. Processor Control Panel, Logic Diagram

Signal RSTPFF also sets the contents of the page register to X'01' by setting flip-flop P6 and resetting flip-flops P1 through P5.

$$\begin{aligned} \overline{\mathrm{P1}} &= \mathrm{RSTPFF} + \ldots \\ \mathrm{CP1} &= \mathrm{CKP} \\ \vdots \quad & \quad \vdots \\ \overline{\mathrm{P5}} &= \mathrm{RSTPFF} + \ldots \\ \mathrm{CP5} &= \mathrm{CKP} \\ \mathrm{P6} &= \mathrm{RSTPFF} + \ldots \\ \mathrm{CP6} &= \mathrm{CKP} \\ \mathrm{CKP} &= \mathrm{CLK}\ (\mathrm{RSTPFF} + \ldots) \end{aligned}$$

With the CPU in the idle mode, all controls and indicators are operative. The CPU is taken out of the idle mode when the START switch is pressed or when an interrupt occurs.

3.107 FILL Switch

The FILL switch activates a read-only diode memory containing a 21-instruction fill routine. Pressing the switch causes the CPU to begin executing the fill routine from address X'8000', the starting address of the diode memory. Loading continues until the entire program is loaded or until the RESET or SINGLE CYCLE switch is pressed. The FILL switch is operative only when the computer is in the idle mode.

Pressing the FILL switch generates signal BSPSW which sets flip-flop BSPFF at the next ODD clock. Signal BSPFF forces 0's into IN11 through IN16 and 1's into IN1 through IN10.

$$\begin{aligned} \mathrm{BSPFF} &= \mathrm{BSPSW} \\ \overline{\mathrm{BSPFF}} &= \overline{\mathrm{BSPSW}} \\ \mathrm{CBSPFF} &= \mathrm{CKSW} \\ \mathrm{CKSW} &= \mathrm{CLK\ XODD} \end{aligned}$$

$$\begin{aligned} \mathrm{IN1} &= \mathrm{BSPSW} + \ldots \\ \vdots \quad & \quad \vdots \\ \mathrm{IN10} &= \mathrm{BSPSW} + \ldots \\ \overline{\mathrm{IN11}} &= \mathrm{BSPSW} + \ldots \\ \vdots \quad & \quad \vdots \\ \overline{\mathrm{IN16}} &= \mathrm{BSPSW} + \ldots \end{aligned}$$

As the CPU is in the idle mode, signal YIDL is true, causing signal DOPIP to be generated at time X1. Signals DOPIP and BSPFF generate gating signal LCS which generates signal INTF gating the IN lines onto the F-input bus to the adder. The data on the IN lines is gated through the adder onto the sum bus and clocked into the L-register during time X11.

$$\mathrm{DOPIP} = \mathrm{XIDL\ X1}$$

$$\mathrm{LCS} = \mathrm{BSPFF\ DOPIP}$$

$$\mathrm{INTF} = \mathrm{LCS}$$

$$\begin{aligned} \mathrm{L1} &= \mathrm{S1\ STL} + \ldots \\ \overline{\mathrm{L1}} &= \ldots \\ \mathrm{CL1} &= \mathrm{CKL} \\ \vdots \quad & \quad \vdots \\ \mathrm{L16} &= \mathrm{S16\ STL} + \ldots \\ \overline{\mathrm{L16}} &= \ldots \\ \mathrm{CL16} &= \mathrm{CKL} \\ \mathrm{STL} &= \mathrm{LCS\ X11} + \ldots \\ \mathrm{CKL} &= \mathrm{CLK}\ (\mathrm{STL} + \ldots) \end{aligned}$$

Signal LADS is generated, gating the address from the L-register onto address lines ADS1 through ADS16. A 1 on address line ADS1 causes signal SDM to be generated. Signal SDM being true and address lines ADS12 through ADS16 containing 0's select the diode memory for readout.

$$\begin{aligned} \mathrm{ADS1} &= \mathrm{YOAS\ X1\ L1} \\ \mathrm{ADS2} &= \mathrm{LADS\ L2} \\ \vdots \quad & \quad \vdots \\ \mathrm{ADS16} &= \mathrm{LADS\ L16} \\ \mathrm{LADS} &= \mathrm{YOAS}\ \overline{\mathrm{X4}} \end{aligned}$$

$$\mathrm{SDM} = \mathrm{ADS1}$$

3.108 SINGLE CYCLE Switch

The SINGLE CYCLE switch is used to step the computer through a program one instruction at a time. It is a push-on, push-off switch with a back-lighted indicator. If the switch is pressed when the computer is in the idle mode, the indicator lights, but nothing occurs until the START switch is pressed or an interrupt becomes active. Pressing the START switch causes the computer to leave the idle mode, execute one complete instruction, and return to the idle mode. Pressing the SINGLE CYCLE switch a second time inhibits single cycle operation and and causes the indicator to go out. If the SINGLE CYCLE switch is pressed when the computer is in the compute mode, the computer enters the idle mode after completing execution of the instruction currently in progress and single cycle operation can then be performed using the START switch.

Pressing the SINGLE CYCLE switch generates signal SCSW which sets single cycle flip-flop SCFF at the next ODD clock.

If the computer is not in the idle mode, at the last clock of the instruction in process, signal ENDI is generated. Signals ENDI and SCFF generate signal GIDL, causing flip-flop LIDL to reset and resetting the start flip-flop STRT at the next clock. The computer is then in the idle mode.

$$\begin{aligned} \mathrm{SCFF} &= \mathrm{SCSW} \\ \overline{\mathrm{SCFF}} &= \ldots \\ \mathrm{CSCFF} &= \mathrm{CKSW} \\ \mathrm{CKSW} &= \mathrm{XODD\ CLK} \end{aligned}$$

GIDL = SCFF ENDI + . . .

$\overline{\text{LIDL}}$ = YIDL
CIDL = CKST

$\overline{\text{STRT}}$ = $\overline{\text{LIDL}}$
CIDL = CKST

$\overline{\text{STRT}}$ = $\overline{\text{LIDL}}$
CSTRT = CKST
CKST = CLK

When the START switch is pressed, flip-flop STRT sets at the next clock. Signal STRT sets flip-flop LIDL causing the computer to leave the idle state and begin executing the next instruction. At the last clock of the instruction, signal ENDI is generated causing signal GIDL to be generated. Flip-flops STRT and LIDL are reset at the next clock and the computer returns to the idle state.

3.109 START Switch

The START switch controls instruction execution. Pressing the switch causes the computer to leave the idle mode, enter the computer mode, and begin executing instructions. If single cycle operation is not in effect, the computer continuously executes instructions until either a Halt instruction is executed or the RESET or SINGLE CYCLE switch is pressed.

Pressing the START switch generates signal STSW. Signals STSW and YIDL (true when the computer is in the idle mode) set start flip-flop STRT at the next ODD clock. Signals STRT STSWP BSPSWP and YIDL set leave idle flip-flop LIDL at time X10.

STRT = STSW XIDL + . . .
$\overline{\text{STRT}}$ = LIDL
CSTRT = CLK

LIDL = STRT X10 (STSWP BSPSWP YIDL)
$\overline{\text{LIDL}}$ = . . .
CLIDL = CLK

Signal LIDL sets flip-flop YA and resets flip-flops YB and YC allowing memory request MGO for the next instruction. The computer then continues executing instructions.

YA = LIDL $\overline{\text{GIDL}}$ + . . .
$\overline{\text{YA}}$ = LIDL RSTY + . . .
CYA = CKY

$\overline{\text{YB}}$ = RSTY
CYB = CKY

$\overline{\text{YC}}$ = RSTY
CYC = CKY
RSTY = LIDL + . . .
CKY = XODD CLK

MGO = LIDL Z1 + . . .
$\overline{\text{MGO}}$ = . . .
CMGO = CLK

3.110 Sense Switches

The eight sense switches are used for external control of a stored program. They are two-position toggle switches with the upper position representing a logical 1 and the lower position representing a logical 0. The status of each switch can be tested and copied into Unequal indicator KU by a Sense instruction with the appropriate address. The hexadecimal addresses of the sense switches are:

SNSW1	X'01'
SNSW2	X'02'
SNSW3	X'03'
SNSW4	X'04'
SNSW5	X'05'
SNSW6	X'06'
SNSW7	X'07'
SNSW8	X'00'

The last three bits of the I-register are decoded and compared with the sense switch signals. If a comparison exists, signal DSCRP is generated.

$$\begin{aligned}\text{DSCRP} = {} & \text{SNSW1 I16 } \overline{\text{I15}}\ \overline{\text{I14}} + \text{SNSW2 } \overline{\text{I16}} \text{ I15 } \overline{\text{I14}} \\ & + \text{SNSW3 I16 I15 } \overline{\text{I14}} + \text{SNSW4 } \overline{\text{I16}}\ \overline{\text{I15}} \text{ I14} \\ & + \text{SNSW5 I16 } \overline{\text{I15}} \text{ I14} + \text{SNSW6 } \overline{\text{I16}} \text{ I15 I14} \\ & + \text{SNSW7 I16 I15 I14} + \text{SNSW8 } \overline{\text{I16}}\ \overline{\text{I15}}\ \overline{\text{I14}}\end{aligned}$$

If a Sense instruction with 0's in bit positions 9 through 13 is programmed, signal DSCRT is generated if DSCRP was true. Signal DSCRT sets Unequal flip-flop KU at the next clock time. Flip-flop KU can be tested with a Branch Equal or Branch Unequal instruction.

DSCRT = SNSP $\overline{\text{I13}}$ DSCRP + . . .
SNSP = SNS GRPP
GRPP = $\overline{\text{I9}}\ \overline{\text{I10}}\ \overline{\text{I11}}\ \overline{\text{I12}}$

KU = SNS DSCRT X30 + . . .
$\overline{\text{KU}}$ = RSUSO
CKU = CKKU
CKKU = CLK

3.111 Register Select Switch

The register select switch is a 15-position thumbwheel switch used primarily to select certain internal registers, indicators, or memory locations for display by the register display indicators. It is also used with the REG TEST position of the function switch to enter data from the 16 data switches into a selected register. Information selected for display by this switch is meaningful only when the computer is in the idle mode.

The operation of displaying the contents of the B-, N-, D-, I-, and P-registers and X, Y, V, and K flip-flops is the same so a description of displaying only the XY position of the register select switch is given. Table 3-34 lists the operations and gating signals for the other displays.

When the register select switch is set to position XY, signal $\overline{XYVSW}$ is forced to ground, generating signal HRT. If the register test switch is not set, indicating a display is desired, signal GRT2 is generated at time X40. Signals HRT and GRT2 generate gating signal STD to gate the contents of the sum bus into the D-register. When Z-state $\overline{Z1}$ $\overline{Z2}$ occurs, gating signal XYVTF occurs and transfers the outputs of X7 through X1, YC through YA, and V1 through V3 to the F-input adder bus, through the adder, and onto the sum bus. Gating term STD then transfers the sum bus contents into the D-register. The output of the D-register is applied to lamp drivers D1L through D16L and is displayed on the control panel indicators.

If the function switch is set to the REG TEST position, data from the 16 data switches is entered into the register or memory location selected on the register select switch. Selecting REG TEST generates signal $\overline{RTSW}$ which resets flip-flop $\overline{RTFF}$ at the next ODD clock, making output RTFF true.

RTFF = RTSW
$\overline{RTFF}$ = $\overline{RTSW}$
CRTFF = CKSW
CKSW = CLK XODD

Signals RTFF YIDL generate gating signal GRT1 at time X7. Signal GRT1 generates signal PIPPA which, in turn, generates signal PIPP. Signal PIPP generates gating signal INTF which transfers the data set into the control panel data switches into the adder.

GRT1 = RTFF YIDL X7

PIPPA = GRT1 + . . .

PIPP = PIPPA + . . .

INTF = PIPP + . . .

The output of the adder is then transferred to the register or memory location selected on the register select switch.

STXYV = GRT1 XYVSW + . . .

STPKW = GRT1 PKWSW + . . .

STD = GRT1 DSW + . . .

STL = GRT1 LSW + . . .

STI = GRT1 ISW + . . .

STB = GRT1 BSW + . . .

STN = GRT1 NSW + . . .

If switch positions A1 through A6 or the R- or X-register are selected, the address is forced onto the address lines as described for register display.

The output of the adder is also applied to output data gates C1 through C16. Signals GRT1 and MRT force memory write signal $\overline{WM}$ to go low generating signal TCM which transfers the data from C1 through C16 into the memory data register. The data from the control panel switches is then written into the addressed memory location.

3.112 Function Switch

The function switch is used primarily for maintenance and program debugging. It is a three-position rotary switch with positions labeled SING INST (single instruction), COMPUTE, and REG TEST (register test). The switch must be set to COMPUTE for normal computer operation. When the switch is set to REG TEST, operation is as described under the register select switch. When the function switch is set to SING INST and the START switch is pressed, the computer executes the instruction specified by the positions of data switches 1 through 16 and then returns to the idle mode.

Setting the function switch to the COMPUTE position generates signal CMPSW making GIDL go false and taking the computer out of the idle mode by making YIDL go false and $\overline{YIDL}$ go high. Signal $\overline{YIDL}$ causes signal $\overline{COMPTL}$ to go low lighting the C (compute) indicator on the control panel.

When the switch is set to SING INST, signal SISW is generated. At the end of the instruction being executed (if the computer is not in the idle mode), signal ENDI is generated. Signals ENDI and SISW generate signal GIDL causing flip-flops YA, YB, and YC to reset, generating signal YIDL. Signal YIDL inhibits memory accesses and forces the computer into the idle mode.

GIDL = SISW ENDI

$\overline{YA}$, $\overline{YB}$, $\overline{YC}$ = GIDL + . . .
CKYA, YB, YC = CLK XODD

YIDL = $\overline{Z2}$ $\overline{YC}$ $\overline{YB}$ $\overline{YA}$

$\overline{MGO}$ = YIDL + . . .
CMGO = CLK

At time X1, signal YIDL generates signal DOPIP. Signals DOPIP and SISW cause gating signals PIPPA and ICS to be generated. Signals ICS and PIPPA generate signal INTF, which gates the data set into the front panel data switches into the adder.

Table 3-34. Register Display, Sequence Chart

Time		Functions Performed	Signals Involved	Description
X4	X40	Display X, Y, V	$\overline{XYVSW}$ = Ground	Switch set to XYV position
		X, Y, V → Adder	XYVTF = $\overline{Z1}$ Z2 + . . .	Transfer output of X, Y, and V flip-flops to adder
		Generate HRT	HRT = $\overline{XYVSW}$ + . . .	
		Generate GRT2	GRT2 = YIDL X40 + . . .	Indicates read and not write
		S1-S16 —/→ D1-D16	STD = HRT GRT2 + . . .	Gate output of adder into D-register
		D1-D16 → D1L-D16L	D1L-D16L = D1-D16	Display contents of D-register
		Display P, K	$\overline{PKWSW}$ = Ground	Switch set to PK position
		Generate GRT2	GRT2 = YIDL X40 + . . .	Indicates read
		K → Adder	KTE = GRT2 PKWSW + . . .	States of KD, KS, KU, KK, KA, KB and KC transferred to adder
		P → Adder	PTF = KTE + . . .	Contents of P-register transferred to adder
		Generate HRT	HRT = $\overline{PKWSW}$ + . . .	
		S1-S16 —/→ D1-D16	STD = HRT GRT2 + . . .	Gate adder output into D-register
		D1-D16 → D1L-D16L	D1L-D16L = D1-D16	Display contents of D-register
		Display D	$\overline{DSW}$ = Ground	Switch set to D position
		Generate HRT	HRT = $\overline{DSW}$ + . . .	
		Generate GRT2	GRT2 = YIDL X40 + . . .	Indicates read
		D → Adder	DTE = GRT2 DSW + . . .	Transfer contents of D-register to adder
		S —/→ D	STD = HRT GRT2 + . . .	Transfer output of adder to D-register
		D → DL	D1L-D16L = D1-D16	Display contents of D-register
		Display L	$\overline{LSW}$ = Ground	Switch set to L position
		Generate HRT	HRT = $\overline{LSW}$ + . . .	
		Generate GRT2	GRT2 = YIDL X40 + . . .	Indicates read

Table 3-34. Register Display, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X4	X40	Display L (Cont.)		
		L ——→ Adder	LTE = GRT2 LSW + . . .	Transfer contents of L-register to adder
		S —/→ D	STD = HRT GRT2 + . . .	Transfer output of adder to D-register
		D ——→ DL	D1L-D16L = D1-D16	Display contents of D-register
		Display I	$\overline{ISW}$ = Ground	Switch set to I position
		Generate HRT	HRT = $\overline{ISW}$ + . . .	
		Generate GRT2	GRT2 = YIDL X40 + . . .	Indicates read
		I ——→ Adder	ITF = GRT2 ISW + . . .	Transfer contents of I-register to adder
		S —/→ D	STD = GRT2 HRT	Transfer output of adder to D-register
		D ——→ DL	D1L-D16L = D1-D16	Display contents of D-register
		Display B	$\overline{BSW}$ = Ground	Switch set to B position
		Generate HRT	HRT = $\overline{BSW}$	
		Generate GRT2	GRT2 = YIDL X40	Indicates read
		B ——→ Adder	BTE = GRT2 BSW	Transfer contents of B-register to adder
		S —/→ D	STD = GRT2 HRT	Transfer output of adder to D-register
		D ——→ DL	D1L-D16L = D1-D16	Display contents of D-register
		Display N	$\overline{NSW}$ = Ground	Switch set to N position
		Generate HRT	HRT = $\overline{NSW}$	
		Generate GRT2	GRT2 = YIDL X40	Indicates read
		N ——→ Adder	NTE = GRT2 NSW	Transfer contents of N-register to adder
		S —/→ D	STD = GRT2 HRT	Transfer output of adder to D-register
		D ——→ DL	D1L-D16L = D1-D16	Display contents of D-register

Table 3-34. Register Display, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X4	X40	Display R	$\overline{\text{RSW}}$ = Ground	Switch set to R position

		Generate MRT16, MRT15, MRT14	MRT16 = $\overline{\text{RSW}}$
			MRT15 = $\overline{\text{RSW}}$
			MRT14 = $\overline{\text{RSW}}$	

		Generate MRT	MRT = MRT16 + MRT15 + MRT14	
		Generate GRT2	GRT2 = YIDL X40 + . . .	
		Generate GRT3	GRT3 = GRT1 + GRT2	

		1 → ADS16,
		1 → ADS15,
		1 → ADS14
			ADS16 = MRT16 GRT3 + . . .
			ADS15 = MRT15 GRT3 + . . .
			ADS14 = MRT14 GRT3 + . . .
				Forces address X'0007' onto memory address lines

		$M \not\rightarrow D$	MTD = GRT2 MRT + . . .	Read contents of roll-arrow register (memory location X'0007') into D-register
		D → DL	D1L-D16L = D1-D16	Display contents of D-register
		Display X	$\overline{\text{XSW}}$ = Ground	Switch set to X position
		Generate GRT2	GRT2 = YIDL X40 + . . .	
		Generate MRT	MRT = $\overline{\text{XSW}}$	
		0 → ADS1-ADS16	$\overline{\text{ADS1}}$ thru $\overline{\text{ADS16}}$ = $\overline{\text{YOAS}}$ $\overline{\text{LADS}}$	Force address lines to 0's to read memory location X'0000'
		$M \not\rightarrow D$	MTD = GRT2 MRT + . . .	Read contents of memory location X'0000' (index register) into D-register
		D → DL	D1L-D16L = D1-D16	Display contents of D-register
		Display A1, A2, A3, A4, A5, or A6	$\overline{\text{A1SW}}$ or $\overline{\text{A2SW}}$ or $\overline{\text{A3SW}}$ or $\overline{\text{A4SW}}$ or $\overline{\text{A5SW}}$ or $\overline{\text{A6SW}}$ = Ground	Switch set to one of A1-A6 positions
		Generate GRT2	GRT2 = YIDL X40	
		Generate GRT3	GRT3 = GRT1 + GRT2	
		If A1 selected, generate MRT16	MRT16 = $\overline{\text{A1SW}}$ + $\overline{\text{A3SW}}$ + $\overline{\text{A5SW}}$	
		If A2 selected, generate MRT15	MRT15 = $\overline{\text{A2SW}}$ + $\overline{\text{A3SW}}$ + $\overline{\text{A6SW}}$	
		If A3 selected, generate MRT16 and MRT15		

Table 3-34. Register Display, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X4	X40	Display A1, A2, A3, A4, A5, or A6 (Cont.)		
		If A4 selected, generate MRT14	MRT14 = $\overline{A4SW} + \overline{A5SW} + \overline{A6SW}$	
		If A5 selected, generate MRT16 and MRT14		
		If A6 selected, generate MRT16 and MRT15		
		If A1, 1 → ADS16	ADS16 = MRT16 GRT3	Selects memory location X'0001'
		If A2, 1 → ADS15	ADS15 = MRT15 GRT3	Selects memory location X'0002'
		If A3, 1 → ADS16, 1 → ADS15		Selects memory location X'0003'
		If A4, 1 → ADS14	ADS14 = MRT14 GRT3	Selects memory location X'0004'
		If A5, 1 → ADS14, 1 → ADS16		Selects memory location X'0005'
		If A6, 1 → ADS14, 1 → ADS15		Selects memory location X'0006'
		M $\nrightarrow$ D	MTD = GRT2 MRT + . . .	Read contents of addressed memory location into D-register
		D → DL	D1L-D16L = D1-D16	Display contents of D-register

DOPIP = X1 YIDL $\overline{\text{DOIRP}}$

PIPPA = SISW DOPIP

ICS = DOPIP SISW

INTF = ICS + PIPPA + . . .

The output of the adder is transferred to the instruction register by signal ICS.

STI = ICS + . . .

When the START switch is pressed, flip-flops STRT and LIDL are set, causing the computer to leave the idle mode and execute the instruction contained in the I-register. At the last clock of the instruction, signal ENDI is generated. As signal SISW is still true, signal GIDL is generated causing the computer to return to the idle mode.

STRT = YIDL STSW + . . .
CSTRT = CLK

LIDL = YIDL STSW BSPSWP STRT X10
CLIDL = CLK

GIDL = SISW ENDI

3.113 CONT (Continuous Switch)

The CONT switch is a two-position toggle switch used primarily for maintenance and troubleshooting. When the switch is set to the upper (continuous) position, the computer starts executing instructions with the instruction at the address specified by the 16 data switches. One millisecond later the computer halts. After a 1-millisecond halt, the computer returns to the address specified by the data switches and again starts executing instructions. This cycle continues until the switch is set to the lower position.

When the switch is set to its upper position, signal CONSW is generated. Signal CONSW is gated with 2-millisecond clock CK2ML. Clock CK2ML is high for

1 millisecond and low for 1 millisecond. When CK2ML goes low, signals CONSW and $\overline{\text{CK2ML}}$ generate signal GIDL resetting flip-flops YA, YB and YC which generate signal YIDL and put the computer in the idle mode.

At time X1, signal YIDL generates signal DOPIP. Signals DOPIP and CONSW generate gating signals LCS and PIPPA. Signal LCS causes gating signal INTF to go true. Signal PIPPA generates signal PIPP which is combined with the signals from the data switches to form IN1 through IN16. The data from the data switches is gated into the adder by INTF and from the adder to the L-register by LCS. At time X11, LCS causes gating term STL to go high gating the address from the data switches into the L-register.

$\text{DOPIP} = \text{X1 YIDL } \overline{\text{DOIRP}}$

$\text{LCS} = \text{CONSW DOPIP } \overline{\text{SISW}} + \ldots$

$\text{PIPPA} = \text{CONSW DOPIP} + \ldots$

$\text{PIPP} = \text{PIPPA} + \ldots$

$\text{INTF} = \text{LCS}$

$\text{IN1} = \text{PIPP PIN1} + \ldots$

$\vdots \qquad \vdots$

$\text{IN16} = \text{PIPP PIN16} + \ldots$

$\text{STL} = \text{LCS X1L} + \ldots$

When the 2-millisecond clock goes high (CK2ML true) flip-flops STRT and LIDL are set, taking the computer out of the idle mode. The computer begins executing instructions starting with the address contained in the L-register. At the end of each instruction, the address in the L-register is increased by one and the next instruction is fetched.

$\text{STRT} = \text{CK2MS CONSW} + \ldots$
$\overline{\text{STRT}} = \text{LIDL}$
$\text{CSTRT} = \text{CKST}$

$\text{LIDL} = \text{STRT X10 (STSWP } \overline{\text{BSPFF}} \text{ YIDL)}$
$\overline{\text{LIDL}} = \ldots$
$\text{CLIDL} = \text{CKST}$
$\text{CKST} = \text{CLK}$

At the end of 1 millisecond (when the 2-millisecond clock goes false) signal GIDL goes true and forces the computer into the idle mode again for 1 millisecond until the 2-millisecond clock goes true and the cycle is repeated.

3.114 INHIBIT INTRP Switch

The INHIBIT INTRP (inhibit interrupt) switch is a two-position toggle switch that controls whether or not an interrupt can occur. When the switch is set to INHIBIT, no interrupts are allowed. When it is set to INTRP, interrupts are allowed to occur provided the interrupt level is armed and enabled.

When the switch is set to INHIBIT, signal INHIRP is generated ($\overline{\text{INHIRP}}$ grounded). Signal INHIRP generates signal STPIRP, which inhibits flip-flop DOIRP from setting when an interrupt request is received.

$\text{STPIRP} = \text{INHIRP} + \ldots$

$\text{DOIRP} = \text{IDOIRP } \overline{\text{STPIRP}}$
$\overline{\text{DOIRP}} = \text{LIDL}$
$\text{CDOIRP} = \text{CLK}$

When the switch is set to INTRP, signal $\overline{\text{INHIRP}}$ goes high causing $\overline{\text{STPIRP}}$ to go high if the SINGLE CYCLE and PARITY HLT switches are not on.

Signal DOIRP normally causes the computer to enter the idle mode at the conclusion of the instruction being executed and generates signal ICS which gates the interrupt address into the adder.

3.115 PARITY HLT Switch

The PARITY HLT (parity halt) switch is a two-position toggle switch that controls whether or not the computer halts when a memory parity error is detected. A parity check is made each time a word is read from memory. If the switch is set to HLT and a parity error occurs, the P indicator lights to indicate the error, but the computer continues executing instructions. If the switch is set to PARITY and a parity error occurs, the P indicator lights and the computer halts after executing the current instruction. The RESET and START switches must then be pressed to restart the program.

When the switch is set to PARITY, signal MPHSW is generated. If a memory request was made and another is not in process ($\overline{\text{MGO}}$ true), flip-flop MGODLY is set. If a parity error occurred (MPE) during the previous readout, flip-flop MPTY sets causing signal $\overline{\text{MPTL}}$ to go low, lighting the P indicator lamp on the control panel. Signals MPTY and MPHSW generate signal GIDL, causing the computer to enter the idle mode at the completion of the instruction being executed. Flip-flops STRT and LIDL are reset.

$\text{MGODLY} = \overline{\text{MGO}}$
$\text{CMGODLY} = \text{CLKB3}$
$\text{E/MGODLY} = \overline{\text{PPRST}}$
$\text{CLKB3} = \text{CLK}$

$\text{MPTY} = \text{MGODLY MPE} + \text{MPTY} + \ldots$
$\text{CMPTY} = \text{CLKB3}$
$\text{E/MPTY} = \overline{\text{PPRST}}$

$\text{MPTL} = \text{MPTY}$

$\text{GIDL} = \text{MPTY MPHSW} + \ldots$

Flip-flop MPTY latches itself with signal MPTY. Signals MPTY and MPHSW force $\overline{\text{STPIRP}}$ high to inhibit recognition of any interrupts.

The computer remains in the idle mode until the RESET switch is pressed. Signal $\overline{\mathrm{PPRST}}$ then directly resets flip-flops MGODLY and MPTY. The START switch must then be pressed to restart the program.

When the PARITY HLT switch is set to HLT, signal $\overline{\mathrm{MPHSW}}$ is generated. If a parity error occurs, flip-flops MGODLY and MPTY are set causing the P indicator on the control panel to light. However, since signal MPHSW is not true, signal GIDL is not generated, and the computer stays in the compute state.

3.116 PROTECT MEM and BLOCK Switches

The PROTECT MEM (protect memory) is a two-position toggle switch that is used with the BLOCK switches to prevent writing into protected areas of memory. When the switch is set to the upper position, writing into the area of memory specified by the setting of the BLOCK switches is inhibited. When the switch is set to the lower position, writing into any area of memory is permitted.

The three BLOCK switches select the area of memory protected. These are two-position toggle switches with the upper position representing a logical 1 and the lower position a logical 0.

When the PROTECT MEM switch is set to PROTECT, signal MPROT goes high. The settings of the BLOCK switches are compared with address bits ADS3, ADS4, and ADS5. If a comparison exists and MPROT is high, signal $\overline{\mathrm{INHWM}}$ is driven low forcing the write memory signal $\overline{\mathrm{WM}}$ to go high and inhibiting writing into any memory location beginning with the address set into the BLOCK switches up to memory location 16,384 (X'3FFF'). Memory locations above X'3FFF' cannot be locked out.

$$\overline{\mathrm{WM}} = \mathrm{INHWM} + \ldots$$

$$\begin{aligned}\mathrm{INHWM} = {} & \mathrm{ADS5\ AD5SW\ S3EA3\ S4EA4} \\ & + \overline{\mathrm{ADS5}}\ \overline{\mathrm{AD5SW}}\ \mathrm{SEA3\ SEA4} \\ & + \overline{\mathrm{ADS5}}\ \overline{\mathrm{AD5SW}}\ \mathrm{SEA3}\ \overline{\mathrm{SEA4}} \\ & + \mathrm{ADS4\ AD4SW\ SEA3} \\ & + \overline{\mathrm{ADS4}}\ \overline{\mathrm{AD4SW}}\ \mathrm{SEA3} + \mathrm{ADS3\ AD3SW}\end{aligned}$$

$$\mathrm{S3EA3} = \overline{\mathrm{AD3SW}}\ \mathrm{ADS3} + \mathrm{AD3SW\ ADS3}$$

$$\mathrm{S4EA4} = \overline{\mathrm{AD4SW}}\ \mathrm{ADS4} + \mathrm{AD4SW\ ADS4}$$

The areas of memory protected by the switches are as follows:

Switch Settings	Protected Memory Area
000	X'0000' to X'3FFF'
001	X'0800' to X'3FFF'
010	X'1000' to X'3FFF'
011	X'1800' to X'3FFF'
100	X'2000' to X'3FFF'
101	X'2800' to X'3FFF'
110	X'3000' to X'3FFF'
111	X'3800' to X'3FFF'

3.117 INPUT/OUTPUT OPERATION

The basic input/output system allows information interchange to occur between the processor and peripheral devices under program control. Single-word parallel transfers are used.

The basic input/output system can be expanded by the addition of optional block-transfer channels that operate through a direct memory access (DMA) feature.

The DMA feature allows control of high-speed I/O devices without program intervention. Two methods of transferring data can be used with the DMA. One method consists of transferring a single block of words. The second method consists of transferring multiple blocks of words and is referred to as chaining.

Communication and data transfers between the processor and external I/O devices are carried out through an I/O bus to which the I/O devices or the DMA is connected. Each I/O device has a unique address to differentiate it from all other devices on the bus. This address is used by each of the input/output instructions to alert a specific device for operation. Since all devices are connected to the same address lines, only the device whose address matches that specified by the instruction responds. Each input/output instruction includes the address field in the eight least significant bits of the instruction. The eight least significant bits of the I-register are used directly to provide the device address.

Peripheral devices connected to the I/O bus are controlled by four input/output instructions: Parallel Input (PIP), Parallel Output (POP), Sense (SNS) and Set (SET).

3.118 Parallel Input (PIP) Instruction

The PIP instruction is used to transfer information from an external device data register into the computer or from the control panel data switches into the B-register. Bits 9 through 16 of the effective address specify the input device from which data is to be loaded. When the effective address is X'01', data is loaded from the data switches. Sign indicator KS is set if the sign of the input data is negative, and Unequal indicator KU is set if the input data is not equal to zero.

Table 3-35 gives the operation sequence for a PIP instruction. Figure 3-39 illustrates the timing.

Table 3-35. Parallel Input (PIP) Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

X2
NY4
	X20	Generate signal IOIO	IOIO = IRST IM1 $\overline{\mathrm{I5}}$ I6
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{\mathrm{I3}}$ I4
				Generate I/O signal if instruction decode is an I/O instruction

		Generate signal PIP	PIP = IOIO I7 $\overline{\mathrm{I8}}$	Generate signal denoting a PIP instruction
		I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of controller from which data input is to be made
		Drive signal DOIO low	DOIO = IOIO X31	
		Drive signal $\overline{\mathrm{ODOIO}}$ high	$\overline{\mathrm{ODOIO}}$ = $\overline{\mathrm{DOIO}}$	Signals to addressed controller
		Drive signal $\overline{\mathrm{OPIP}}$ low	OPIP = PIP	
	X21	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of controller for data input
		Generate X30	X30 = X21 SKPX + . . .	X30 follows X21 normally if no X-time has been gated
X3 NY4	X30	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of controller for data input
		Data received from controller on $\overline{\mathrm{UIN1}}$-$\overline{\mathrm{UIN16}}$ data lines		
		Generate signal PIPU if data input is not from control panel ($\overline{\mathrm{GRPP}}$ true)	PIPU = PIP $\overline{\mathrm{GRPP}}$ + . . .	Data input is from a peripheral device and not from control panel
		Generate signal PIPCYU	PIPCYU = PIPU	

		If the data is from the control panel, generate signal PIPP	PIPP = PIP GRPP I16 + . . .
			GRPP = $\overline{\mathrm{I9}}$ $\overline{\mathrm{I10}}$ $\overline{\mathrm{I11}}$ $\overline{\mathrm{I12}}$
				If effective address is X'01', data is from the control panel

		Gate data into IN1-IN16	IN1-IN16 = ($\overline{\mathrm{UIN1}}$-$\overline{\mathrm{UIN16}}$) PIPCYU + ($\overline{\mathrm{PIN1}}$-$\overline{\mathrm{PIN16}}$) PIPP	
		Generate signal INTF	INTF = PIPP + PIPCYU	Gates IN1-IN16 into adder

		If data word is negative (S1 true), set flip-flop KS; otherwise reset it	KS = PIP BINSIN X30 + . . .
			$\overline{\mathrm{KS}}$ = RSUSO
			BINSIN = S1 $\overline{\mathrm{BINOVF}}$
			RSUSO = PIP X30 + . . .
				Flip-flop KS is used to indicate sign of data word

		S $\not\longrightarrow$ B	STB = PIP X30 + . . .	Data word clocked into B-register

Table 3-35. Parallel Input (PIP) Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X3 NY4	X30	If data word is not all zeros, set flip-flop KU; otherwise reset it	KU = KUTME $\overline{\text{SZRO}}$ + . . . $\overline{\text{KU}}$ = RSUSO $\overline{\text{SZRO}}$ = $\overline{\text{SZRO1}}$ + $\overline{\text{SZRO2}}$ + $\overline{\text{SZRO3}}$ + $\overline{\text{SZRO4}}$ $\overline{\text{SZRO1}}$ = S1 + S2 + S3 + S4 $\overline{\text{SZRO2}}$ = S5 + S6 + S7 + S8 $\overline{\text{SZRO3}}$ = S9 + S10 + S11 + S12 $\overline{\text{SZRO4}}$ = S13 + S14 + S15 + S16	Indicates data word is not all zeros
	X31	I9-I16 ⟶ OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of controller for input
		Generate signal DOIO	DOIO = IOIO X31	
		Generate signal ENDI	ENDI = IOIO X31 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register

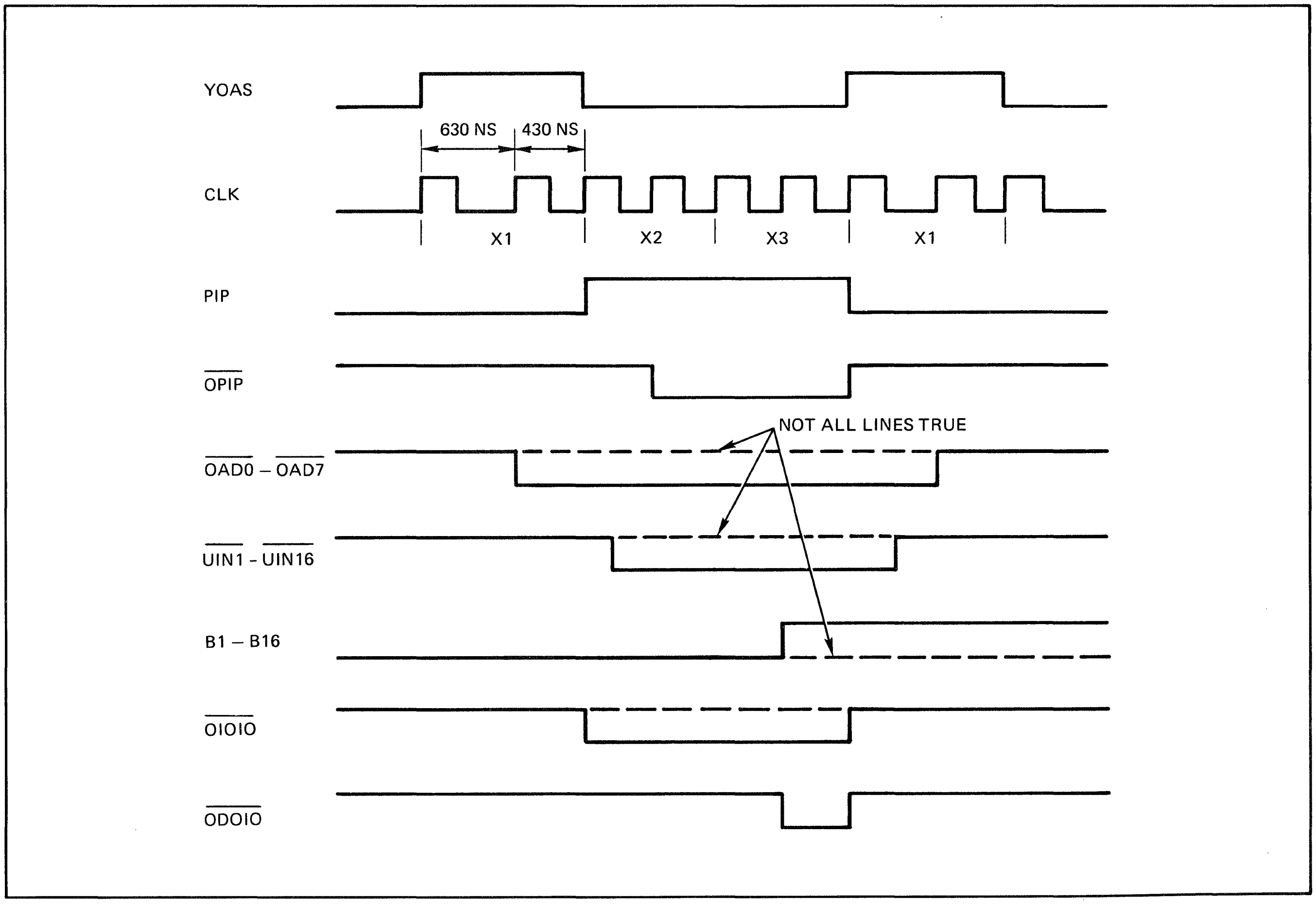

YOAS
630 NS
430 NS
CLK
X1
X2
X3
X1
PIP
$\overline{\text{OPIP}}$
NOT ALL LINES TRUE
$\overline{\text{OAD0}}$ – $\overline{\text{OAD7}}$
$\overline{\text{UIN1}}$ - $\overline{\text{UIN16}}$
B1 – B16
$\overline{\text{OIOIO}}$
$\overline{\text{ODOIO}}$

Figure 3-39. Parallel Input (PIP) Instruction, Timing Diagram

3.119 Parallel Output (POP) Instruction

The POP instruction is used to transfer information from the B-register to an external device or to display the eight most significant bits of the B-register on the programmable display indicators on the control console. Bits 9 through 16 of the effective address specify the device to which the data is to be transferred. When the effective address is X'00', bits 1 through 8 of the B-register are displayed on the control panel.

The POP instruction is also used to arm and disarm the priority interrupts. The interrupt levels are individually armed or disarmed 16 levels at a time by executing a POP instruction with an effective address of X'EA' for levels 1 through 16 or X'EC' for levels 17 through 32. The word in the B-register specifies the state to which each level is set. If the bit is a 0, the level is disarmed; if the bit is a 1, the level is armed.

Table 3-36 gives the operation sequence for a POP instruction. Figure 3-40 illustrates the timing.

3.120 Sense (SNS) Instruction

The SNS instruction is used to test the operational status of an I/O device. When the instruction is executed, Unequal indicator KU is set or reset to indicate the status of the condition specified by the instruction. Table 3-37 gives the operation sequence for a SNS instruction. Figure 3-41 illustrates the timing.

3.121 SET Instruction

The SET instruction transmits a discrete control signal to an I/O device to perform such operations as resetting operating conditions, starting a mechanical operation, rewinding tape, or stopping an operation.

The SET instruction is also used to enable and reset the priority interrupts. The interrupt levels are enabled by executing a SET instruction with an effective address of X'EA'. This enables the interrupt levels as a whole, but only the highest priority waiting level can advance to the active state. An active level can be reset by executing a SET instruction with an effective address of X'EB'. This resets the currently active interrupt level and permits the next highest waiting and enabled level to advance to the active state.

Table 3-38 gives the operation sequence for a SET instruction. Figure 3-42 illustrates the timing.

3.122 GLOSSARY OF LOGIC SIGNALS

Table 3-39 is a glossary of logic signals used in the RC 70 Computer logic equations. The logic signals are arranged in alphanumeric order and a definition is given for each.

Table 3-36. Parallel Output (POP) Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

X2
NY4
	X20	Generate signal IOIO	IOIO = IRST IM1 $\overline{\text{I5}}$ I6
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{\text{I3}}$ I4
				Generate I/O signal if instruction decode is an I/O instruction

		Generate signal POP	POP = IOIO I7 I8	Generate signal denoting a POP instruction
		I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of device where data is to go
		Drive signal DOIO low	DOIO = IOIO X31	
		Drive signal $\overline{\text{ODOIO}}$ high	$\overline{\text{ODOIO}}$ = $\overline{\text{DOIO}}$	Signals to addressed device
		Drive signal $\overline{\text{OPOP}}$ low	OPOP = POP	
		B → Adder	BTE = POP	Transfer output data to adder
		E → C	C1-C15 = $\overline{\text{E1-E15}}$ + $\overline{\text{F1-F15}}$	Data transferred from adder to output data lines
		C → $\overline{\text{OUT}}$	$\overline{\text{OUT1}}$-$\overline{\text{OUT16}}$ = C1-C16	

Table 3-36. Parallel Output (POP) Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X2 NY4	X21	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of peripheral device where data is to go
		C → $\overline{\text{OUT}}$	$\overline{\text{OUT1}}$-$\overline{\text{OUT16}}$ = C1-C16	Output data
		Generate X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 follows X21 normally if no X-time is gated
X3 NY4	X30	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of peripheral device where data is to go
		C → $\overline{\text{OUT}}$	$\overline{\text{OUT1}}$-$\overline{\text{OUT16}}$ = C1-C16	Output data

		If effective address is X'00', generate signal RLCH	RLCH = POP GRPP X3
			GRPP = $\overline{\text{I9}}$ $\overline{\text{I10}}$ $\overline{\text{I11}}$ $\overline{\text{I12}}$
				Resets B-register latches

		Generate signal UNLCH	UNLCH = RLCH	
	X31	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Address of peripheral device where data is to go
		C → $\overline{\text{OUT}}$	$\overline{\text{OUT1}}$-$\overline{\text{OUT16}}$ = C1-C16	Output data
		Generate signal DOIO	DOIO = IOIO X31	

		If effective address is X'00', generate GRPP and SLCH	SLCH = POP GRPP DOIO
			GRPP = $\overline{\text{I9}}$ $\overline{\text{I10}}$ $\overline{\text{I11}}$ $\overline{\text{I12}}$
				Latch data from B-register into display latches $\overline{\text{LCH1}}$-$\overline{\text{LCH8}}$ and display on control panel indicators

		Generate signal ENDI	ENDI = IOIO X31 + . . .	End instruction execution
		N $\xrightarrow{/}$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address into L-register

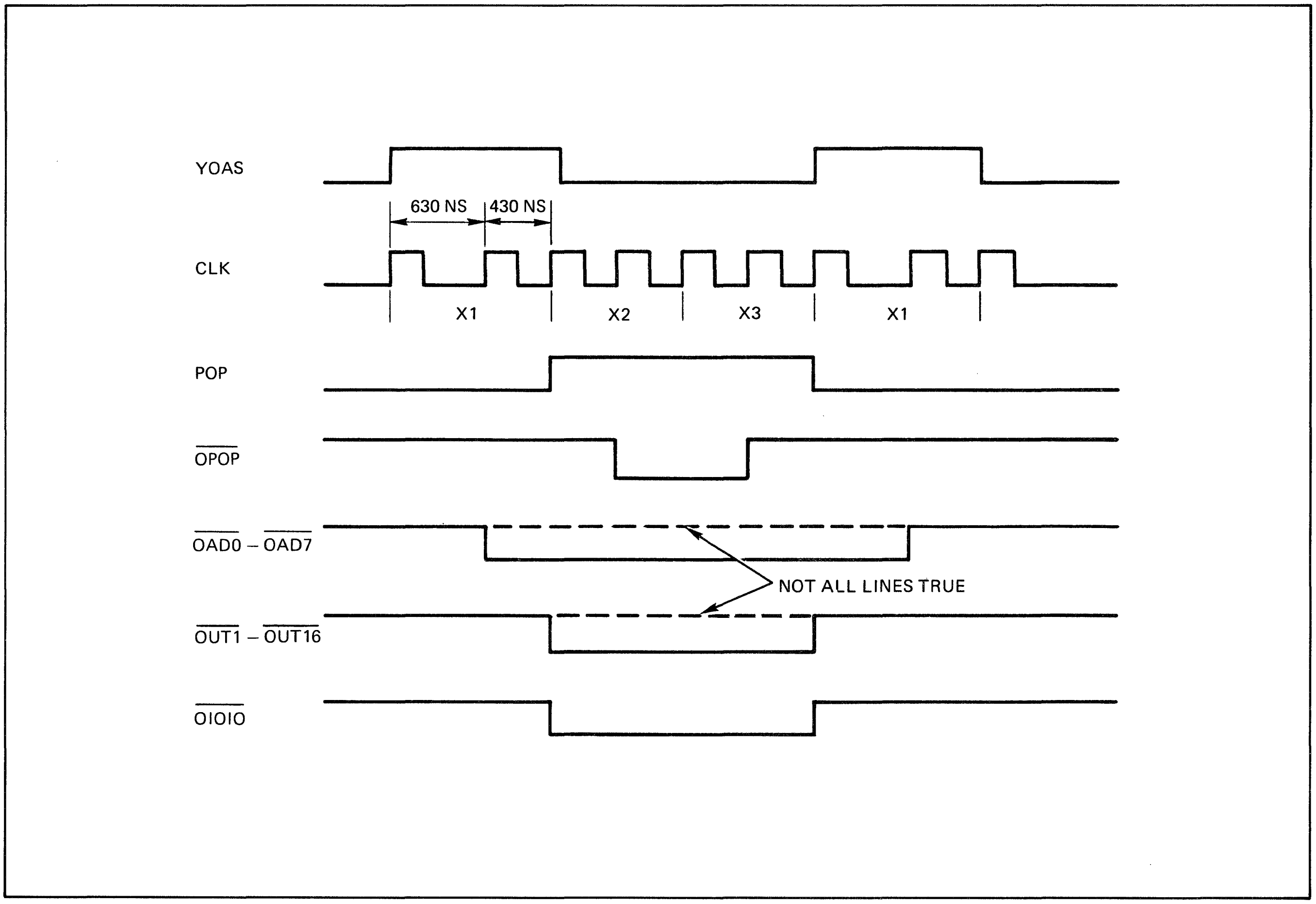

YOAS
630 NS
430 NS
CLK
X1
X2
X3
X1
POP
$\overline{\text{OPOP}}$
$\overline{\text{OAD0}}$ – $\overline{\text{OAD7}}$
NOT ALL LINES TRUE
$\overline{\text{OUT1}}$ – $\overline{\text{OUT16}}$
$\overline{\text{OIOIO}}$

Figure 3-40. Parallel Output (POP) Instruction, Timing Diagram

Table 3-37. Sense (SNS) Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

X2
NY4
	X20	Generate signal IOIO	IOIO = IRST IM1 $\overline{\text{I5}}$ I6
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{\text{I3}}$ I4
				Generate I/O signal if instruction decode is an I/O instruction

		Generate signal SNS	SNS = IOIO ILL1
			ILL1 = $\overline{\text{I7}}$ I8
				Generate signal denoting a Sense instruction

		I9-I16 ——► OAD0-OAD7	OAD0-OAD7 = I9-I16	Transfer controller address and sense status request
		Drive signal DOIO low	DOIO = IOIO X31	
		Drive signal $\overline{\text{ODOIO}}$ high	$\overline{\text{ODOIO}}$ = $\overline{\text{DOIO}}$	Signals to controller
		Drive signal $\overline{\text{OSNS}}$ low	OSNS = SNS X20	

Table 3-37. Sense (SNS) Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description

X2
NY4
	X20	If bits I9-I12 are 0's, generate signal SNSP	SNSP = SNS GRPP
			GRPP = $\overline{\text{I9}}$ $\overline{\text{I10}}$ $\overline{\text{I11}}$ $\overline{\text{I12}}$
				Indicates Sense instruction is not I/O but to test sense switches on control panel

		If I13 is a 0 and SNSP is true, test sense switches. If coincidence is found, generate DSCRT	DSCRT = SNSP $\overline{\text{I13}}$ DSCRP	
	X21	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Controller address and sense status request
		Generate signal X30	X30 = X21 $\overline{\text{SKPX}}$ + . . .	X30 normally follows X21 if no other X-time is gated
X3 NY4	X30	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Controller address and sense status
		When status signal $\overline{\text{DSCRU}}$ received from addressed device, generate signal DSCRT	DSCRT = SNS $\overline{\text{GRPP}}$ $\overline{\text{DSCRU}}$+ . . .	Status signal received from addressed device. If signal $\overline{\text{DSCRU}}$ is true, requested status is true
		Set flip-flop KU	KU = SNS DSCRT X30 + . . .	Flip-flop KU is set if the received status is true or a coincidence occurred between sense switch and address during time X20
	X31	I9-I16 → OAD0-OAD7	OAD0-OAD7 = I9-I16	Controller address and sense status requested
		Hold signal DSCRT	DSCRT = SNS $\overline{\text{GRPP}}$ $\overline{\text{DSCRU}}$+ . . .	
		Hold flip-flop KU set	$\overline{\text{KU}}$ = $\overline{\text{RSUSO}}$	
		Generate signal ENDI	ENDI = IOIO X31 + . . .	End instruction execution
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register

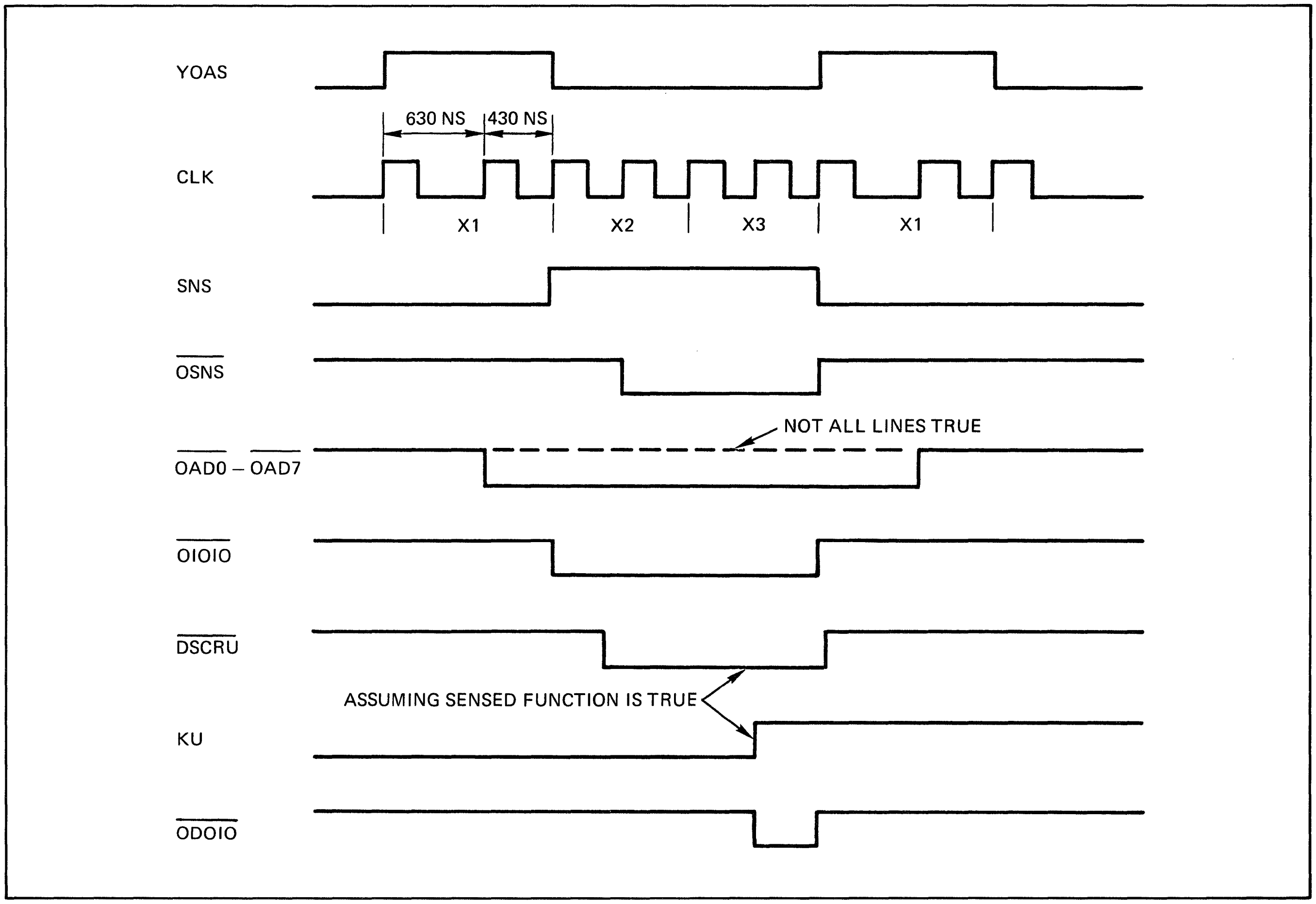

YOAS
630 NS
430 NS
CLK
X1
X2
X3
X1
SNS
$\overline{\mathrm{OSNS}}$
NOT ALL LINES TRUE
$\overline{\mathrm{OAD0}} - \overline{\mathrm{OAD7}}$
$\overline{\mathrm{OIOIO}}$
$\overline{\mathrm{DSCRU}}$
ASSUMING SENSED FUNCTION IS TRUE
KU
$\overline{\mathrm{ODOIO}}$

Figure 3-41. Sense (SNS) Instruction, Timing Diagram

Table 3-38. SET Instruction, Sequence Chart

Time		Functions Performed	Signals Involved	Description

X2
NY4
	X20	Generate signal IOIO	IOIO = IRST IM1 $\overline{\mathrm{I5}}$ I6
			IRST = NY4 IH3
			IH3 = I1 I2
			IM1 = $\overline{\mathrm{I3}}$ I4
				Generate I/O signal if instruction decode is an I/O instruction

		Generate signal SET	SET = IOIO $\underline{\mathrm{IL}}\mathrm{L0}$
			ILL0 = $\overline{\mathrm{I7}}$ $\overline{\mathrm{I8}}$
				Generate signal denoting a SET instruction

		I9-I16 ⟶ OAD0-OAD7	OAD0-OAD7 = I9-I16	Transmit discrete control signal to I/O bus
		Drive signal DOIO low	DOIO = IOIO X31	
		Drive signal $\overline{\mathrm{ODOIO}}$ high	$\overline{\mathrm{ODOIO}}$ = $\overline{\mathrm{DOIO}}$	

Table 3-38. SET Instruction, Sequence Chart (Cont.)

Time		Functions Performed	Signals Involved	Description
X2 NY4	X21	I9-I16 ⟶ OAD0-OAD7	OAD0-OAD7 = I9-I16	Control signal to I/O bus

		Set flip-flop MIDIO	MIDIO = IOIO $\overline{\text{MIDIO}}$
			$\overline{\text{MIDIO}}$ = . . .
			CMIDIO = XODD	

		Drive signal $\overline{\text{OSET}}$ low	OSET = SET $\overline{\text{MIDIO}}$	
X3 NY4	X30	I9-I16 ⟶ OAD0-OAD7	OAD0-OAD7 = I9-I16	Control signal to I/O bus
	X31	I9-I16 ⟶ OAD0-OAD7	OAD0-OAD7 = I9-I16	Control signal to I/O bus
		Drive signal DOIO high	DOIO = IOIO X31	
		Drive signal $\overline{\text{ODOIO}}$ low	$\overline{\text{ODOIO}}$ = $\overline{\text{DOIO}}$	
		Generate signal ENDI	ENDI = IOIO X31 + . . .	End execution sequence of instruction
		N $\nrightarrow$ L	NTL = NY4 $\overline{\text{IBX}}$ $\overline{\text{IBSV}}$ ENDI	Transfer next instruction address to L-register

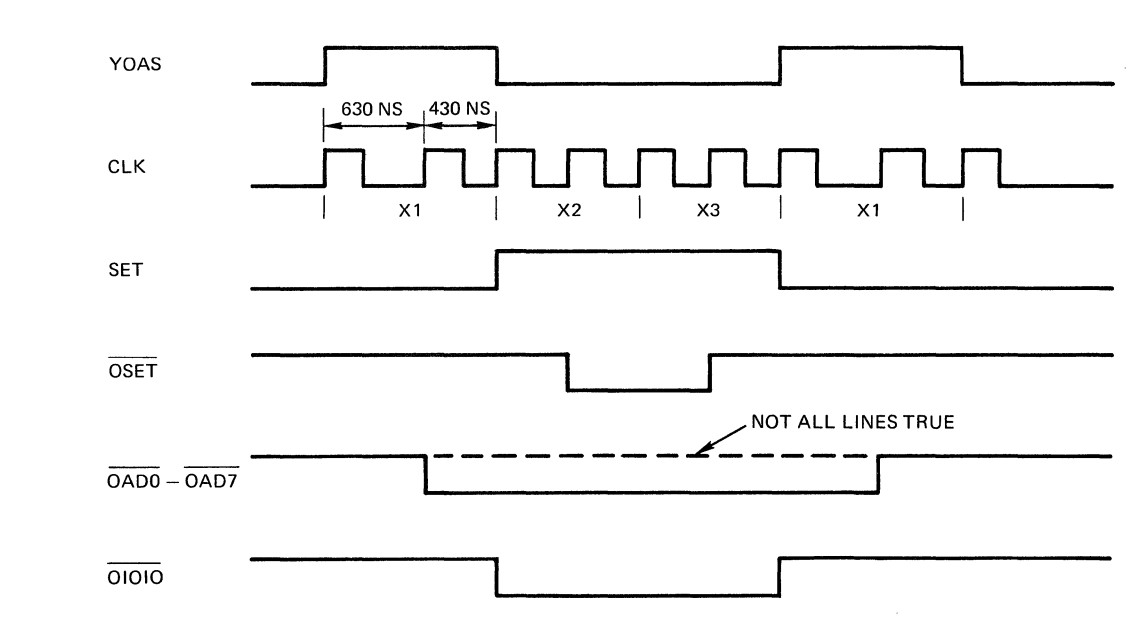

YOAS
630 NS
430 NS
CLK
X1
X2
X3
X1
SET
$\overline{\text{OSET}}$
NOT ALL LINES TRUE
$\overline{\text{OAD0}}$ – $\overline{\text{OAD7}}$
$\overline{\text{OIOIO}}$

Figure 3-42. Set (SET) Instruction, Timing Diagram

Table 3-39. Glossary of Logic Signals

Signal	Definition
0KAD	Clock reset logic signal for flip-flop KA
0KCD	Clock reset logic signal for flip-flop KC
1KA	Clock set logic signal for flip-flop KA
1KC	Clock set logic signal for flip-flop KC
1KSD	Clock set logic signal for flip-flop KS
1DOIRP	Clock set logic signal for flip-flop DOIRP
A1SW thru A6SW	Signals from control panel display thumbwheel switch associated with positions A1 through A6
AD3SW thru AD5SW	Signals from three control panel memory protect switches to the right of the master memory protect switch
ADS1 thru ADS16	Address lines to memory to specify the location being accessed
AND	Logic signal true during the logical And instruction
B0 thru B16	Stages of the B-register (binary accumulator). The 16 stages associated with the 16 bits the computer uses are B1 through B16. B1 is the most significant stage
BAET	Signal from control panel switch used to periodically simulate a power shutdown and power restart for maintenance test purposes
BBK	Instruction decode signal for Branch Back instruction
BINOVF	Binary overflow. Decoding of a situation which occurs when an arithmetic operation such as an add or subtract results in an overflow of the capacity of the 16 allowed bits
BINSIN	Binary sign. This signal is developed as a true sign of certain arithmetic operations such as addition and subtraction, and is a true sign of this operation as opposed to a sign that may be apparent as a result of an overflow situation
BLK	Instruction decode signal for Branch and Link instruction
BNO	Instruction decode signal for Branch No Overflow instruction
BPT	Instruction decode signal for Branch and Put instruction
BRSAT	Signal that comes true when conditional branches are satisfied. The conditional branches are BUC (Branch Unconditional), BNO (Branch No Overflow), BLE (Branch Less Than or Equal), BGT (Branch Greater Than), BGE (Branch Greater Than or Equal), BEQ (Branch Equal)
BSPFF	Signal from reset output of Fill flip-flop
BSPSW	Signal from FILL switch on control panel
BSW	Signal from the B position of the thumbwheel display switch on the control panel

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
BTE	Gating signal that transfers the B-register to the E-adder input bus
BTED	Option input signal to BTE
BTF	Signal that gates the transfer of the B-register onto the F-adder input
BXB	Instruction decode signal for Branch Index instruction
BZRO	True when the binary portion (bits I15 and I16) of the I-register counter is zero
C0 thru C16	Data lines to memory
CDBR	Signal that is true when a conditional branch instruction is decoded
CK2ML	A form of the 2-millisecond clock. This signal is a square wave signal that is true for 1 millisecond and false for 1 millisecond in phase with the 2-millisecond clock
CK2MS	Another form of the 2-millisecond clock. This signal is true for one memory cycle (approximately 1 microsecond) once every 2 milliseconds
CKB	Clock signal for the B-register
CKD	Clock signal for the D-register
CKI	Clock signal for I6, I7, I8
CKIB	Clock signal for I16 and I15
CKID	Clock signal for I14 and I13
CKIH	Clock signal for I0 through I15
CKIW	Clock signal for I12, I11, I10, and I9
CKL	Clock signal for the L-register
CKN	Clock signal for the N-register
CKP	Clock signal for the P-register (page register)
CKX	Clock signal for the X flip-flops
CKY	Clock signal for the YA, YB, and YC flip-flops
CLK	The 430-nanosecond system clock
CLKG	Clock signal for the G-register
CLKJ	Clock signal for the J-register
CMPSW	Compute position of a three-position switch on the control panel
CMPTL	Signal that drives the compute light on the control panel

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
CONSW	Signal from the continuous switch (CONT) on the control panel
CTI	Signal used to count the last four bits of the I-register
CTID	Signal used to count bits 13 and 14 of the I-register
CYSTL	Cycle steal signal
CYSTW	Cycle steal write signal. Used by direct memory access feature when a write into memory operation is required
D0 thru D16	Stages of the D-register. Only D1 through D16 are used by the computer
D0L thru D16L	Light drivers for control panel display indicators
DG1	Clock set logic signal for the first stage of the G-register
DLYCK	Signal that is true when a delay clock is necessary. The delay clock allows interrogation of the memory during the first clock time of the memory cycle
DOIO	Signal true during the last clock time of an I/O instruction
DOIRP	Output of interrupt flip-flop that comes true when an interrupt has been called
DOX	Control signal meaning do index, indicating that the instruction being performed is indexed
DSCRP	Signal resulting from sensing of the control panel sense switches
DSCRT	Discrete line or signal to be sensed
DSCRU	Discrete line signal from I/O device
DSFT	Signal indicating a four-bit or decimal shift
DSW	Signal from the display panel thumbhweel switch, indicating display of the D-register
DTE	Signal that gates the D-register onto the E-adder input bus
DTED	Option input signal to DTE
DVA-DVB	Instruction signal for Divide instruction
DVSF	Signal true for both the Divide instruction and the Double Length Shift instruction
DVBIT	Signal used during a Divide instruction. Normal process in the division is to develop the quotient in the 1's complement form. Term DVBIT is added to this 1's complement form to form the 2's complement
E1 thru E16	Adder input bus lines
E1A thru E16A	Adder input from E1 through E16. At this stage, the normal shifting is done, including the right and left shifting at one and four places

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
EFTF	Special signal used during the logical And instruction to obtain logical product of the E- and F-adder buses
ELB	Instruction decode signal for End Left instruction
ELD	Instruction decode signal for Double Length Shift instruction
ENDI	Signal that comes true at the end of every instruction and is true during the last clock of every instruction
ENDID	Option input signal to ENDI
EQF1 thru EQF16	The E-adder input stage and the F-adder input at that stage
ETADR	Control signal for the adder. It comes true during a normal adder operation when there is an E-input to the adder opposed to a logical And
EZRO	Signal is true when all of the E-adder input signals are false
EZRO1 thru EZRO4	Decoded signals used to make up EZRO
F1 thru F16	First stage of the F-adder input bus
F1A thru F16A	The second stage of the F-adder input bus
FYT	Fine Y timing signal
G1 thru G16	G-register stages used for fast multiplication, fast division and double length shift operations
GIDL	Timing control signal indicating go to the idle stage
GND	Ground signal
GRT1	Control signal for register test period. This signal allows writing into the selected register
GRT2	Control signal for register test and display. This timing stage is read back into the D-register for display purposes
GRT3	Signal indicating GRT1 or GRT2
GTF	Gate that transfers the G-register onto the F-adder input bus
GX10, 20, 30, 40, 50, 60, 70	Timing control signals. A forcing signal that causes the logic to jump to selected time from wherever it happens to be at the present time
GYA	Gate YA. Used to control setting YA flip-flop
GYB	Gate YB. Used to control setting YB flip-flop
GYC	Gate YC. Used to control setting YC flip-flop

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
H1 thru H4	Control signals used for shifting. During an end around shift the data that shifted off the end is stored in H1 through H4 preparatory to shifting it into the other end the next time around
HLT	Instruction decode signal for the Halt instruction
I1 thru I16	Bits of the I-register. In addition to serving as the instruction register, the lower seven bits of the I-register constitute a counter. This counter is used to control shifting, and for certain timing controls such as the multiply and divide
IATH	Decode signal for arithmetic instructions. This includes Add, Subtract, Compare, Load, Store, and logical And
IBLPK	Decode signal for a group of instructions including Branch and Link, Branch and Put, and Branch Back
IBSV	Decode of a group of instructions including Branch and Link and Branch and Put
IBX	Instruction decode signal for the Branch Index instruction
ICM	Instruction decode signal for the Compare instruction
ICS	Control signal that gates external source adder inputs into the I-register
IDONE	ID portion of the decimal counter is equal to binary 1
IDZRO	Bits 13 and 14 of the I-register equal zero
IGX10	Signal that causes the X-register to jump to 10. Similarly IGX20 and IGX30 have analogous definitions. These signals are used when passing from YOAS of the operand address select state to the normal instruction
IGYC	Instruction signals which are peculiar to the time at which the Y-counter is passed on to NY for the normal instruction implementation time
IH0 thru IH3	Decodes of the I-register used in the instruction decode. These signals are decodes of I1, and I2 and the 0, 1, 2, and 3 relate to the corresponding binary numbers generated by these two bits
ILD	Instruction decode signal for the Load instruction
ILPW	Decode signal for a group of instructions including Inclusive Or, Exclusive Or, and Load Page
ILTF	Signal that gates the lower eight bits of the I-register onto the F-adder bus
ITF	Signal that causes the enitre 16-bit I-register to be gated onto the F-input adder bus
ILTIH	Signal that causes the lower five bits of the I-register to be transferred to the upper five bits of the I-register
IM0 thru IM3	Decodes of bits I3 and I4 used during the instruction decode; the numbers relate to the binary numbers generated by bits I3 and I4

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
IMDB	Instruction decode signal for the Multiply and Divide instructions
IMSC	Miscellaneous decode instruction signals consisting of the following: Load Index, Branch Index, Load Page, Exchange Memory and Accumulator, Halt, Inclusive Or, and Exclusive Or
IN1 thru IN16	Sixteen input lines to the F-adder input from an I/O device
INHWM	Indicates inhibit write memory. Signal is true when coincidence between the memory address and memory protect switches is such that memory address is in protected area. Inhibits signal WM, write memory
INTF	Control gate for gating external signals onto the F-adder input bus
INHIRP	Signal derived from the control panel INHIBIT INTRP switch
INHPS	Inhibit power shutdown. Signal that inhibits power shutdown until completion of a fast multiply or fast divide instruction
INVF	Signal which causes the inversion of the F-adder input to occur
IOIO	Decode signal for a group of I/O instructions: PIP, POP, SET, and SNS
IP0 thru IP16	Inhibit winding termination
IR	Read current
ISFL	Control signal for shift instructions. It is true during any shift left instruction, with the exception of ELD
ISFR	Control signal for shift instruction. It is true during any shift right instruction
ISFT	Control signal for shift instruction. It is true during any shift instruction, with the exception of ELD
ISHMSC	Instruction decode for a group of short miscellaneous instructions
IST	Instruction decode signal for the Store instruction
ISUB	Instruction decode signal for Subtract instructions. The two instructions which specifically include a Subtract instruction are Subtract and Compare
ISW	Signal derived from the I-switch position of the control panel display thumbwheel
ITP	Special control signal that causes the transfer of the lower six bits of the I-register to the P-register
IW	Write current
J0 thru J16	Stages of the J-register. The J-register is a special working register used for fast Multiply, Divide, and Double Length Shift instructions
J1 thru J16	Control logic within the adder itself

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
K1 thru K16	Normal carry signals used by the adder. K17 is initial carry or carry-in stage. Used to turn the inverted F-adder input 1's complement form into a 2's complement form
KA, KB, KC	Working flip-flops for miscellaneous logic functions
KAQKC	Exclusive-Or of the KA, KC flip-flops
KK	Flip-flop used for temporary storage of add or carry when necessary
KO	Flip-flop used to store the results of an arithmetic operation. It is used to store the fact that an overflow occurred from an operation. KO is also used when an illegal division is attempted
KS	Flip-flop used to store the sign of an arithmetic operation
KU	Flip-flop used to indicate that result of operation was not equal to zero
KSTME	Control signal used in recognizing the KS logic
KTE	Control signal used to gate the K flip-flops onto the E-adder input bus
KUTME	Control signal for the KU flip-flop
L1 thru L16	Stages of the L-register. The L-register is the memory location register
LADS	Control signal indicating that the L-register is supplying the address to the memory
LADSD	Option input signal to LADS
LCH1 thru LCH8	Latches used to control the display of the latch display lights on the control panel
LCS	Control signal used to gate the output of the adder into the L-register. Its external source is destined for the L-register, such as the data switches on the control panel
LIDL	Timing control signal meaning leave idle. It comes true whenever the computer leaves the idle state
LNG	Instruction decode signal that is true whenever a long instruction is encountered
LS1	Shift control signal meaning left shift 1
LS4	Shift control signal meaning left shift 4
LSG1	Control signal used for left shift G1
LSHFT	Control signal indicating left shift
LSW	Signal from the control panel display thumbwheel when the L-register is selected for display purposes or register test
M0L thru M16L	Memory display latch outputs

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
M0 thru M16	Memory display latches combined with other memory input signals such as the diode memory
MA1 thru MA12	Memory address register
MGO	Control signal to the memory from the computer which specifies that a memory cycle is desired the next possible memory cycle time
MPA-MPB	Instruction decode signals for Multiply instruction
MPE	Memory parity error signal
MPHSW	Memory parity halt switch signal from the control panel
MPROT	Memory protect switch signal from the control panel
MPTL	Memory parity error signal to the control panel (the light driving signal)
MPTY	Flip-flop or latch signal that is holding the fact that a memory parity error has occurred
MRT	Control signal from the display thumbwheel on the control panel that indicates that a memory register test is in process
MRT14, 15, 16	Address control signals from the thumbwheel display on the control panel. These are decoded to supply the address for the memory register to be displayed. The 14, 15, and 16 result from the fact that these three bit positions correspond to the memory locations to be displayed on the 0 through 7
MTD	Special control gate allowing direct transfer of memory to D-register
MTF	Control signal allowing transfer of memory data to F-adder input bus
MTFD	Option input signal to MTF
MTI	Gate allowing direct transfer of memory into the I-register
MY5, DY6, IMD2, DY7	Invalid Y-states
N1 thru N16	Stages of the N-register. N-register is the next instruction register
NEGF	Signal generated when INVF and K17 occur, such that a true 2's complement occurs as the information is passed through the adder
NOSFT	No shift signal that comes high during a shift instruction when the total specified shift has occurred. Total specified shift is contained in the lower seven bits of the I-register. When this has been counted down to 0, NOSFT comes true, indicating shift operation is over
NSW	N position on the display panel thumbwheel. Signal is true when N register display position is selected

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Description
NTE	Signal to transfer N-register contents to the E-adder input bus
NTL	Control signal that gates the N-register into the L-register directly
NY4	Y timing stage. NY4 is the normal operand time when the instruction is actually performed
OAD0 thru OAD7	Eight address lines to I/O devices from the computer. These originate from the lower eight bits of the I-register
ODOIO	Output buffered signal indicating I/O
OKIRP	Latch signal that is true when an interrupt is allowed. This latch is set by a SET instruction
OPIP	Output signal that is the buffered version of the PIP signal
OPOP	Output signal that is the buffered version of the POP signal
OSET	Output signal that is the buffered version of the SET signal
OSNS	Output signal that is the buffered version of the SNS signal
OUT1 thru OUT16	Output lines from the computer. These originate from the B-register in the case of a programmed I/O operation or from the memory itself in the case of a direct memory transfer operation
P1 thru P6	Stages of the P-register (page register). They are used for operand address selection
PAN0 thru PAN7	Binary code of I flip-flop stages I16-I14, with I16 being the least significant stage
PASA, PASB	Carry locate adder control signals
PIN0 thru PIN16	Inputs from the control panel data switches
PIP	Instruction decode signal for Parallel Input instruction
PIPCYU	Control signal which gates the I/O data lines through the adder to their destination
PIPP	Control signal which gates the control panel data switches to the input of the adder. PIPP is also used to gate all the control panel functions that go through the adder, such as the display readback
PIPPA	Contributing signal to PIPP
PKWSW	Signal from the control panel thumbwheel for the position PK, which allows display of the page register and the indicator flip-flops, KO, KU, KS
PLUPIN	Pull-up for the input lines. Allows the UIN lines from the I/O devices to be pulled up
POK	CPU power OK signal
POP	Instruction decode signal for Parallel Output instruction

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
PRST	Power reset signal
PPRST	Control signal that combines power reset signal and reset switch signal from the control panel
PTF	Control signal that gates the P-register onto the F-adder input bus
PWRL	Control panel power light
PWRSW	Power switch signal
PWRSWR	Return for the power switch signal
RABCK	Control signal for the KA, KB, KC, and KK flip-flops. Used on the clock reset side of these flip-flops
$\overline{\text{RAS}}$	Memory timing control signal that loads the M-register with word read from memory
RLCH	Reset latch signal
RS1	Shift control signal indicating right shift one bit place
RS4	Shift control signal indicating right shift four bit places
RSG2	Special control signal for the G-register, meaning right shift G-register two bit places
RSHFT	Control signal for all right shift instructions except ELD
RSJ1	Control signal for the J-register, meaning right shift J-register one bit position
RSTJ	Unclocked reset signal for the J-register
RSTKO	Reset control signal for the Overflow, or KO, flip-flop
RSTPFF	Flip-flop that receives the switch signal from the panel reset switch
RSTYA	Reset YA signal. Control signal for the Y-state counter
RSUSO	Control signal for resetting indicator flip-flops KS, KU, and KO
RSW	Signal originating from the display panel thumbwheel. It is true when the R-register (roll-arrow register) has been selected
RT	Read timing
RTFF	Register test switch control panel flip-flop
RTSW	Signal resulting from the register test position (REG TEST) of the control panel switch
S0 thru S16	Final adder output stage outputs
SCSW	Single cycle switch signal from the control panel

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
SDM	Signal that selects the memory diode
SDMG	Select diode memory gate
SET	Instruction decode signal for the Set instruction
SFTTME	Indicates shift time and is a special control signal for shift instructions
SHIFT	Control signal for the shift instructions. True during all shift instructions except ELD
SHRT	Indicates single word instruction
SISW	Switch signal from the control panel, single instruction switch (SING INST)
SIT	Signal that selects inhibit timing
SITO	Signal that disables inhibiting circuit
SNSW1 thru SNSW8	Signals from control panel sense switches 1 through 8
SPMS	Fast multiply control signal for shift plus minus shift
SP0 thru SP17	Sense winding termination
SRAS	Sense amplifier strobe
SSPM	Fast multiply control signal for shift shift plus minus
STB	Adder output to B-register control gate
STBD	Option input signal to STB
STD	Adder output to D-register control gate
STDD	Option input signal to STD
STG	Adder output to G-register control gate
STI	Adder output to I-register control gate
STIL	Lower eight bits of the adder to the lower eight bits of the I-register control gate
STL	Adder output to L-register control gate
STN	Adder output to N-register control gate
STPKW	Adder output to page register and indicator flip-flops control gate. Also control gate to flip-flops KA, KB, KC, and KD
STPIRP	Inhibit signal for interrupts. Inhibits setting the DOIRP flip-flop
STRT	Start flip-flop. Used with signal LIDL to initiate computer operation in the compute mode.

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
STSW	Start switch signal from the control panel
STXYV	Adder output to the X-, Y-, and V-registers
SVNV	Control signal for power shutdown and power restart. Equals 7V
SZRO	Adder output consisting of all zeros
SZRO1 thru SZRO4	Decoding signals making up SZRO
T1 thru T9	Memory timing and control logic one-shots
TA	Signal that loads the memory address register
TAB	Signal that clocks ADS5 through ADS16 into the MA-register
TCM	Transfers the C0 through C16 outputs of the adder to the M-register
TP	Transfer parity signal
TTE	Control gate gating the T-register to the E-adder input bus
U2 thru U16	Adder input terms from DMA
UADS	Control gate for U2 through U16
UIN1 thru UIN16	Data input lines from I/O devices
US	Memory select signal
USES0	Signal indicating use of S0 to write into the zero-bit of memories as opposed to some other forced source of input to the zero stage of memory
V1 thru V3	Stages of the V-register. The V-register is used for timing and to supply address during power shutdown and power restart
VADJ	Memory temperature tracing signal
VADS	Control signal which allows the V flip-flops to supply the address during power shutdown and power restart
VTH	Sense amplifier voltage threshold signal
VZRO	Control signal which is true when each of the V flip-flops is false
WD00 thru WD14	Each signal selects a diode memory word
WM	Write memory signal. This term is true whenever a write into memory operation is required
WMD	Option input signal to WM
WT	Write timing signal

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
X1 thru X4	Diode memory X-coordinate signals
X1 thru X7	X timing state flip-flops. These seven flip-flops normally constitute a ring counter. X1 through X7 are further broken up into timing states. Namely, X1 is broken up into X10, X11; X2 is broken up into X20, X21, etc. Each X-state is one memory cycle in length. Each partial X-state (X10) is one clock time in length
XADSA	Address decode for address 0. Used for control purposes in loading the hardware index register
XE0 thru XE15	Outputs produced by the X-line voltage selection switches
XIR0 thru XIR7	Read current outputs produced by the X-line current selection switches
XIRP	External interrupt signal
XIW0 thru XIW7	Write current outputs produced by the X-line current selection switches
XMB	Instruction decode signal for Exchange Memory and Accumulator instruction
XODD	Means X odd. This signal is true during the odd half of one of the X-states
XRD1 thru XRD4	Outputs produced by the X-line selection switch drivers
XRT	X read timing signal
XRT01 thru XRT04	Signals disable the read section of the X-line voltage selection switches
XSW	Signal received when the control panel thumbwheel switch is set to select the X-register (index register)
XWD1 thru XWD4	Outputs produced by the X-line selection switch drivers
XWT	X write timing signal
XWT01 thru XWT04	Signals disable the write section of the X-line voltage selection switches
XYVTF	Gates the X-, Y-, and V-registers into the F-adder input
XYVSW	Selects signals from the thumbwheel on the display panel, control panel display. Selects the X-, Y-, and V-registers
Y1 thru Y4	Diode memory Y-coordinate signals
Y1S thru Y31S	Outputs produced by the Y-line drive transformers
YA, YB, YC	Timing states which select the operating mode, such as idle, normal instruction processing, operand and address. This timing state is one level coarser than the X-states; that is, the X-state times are within the Y-state times
YBGN	Y decode state for power restart
YBGNF	Contributing signal to YBGN

Table 3-39. Glossary of Logic Signals (Cont.)

Signal	Definition
YEND	Y decode state for power shutdown
YER0 thru YER7	Read voltage outputs produced by the Y-line voltage selection switches
YEW0 thru YEW7	Write voltage outputs produced by the Y-line voltage selection switches
YIDL	Y decode state for computer idle
YIR0 thru YIR31	Read current outputs produced by the Y-line current selection switches
YIW0 thru YIW31	Write current outputs produced by the Y-line current selection switches
YOAS	Y decode state for operand address select
YRI	Y-line read current
YWI	Y-line write current
Z0 thru Z2	Further timing states that are coarser than the Y-states. In general, they are power on, power off, and inhibit the Y-states for the purposes of cycle stealing
ZMGO	Control signal for memory go flip-flop. Controls memory accessing MGO signal

SECTION IV
INSTALLATION AND MAINTENANCE

4.1 GENERAL DESCRIPTION

This section consists of two parts: installation and maintenance. The first part describes installation requirements, unpacking procedures, procedures for setting up the equipment and connecting its cables, and checkout procedures. The second part of this section presents preventive and corrective maintenance procedures. Preventive maintenance includes inspection and cleaning procedures. Corrective maintenance comprises adjustment, removal, and replacement procedures.

The procedures in this chapter function as a guideline and service aid for the Field Service Engineer. However, the ultimate success of any RC 70 Computer installation depends largely upon two factors: a thorough and continuing preventive maintenance program and a comprehensive analysis of any problems which may arise out of any component failure in a unit. In order to implement both areas of maintenance, the following steps are outlined.

a. Maintenance Service Reports. The Field Service Engineer will be provided with maintenance service report forms for use in establishing a case history of each computer in his maintenance area. These records must be kept up to date in order to serve the Field Service Engineer as a useful tool in maintaining an RC 70 Computer.

1. Preventive Maintenance. By keeping an accurate, up-to-date record of preventive maintenance performed on each RC 70 Computer, the Field Service Engineer will establish a "feel" for the overall condition of any particular RC 70 unit in his maintenance area. In addition, he can observe any trends toward possible problems and take steps to prevent any major problems before they occur.

2. Corrective Maintenance. In addition to the benefits obtained by keeping accurate preventive maintenance records, the recording of all troubles which occur together with the corrections and/or repairs made will prove invaluable to the Field Service Engineer. It will expose any chronic trouble spots and enable him to make a decision as to whether further maintenance is required in these areas. Should a troublesome component become evident, he can schedule the replacement of that component during one of his regularly scheduled preventive maintenance calls.

b. Customer Logs. Coordination of activities with the customer is the keynote of a successful maintenance program. The customers who operate the computer should keep operational type logs for each individual piece of equipment on a daily basis, showing operating times, downtimes of various units, and specific trouble symptoms which have occurred. Upon entering the installation, the Field Service Engineer should consult the customer's log to ascertain any specific problems which might have been encountered with the installation's RC 70 Computer. He should endeavor to obtain all of the available pertinent information from the customer's log. This action will minimize the time required to locate the trouble. Full use should be made of this manual and the schematics and troubleshooting charts contained in it to try to determine the problem before actually taking the unit away from the customer for service. By application of a cooperative effort on the part of the Field Service Engineer, full maintenance of his equipment will be realized by the customer, and a resulting confidence in the Field Service Engineer will be generated.

4.2 INSTALLATION

Subsequent paragraphs contain the information necessary to install, to set up, and to check out the RC 70 Computer.

4.3 SPACE REQUIREMENTS

The RC 70 Computer, including its self-contained power supplies, can be mounted in a standard 19-inch rack. Its dimensions are 19 inches wide, 19 inches high, and 19 inches deep. There should be sufficient area around the machine for free air flow.

4.4 SPECIAL TOOLS

The tools required to mount the computer are a standard slot screwdriver, a Phillips screwdriver, and a crescent wrench. No other tools are required.

4.5 ENVIRONMENTAL REQUIREMENTS

The immediate enclosure of the computer, whether a building or an office, should be climatically controlled to provide the necessary environment. The temperature requirement for the computer is 0^{o} to 55^{o} Celsius (32^{o} to 131^{o} Fahrenheit). The relative humidity requirement is 0 to 90 percent.

4.6 UNPACKING PROCEDURE

The following procedure is provided for unpacking the RC 70 Computer in preparation for installation.

a. Read the shipping notice. It identifies the customer, indicates the purchase order number, and lists all the components that have been shipped to the customer.

b. Refer to figure 4-1. Remove the three straps that secure the computer to the wooden pallet.

c. Remove the polyethylene plastic cover that encloses the RC 70 Computer. Thoroughly inspect the equipment to determine if any damage has occurred during shipment. If there is damage, a claim should be filed with the carrier. A full report of the damage should be forwarded to:

Manager of Customer Service
REDCOR Corporation

The manager of Customer Service will arrange the necessary disposition of the equipment.

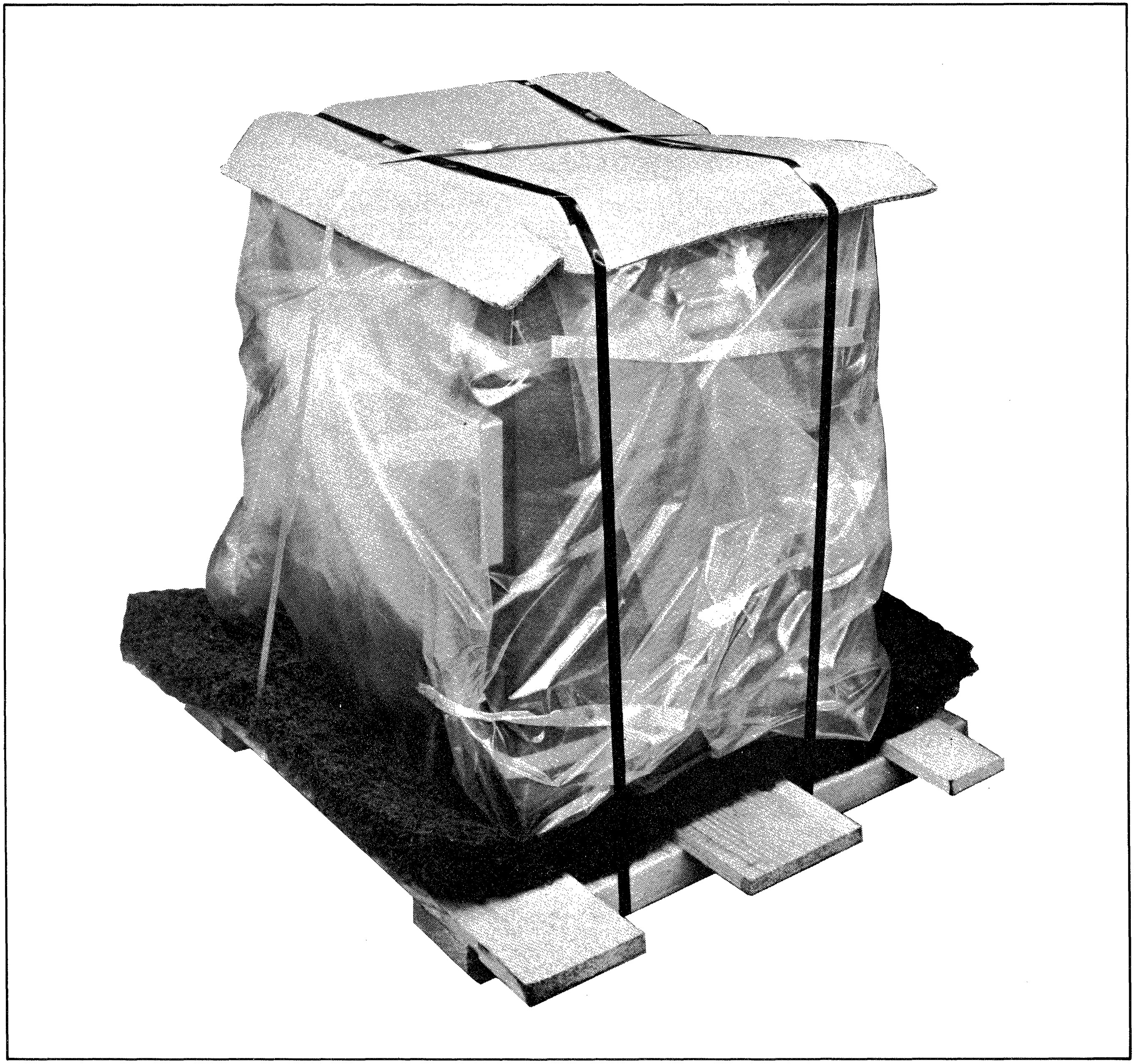


Figure 4-1. RC 70 Computer Shipment Configuration

d. Using the handhold cutouts in the plywood side panels, remove the RC 70 Computer from the wooden pallet. Place the machine on a work bench or in a clean area.

CAUTION

> The computer weighs approximately 90 lb. It should be handled by two men.

e. Refer to figure 4-2. Remove the two screws, located on the rear of the computer, that hold the power supply assembly in place. Using a crescent wrench, loosen the nut on the screw located on the right-hand side of the power supply assembly.

f. Grasp the bottom of the power supply assembly and slide it back (see figure 4-3). Pivot the power supply assembly on its hinge (figure 4-4) so that there is complete access to all of the machine's printed circuit card assemblies.

g. Remove any foam rubber or cardboard that is packed in the printed circuit card area.

h. Remove any fiberglass tape that is used to hold the card assemblies in place.

i. Visually check the condition of the card assemblies. They should not be physically dislodged from their slots.

j. Inspect the interior of the computer assembly to ensure that all mounted components are mechanically secure.

k. Inspect each fan assembly for damage or obstructions. Move the blades of each fan to ensure that they move freely.

l. Visually inspect the power supply assembly to ensure that none of the wiring is broken or that none of the connectors is damaged. Any damage should be reported to the manager of Customer Service.

m. Pivot the power supply assembly on its hinge and slide it back into its proper place.

n. Replace the screws that were removed in step e of this procedure. Tighten the nut that was loosened in step e.

o. Inspect the memory card assemblies. They are located on the left-hand side of the computer, when viewing the machine from the rear. Ensure that each module is seated properly in its slot. Any damage should be reported.

p. Check the mechanical operation of the control panel switches. There should be no mechanical hindrance to the movement of any of the switches.

q. The machine is packed using two plywood side panels. These side panels support the computer in a vertical position. Do not remove the plywood side panels and place the computer down vertically because the two fans mounted on the bottom of the unit will be damaged. Three men are required to install the computer in a 19-inch rack: two men to support the computer, while a third removes the eight bolts that secure the side panels to the machine. With the side panels removed, two men can hold the computer in place, and the third man can bolt the unit into the 19-inch rack.

4.7 SETUP PROCEDURE

When ordered, a Teletype unit is shipped to the customer with the RC 70 Computer. The type of unit purchased can be an ASR-33 Teletype, an ASR-35 Teletype, or an IBM Selectric typewriter. Each of these input/output devices has an interface card assembly, which is housed inside the device cabinet. The ASR-33 and ASR-35 Teletypes each have the interface card mounted on the inside of the pedestal which supports the keyboard assembly.

There are five toggle switches mounted on the interface card. These switches are used for addressing the Teletype. During the following procedure, it is assumed that an ASR-33 Teletype has been shipped with the computer. After both the Teletype* and the RC 70 Computer have been unpacked, proceed with the setup procedure.

a. To gain access to the interface card, remove the two sheet metal screws that hold the rear panel of the Teletype in place.

b. Refer to figure 4-5. The interface card is mounted within the unit. The switches are located on the left-hand side of the card. The switch located to the extreme right is designated switch 0, the next switch is designated switch 1, and so on. These switches must be placed in a different position when the RC 70 diagnostic program is performed and when the same program is loaded into the machine. (A loading procedure is presented in step m of this paragraph.)

c. There are three Teletype cables associated with the interface card (see figure 4-5). Each cable consists of a twisted pair of color-coded wires. In addition, each cable connector has a key to ensure that the cable cannot be incorrectly connected to the interface card. Connect these cables to the interface card as follows:

1. Connect the cable with the red and white wires to the connector located at the top right-hand side of the interface card.

*Unpack the ASR-33 Teletype in accordance with the unpacking procedure given in the vendor's manual.

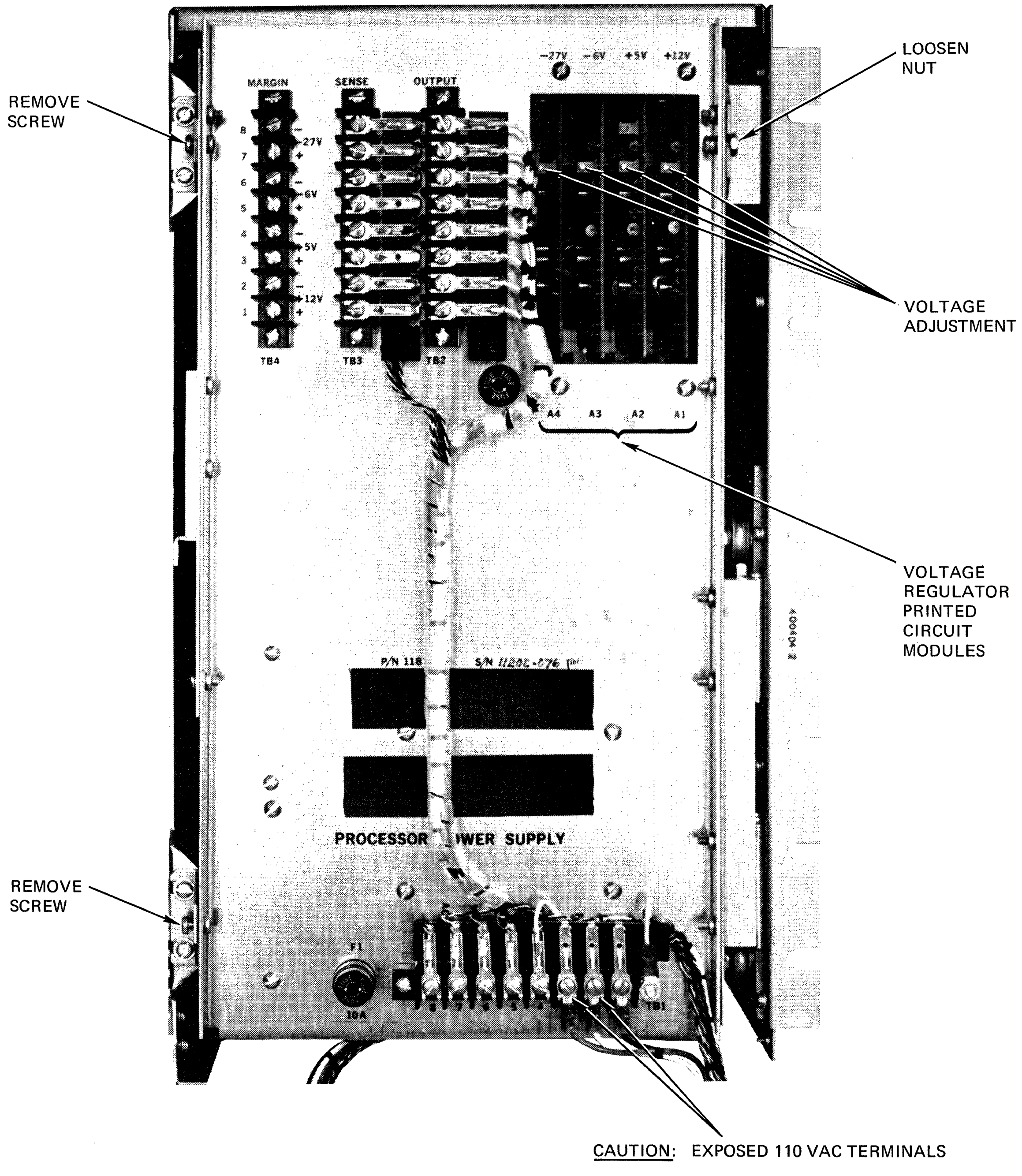
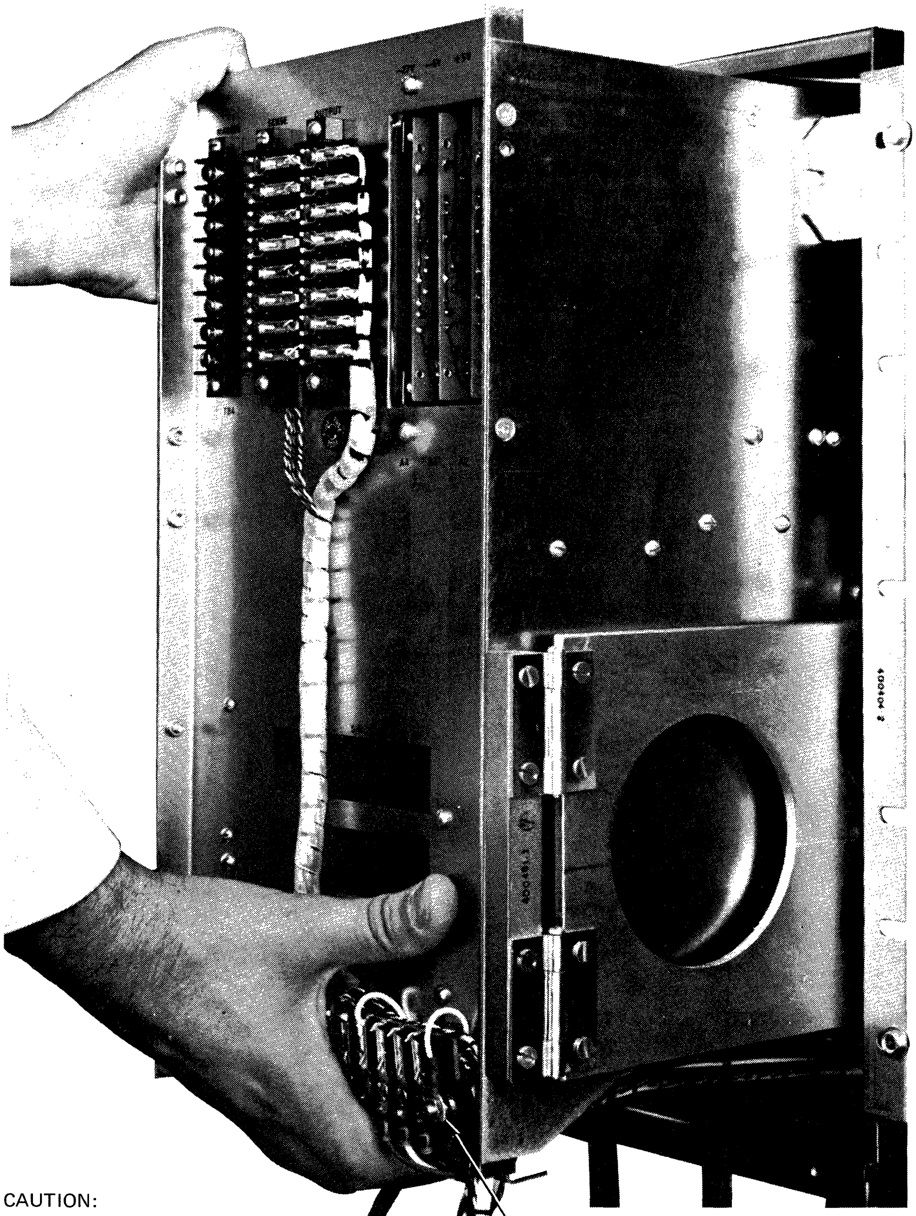


Figure 4-2. Power Supply Assembly, Rear View



CAUTION:
REMOVE AC POWER CABLE FROM AC POWER SOURCE BEFORE SLIDING THE POWER SUPPLY ASSEMBLY OUT OF THE COMPUTER ASSEMBLY

110 VAC TERMINALS

Figure 4-3. Sliding the Power Supply Assembly Out of the Card Cage Assembly

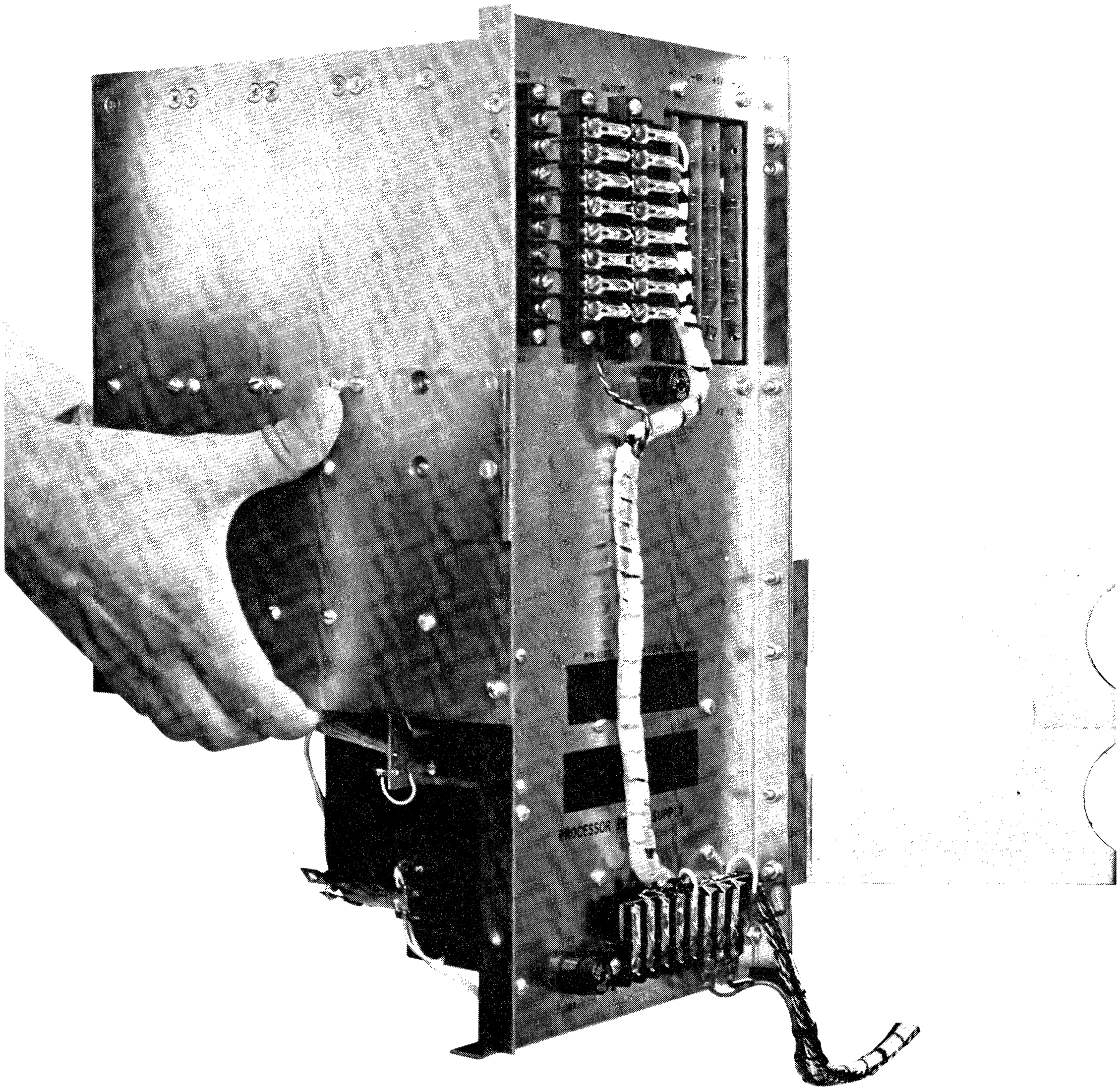


Figure 4-4. Pivoting the Power Supply Assembly on Its Hinge

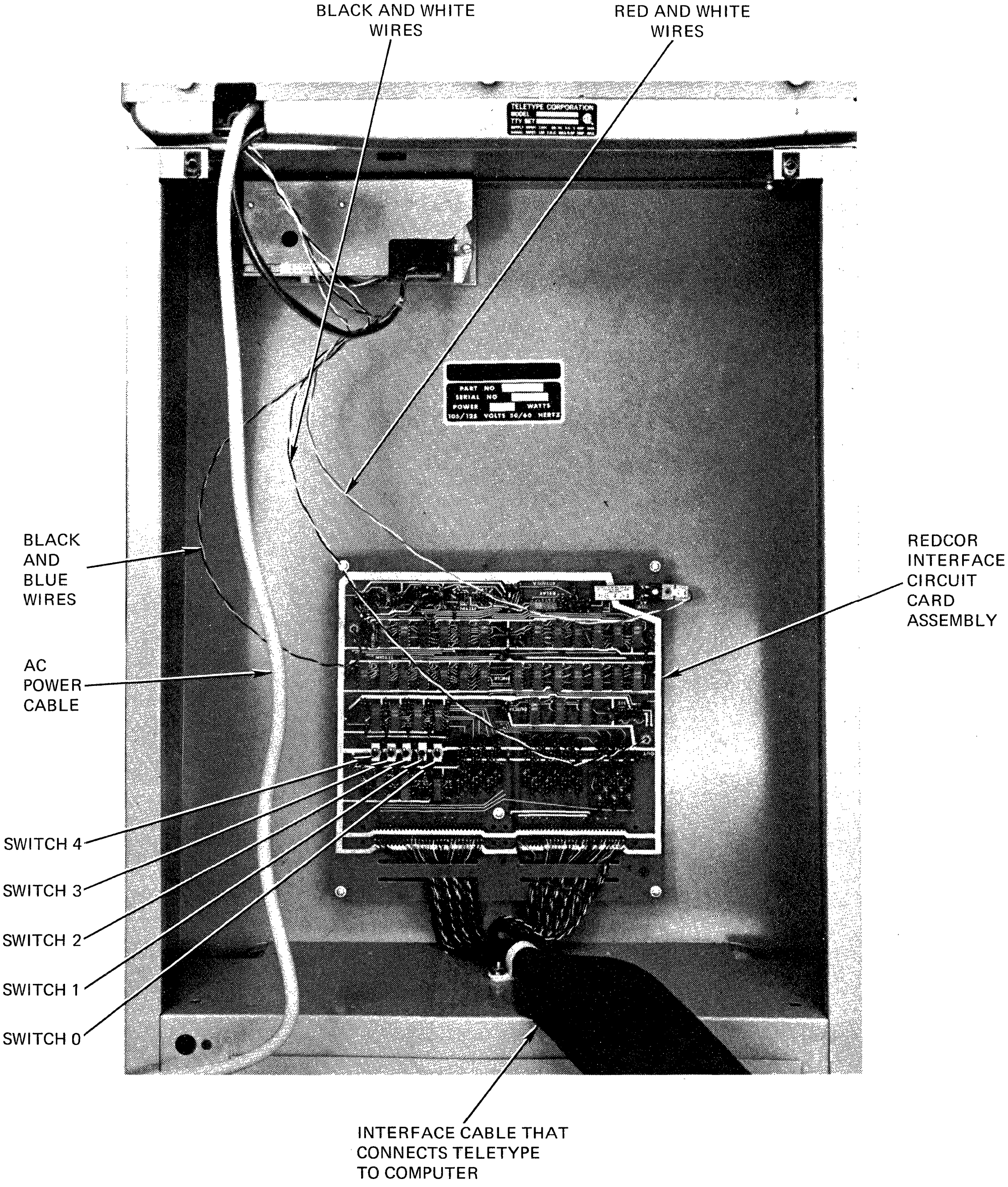


Figure 4-5. ASR-33 Teletype Interface Card Location

2. Connect the cable with the black and white wires to the connector located directly below the connector mentioned in step 1.

3. Connect the cable with the black and blue wires to the connector located at the top left-hand side of the interface card.

d. Set the LINE/OFF/LOCAL switch on the Teletype control panel to OFF (see figure 4-6).

e. Press the paper tape punch OFF pushbutton switch on the Teletype control panel.

f. Bring the interface cable and the Teletype power cable out through the rear of the unit.

g. Refer to figure 4-7. Connect the interface cable to either of the computer's I/O connectors (J69 or J70).

h. Connect the Teletype and RC 70 Computer power cables to 117V ac power sources.

i. Set the LINE/OFF/LOCAL switch on the Teletype control panel to LINE.

j. Press the POWER pushbutton switch on the computer control panel. The switch indicator goes on. Pressing this switch applies power to both the RC 70 Computer and the ASR-33 Teletype.

k. Normally the computer is shipped to the customer with the RC 70 diagnostic program loaded in memory. If the RC 70 diagnostic program is stored in memory, set the switches on the interface card as follows:

Switch 0: up

Switch 1: down

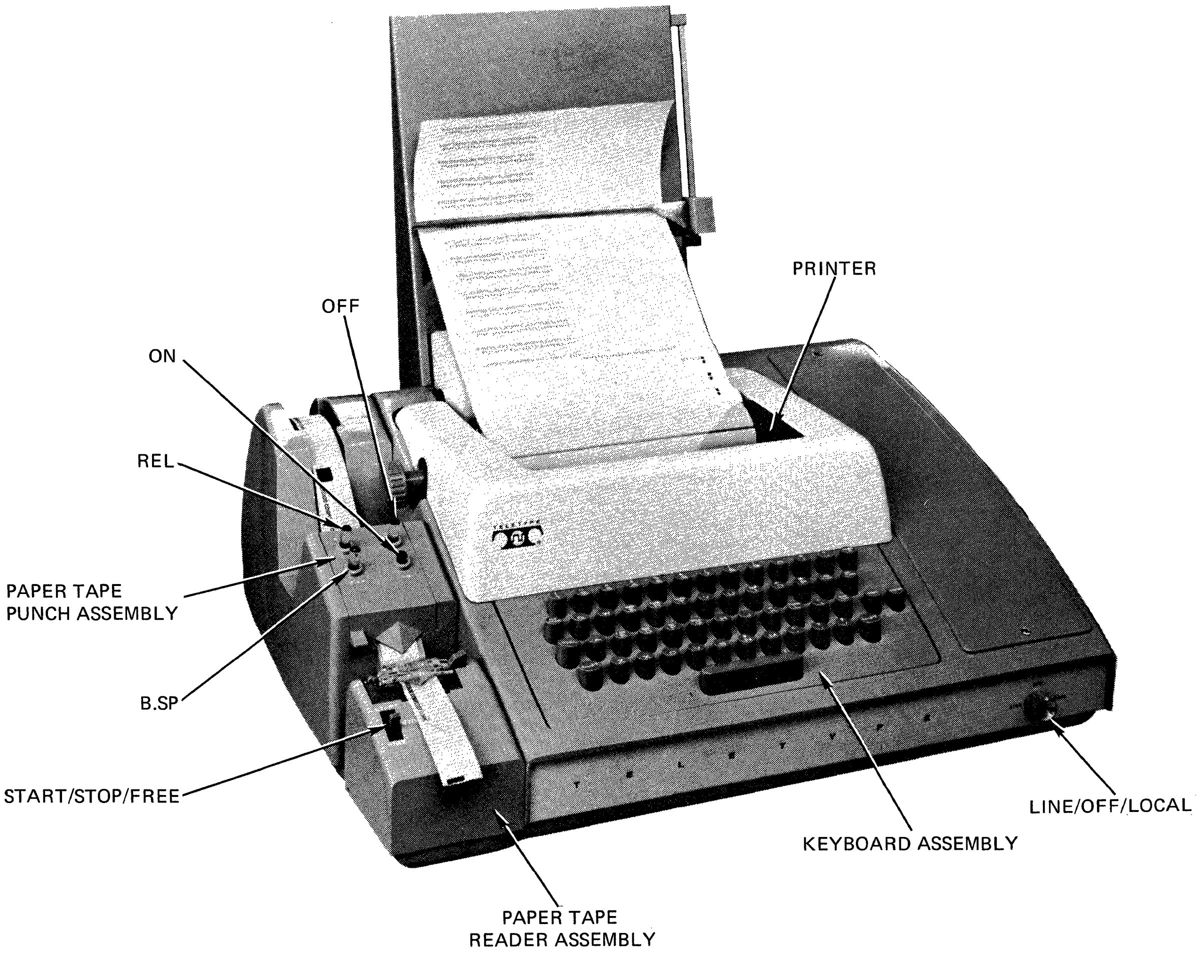


Figure 4-6. ASR-33 Teletype Control Panel

Switch 2: up

Switch 3: up

Switch 4: up

l. Replace the rear panel of the Teletype.

m. If the RC 70 diagnostic program is not resident in memory, load the program as follows:

1. Remove the rear panel of the Teletype, and place the switches on the interface module in the following positions:

Switch 0: up

Switch 1: up

Switch 2: up

Switch 3: down

Switch 4: up

2. Replace the rear panel of the Teletype.

3. Load the RC 70 diagnostic program paper tape into the paper tape reader.

4. Set the START/STOP/FREE switch on the Teletype control panel to START.

5. Position the diagnostic program tape so that approximately 6 to 10 inches of blank leader tape occur before the first loader data bit.

6. Place each of the data switches on the processor control panel in the lower position, except switches 12, 13, and 16. These switches are placed in the upper position.

7. Set the register select switch on the processor control panel to L.

8. Press the RESET pushbutton switch on the processor control panel.

9. Press the FILL pushbutton switch on the processor control panel. The C indicator on the processor control panel goes on.

10. Set the STOP/START/FREE switch on the Teletype control panel to START. The paper tape reader should read all of the data on the tape into memory. When the reader reads the first character on the tape, the register display follows a binary count. A halt should not occur until all the data on the tape has been processed by the computer. If a halt does occur, a checksum error is the probable cause. Therefore, reload the diagnostic program paper tape into the paper tape reader, and repeat this procedure.

11. If the paper tape reader fails to operate, perform steps m11 through m16 of this procedure. If the paper tape reader is functioning in a normal manner, continue to step n.

12. Place each of the data switches on the processor control panel in the lower position, except switches 1, 2, 4, 7, and 12. These switches are placed in the upper position.

13. Set the function switch on the processor control panel to SING INST.

14. Press the SINGLE CYCLE pushbutton switch on the processor control panel.

15. Press the START pushbutton switch on the processor control panel. The I/O bus is now reset.

16. Press the SINGLE CYCLE pushbutton switch on the processor control panel. The machine is now operating in the compute mode.

17. Repeat steps m6 through m10 of this procedure.

n. Note the reading on the ELAPSED TIME indicator for future reference.

4.8 CHECKOUT PROCEDURES

The computer checkout procedure is given below. Unless otherwise indicated, all the controls in this procedure are located on the processor control panel.

a. Set up the computer and the ASR-33 Teletype in accordance with the procedure given in paragraph 4.7.

b. Set the LINE/OFF/LOCAL switch on the Teletype control panel (figure 4-5) to LINE. The Teletype activates.

c. Place each of the eight sense switches in the lower position. These toggle switches are labeled 1 through 8.

d. Set the register select switch to L.

e. Set the function switch to COMPUTE.

f. Place the CONT toggle switch in the lower position.

g. Place the PROTECT MEM toggle switch in the lower position.

h. Place each of the BLOCK toggle switches in the lower position.

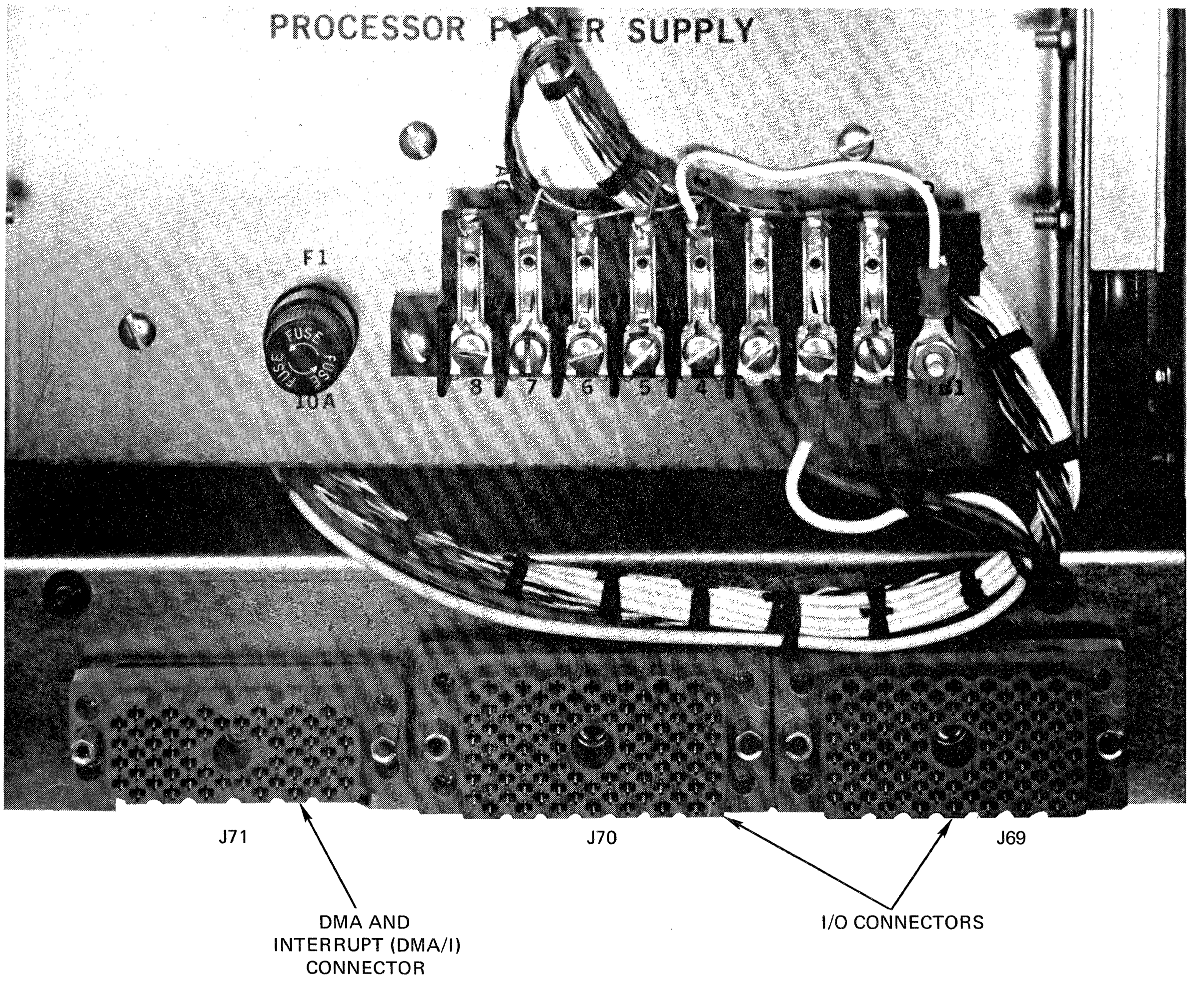


Figure 4-7. I/O Connectors

i. Place the INHIBIT INTRP toggle switch in the lower position.

j. Place the PARITY HLT toggle switch in the lower position.

k. Place the B-E TEST toggle switch in the lower position.

l. Place each of the data toggle switches, labeled 0 through 16, in the lower position, except data switch 5. It is placed in the upper position.

m. Press the RESET pushbutton switch. The contents of the data switches, representing instruction X'0800' are loaded into the L-register.

n. Press the START pushbutton. Since the diagnostic program is resident at location X'0800', the program is executed by the machine. At the same time, the C indicator goes on, indicating that the computer is functioning normally.

Note

> The status of the register display indicators are meaningful only when the computer is in the idle mode of operation.

o. Permit the computer to operate for two minutes. If the C indicator does not go out, proceed to the next step. However, if the C indicator does go out, refer to the RC 70 diagnostic troubleshooting procedure in section V.

p. Place sense switch 1 in the upper position. The computer halts, and the C indicator goes out.

q. Place sense switch 3 in the upper position.

r. Place sense switch 6 in the upper position.

s. Press the ON pushbutton switch on the paper tape punch control panel.

t. Place sense switch 1 in the lower position.

u. Press the START pushbutton switch. The paper tape punch activates. Data, which is being read out of memory, is punched into the tape and recorded by the Teletype.

v. Place sense switch 1 in the upper position. The diagnostic program halts at the end of one type cycle.

w. Place sense switch 1 in the lower position.

x. Place sense switch 7 in the upper position.

y. Load the paper tape, which has just been punched, into the paper tape reader. The first punched character should be directly over the reader pins.

z. Press the RESET pushbutton switch.

aa. Set the register select switch to L.

ab. Place data switch 5 in the upper position. Place each of the remaining data switches in the lower position.

ac. Press the START pushbutton switch.

ad. Place the START/STOP/FREE switch on the Teletype control panel in the START position.

ae. The paper tape reader reads the first character on the tape. The data is transferred into memory, where it is compared to data in a table. After the comparison is made, the computer outputs the character just read to the Teletype. This sequence is repeated as each character is read by the paper tape reader and continues to be read until an error occurs or until sense switch 1 is placed in the upper position.

af. If there is a difference in the data stored in the tape and the data recorded by the Teletype, the diagnostic program halts, and X'C9' is displayed by the register display. (The 8 and the 4 indicators in the extreme left-hand column of the register display are on. The 8 and the 1 indicators in the next column to the right are on. All of the remaining indicators are out.) This error message is transmitted to the Field Service Engineer if the register select switch is in the I position.

4.9 PREVENTIVE MAINTENANCE

Preventive maintenance consists of procedures to check out the electronic and mechanical operation of the RC 70 Computer. Preventive maintenance for the electronic circuits includes periodically checking the computer clock, running the RC 70 diagnostic program, and checking the voltage levels of the power supplies at the computer backplane. The fans and the control panel switches are checked mechanically. In addition, the machine should be cleaned at regular intervals.

It is recommended that a log showing the power supply voltages and the pulse widths of the one-shot circuits be maintained. Such a log may prove helpful in anticipating or locating malfunctions.

4.10 CHECKING THE COMPUTER CLOCK

The computer clock should be checked once every three months. This check ensures that the switching times of the clock are not drifting. To check the operation of the computer clock, perform the following procedure:

a. Place the horizontal sweep control on the oscilloscope in the .1 μs position.

b. Place the vertical gain control on the oscilloscope in the 2 volts position.

c. Press the POWER pushbutton switch on the processor control panel. The switch indicator lights.

d. Place each of the toggle switches on the processor control panel in the lower position.

e. Set the function switch on the processor control panel to COMPUTE.

f. Press the RESET pushbutton switch on the processor control panel.

g. Connect the sync probe of the oscilloscope to test jack X1 on the computer control panel. Use an externally or internally generated negative trigger.

h. Connect the test probe of the oscilloscope to test jacks X2 through X7 and to test jack CLK. Observe the waveforms shown in figure 4-8.

Note

The time values of the waveforms shown in figure 4-8 may vary from machine to machine. These variations occur because the computer clock is RC time constant controlled, not crystal controlled.

4.11 RUNNING THE RC 70 DIAGNOSTIC PROGRAM

The RC 70 diagnostic program should be performed once every three months. Executing the program ensures that the machine's input/output unit, arithmetic unit, control unit, and memory are operating normally. The program also ensures that the switches on the processor control panel are operational. Inspect the toggle switches; they should not be bent or loose. Also check the other switching assemblies on the computer control panel. They should be mechanically functional and secure.

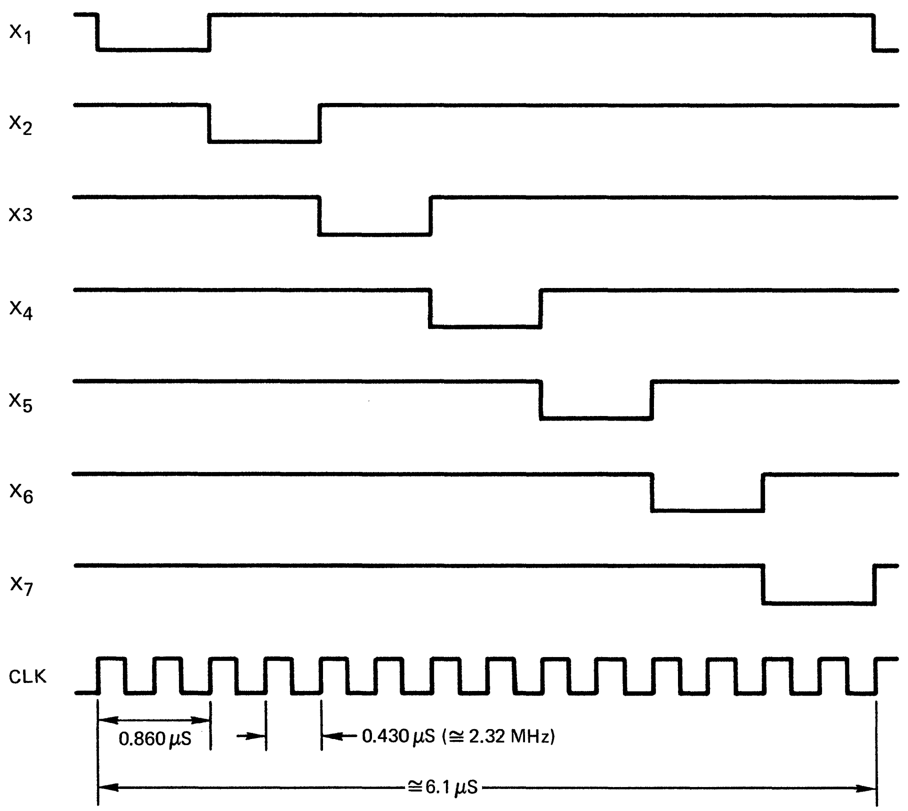


Figure 4-8. Computer Clock Signals

4.12 CHECKING THE POWER SUPPLY ASSEMBLY

The power supply assembly, consisting of four power supplies, should be checked once every three months and before the Field Service Engineer troubleshoots the computer. All power supply outputs must be within ±5% of the values specified in the following procedure:

a. Place the black probe of a VOM (volt-ohmmeter) on pin 79 of connector J3. This connector, as well as the other connectors mentioned in this procedure, is located on the backplane of the computer (figure 4-9).

b. Place the red probe of the VOM on pin 40 of J3. Observe a meter reading of +5 (±0.25)V.

c. Place the red probe of the VOM on pin 71 of J3. Observe a meter reading of +12 (±0.60)V.

d. Place the red probe of the VOM on pin 79 of J3.

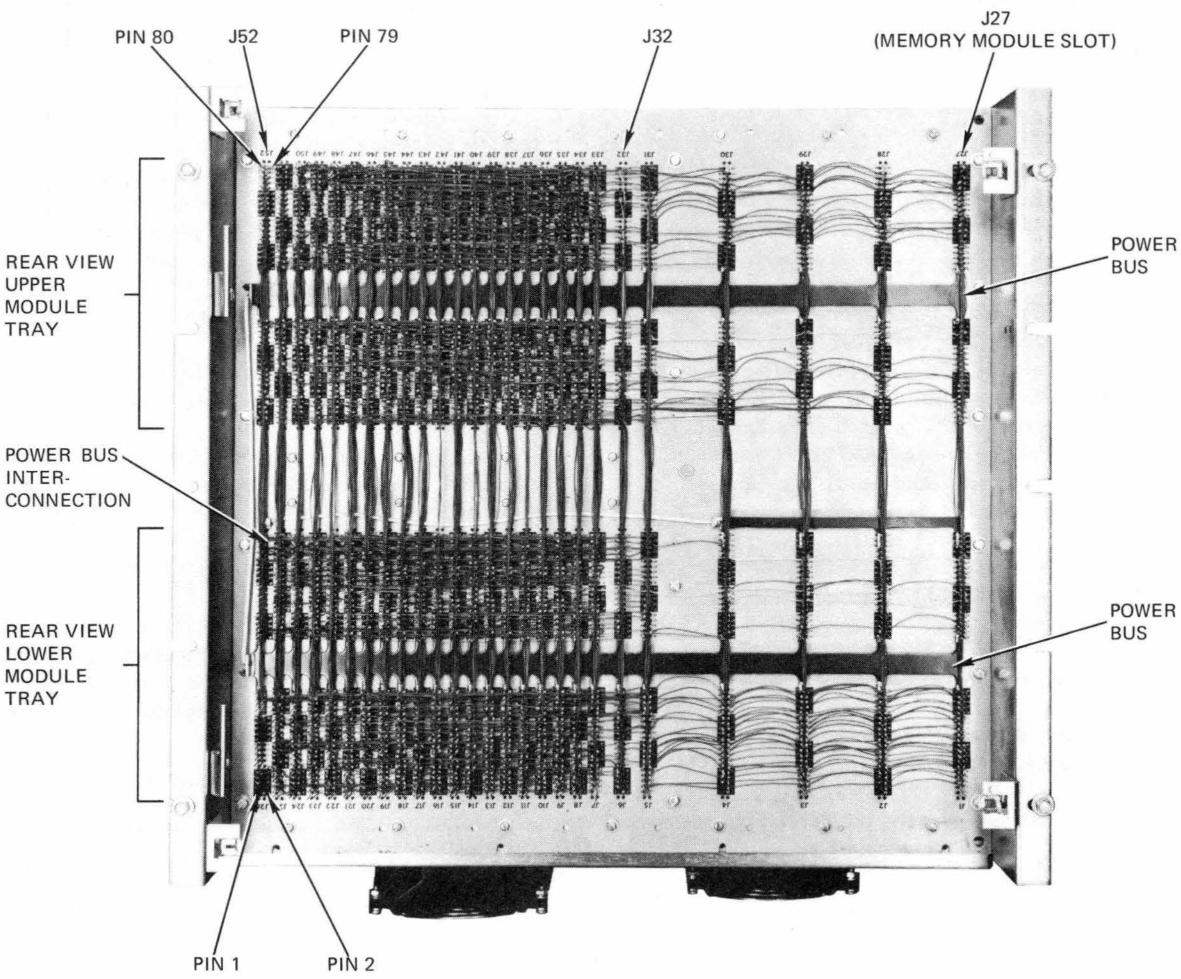


Figure 4-9. Computer Backplane

e. Place the black probe of the VOM on pin 55 of J3. Observe a meter reading of -6 (±0.30)V.

f. Place the black probe of the VOM on pin 78 of J4. Observe a meter reading of -27 (±1.35)V.

Note

> The voltage measured in step f may vary a volt or two because the voltage is temperature compensated. The -27V power supply accomplishes this control using diodes as temperature sensors.

4.13 CHECKING THE FANS

There are two fans mounted on the bottom of the computer assembly. These fans should be checked once every six months to ensure that they function properly.

4.14 CLEANING

In order to reduce the cost of repairs and keep computer downtime to a minimum, the general area in which the computer is operated should be kept clean, neat, and orderly.

It is recommended that good housekeeping regulations be set up and observed. For example, once every six months vacuum and blow out the dirt collected by the fans. If the environment is not clean, dirt collects on the upper and lower row of the printed circuit modules, blocking the flow of air. The reduced airflow causes excessive heat in the circuit modules. The heat, in turn, reduces component life.

Doors and panels should be kept closed or secured. Dust covers, when available, should be used on the equipment when it is not in service.

4.15 LUBRICATION

Since there are very few moving parts in the computer, no special lubrication procedures are required. The fans have sealed bearings. All hinges, rollers, pins, and similar type hardware may be lubricated using any lightweight oil.

4.16 CORRECTIVE MAINTENANCE

Subsequent paragraphs provide adjustment, removal and replacement, and repair procedures.

4.17 POWER SUPPLY VOLTAGE ADJUSTMENTS

All the power supply voltage adjustments are made on the voltage regulator printed circuit cards. These cards are housed in the power supply assembly (see figure 4-2). There are three types of voltage regulator cards: The first type has three potentiometers; the second type has two potentiometers; and the third type has one potentiometer. Each control is accessible from the front of the power supply assembly.

The number of potentiometers that exist on a card does not affect the manner in which the adjustment procedure is performed. If the voltage regulator card has three controls, use the middle potentiometer during the procedure. If the card has two potentiometers, use the top one. If there is only one potentiometer on the card, that is the control that is adjusted during the procedure.

To adjust the outputs of the power supply assembly, perform the following procedure:

a. Press the POWER pushbutton switch on the processor control panel. The switch indicator goes on. Permit the machine to operate for 30 minutes so that the operating temperature of the computer stabilizes.

b. Place the red and black probes of the VOM on the pin locations listed in table 4-1. All voltage measurements are taken at the computer backplane (figure 4-9). This action eliminates the need to compute the voltage drop between the output of the power supply and the backplane connector.

Note

> If a digital voltmeter is available, use it for all voltage measurements.

c. Adjust the proper control on the proper voltage regulator module, obtaining the voltages listed in table 4-1.

4.18 COMPUTER CLOCK ADJUSTMENT

The adjustments made in the following procedure are made only when a defective component has been replaced in the computer clock circuit or when the clock drifts out of tolerance. All measurements and adjustments

Table 4-1. Voltage Regulator Adjustments

Slot Designator	Black Probe	Red Probe	Voltage Reading
A1	J3, Pin 79*	J3, Pin 71	+12 (±0.60)V
A2	J3, Pin 79	J30, Pin 40	+5 (±0.25)V
A3	J3, Pin 55	J3, Pin 79	-6 (±0.30)V
A4	J4, Pin 78	J3, Pin 79	-27 (±1.35)V

*This pin is ground.

are made on the computer clock printed circuit card (see figure 4-10). The card is housed in slot J32 (533). To adjust the computer clock, proceed as follows:

a. Press the POWER pushbutton switch on the computer control panel. The switch indicator is off.

b. Disconnect the computer ac power cable from the 117V ac power source.

c. Remove the two screws that hold the power supply assembly in place. These screws are located on the rear of the processor assembly (figure 4-2). Using a crescent wrench, loosen the nut on the screw that is located on the right-hand side of the power supply assembly.

d. Grasp the bottom of the power supply assembly and slide it back (figure 4-3). Pivot the power supply assembly on its hinge (figure 4-4) so there is complete access to all of the computer's card assemblies.

e. Remove the computer clock printed circuit card from slot J32.

f. Insert a card extender assembly into slot J32.

g. Connect the computer clock circuit card to the card extender.

h. Place each of the toggle switches on the processor control panel in the lower position.

i. Set the function switch on the processor control panel to COMPUTE.

j. Connect the computer ac power cable to the 117V ac power source, and press the POWER pushbutton switch on the processor control panel. The switch indicator goes on.

k. Press the RESET pushbutton switch on the processor control panel.

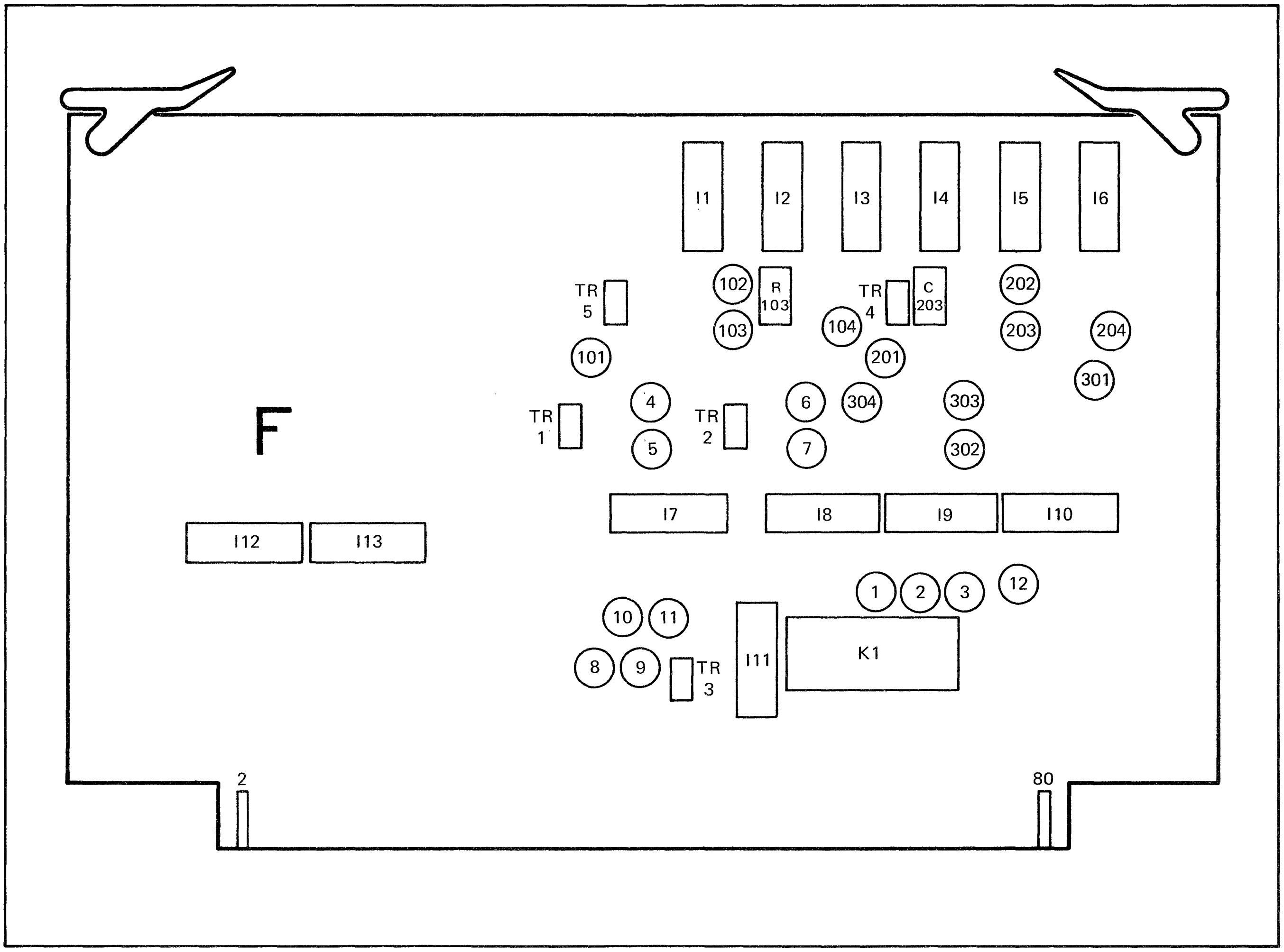


Figure 4-10. Computer Clock Printed Circuit Module

l. Place the horizontal sweep control on the oscilloscope in the .1 µs position.

m. Place the vertical gain control on the oscilloscope in the 2 volts position.

n. Connect the input probe of the oscilloscope to pin 6 of component I9.

o. Adjust potentiometers TR4 and TR5 to obtain the waveshape shown in figure 4-11.

p. Remove the input probe of the oscilloscope from pin 6 of component I9.

q. Press the POWER pushbutton switch on the processor control panel. The switch indicator goes out.

r. Disconnect the computer power cable from the 117V ac power source.

s. Remove the computer clock card assembly from the extender, and remove the extender from the slot.

t. Insert the computer clock card into slot J32.

u. Pivot the power supply assembly on its hinge and slide the assembly back into its proper position.

v. Replace the screws that were removed in step c of this procedure. Tighten the nut that was loosened in step c.

4.19 ONE- AND TWO-MILLISECOND CLOCK ADJUSTMENT

The following adjustments are made only when a defective component has been replaced in the 1- and 2-millisecond clock circuit or when the clock drifts out of tolerance. The clock has a ±20% tolerance.

To adjust the 1- and 2-millisecond clock, perform the following procedure:

a. Perform steps a through k of the procedure given in paragraph 4.18.

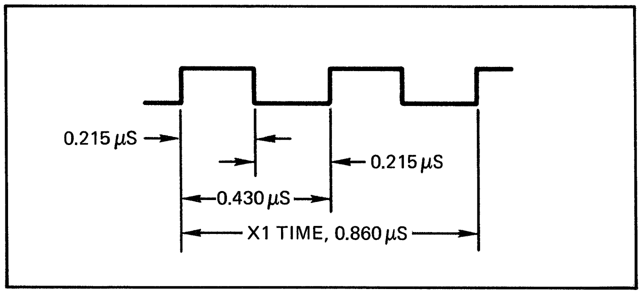


Figure 4-11. Computer Clock Output

b. Place the sweep control of the oscilloscope in the .2 µs position.

c. Place the vertical gain control of the oscilloscope in the 2 volts position.

d. Connect the input probe of the oscilloscope to pin 3 of component I3 (see figure 4-10).

e. Adjust potentiometers TR1 and TR2 to obtain the waveshape shown in figure 4-12.

f. Remove the input probe of the oscilloscope from pin 3 of component I3 and connect it to pin 3 of component I12.

g. Adjust potentiometers TR1 and TR2 to obtain the symmetrical waveshape shown in figure 4-13.

h. Remove the input probe of the oscilloscope from pin 3 of component I12.

i. Since potentiometers TR1 and TR2 are both used to obtain CK1M and CK2M, repeat steps d through g of this procedure to ensure that the clocks meet the desired requirements.

j. Perform steps q through v of the procedure given in paragraph 4.18.

4.20 POWER FAIL-SAFE ADJUSTMENT

The following procedure is provided to adjust the power fail-safe circuit on the computer clock card assembly.

a. Disconnect the computer ac power cable from the 117V ac power source.

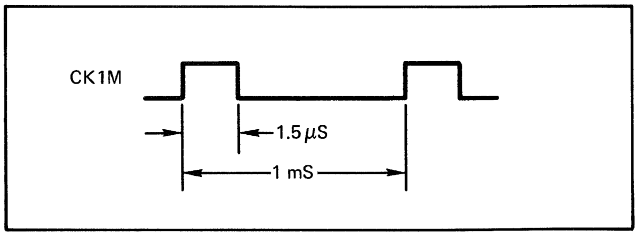


Figure 4-12. One-Millisecond Clock

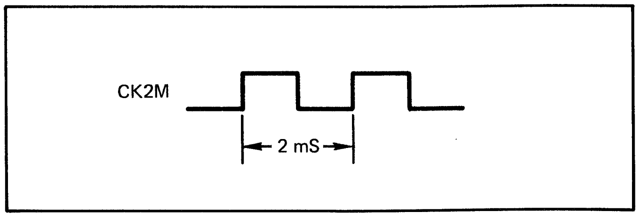


Figure 4-13. Two-Millisecond Clock

b. Connect a Variac to the 117V ac power source.

c. Rotate the voltage selector control on the Variac to the 117V ac position.

d. Connect the computer ac power cable to the Variac.

e. Connect an ac voltmeter across the 117V ac power source.

f. Perform steps a through g of the procedure given in paragraph 4.18, except disconnect the computer ac power cable from the Variac.

g. Connect the computer ac power cable to the Variac.

h. Connect the red probe of a VOM to pin 79 of connector J32. The connector is located on the computer backplane (figure 4-9).

i. Connect the black probe of the VOM to pin 14 of J32.

j. Observe a positive reading on the VOM.

k. Using the voltage selector control on the Variac, slowly decrease the Variac's output. When this voltage reaches a potential of 90V ac, a 0V dc reading should be observed on the VOM. If this reading is not obtained when the output of the Variac is 90V ac, adjust TR3 until it is.

l. Disconnect the ac voltmeter from the 117V ac power source.

m. Disconnect the VOM probes from pins 79 and 14 of J32.

n. Press the POWER pushbutton switch on the computer control panel. The switch indicator goes off.

o. Disconnect the ac power cable of the computer from the Variac and disconnect the Variac's power cord from the 117V ac power source.

p. Perform steps s through v of the procedure given in paragraph 4.18.

4.21 REPLACEMENT OF DEFECTIVE POWER SUPPLY SEMICONDUCTORS

The following procedure is provided to remove and replace a defective semiconductor on the power supply assembly:

a. Perform steps a through d of the procedure given in paragraph 4.18.

b. Remove the six Phillips screws and two standard single slot screws that secure the rear plate to the power supply assembly. The rear plate is opposite the side of the power supply assembly that has the terminal boards (figure 4-2).

c. To remove a heat sink which is located adjacent to the rear plate, remove the two outside screws (figure 4-14) using a Phillips screwdriver and a crescent wrench.

Note

> The plastic spacers are used as electrical insulators. They should be handled carefully.

d. Slide the heat sink out of the power supply assembly, and remove the defective transistor.

e. To remove a heat sink located in the interior of the power supply assembly, remove the right-hand plate of the assembly (see figure 4-15). This step requires that one man hold the side plate, while another man removes the five screws that secure the plate to the power supply assembly (figure 4-14).

Note

> Once the side plate has been removed, the plate is still attached to the power supply by a cable.

f. Using a 5-inch socket extender and socket, remove the outside screws that secure the heat sink upon which the defective transistor is mounted.

g. Remove the heat sink, and replace the defective transistor.

CAUTION

> While removing a defective component, do not touch the electrolytic capacitors or the choke located at the bottom of the power supply assembly. A capacitor may still be charged if its associated bleeder resistor is defective.

4.22 REGISTER DISPLAY INDICATOR REPLACEMENT

To remove a defective register display indicator, proceed as follows:

a. Disconnect the computer ac power cable from the 117V ac power source.

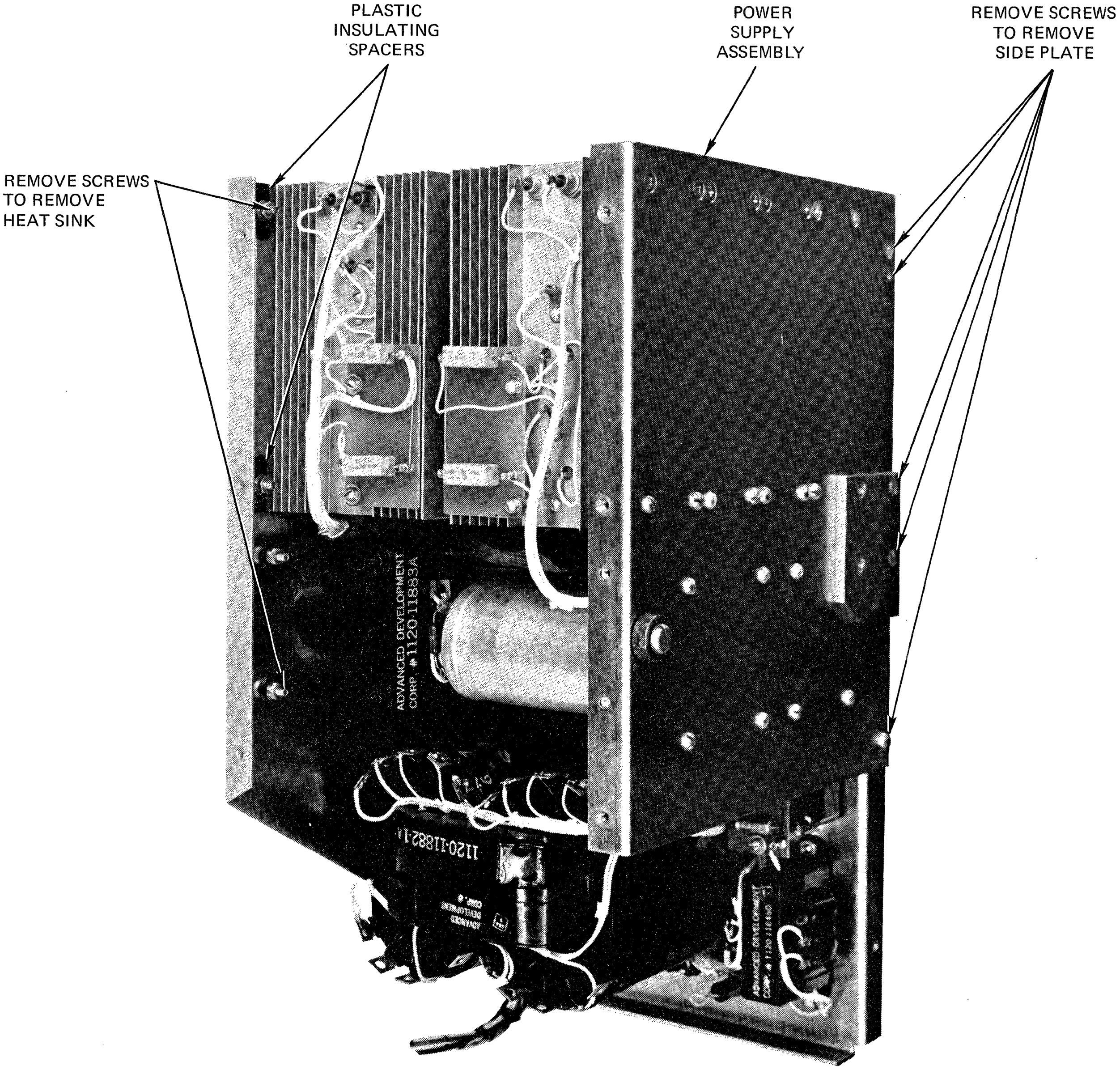


Figure 4-14. Removing a Power Supply Heat Sink

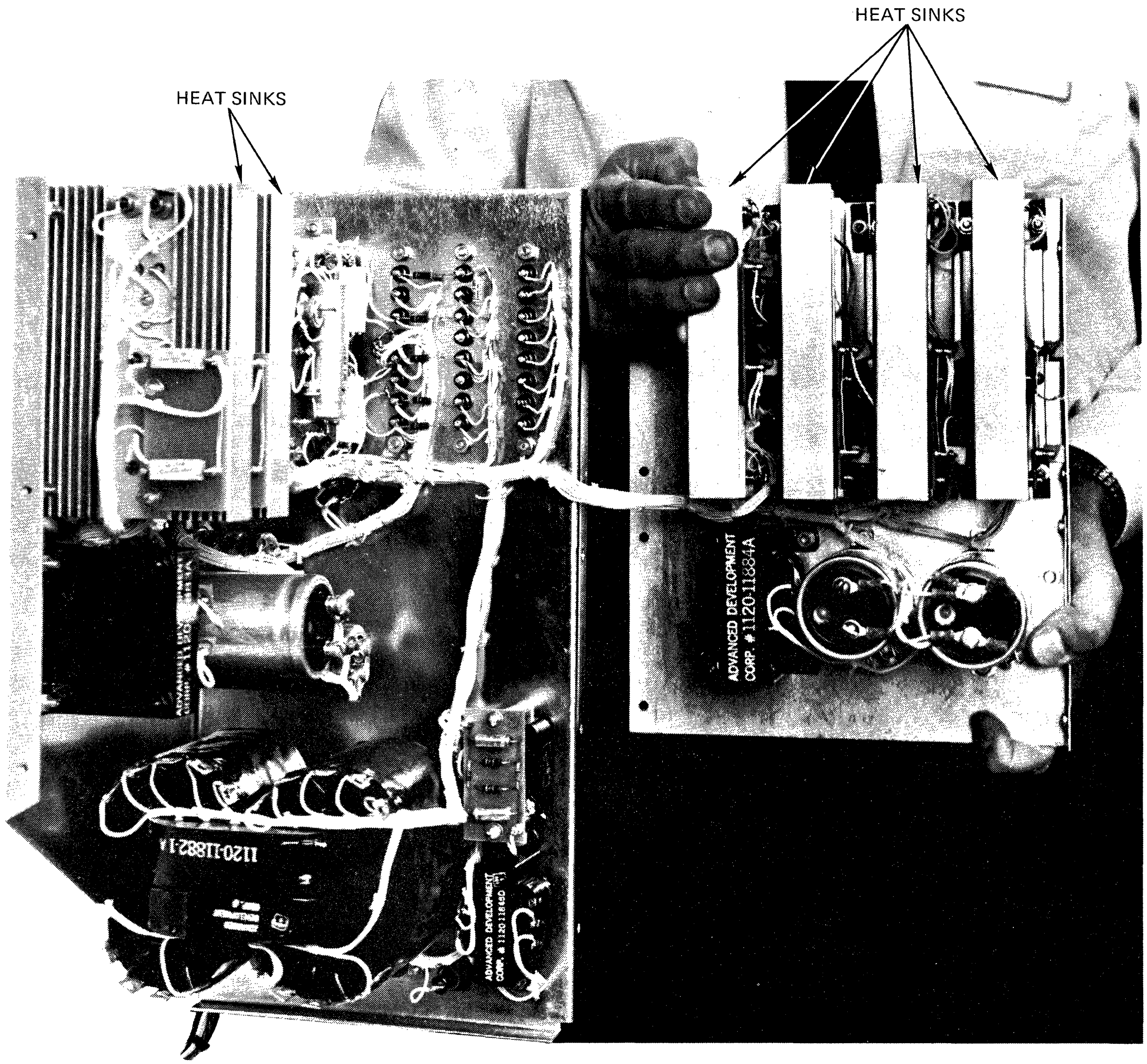


Figure 4-15. Disassembling the Power Supply Assembly

b. If the computer is mounted in a rack and the control panel is mounted in the front panel, open the front panel.

c. Remove the three screws that secure the register display assembly to the control panel (see figure 4-16). The defective indicator lamp is now accessible.

Note

Do not misplace the fiber insulated washers used on the screws.

d. Remove the defective indicator from its socket.

e. Place the indicator in the empty socket.

f. Secure the register display assembly to the control panel. Do not tighten the screws excessively.

g. Close the front panel of the computer.

h. When the machine is housed in a desk, the control panel is mounted on top of the desk. To gain access to the register display assembly, remove the 10 Phillips screws around the perimeter of the control panel.

i. Slide the control panel out of its housing.

j. Repeat steps c through f of this procedure.

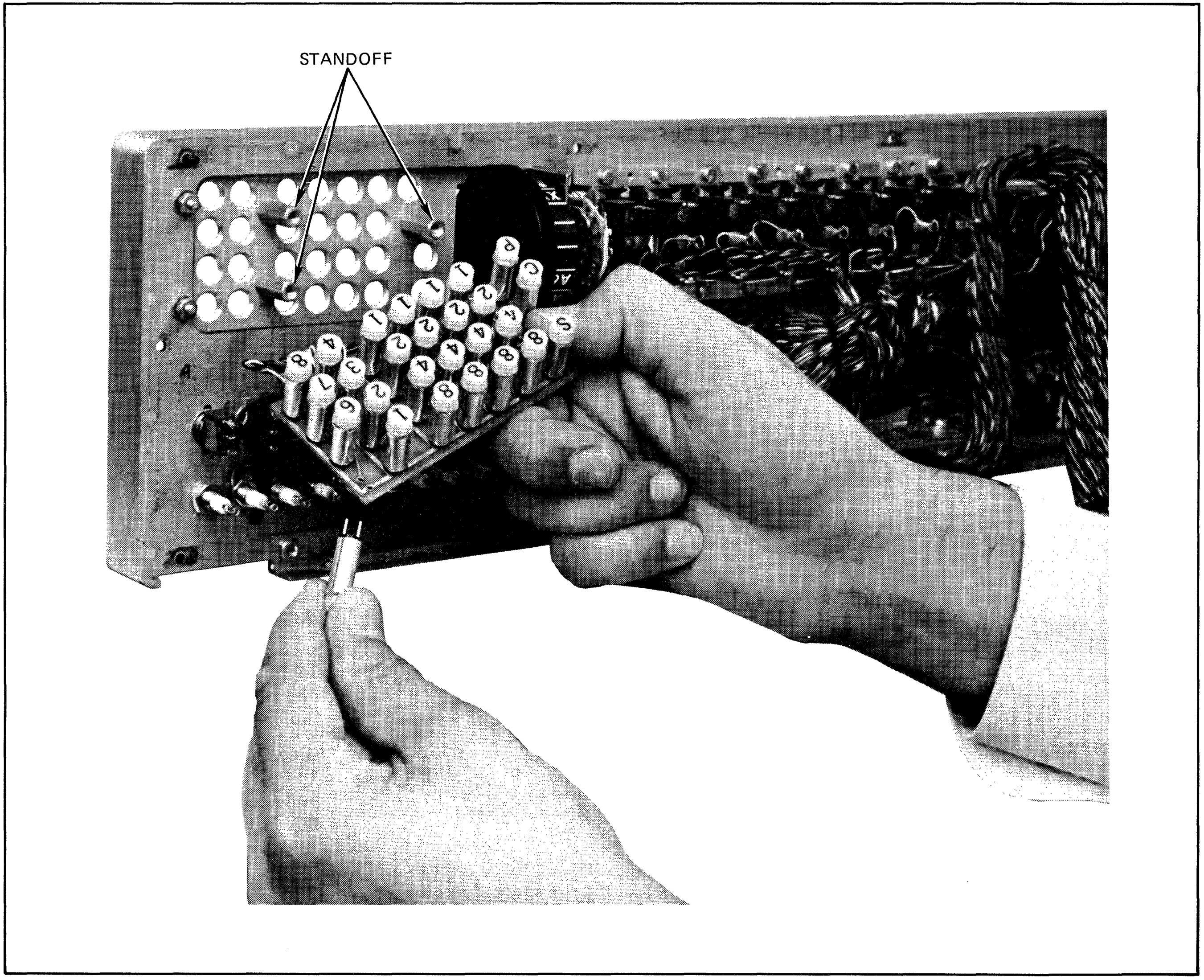


Figure 4-16. Removing a Defective Register Display Indicator

4.23 REPLACING CIRCUIT COMPONENTS

The computer uses printed circuit cards that are manufactured using the REDCOR NT-1 process. The NT-1 process consists of coating the surface of a printed circuit card with a layer of epoxy, before holes are drilled into the card. The epoxy coating is rugged and provides excellent protection for the printed circuit etch.

Due to the special characteristics of the NT-1 process, field repair of a printed circuit card is possible only with special tools. All repair, without these tools, is considered by REDCOR as an emergency repair. It is therefore suggested that defective circuit cards be returned to REDCOR for repair. Each district office is equipped with the special tools and replacement parts. If emergency repair of a printed circuit card is required, then refer to the procedures presented in the following paragraphs.

4.24 REPLACING MICRO ELEMENTS

The following is a recommended emergency procedure for removing and replacing a defective micro element. No special equipment is required to perform the procedure.

a. Cut the leads of the defective micro element as shown in figure 4-17a.

b. Cut the stems of the replacement micro element as shown in figure 4-17b.

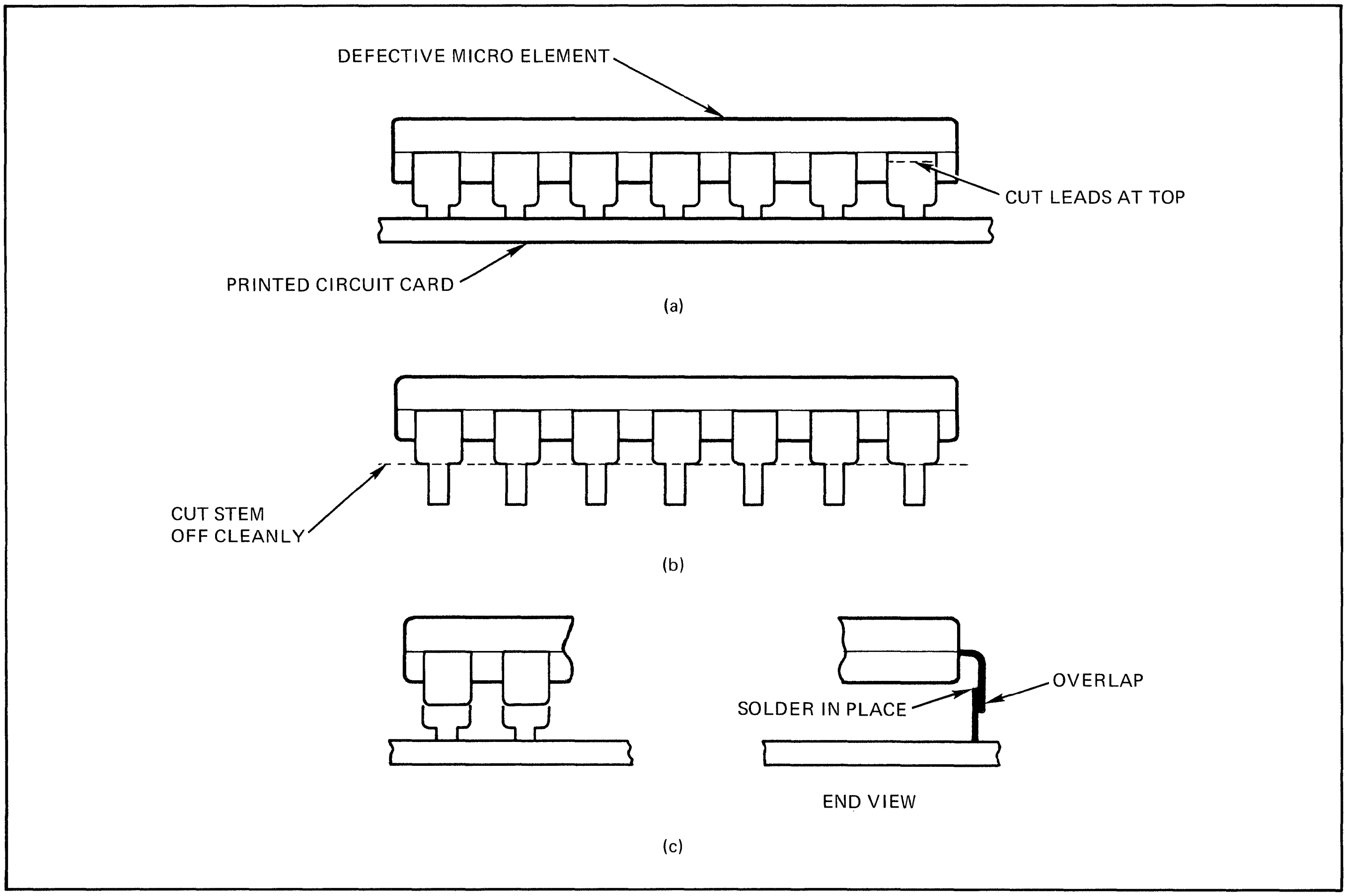


Figure 4-17. Replacing Defective Micro Element

c. Mount and solder the new micro element in place as shown in figure 4-17c.

4.25 REPLACING DIODES, RESISTORS, OR CAPACITORS

The following is a recommended emergency procedure for removing and replacing a defective diode, resistor, or capacitor. No special equipment is required to perform the procedure.

a. Cut the leads of the defective part as shown in figure 4-18a.

b. Straighten the leads.

c. Wrap the leads of the new part around the straightened leads as shown in figure 4-18b, and solder.

4.26 REPLACING TRANSISTORS

The following is a recommended emergency procedure for removing and replacing a defective transistor. No special equipment is required to perform the procedure.

a. Cut the leads of the defective transistor as shown in figure 4-19a.

b. Using a soldering iron, heat each of the remaining leads and remove the leads as shown in figure 4-19b.

CAUTION

Use the soldering iron and the lead itself to clean out the hole left by the lead. Do not drill out the hole as this will damage the printed circuit card.

c. Solder the new transistor into place.

4.27 WIRING WRAPPING

A wire-wrap tool (Gardner-Denver No. 1411-1C or equivalent) is required to make solderless connection to terminal posts on the printed circuit card mounting receptacles (see figure 4-20). To assure reliable

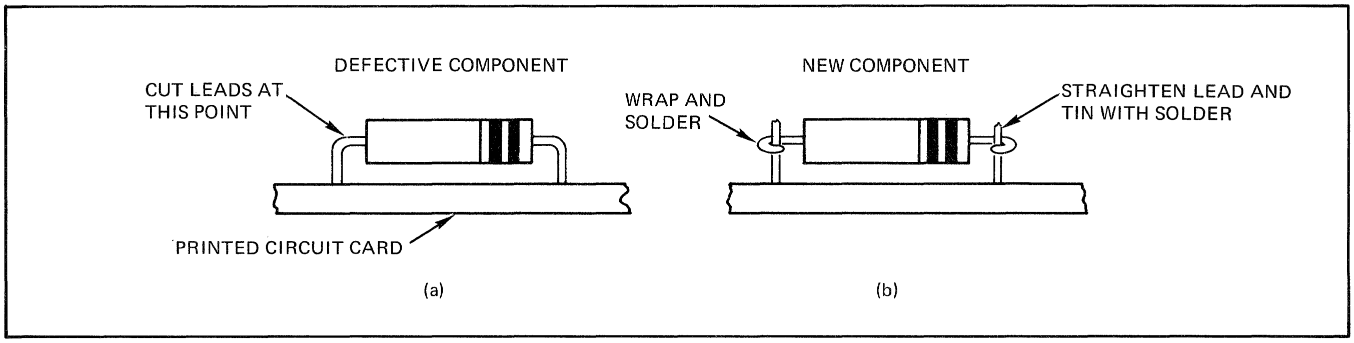


Figure 4-18. Replacing Defective Diode, Resistor, or Capacitor

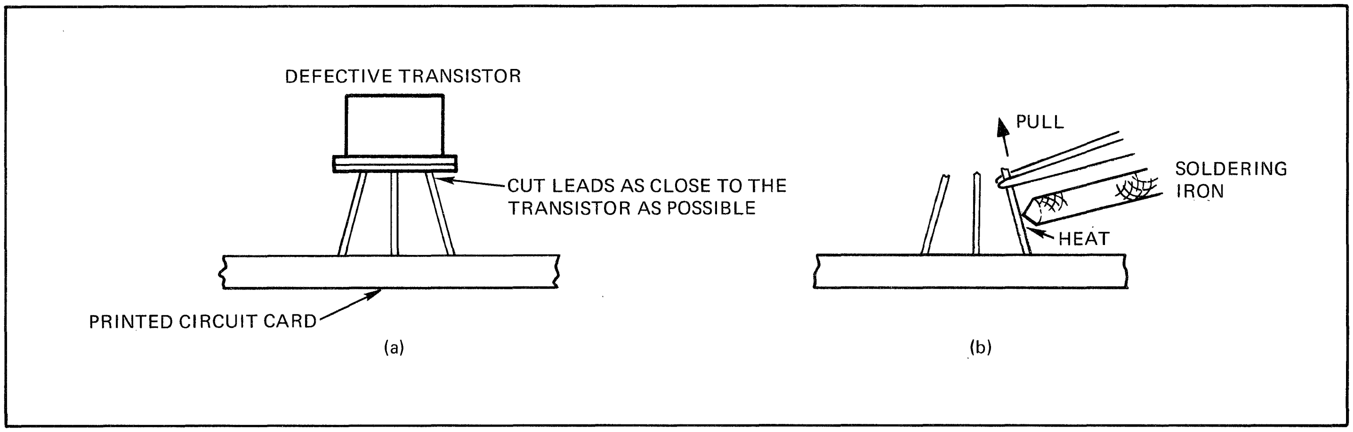


Figure 4-19. Replacing Defective Transistor

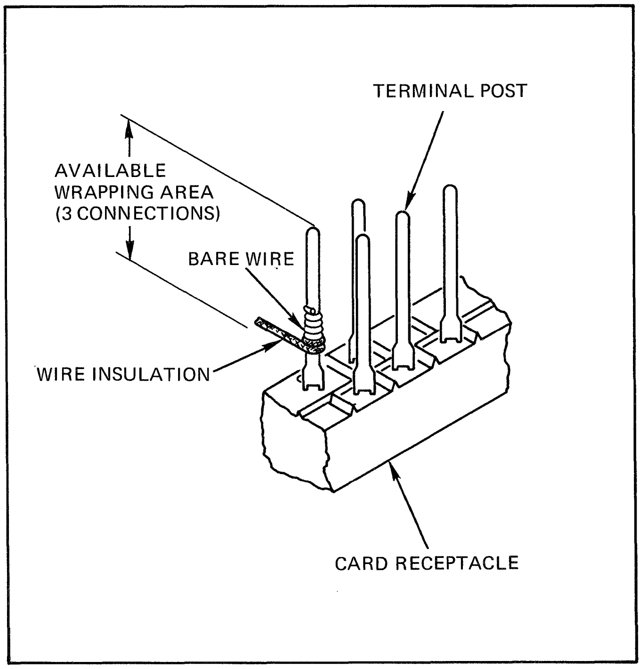


Figure 4-20. Printed Circuit Card Receptacle

connections, the following general requirements should be observed:

a. Do not wrap the end of a wire a second time. Always cut away the used wire end.

b. Do not make any connections with fewer than the minimum number of turns specified in table 4-2.

Table 4-2. Wire-Wrap Turns

WIRE GAUGE	BARE WIRE TURNS		INSULATED WIRE TURNS	
	Maximum	Minimum	Maximum	Minimum
30	10	7	1-1/2	1
28	8-1/2	7	1-1/2	1
26	7	5-1/2	1-1/2	1
24	6	4-1/2	1-1/2	1
22	6	4-1/2	1-1/2	1

c. Do not wrap more than two connections on any one terminal post.

d. Route the wire to the wrapped connection with sufficient slack to prevent strain on the post and in a direction that will tend to prevent unwrapping of the connection.

4.28 Solid Wire-Wrap Procedure

The following procedure explains the steps required to install a connection to a printed circuit card receptacle using insulated solid conductor wire:

a. Strip the insulation from the end of the wire, exposing a sufficient length of bare wire to permit the number of turns specified in table 4-2.

b. Insert the stripped end of the wire into the wire funnel of the wire-wrapping tool (see figure 4-21). The wire funnel will accept a sufficient length of the insulated wire to fulfill the insulated wire-wrapping requirements.

c. Route the wire through the anchor notch and back alongside the tool (B, figure 4-21).

Note

> When wrapping insulated wire, include at least one turn of insulated wire in the connection in addition to turns of bare wire specified in table 4-2.

d. Place the tool over the terminal post with the post inserted into the rotating spindle (C, figure 4-21).

e. While applying only the minimum force required to hold the tool in position, squeeze the trigger to make the wrap connection.

4.29 Stranded Wire-Wrap Procedure

The following procedure explains the steps required to install a connection to a circuit and to a receptacle using insulated stranded conductor wire:

a. Strip approximately one-half inch of insulation from the end of the stranded wire to be connected. Thoroughly tin the clean stranded wire end.

b. Prepare a length of solid wire for wrapping as described in the preceding paragraph. Strip sufficient insulation from the solid wire so that when the wire is inserted into the wire funnel no insulation will be on the wire funnel side of the anchor notch.

c. Insert the stripped and tinned end of the stranded wire into the rotating spindle of the wire-wrap tool.

d. Hold the stranded wire in place in the rotating spindle and, with the solid wire inserted in the wire funnel, position the tool over the terminal post and make the connection.

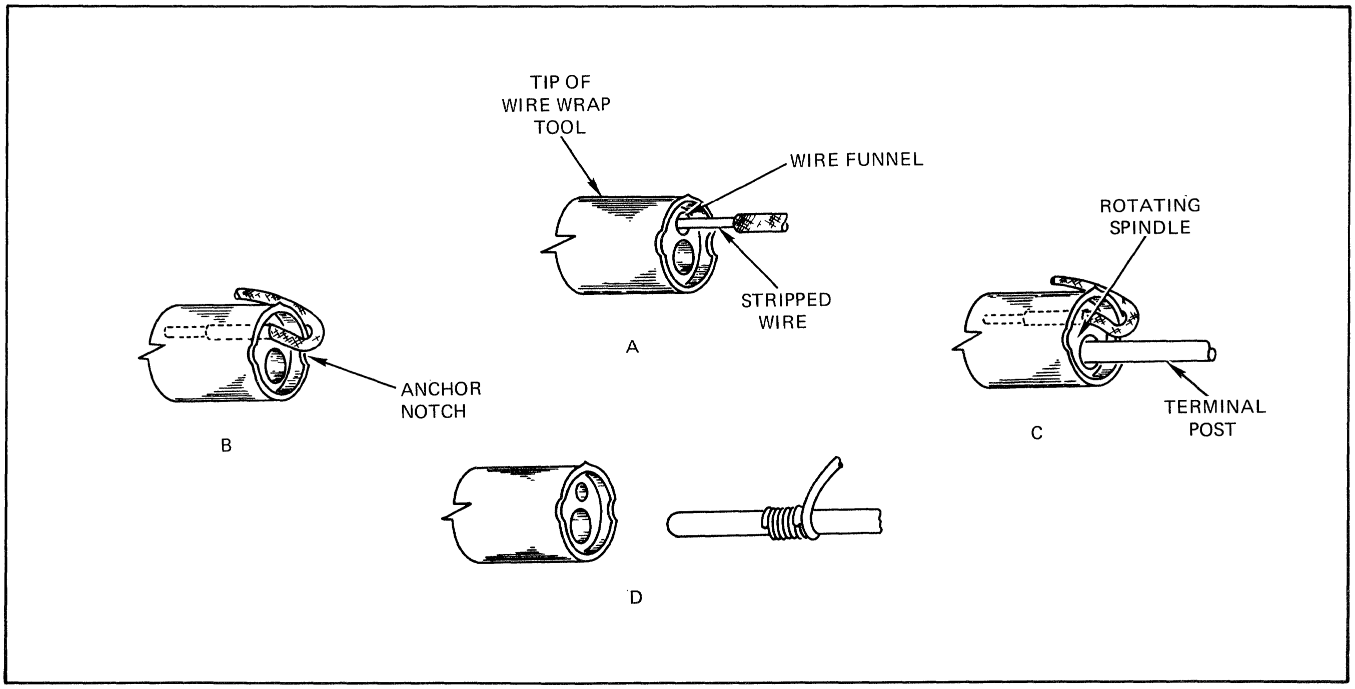


Figure 4-21. Termination of Solid Wire

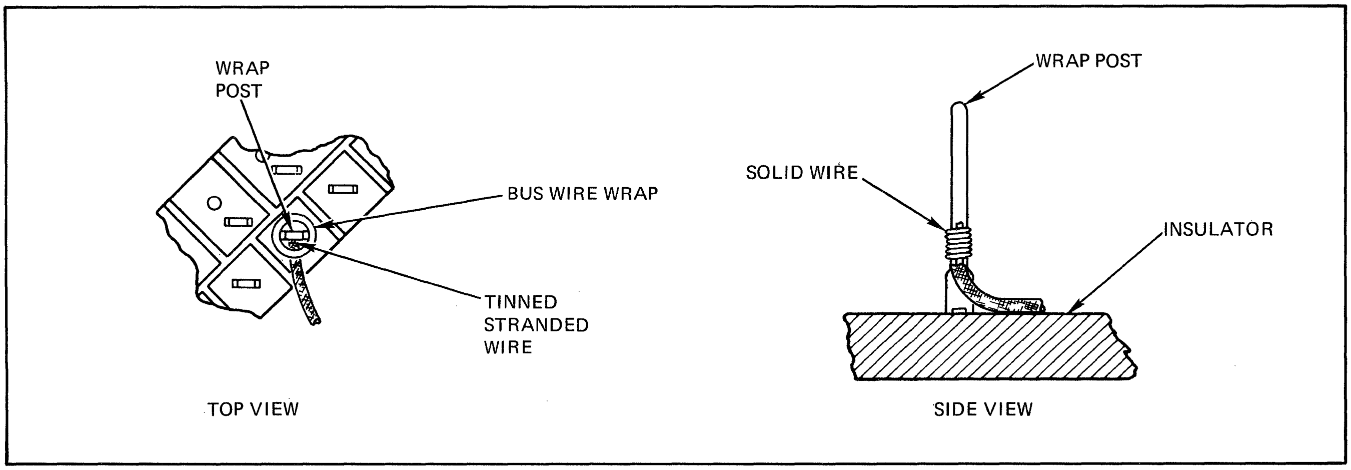
e. Cut away the solid wire leading away from the connection.

f. Solder the connection.

g. Inspect the connection for conformance to figure 4-22. Also check that the solid wire connection fulfills all of the requirements for solid wire connections.

4.30 RECOMMENDED CARD ASSEMBLY SPARES

Table 4-3 lists the recommended number of spares for each card assembly used in the computer.



TOP VIEW

SIDE VIEW

Figure 4-22. Termination of Stranded Wire

Table 4-3. Spare Card List

Card Assembly	Part Number	Quantity Used	Recommended Spares	Card Assembly	Part Number	Quantity Used	Recommended Spares
453	400453	1	0	520	400520	1	1
501	400501-2	1	0	521	400521	1	1
502	400502	1	0	523	400523	1	1
503	400503	1	0	524	400524	1	1
504	400504	1	1	525	400525	1	1
505	400505	1	1	526	400526	1	1
506	400506	1	1	527	400527	1	1
507	400507	1	1	528	400528	1	1
508	400508	1	1	530	400530	1	1
509	400509	8	2	532	400532	1	1
510	400510	2	1	533	400533	1	1
511	400511	1	1	534	400534	1	1
512	400512	1	1	542	400542	2	1
514	400514	1	0	545 (Option)	400545	2	1
515	400515	1	1	546 (Option)	400546	1	1
516	400516	1	1	547	400547	1	1
519	400519	1	1	548	400548	1	1

Note: Card assemblies 547 and 548 are housed in slots J48 and J42, respectively.

SECTION V
PERFORMANCE TESTING AND TROUBLE ANALYSIS

5.1 INTRODUCTION

This section contains information pertaining to testing and troubleshooting the RC 70 Computer. The performance testing portion of this section describes, in general terms, the diagnostics used to check out the operation of the machine. The trouble analysis portion of this section presents the diagnostics and explains how they are used in isolating malfunctions. In addition, the tests that constitute the diagnostics are described in detail. Troubleshooting aids are also referenced.

5.2 PERFORMANCE TESTING

Two diagnostic programs are used to test the performance of the RC 70 Computer. They are the RC 70 diagnostic program and the memory tests diagnostic program.

The RC 70 diagnostic program consists of two parts: automatic mode tests and manual tests. The automatic mode tests include:

a. A checksum test

b. An arithmetic test

c. A memory test

d. A panel and buffer test

e. A 2-millisecond interrupt test

f. A basic input/output test

The manual mode tests consist of:

a. Keyboard and typewriter input tests

b. A memory protection test

c. A memory parity error detection test

d. A basic input/output test

The memory tests program contains a memory parity test, memory test 1, and memory test 2. The memory parity test checks each word of the memory for correct parity. Memory test 1 tests 4096 words of memory by comparing the preloaded contents of each location with a similar bit configuration. This test is performed using two bit patterns.

Memory test 2 determines if a one can be written into and read out of each bit position in memory locations X'006B' through X'0FFF'. This test also checks the parity bit.

5.3 TROUBLE ANALYSIS

The RC 70 diagnostic program and the memory tests diagnostic program can be used to troubleshoot the RC 70 Computer. Other troubleshooting aids are also available in volume II of this manual. These aids consist of an RC 70 diagnostic program listing, a memory tests diagnostic program listing, an illustration of each computer printed circuit card together with a parts list, assembly drawings, logic diagrams, and power supply drawings.

A sort-by-name listing is also presented in volume II. This listing enables the Field Service Engineer to trace each logic function from its origin to its destination. It lists, in alphabetical order, every computer signal. If the complement of a signal is listed, an asterisk (*) appears to the right of the signal mnemonic.

Adjacent to a signal name are two entries. An entry consists of a jack designator and a pin reference. If an asterisk appears to the right of an entry, it indicates that the signal is generated at that location. For example: J33/78* indicates that signal X is generated at pin 78 of connector J33.

A discussion on how to interpret the REDCOR logic symbols is provided in appendix A of this volume. Supplementing this discussion is a table converting REDCOR symbols into the more familiar MIL-STD-806B logic symbols.

Before the RC 70 diagnostic program is described, the troubleshooting process is discussed.

5.4 TROUBLESHOOTING PROCESS

The proper method for troubleshooting consists of four steps:

a. Localization. In servicing a defective piece of digital equipment, localize the trouble by tracing the fault to the component responsible for the existing problem.

b. Isolation. Isolate by tracing the trouble to the defective printed circuit card responsible for the defective condition. Some malfunctions can be located by sight, smell, or hearing; others must be isolated by checking voltages, waveforms, and timing relationships.

c. Inspection. Inspect the problem area visually to locate any trouble such as loose, broken, or charred parts. Through this inspection, the Field Service Engineer frequently discovers the trouble or determines the circuit in which the trouble exists.

d. Repair. The final step of the troubleshooting process is repair. Take the necessary steps to correct the malfunction.

5.5 RC 70 DIAGNOSTIC PROGRAM

The RC 70 diagnostic program is used to diagnose any malfunction that may occur in the RC 70 Computer. It can also be used for computer checkout prior to running a program.

Although the diagnostic program does not test the machine on a state-by-state basis, a high degree of confidence can be placed in any RC 70 Computer that successfully completes the tests.

If a halt occurs during the program, refer to table 5-1. Table 5-1 lists the halts that may occur during a given test of the program. It also directs the Field Service Engineer to a paragraph that describes that portion of the diagnostic in which the halt has occurred. This paragraph not only describes the test in detail, but also lists all the instructions executed during the test and the paragraphs in section III where the instructions are explained.

Table 5-1. Error Halts

Error Halt*	Test	Test Description
1C903	Checksum	Paragraph 5.9
1C901-1C913	Conditional Branches, KU, KS	Paragraph 5.11
1C914-1C92C	Binary Adder	Paragraph 5.12
1C92D-1C935	Branch No Overflow	Paragraph 5.13
1C936-1C946	Operand Address Selection	Paragraph 5.14
1C947-1C94D	Paged Addressing	Paragraph 5.15
1C94E-1C951	Exchange (B), (UBA)	Paragraph 5.16
1C956, 1C957	Binary Logical Product	Paragraph 5.17
1C958-1C95E	Branch and Save (BLK, BPT, and BBK)	Paragraph 5.18
1C95F-1C962, 1C9F9-1C9FD	Binary Shifting	Paragraph 5.19
1C963-1C973	Binary Multiply	Paragraph 5.20
1C974-1C98C	Binary Divide	Paragraph 5.21
1C930-1C933	MM Start 1	Paragraph 5.26
1C936, 1C937, 1C9BB	MM Start 2 and MM Start 3	Paragraph 5.27
1C904	Memory Parity Error Detection	Paragraph 5.30
1C906-1C90A	Two-Millisecond Interrupt	Paragraph 5.31
1C90B, 1C90C	I/O Buffer	Paragraph 5.32
1C910-1C913, 1C918	Typewriter	Paragraph 5.34
1C914-1C917, 1C919	Reader	Paragraph 5.36
1C919	Teletype	Paragraph 5.37
1C9C5-1C9C7	Manual Mode Parity Error Detection	Paragraph 5.42

*The register display portrays the error halt when the register select switch is set to I.

5.6 PROGRAM LOADING

The RC 70 diagnostic program is loaded by performing the procedure given in paragraph 4.8, steps b through n.

5.7 PROGRAM CONTROL

The RC 70 diagnostic program is controlled by sense switch settings. The switches and their functions are listed in table 5-2.

5.8 AUTOMATIC MODE TESTS

The automatic mode tests are discussed in order of their occurrence. Note that the console keyboard light is not on in the automatic mode.

5.9 Checksum Test

When the checksum test is being performed, programmable register display lamp 1 should be on.

The binary sum of all instructions and constants used by the program is calculated and added to the checksum constant. The result should be zero. Failure to sum to zero results in an error halt.

The following instructions are used during this test:

Instruction	Description
LDX	Paragraph 3.96
LDB	Paragraph 3.94
ADB	Paragraph 3.79
BXB	Paragraph 3.87
BEQ, BUC	Paragraph 3.84
HLT	Paragraph 3.92

5.10 Arithmetic Tests

There are 16 arithmetic tests. During each of these tests, lamp 2 of the programmable display should be on. Subsequent paragraphs describe the arithmetic tests.

5.11 Conditional Branches – KU, KS Test

The three obtainable combinations of flip-flops KU and KS (00, 10, or 11) are established using the Load instruction (LDB) and appropriate constants. Conditional branch instructions, excluding the Branch No Overflow instruction (BNO) follow each LDB instruction. The branch instructions should be executed in accordance with the settings of KU and KS.

Table 5-2. Sense Switch Functions

Sense Switch	Functions
1	When sense switch 1 is in the lower position, the START pushbutton switch causes the program to cycle continuously through the automatic mode tests. If sense switch 1 is up, the program completes one cycle through the automatic mode tests and then halts. When this halt occurs, the program is put into the manual mode by pressing the START pushbutton switch. If the program is running and it is necessary to change sense switch settings, the program must first be halted by setting sense switch 1 in the upper position. When the program halts, set the sense switches as desired, and then press the RESET and START pushbutton switches in that order.
2	In the automatic mode, setting sense switch 2 in the upper position causes the program to repeat each automatic mode test for approximately 15 seconds before proceeding to the next test. In the manual mode, sense switch 2 halts the program at the memory protection test.
3	Setting sense switch 3 in the upper position selects the Teletype tests in both the automatic and manual modes. Sense switches 6 and 7 are used with sense switch 3 to select the Teletype punch and reader, respectively.
4	Setting sense switch 4 in the upper position selects the basic I/O tests in both the automatic and manual modes. Sense switches 5, 6, and 7 are used with sense switch 4 to select the typewriter, punch, and reader, respectively.
5	Setting sense switch 5 in the upper position selects the typewriter tests in both the automatic and manual modes.
6	Setting sense switch 6 in the upper position selects the punch tests in both the automatic and manual modes.
7	Setting sense switch 7 in the upper position selects the reader tests in both the automatic and manual modes.
8	In the automatic mode, setting sense switch 8 in the upper position selects the automatic power shutdown test. In the manual mode, sense switch 8 must be in the lower position.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
HLT	Paragraph 3.92
BUC, BLE, BEQ, BGE, BLT, BNE, BGT	Paragraph 3.84

5.12 Binary Adder Test

A 1 is loaded into the low end of the B-register. The contents of the B-register are put in temporary storage, then added to the B-register. The resultant carry should leave a 1 in the next most-significant bit position of the B-register. The store, add, and carry operations are repeated until each bit position in the B-register has been checked. This test also contains a series of additions which checks the binary adder logic.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
BGT, BEQ	Paragraph 3.84
HLT	Paragraph 3.92
STB	Paragraph 3.100
ADB	Paragraph 3.79
CMB	Paragraph 3.88
SBB	Paragraph 3.99

5.13 Branch No Overflow Test

Several additions are performed that should result in a series of overflow and no overflow conditions for the binary adder. The Branch No Overflow instruction (BNO) monitors the results of these operations.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
BNO, BUC, BGT, BLT	Paragraph 3.84
HLT	Paragraph 3.92
ADB	Paragraph 3.79

5.14 Operand Address Selection Test

The indirect, relative, and indexed operand address selection options are tested for short instructions, long instructions, and conditional branches. The various options are checked individually and in combination.

The following instructions are used during this test:

Instruction	Description
ADB	Paragraph 3.79
BEQ, BLT, BGT, BUC	Paragraph 3.84
HLT	Paragraph 3.92
LDB	Paragraph 3.94
CMB	Paragraph 3.88
LDX	Paragraph 3.96

5.15 Paged Addressing Test

Paged operand address selection is verified during this test. The page register is loaded with bit patterns that check each bit position with a 1 and a 0. The I-register is loaded with the eight low-order bits of the operand address during the long instruction and during indirect operand address selection.

A Load Page instruction (LDP) is used to load the page register. The tests for LDP are done using long, and long indirect operand address selection to verify that the I-register is being loaded correctly.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
BEQ, BUC, BGE	Paragraph 3.84
CMB	Paragraph 3.88
HLT	Paragraph 3.92
LDP	Paragraph 3.95
SSB	Paragraph 3.99
BXB	Paragraph 3.87
LDX	Paragraph 3.96

5.16 Exchange (B), (UBA) Test

Known values are loaded into the B- and UBA-registers. These values are exchanged, and the 17-bit exchange is verified. The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
XMB	Paragraph 3.101
CMB	Paragraph 3.88
HLT	Paragraph 3.92
BEQ, BNO, BUC	Paragraph 3.84

5.17 Binary Logical Product Test

Binary logical products are checked using the four possible bit patterns for each bit (B1 through B16) of the B-register. The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
ANB	Paragraph 3.81
CMB	Paragraph 3.88
BEQ	Paragraph 3.84
HLT	Paragraph 3.92

5.18 Branch and Save (BLK, BPT, and BBK) Test

A subprogram using Branch and Link (BLK), Load (LDB), and Branch Back (BBK) instructions is executed. Failure to branch or failure to load the B-register properly causes an error halt.

A subprogram using Branch and Put (BPT) and BBK instructions is executed. The index register, the indicators, and the page register are set up before the BPT, changed after the BPT, and checked for the original values after the BBK.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
BUC, BEQ, BNO	Paragraph 3.84
BBK	Paragraph 3.83
CMB	Paragraph 3.88
LDX	Paragraph 3.96
ADB	Paragraph 3.79
LDP	Paragraph 3.95
BPT	Paragraph 3.86
HLT	Paragraph 3.92
BLK	Paragraph 3.85

5.19 Binary Shifting Test

An End Left instruction (ELB) specifying a 15-bit shift is performed. The value shifted fully exercises both the one-bit and four-bit shifts for various bit patterns. Fifteen-bit shifts are executed using a Logical Right instruction (LRB), then an Arithmetic Left instruction (ALB). Finally, a series of Arithmetic Right instructions are performed using first a negative constant, then a positive constant.

The proper settings of indicators KU and KS are verified for the shifts. The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
ELB	Paragraph 3.90
BGT, BEQ, BLT	Paragraph 3.84
HLT	Paragraph 3.92
CMB	Paragraph 3.88
LRB	Paragraph 3.97
ALB	Paragraph 3.80
ARB	Paragraph 3.82

5.20 Binary Multiply Test

Binary multiplications are performed using constants selected to exercise all of the logical paths within the multiplication algorithm. The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
CMB	Paragraph 3.88

Instruction	Description
MPB	Paragraph 3.98
BEQ, BGT, BLT	Paragraph 3.84
HLT	Paragraph 3.92

5.21 Binary Divide Test

Binary divisions are performed using constants selected to exercise all of the logical paths within the division algorithm. Whenever the dividend is less than or equal to the divisor, KO is set and the division is not attempted. The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
DVB	Paragraph 3.89
BNO, BUC, BGT, BEQ, BLT	Paragraph 3.84
HLT	Paragraph 3.92
XMB	Paragraph 3.101
CMB	Paragraph 3.88

5.22 Inclusive Or Test

The inclusive Or operation is verified for the possible bit patterns for each bit of the B-register. (See the RC 70 fixed listing at the end of the RC 70 diagnostic program listing in volume II.)

5.23 Exclusive Or Test

The exclusive Or operation is verified for the possible bit patterns for each bit of the B-register. (See the RC 70 fixed listing at the end of the RC 70 diagnostic program listing in volume II.)

5.24 End Around Double Length Shift (Left) Test

The shift operation is verified in both the B- and UBA-registers. (See the RC 70 fixed listing at the end of the RC 70 diagnostic program listing in volume II.)

5.25 Memory Test

The memory test is the third automatic mode test. It verifies that both 0's and 1's can be written into and read out of memory locations 0 through 6 and from memory location 20 through the highest address of available unprotected memory.

Locations 7 through 19 are used frequently during the remainder of the program and, therefore, are not included in this test. In particular, power shutdown locations 10 through 17 must remain free at all times.

During the memory test, lamp 3 of the programmable display should be on.

5.26 MM Start 1 Test

This test consists of two parts: First, all 0's are written into, then read out of memory locations 0 through 6. Second, all 1's are written into, then read out of the same locations. These locations are used for temporary storage during the remainder of the memory test. The following instructions are used during this test:

Instruction	Description
LDP	Paragraph 3.95
LDB	Paragraph 3.94
STB	Paragraph 3.100
CMB	Paragraph 3.88
BEQ, BNO, BNE	Paragraph 3.84
HLT	Paragraph 3.92
LDX	Paragraph 3.96
BXB	Paragraph 3.87
BLK	Paragraph 3.85

5.27 MM Start 2 and MM Start 3 Tests

The instructions contained in tests MM start 2 and MM start 3 are essentially identical. The contents of the locations under test are temporarily inverted, thereby making it impossible to check locations containing the instructions used during the test. Therefore, the instructions used in the MM start 2 test are used to check the area occupied by MM start 3, and the instructions in the MM start 3 test are used to check the area occupied by MM start 2.

The contents of each location are read and stored in location X'0001'. The original value is inverted, stored in X'0002', and written into the test location. The inverted value is read out of the test location, stored in X'0003', and then complemented and stored in X'0004'. The contents of X'0004' are compared with the original value, then returned to the test location. If the comparison is not successful, an error halt occurs. If an error halt does occur, the index register contains the address of the test location, X'0001' contains the original value, and X'0002' contains the error value.

The RC 70 diagnostic program starts loading at X'0800'. Therefore, if a 4096-word memory is used, the end of memory is reached before the program is loaded. The memory wraparound feature compensates for this. This feature causes that portion of the program having addresses above X'0FFF' to load into memory locations with addresses 1000 less than the addresses specified in the program. Thus, the instruction loaded into location X'1000' is actually loaded into location X'0000'. The test is terminated at the highest available address in unprotected memory. The last address of the program cannot be reached.

With memories larger than 4096 words, each location beyond the last word of the program is loaded into the address specified in the program. The normal memory test is performed on these locations. In addition, the content of each of these locations is compared with the location address. If an error results, a halt occurs. The index register then contains the original value of the location, and the B-register contains the error value.

The following instructions are used during this test:

Instruction	Description
ADB	Paragraph 3.79
STB	Paragraph 3.100
LDB	Paragraph 3.94
BLK	Paragraph 3.85
CMB	Paragraph 3.88
BEQ, BLT, BGT, BNE, BUC, BNO	Paragraph 3.84
SSB	Paragraph 3.99
BBK	Paragraph 3.83
XMB	Paragraph 3.101
HLT	Paragraph 3.92

5.28 Panel and Buffer Test

The panel and buffer test is the fourth automatic mode test. During the test, lamp 4 of the programmable display should be on.

5.29 Data Switches Test

The contents of the data switches on the computer control panel are loaded into the B-register by a Parallel Input instruction (PIP). Whenever any of the 17 bits stored in the data switches differs from the bit configuration in location X'0800', lamp 8 of the programmable display goes on.

The latch circuit that drives lamp 8 is reset when the test following the panel test occurs. Therefore, it is necessary to place sense switch 2 in the upper position to continue the panel test. This action is required to determine the state of lamp 8. Sense switch 2 causes each automatic mode test to cycle for approximately 15 seconds before proceeding to the next test.

The following instructions are used during this test:

Instruction	Description
LDP	Paragraph 3.95
LDB	Paragraph 3.94
STB	Paragraph 3.100
PIP	Paragraph 3.118
BNO, BUC, BNE	Paragraph 3.84
CMB	Paragraph 3.88
POP	Paragraph 3.119
HLT	Paragraph 3.92

5.30 Memory Parity Error Detection Test

The memory parity error flip-flop is sensed to determine if a parity error has occurred. The following instructions are used during this test:

Instruction	Description
SNS	Paragraph 3.120
HLT	Paragraph 3.92
BEQ, BNE	Paragraph 3.84

5.31 Two-Millisecond Interrupt Test

The 2-millisecond test is sensitive to real time and, therefore, is skipped if sense switch 8 is in the upper position. During this test, the INHIBIT/INTRP switch on the computer control panel must be in the INTRP position.

An interrupt forces an indirect Branch and Put instruction (BPT) from location X'000F'. For the purpose of this test, the BPT instruction causes the program to branch to a particular subroutine. This subroutine decreases the contents of the roll-arrow register by a count of three and causes the program to branch to or around error halts, depending upon the feature being checked.

Initially, interrupt enabled latch OKIRP is reset. A test is made to verify that no interrupt occurs within a

2.4-millisecond period. Latch OKIRP is then set, and a test is made to verify that an interrupt does occur within 2.4 milliseconds. This interrupt takes the program directly into a test that verifies that a second interrupt does not occur within 1.6 milliseconds, but does occur within 2.4 milliseconds. Finally, OKIRP is reset and a test is made verifying that an interrupt has not occurred within the following 2.4-millisecond period.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
SBB	Paragraph 3.99
BGT, BUC	Paragraph 3.84
SET	Paragraph 3.121
HLT	Paragraph 3.92

5.32 I/O Buffer Test

During this test, a buffer tester box is connected as an I/O device to the machine. The presence of the tester can be sensed and, if it is not attached, the panel test is complete.

The tester has a 14-bit address register and a 17-bit data register. Each of these registers has a parallel output and a parallel input address associated with it. A series of Parallel Input (PIP) and Parallel Output (POP) instructions are performed to verify that the registers can be loaded correctly.

Once the registers are checked, they are used with the necessary Set instructions to exercise the direct memory access (DMA) capability. The address register supplies the DMA addresses and the data register supplies or receives the memory data. When writing, the POP instructions are used to output the addresses and data to the tester. When reading, the POP instructions provide the address register with addresses, and a PIP instruction retrieves the data word from the data register.

The external interrupt is tested during the I/O buffer test. A Set instruction indirectly activates the tester, generating an external interrupt. This interrupt is similar to the interrupt described in paragraph 5.31. The successful operation of the interrupt results in branching past an error halt.

The following instructions are used during this test:

Instruction	Description
SNS	Paragraph 3.120
BEQ, BUC, BNO, BNE	Paragraph 3.84
LDB	Paragraph 3.94
STB	Paragraph 3.100
POP	Paragraph 3.119
PIP	Paragraph 3.118
BLK	Paragraph 3.85
SET	Paragraph 3.121
CMB	Paragraph 3.88
BBK	Paragraph 3.83
SBB	Paragraph 3.99
HLT	Paragraph 3.92

5.33 Basic Input/Output Test

The input/output test is the fifth automatic mode test. Independent selection of those portions of the test which exercise the various devices and their associated logic is made by setting the sense switches. When the following tests are performed, refer to table 5-2 for the proper sense switch settings.

The I/O devices (typewriter, reader, and punch) can be operated singly or together. An I/O unit can be tested by addressing it properly. To address the unit, perform the following procedure:

a. Disconnect the I/O power cable from the 117V ac power source.

b. Remove the two sheet metal screws that hold the rear panel of the I/O unit in place.

c. Refer to figure 4-5. Set the switches on the interface card as follows:

Switch 0:	up
Switch 1:	down
Switch 2:	up
Switch 3:	up
Switch 4:	up

d. Replace the rear panel of the I/O unit.

e. Connect the I/O power cable to the 117V ac power source.

f. Proceed with the input/output test.

5.34 Typewriter Test

The typewriter is sent 130 characters consisting of all output characters, as well as functions, such as carriage return, tab, backspace, and so forth. Each output character is used with the typewriter set for both lower and upper case. The characters are transmitted to the typewriter output latch circuits by means of Parallel Output (POP) instructions.

As the typewriter acts on this information, the typewriter input latch circuits are loaded with the typed characters. Parallel Input instructions (PIP) return the contents of the characters, thereby checking the typewriter input capability.

A series of tests are performed to verify that the typewriter can shift from lower case to upper case, from upper case to upper case, from upper case to lower case, and from lower case to lower case.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
BUC, BNE, BEQ, BGT	Paragraph 3.84
LDP	Paragraph 3.95
SNS	Paragraph 3.120
POP	Paragraph 3.119
LDX	Paragraph 3.96
BLK	Paragraph 3.85
ALB	Paragraph 3.80
BXB	Paragraph 3.87
HLT	Paragraph 3.92
SBB	Paragraph 3.99
SET	Paragraph 3.121
ANB	Paragraph 3.81
PIP	Paragraph 3.118
CMB	Paragraph 3.88
LRB	Paragraph 3.97
ADB	Paragraph 3.79
BBK	Paragraph 3.83

5.35 Punch Test

During this test, 130 characters are transmitted to the paper tape punch. Each character is recorded on paper tape. In addition, a second set of 32 characters is sent to and punched by the paper tape punch. Both sets of characters are used in the reader test described in paragraph 5.36. The punched paper tape is easily verified using the reader test.

5.36 Reader Test

The paper tape reader is tested by reading the tape generated during the punch test. Care must be taken to synchronize the punch output frames with the reader input tape frames. To accomplish this, punch a sufficient amount of tape to reach the reader; then set sense switch 1 in the upper position. With sense switch 1 in this position, the program stops after the punch test has been completed.

The first punched tape frame must then be positioned over the read pins. With both the punch and the reader selected, the program is restarted by pressing the RESET and START pushbutton switches on the processor control panel. The characters being read back are compared with the characters being supplied to the punch. Thirty-two characters with alternately correct and incorrect parity are included in those being punched and read back. A series of tests verifies that the input parity checking logic can be disabled and that, when enabled, the parity check is correct.

The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
BUC, BEQ, BNE	Paragraph 3.84
LDX	Paragraph 3.96
SNS	Paragraph 3.120
SET	Paragraph 3.121
BLK	Paragraph 3.85
ALB	Paragraph 3.80
BXB	Paragraph 3.87
ANB	Paragraph 3.81
PIP	Paragraph 3.118
POP	Paragraph 3.119
CMB	Paragraph 3.88
LRB	Paragraph 3.97
ADB	Paragraph 3.79
HLT	Paragraph 3.92
BBK	Paragraph 3.83

5.37 Teletype Test

The Teletype test is similar to the basic I/O punch and reader tests. The only exceptions are that a separate set of characters is used and that the input parity tests are not repeated. The character set includes all of the output characters and the function characters used by the Teletype. The characters are typed or otherwise used by the Teletype typewriter as they are punched.

Sense switch 7 must be in the upper position to select the reader during the Teletype test. The reader is enabled and disabled by Set instructions (SET). The reader reads one frame after it is disabled; therefore, it is enabled only every other frame. Failure to read the extra frame or the reading of more than one extra frame results in an error halt.

The following instructions are used during this test:

Instruction	Description
SNS	Paragraph 3.120
BEQ, BNE	Paragraph 3.84
PIP	Paragraph 3.118
SET	Paragraph 3.121
CMB	Paragraph 3.88
LRB	Paragraph 3.97
ADB	Paragraph 3.79
HLT	Paragraph 3.92
BBK	Paragraph 3.83
LDP	Paragraph 3.95
LDB	Paragraph 3.94
STB	Paragraph 3.100
LDX	Paragraph 3.96
BLK	Paragraph 3.85
ANB	Paragraph 3.81
POP	Paragraph 3.119
ALB	Paragraph 3.81
BXB	Paragraph 3.87
BUC	Paragraph 3.84

5.38 Power On-Off Test

The power on-off test is selected by setting sense switch 8 in the upper position. During this test, lamp 7 of the programmable display should be on.

When a power shutdown occurs during the program, the contents of the computer's volatile registers and the states of various flip-flops are stored in locations X'0010' through X'0017'. On the next pass through the test, the contents of these memory locations are typed out. The typeout includes letters that identify the registers from which the locations were loaded. The identifiers and data format are shown in figure 5-1.

Power shutdowns can be simulated at a rate of one every 16 milliseconds by placing the B-E TEST switch on the processor control panel in the B-E position. Sense switch 8 must be placed in the upper position before setting the B-E TEST switch.

5.39 MANUAL MODE TESTS

The manual mode is entered by setting sense switch 1 in the upper position and, after the program halts, by pressing the START pushbutton switch. In the manual mode, sense switch functions and indicator meanings are different from those in the automatic mode. Once in the manual mode, the program can be halted by setting sense switch 1 in the lower position.

This halt feature must be used if it is desired to change sense switch settings. The program can then be returned to the automatic mode by pressing the RESET and START pushbutton switches in that order, or continued in the manual mode by setting sense switch 1 in the upper position and by pressing the START pushbutton switch.

5.40 Keyboard and Typewriter Input Tests

These tests are performed with sense switches 2 through 8 in the lower position. In addition, the I/O unit is addressed by setting the switches on the interface card in accordance with the procedure given in paragraph 5.33. Note that the console keyboard indicator lamp is on during these tests.

A character generated at the console, typewriter, or Teletype keyboards is displayed by lamps 3 through 8 of the programmable display. When using the console keyboard, a count (modulo 4) is also displayed by lamps 1 and 2 of the same display. This feature is a check for switch contact bounce. The count should advance by one each time a console keyboard key is pressed.

5.41 Memory Protection Test

The memory protection test is entered by setting sense switch 2 in the lower position. This action halts the program. The memory protect BLOCK switches and the PROTECT MEM switch should be set as desired.

If the PROTECT MEM switch is placed in the lower position, none of the memory is protected. Pressing the START pushbutton switch turns on lamp 5 of the programmable display. If the PROTECT MEM switch is placed in the upper position, pressing the START pushbutton switch causes lamps 6, 7, and 8 of the programmable display to match the settings of the BLOCK switches.

IDENTIFIER	STORAGE/TYPED FORMAT					CELL
	BIT 0	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	
X	0	X7, X6, X5, X4	/ X3, X2, X1, YC	/ YB, YA, V1, V2	/ V3, 0, 0, 0	10_{16}
P	KO	/ KS, KU, P1, P2	/ P3, P4, P5, P6	/ KK, KC, KB, KA	/ -	11_{16}
T	0	/ 0 0 0 0	/ 0 0 0 0	/ 0 0 0 0	/ T Reg.	12_{16}
I	IO	/ I Reg. ———	/———	/———	/———►	13_{16}
L	*	/ L Reg. ———	/———	/———	/———►	14_{16}
B	BO	/ B Reg. ———	/———	/———	/———►	15_{16}
D	DO	/ D Reg. ———	/———	/———	/———►	16_{16}
N	0	/ N Reg. ———	/———	/———	/———►	17_{16}

*INTERRUPT ENABLE LATCH

Figure 5-1. Power Shutdown Typeout

The program then halts and waits for new switch settings and for the START pushbutton switch to be pressed. The test can be terminated by placing sense switch 2 in the lower position and by pressing the START pushbutton switch.

5.42 Memory Parity Error Detection Test

The memory parity error flip-flop is sensed to determine if a parity error has occurred. A Branch and Link (BLK) instruction activates the diode memory, generating instruction X'8013'. The instruction is routed to the I-register, decoded, and produces signals that represent a Branch Back (BBK) instruction with incorrect parity. The detection of this parity error is verified. A third parity error check ensures that the parity error flip-flop is reset by the parity error Sense (SNS) instruction.

The following instructions are used during this test:

Instruction	Description
SNS	Paragraph 3.120
BEQ, BNE	Paragraph 3.84
BLK	Paragraph 3.85
HLT	Paragraph 3.92

5.43 Basic Input/Output Test

The basic input/output test is similar to the automatic mode test and to those portions of the automatic mode typewriter, punch, and reader tests that use the set of 130 typewriter test characters.

The difference between the two modes of operation is that while the automatic tests use a sense device ready and response approach, the manual mode assumes device readiness at timed intervals. While a malfunctioning sense loop may leave the program waiting, the timed approach allows repetition of the operations, right or wrong, for observation.

In the manual mode, the typewriter outputs are typed and the various punch and reader characters are portrayed by the register display. The Teletype reader test does have a sense loop in addition to a time delay. The Teletype reader is enabled for every other character, and the sense loop allows time for the reader enabling relay to operate.

5.44 MEMORY TESTS DIAGNOSTIC PROGRAM

The memory tests diagnostic program consists of three tests: memory parity, memory test 1, and memory test 2. If an error halt occurs during any of these tests, the register display portrays the error word. To correct a malfunction, refer to the paragraphs listed in the test description that describes the instructions used during the test. Then use the necessary steps to localize, isolate, and repair the defective component that caused the error halt.

5.45 MEMORY PARITY TEST

The memory parity test checks each word of the memory for correct parity. The index register is loaded with address X'8000', and a load B-register indexed (LDBX) instruction is used to address locations X'0000' through X'FFFF'. If a parity error exists, the computer halts with the address in the index register. Note that because of the wraparound feature of the memory, a parity error in location X'0100' of a 4096-word memory would cause halts at address X'8100', X'9100', X'A100', . . . X'E100', and X'F100'. The following instructions are used during this test:

Instruction	Description
LDX	Paragraph 3.96
LDB	Paragraph 3.94
SNS	Paragraph 3.120
BEQ, BUC	Paragraph 3.84
HLT	Paragraph 3.92
BXB	Paragraph 3.87

To perform the parity test, proceed as follows:

a. Place the program tape in the paper tape reader.

b. Set the LINE/OFF/LOCAL switch on the I/O unit control panel to LINE.

c. Place each of the sense switches in the lower position.

d. Set the register select switch to X.

e. Set the function switch to COMPUTE.

f. Place the CONT toggle switch in the lower position.

g. Place the PROTECT MEM toggle switch in the lower position.

h. Place each of the BLOCK toggle switches in the lower position.

i. Place the PARITY HLT toggle switch in the upper position.

j. Place the INHIBIT INTRP toggle switch in the lower position.

k. Place the B-E TEST toggle switch in the upper position.

l. Press the RESET pushbutton switch.

m. Press the FILL pushbutton switch. After the tape is loaded, the program starts running automatically.

n. If there are no parity errors, the machine stays in the compute mode (C indicator lit) of operation. If an error exists, the computer halts and the register display portrays the error word. The parity latch is reset when it is sensed; therefore, the parity lamp is off when the halt occurs.

o. The test is continued by pressing the START pushbutton switch.

5.46 MEMORY TEST 1

Memory test 1 tests 4096 words of memory and is in the diode memory bootstrap format. It initially stores pattern 1 (10000) in locations X'0050' through X'0FFF'. The program then tests each location from X'0FFF' with the Compare instruction. If the contents of the location under test match pattern 1, pattern 2 (1FFFF) is then stored in that location.

When location X'0050' has been tested, the program returns to location X'0FFF' and begins comparing each location for pattern 2. If the contents of the location under test match pattern 2, pattern 1 is again written in the location.

If an error occurs, the program halts with the error address in X'0001' and the data for that word in the B-register. The following instructions are used during this test:

Instruction	Description
LDB	Paragraph 3.94
STB	Paragraph 3.100
LDX	Paragraph 3.96
BXB	Paragraph 3.87
CMB	Paragraph 3.88
BEQ, BUC	Paragraph 3.84
ADB	Paragraph 3.79
HLT	Paragraph 3.92

To perform memory test 1, proceed as follows:

a. Perform steps a through m of the procedure given in paragraph 5.45, except for the following differences:

1. Set the register select switch to A1.

2. Place the PARITY HLT toggle switch in the lower position.

b. If there are no errors, the machine stays in the compute mode of operation. If an error occurs, the computer halts with the error address in X'0001' and the data in the B-register. The register display portrays the data.

c. The program is continued by pressing the START pushbutton.

5.47 MEMORY TEST 2

Memory test 2 determines whether or not a 1 can be written into and read out of each bit position in memory locations X'006B' through X'0FFF'. This test also checks the parity bit.

First, a 1 is written into each location, beginning with X'0FFF' and proceeding in descending order to X'006B'. The bit positions of the 1's are: bit 16 in X'0FFF'; bit 15 in X'0FFE'; bit 14 in X'0FFD'; and so on. All 0's are written into X'0FEE' so that parity can be checked. This pattern is repeated through location X'006B'.

The read cycle is activated and the bit patterns are compared with the contents of an independent register.

Upon completion of the read cycle, a write cycle occurs. During this cycle, the 1's are shifted one bit position to the left. Those locations which have a 1 in bit position 0 prior to the shift contain all 0's after the shift. Those locations which contain all 0's prior to the shift have a 1 in bit position 16 after the shift.

The read and write cycles continue repetitively until a comparison error is encountered. If an error occurs, the program halts with the error address in X'0001'. The correct word appears in the B-register, and the error word is stored in location X'0002'.

The following instructions are used during this test:

Instruction	Description
LDX	Paragraph 3.96
LDB	Paragraph 3.94
STB	Paragraph 3.100
BLK	Paragraph 3.85
BXB	Paragraph 3.87
BNO, BEQ, BUC	Paragraph 3.84
CMB	Paragraph 3.88
ADB	Paragraph 3.79
HLT	Paragraph 3.92
ELB	Paragraph 3.90
BBK	Paragraph 3.83

To perform memory test 2, proceed as follows:

a. Perform steps a through m of the procedure given in paragraph 5.45, except for the following differences:

1. Set the register select switch to L.

2. Place the PARITY HLT toggle switch in the lower position.

b. When the tape stops, the register display portrays X'0019'.

c. Press the START pushbutton switch to begin the program.

d. If an error halt occurs, the register display portrays X'004B'.

e. To continue the program after an error halt, press the START pushbutton switch.

APPENDIX A
ENGINEERING DRAWING REFERENCE DATA

A.1 INTRODUCTION

Appendix A provides engineering drawing information. This information consists of engineering drawing numbering schemes, logic element symbology, symbology conversion information, and equation usage.

A.2 ENGINEERING DRAWING NUMBER SCHEMES

The engineering drawings presented in volume II of this manual have two types of numbers: a 500 number or an 800 number. A drawing which has a 500 number is either a schematic or a logic diagram or a combination of the two. A drawing which has an 800 number is an assembly drawing.

For example, drawing 400833 is a schematic and logic diagram of the computer clock; whereas, drawing 400533 is the computer clock printed circuit card assembly drawing.

A.3 LOGIC ELEMENT SYMBOLOGY

The following paragraphs describe the REDCOR gating, flip-flop, and inverter symbols.

A.4 GATING SYMBOLOGY

The basic logic element of the RC 70 Computer is the Nand gate. The symbol used for this gate is a half-moon configuration (see figure A-1). The inputs to the gate are shown entering the flat side of the symbol, and the output of the gate is shown leaving the convex portion of the symbol.

The numbers inside the circles are the micrologic element pin numbers. Pins 1 and 2 are the input pin numbers of the element, and pin 3 is the output pin number. An I6 is shown within the periphery of the half moon. It is the micrologic element number and is found on both the assembly and logic drawings.

The X below the I6 indicates that the output transistor of the gate does not have a pull-up resistor in its collector circuit. The star shown to the right of the Nand gate symbol means that the resistor connected to pin 3 is shared with another circuit on a different assembly.

A.5 FLIP-FLOP SYMBOLOGY

Figure A-2 shows the REDCOR flip-flop logic symbols. The symbol used is a rectangle. A micrologic element number is shown within the periphery of the rectangle.

The flip-flop shown in figure A-2a has a single input. A signal, representing some function, is applied to pin 2. Pin 3 is the clock input. If a +5V signal is transmitted to pin 2 and the voltage at pin 2 goes from a 0V level to a +5V level, the flip-flop sets. However, if a 0V signal is applied to pin 2 and clock input goes high, the flip-flop resets.

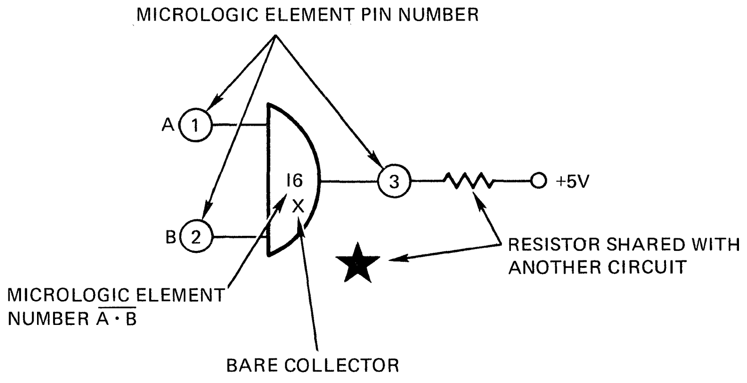
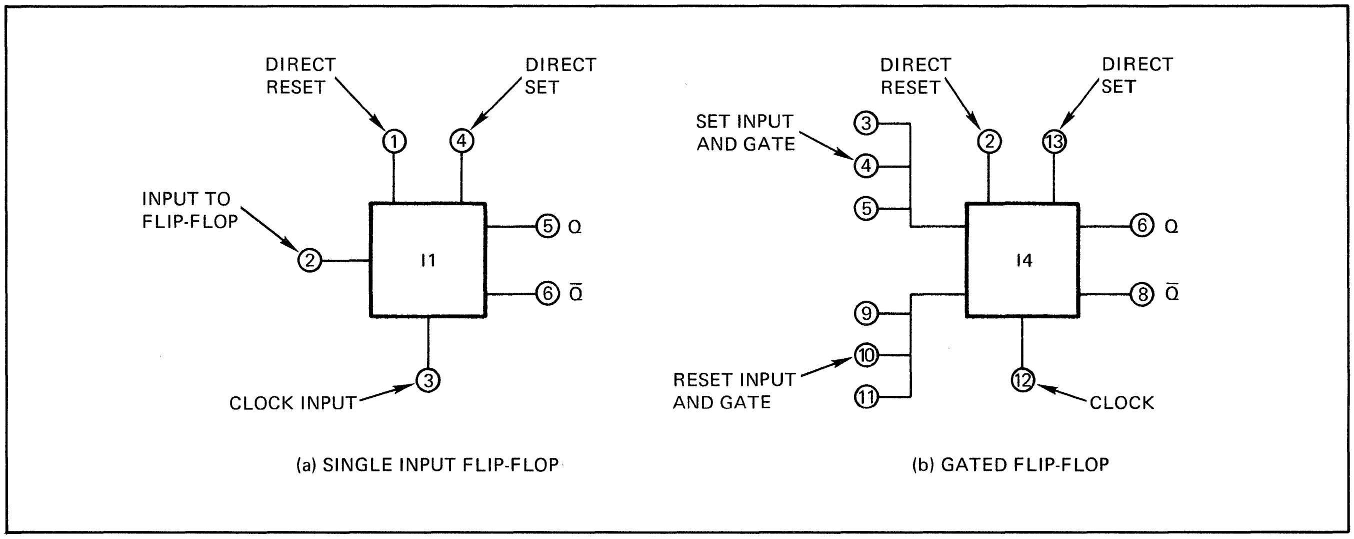

MICROLOGIC ELEMENT PIN NUMBER
A
1
B
2
I6
X
3
+5V
RESISTOR SHARED WITH ANOTHER CIRCUIT
MICROLOGIC ELEMENT NUMBER $\overline{A \cdot B}$
BARE COLLECTOR

Figure A-1. Nand Gate Symbol



(a) SINGLE INPUT FLIP-FLOP

(b) GATED FLIP-FLOP

Figure A-2. Flip-Flop Symbols

Pins 4 and 1 are the direct set and the direct reset inputs, respectively. These inputs are sensitive to negative voltage changes. The true output (Q) of the flip-flop appears at pin 5; the false output ($\overline{Q}$) at pin 6.

The flip-flop shown in figure A-2b is gated. It has a set input And gate and a reset input And gate. When the set input And gate is activated and the clock input goes high, the flip-flop sets. When the reset input And gate is activated and the clock occurs, the flip-flop resets.

Pin 5 of the micrologic element shown in figure A-2b, as well as pin 9, requires a 0V signal at that pin to enable or prime the gate. An inverter, which is not shown on the logic symbol, exists between pin 5 and the input of the gate associated with that pin. Thus, to set the flip-flop, a 0V signal must be applied to pin 5, +5V signals must be supplied to pins 3 and 4, and the signal applied to pin 12 must go high.

Pins 2 and 13 are the direct reset and the direct set inputs of the flip-flop. These inputs are sensitive to negative changes. The true output of the flip-flop appears at pin 8; the false output at pin 6.

A. 6 INVERTER SYMBOLOGY

The symbol used for an inverter is the same as the one used for a Nand gate, except the input pins are tied together (see figure A-3).

A. 7 SYMBOLOGY CONVERSION INFORMATION

The logic diagrams presented in the engineering drawings in volume II use REDCOR logic symbols. Table A-1 is provided as a means of translating the REDCOR logic into the more familiar MIL-STD-806B logic.

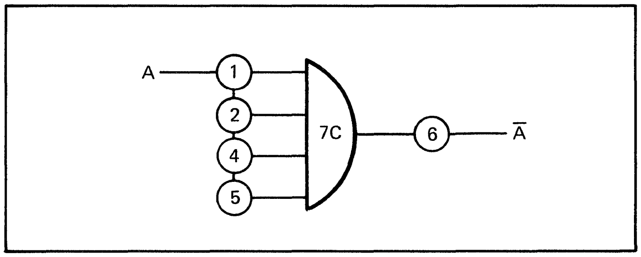


Figure A-3. Inverter Symbol

A. 8 EQUATION USAGE

The equations used in this manual conform to the following conventions. Logic inversion of a signal is denoted by a bar over the simplified logic term. For example:

An inversion of AB would be $\overline{AB}$.

The Or function is noted by a + sign between two logic terms. For example:

A Or B is noted A+B.

The And function is indicated by a simple conjunction of terms or by the use of parentheses between two logic terms. For example:

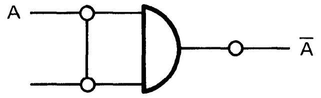
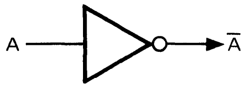
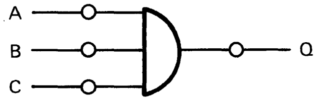
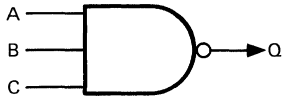
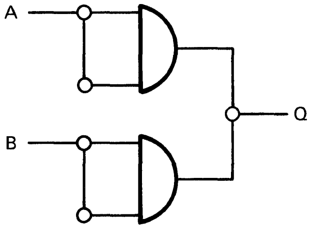
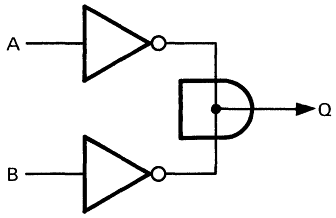
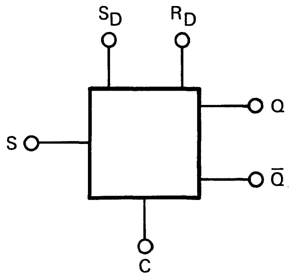
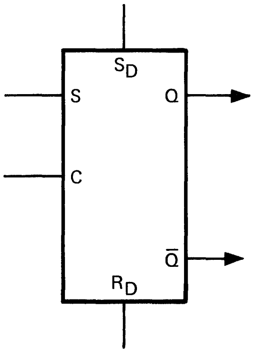
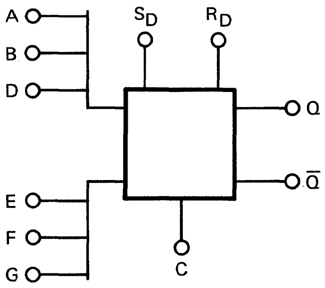
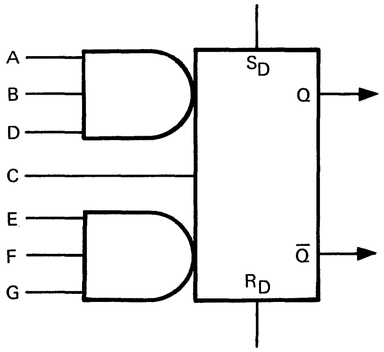
A And B are noted A B or (A)(B).

A flip-flop requires two equations to define its operation: a set equation and a reset equation. The reset equation always follows the set equation. For example:

$$\text{CK2MS} = \text{REVB} + \text{CLK CK2M } \overline{\text{CK2ML}}$$

$$\overline{\text{CK2MS}} = \text{REVB} + \text{CLK } \overline{\text{0V}} \text{ CK2ML}$$

Table A-1. Conversion of REDCOR Logic to MIL-STD-806B Logic

REDCOR SYMBOLS	EQUATION	MIL-STD-806B EQUIVALENT
	$A \Rightarrow \overline{A}$	
	$\overline{Q} = A \cdot B \cdot C$ OR $Q = \overline{A \cdot B \cdot C}$ OR $Q = \overline{A} + \overline{B} + \overline{C}$	
	$Q = \overline{A} \cdot \overline{B}$ OR $\overline{Q} = A + B$	
	$Q = S_D + (S \cdot C)$ $\overline{Q} = R_D + (\overline{S} \cdot C)$	
 A S_D R_D B D Q E $\overline{Q}$ F G C	$Q = S_D + (ABCD)$ $\overline{Q} = R_D + (EFGC)$	 A S_D B Q D C E F $\overline{Q}$ R_D G

NOTES:

1. THE GATES SHOWN ARE ON THE SAME CARD AS THE FLIP-FLOP.
2. C IS THE RISING EDGE OF THE CLOCK.
3. S_D IS THE FALLING EDGE OF THE DIRECT SET PULSE.
4. R_D IS THE FALLING EDGE OF THE DIRECT RESET PULSE.
5. THE SYMBOL $\Rightarrow$ MEANS IMPLIES.

USERS' REMARKS FORM

TITLE: RC 70 COMPUTER MAINTENANCE MANUAL
NO. M-7001A

DATED:

ERRORS NOTED:

Fold

SUGGESTIONS FOR IMPROVEMENT :

Fold

FROM: NAME ______________________________

COMPANY ______________________________

TITLE ______________________________

ADDRESS ______________________________

____

DATE ______________

FIRST CLASS
Permit No. 164
Canoga Park, Calif.

(No postage necessary if mailed in the United States)

BUSINESS REPLY CARD

Postage will be paid by addressee

REDCOR CORPORATION
7800 DEERING AVENUE
P.O. BOX 1031
CANOGA PARK, CALIF. (91304)

ATT'N: PUBLICATIONS DEPARTMENT

REDCOR CORPORATION