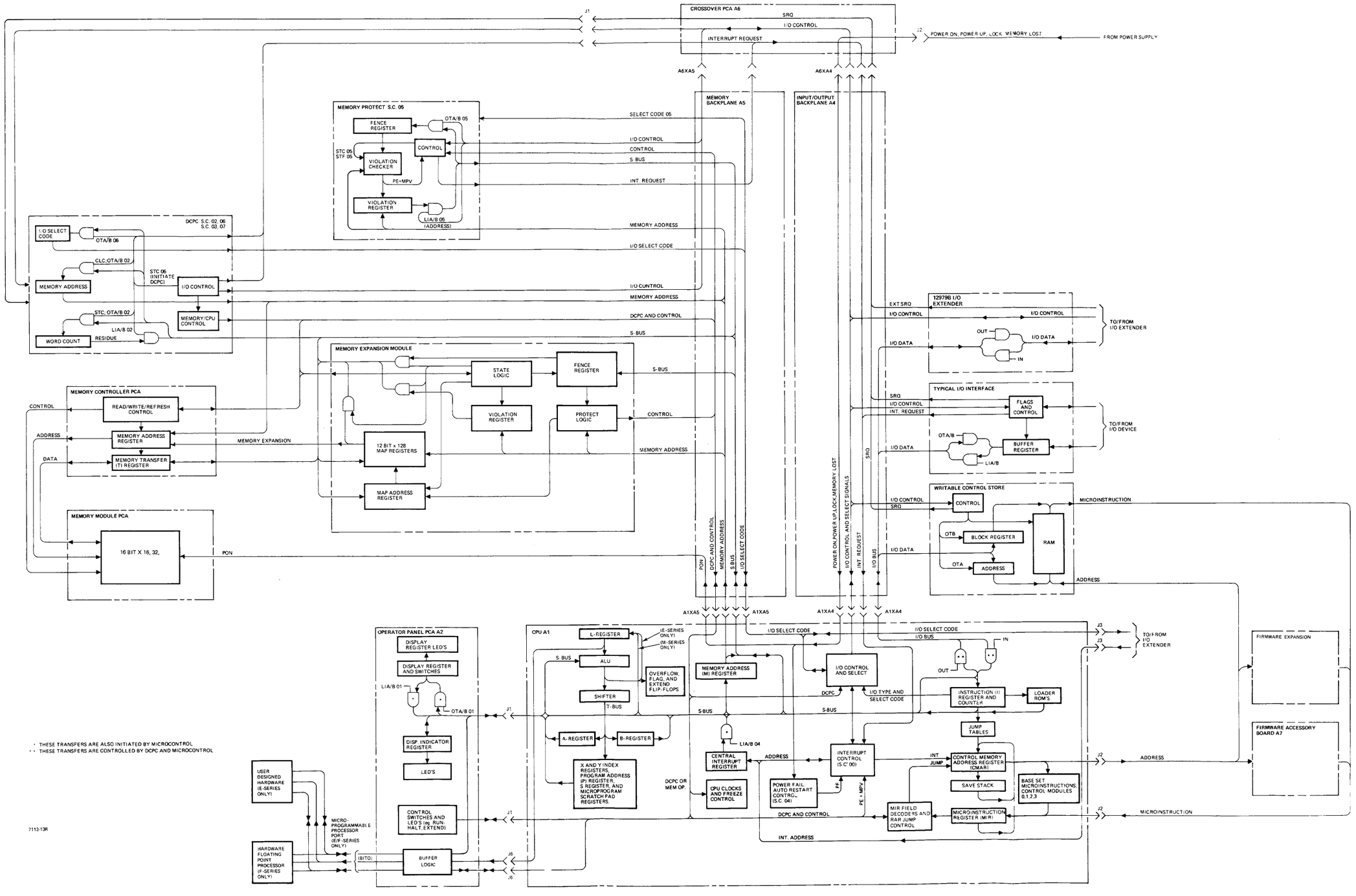


M, E, and F-Series Block Diagram

HP 1000 E AND F-SERIES COMPUTER CENTRAL PROCESSING UNIT (CPU)

THEORY OF OPERATION

NOTE

This document is part of the HP 1000 M, E, and F-Series Computers Engineering and Reference Documentation and is not available separately.

Table of Contents

INTRODUCTION	SECTION I

This document is the Theory of Operation for the HP 1000 E and F-Series Computer Central Processing Unit (CPU) and describes the processor hardware and base-set microprograms in detail.

The 12740A Hardware Floating Point Processor for the F-Series computer is described seperately in this section immediately following this description and associated diagrams.

Memory, I/O, power supply, and other accessories are not discussed in this section except when they directly affect the operation of the CPU. This discussion is conducted on a functional level using block diagrams and references to the schematic diagrams, 5061-1400-51 thru -59*, following the text. Understanding this theory is essential for performing detailed troubleshooting and repair of the computer. A thorough understanding of the following publications is necessary for complete comprehension of this theory:

a. HP 1000 E-Series Computer HP 2109B and HP 2113B Operating and Reference Manual, part no. 02109-90014, or HP 1000 F-Series Computer Operating and Reference Manual, part no. 02111-90001.

b. HP 1000 E-Series Computer HP 2109B and HP 2113B Installation and Service Manual, part no. 02109-90015, or HP 1000 F-Series Computer Installation and Service Manual, part no. 02111-90002

c. HP 1000 E and F-Series Computer Microprogramming Reference Manual, part no. 02109-90004.

d. HP 1000 M, E, and F-Series Computer I/O Interfacing Guide, part no. 02109-90006.

* The 5061-1400 CPU assembly (with schematic 5061-1400-51 thru 59) was introduced with the F-series computer and later used on the E-series computer. Earlier E-series computers used the 5061-1341 CPU assembly. Boards with date codes 1809 or later are identical.

REFERENCE INFORMATION — SECTION II

The HP 1000 E and F-Series CPU including its base-set control memory resides on nine sheets of schematics 5061-1400-51 through 59. They are referred to as sheets 1 through 9 throughout this text. Schematics, assembly drawing, and parts list are located immediately following this text. A microprogram listing of the E and F-Series base-set firmware is provided in Appendixes D and E.

2-1. Binary Signal Levels

Most of the logic used in the computer is implemented with standard or Schottky TTL. High logic levels are approximately +2.5 Vdc to +4.5 Vdc. Low logic levels are 0.0 +0.8V and varies due to the type of device, the load, and the condition of the device. When using positive logic a high is "true" and a low is "false".

2-2. Schematic Reading

Logic symbols are drawn to aid in understanding the logical functions being represented. A circle or "bubble" on an input or output indicates an active low logic level. For example a NAND gate may be represented by either logic symbol shown in figure 2-1.

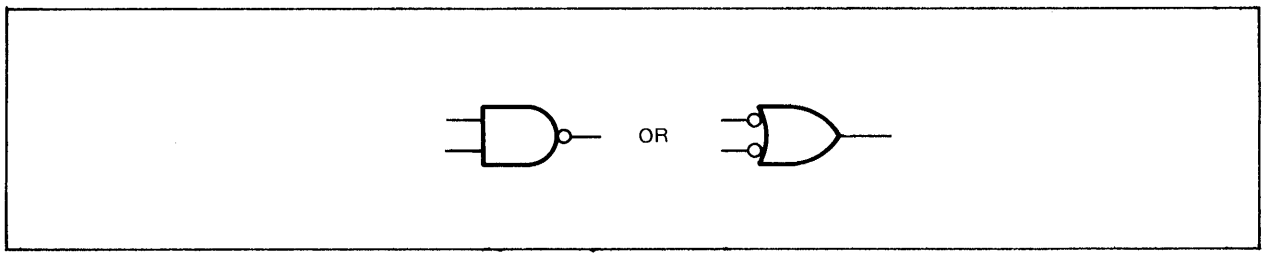


Figure 2-1. NAND Gate Functional Symbols

A circle or "bubble" on the ENABLE input of a bus driver or a LOAD input of a register indicates that the function of the device is enabled only when that input line is low. A circle on the clock input of a flip-flop means that a negative-going edge of the clock signal will clock the flip-flop.

2-3. Signal Names

Signal names are alphanumeric identifiers selected to aid in the understanding of the signal function. Not all signals are labeled, but all signals running

between schematic sheets are labeled. Any signal mnemonic may have a bar over it to indicate that the signal is active low, however in text the "not" bar is replaced with a minus sign (IBL-). For example, the IBL- signal is low only when the IBL button is pressed. When the IBL button is not pressed, the IBL- signal is high. Buses are named by a sequence of letters, followed by a number indicating bit significance. Many signal names have suffixes that help distinguish the origin or purpose of the signal. These signals include the following:

B	-	Buffered or Bus
EN	-	Enable (S-bus field micro-order)
FF	-	Flip-flop
OP	-	Operation (operation field micro-order)
SP	-	Special (special field micro-order)
ST	-	Store (store field micro-order)
SB0	-	S-bus bit 0
EXFF	-	Extend flip-flop
SOVSP-	-	Not set overflow special

2-4. Cross References

There are many signals that run from sheet to sheet in the schematics. To aid in locating a signal each schematic sheet is divided into 24 segments (six numbers horizontal and four letters vertical). Locations on the schematics are indicated brackets [ ]. For example, this location is found on sheet 2 in the upper right corner. Signals leaving one sheet to travel to other sheets are coded with the sheet number or numbers adjacent to the signal name. For example, a signal leaving sheet 1 for sheet 7 to sheet 1 would be coded as follows: (7) CPUTEN. If the signal is going to several sheets from sheet 1 it would be coded as follows: DMACYC.T5 (3,4,7). If the signal is going to the edge connector, it would be coded as follows: DMALO XA5-69.

2-5. Abbreviations

There are abbreviations used many times in this theory. The first reference is written out, but subsequence references may be abbreviated. Appendix A lists selected abbreviations used in the theory.

CPU BLOCK DIAGRAM	SECTION III

A simplified block diagram of the HP 1000 E and F-Series computer is illustrated in figure 3-1. The CPU will be functionally analyzed in Part 5 of this section in the following manner:

Timing: The purpose of the timing logic is to provide the necessary timing pulses to both the control processor and the input/output sections.

Control Processor: The heart of the computer, the control processor, controls the execution of microinstructions by presenting specific signals to the other processor sections.

Arithmetic/Logic: The Arithmetic/Logic section includes the primary hardware actually required to carry the arithmetic and data manipulative commands of the control processor.

Input/Output: The Input/Output (I/O) section serves as an interface and communication link between the computer and the external devices.

Operator Panel: The operator panel is the basic machine level interface between the computer and the operator.

MPP The Microprogrammable Processor Port (MPP) for the provides a direct access to the CPU for user-designed hardware processors. The MPP provides address,data, and control capability, so that external processors can be controlled and can transfer data at burst rates up to 11.4M bytes/second. The F-Series uses the MPP as the interface to the Hardware Floating Point Processor (FPP).

FPP F-Series Hardware Floating Point Processor - high Speed processor for floating point single, extended, and double precision integer operations.

Because both the Control Processor and the Arithmetic/Logic sections contain many registers, they are briefly discussed in the paragraphs that follow.

3-1. Control Processor

The Control Processor executes microinstructions in a time interval called a microcycle. A microinstruction is a 24-bit coded word that defines a very specific hardware operation to be performed by the computer.

Microinstructions physically reside in control memory and are the basic building blocks for defining microprograms. The operation of the Control Processor may be better understood with reference to figure 3-2 and the following brief definitions:

IR: The 16-bit Instruction Register is loaded from the main computer data bus and usually contains a machine language instruction for execution.

CMAR: The 14-bit Control Memory Address Register contains the address of the next microinstruction to be executed by the Control Processor. This is incremented at the beginning of every microcycle, but may be parallel loaded by the Microjump logic.

MJL: The Microjump Logic is controlled by the presently executing microinstruction and anticipates if and how the CMAR is to be loaded to set up the next microinstruction.

SAVE: A 16-bit subroutine Save Register Stack is pushed with the contents of the CMAR on every microsubroutine jump. It is popped with the contents loaded into the CMAR when a subroutine return is executed.

CM: The Control Memory contains up to 16K of 24-bit words implemented in either ROM, PROM, or RAM (WCS).

MIR: The 24-bit Microinstruction Register holds the microinstruction under Control Processor execution.

DECODERS: Timing signals are merged with the decoding logic from MIR to generate the signals needed to perform data functions through the microcycle.

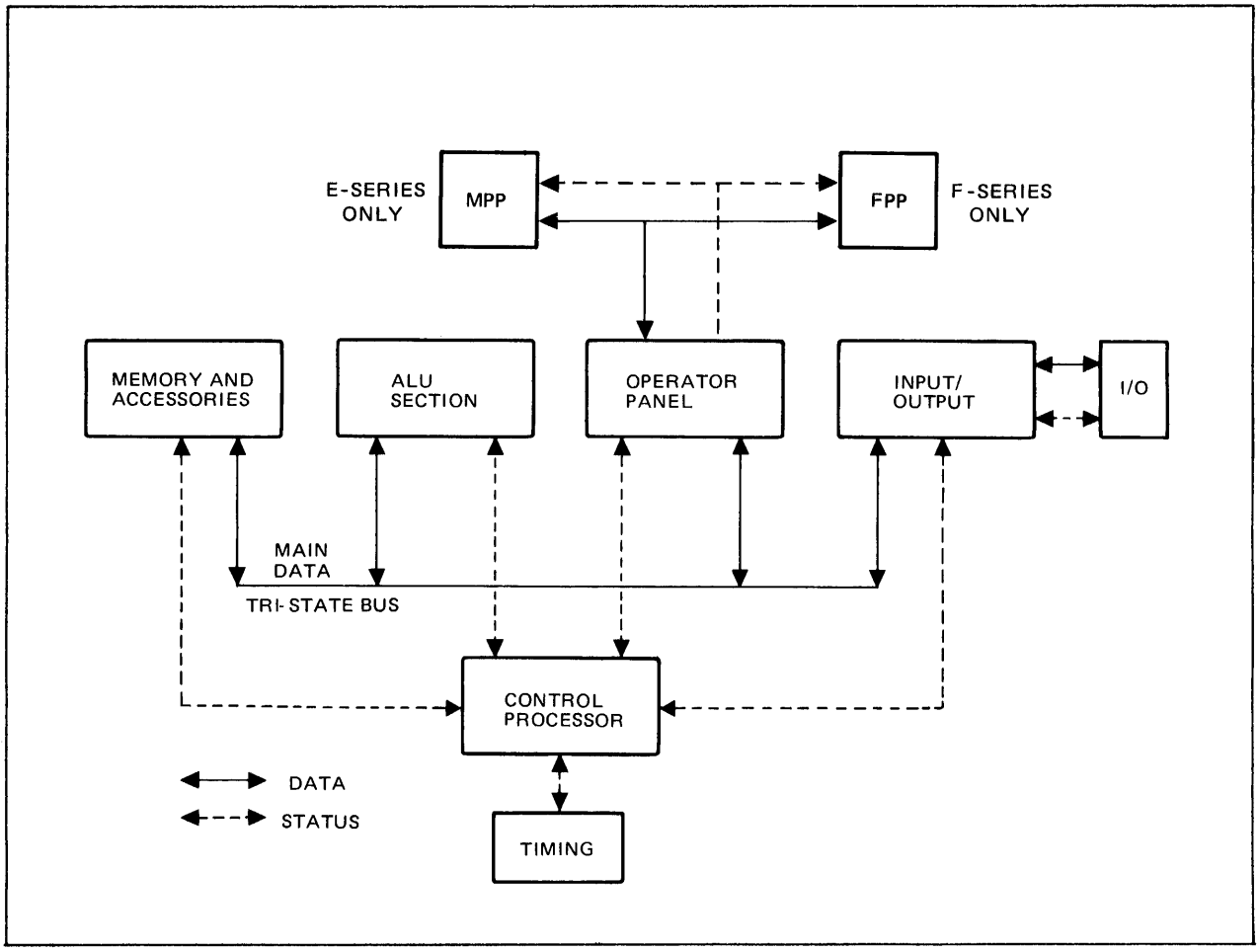


Figure 3-1. E/F-Series Computer Simplified Block Diagram

While a microinstruction presently residing in the MIR is being executed, the CMAR is incremented to present a new address to the CM to access the next microinstruction. At the next microcyle, the MIR is clocked with a new microinstruction to begin its execution and the procedure is repeated. Sequential microinstruction processing is interrupted by the MJL where in the CMAR is parallel loaded with a new CM branch address.

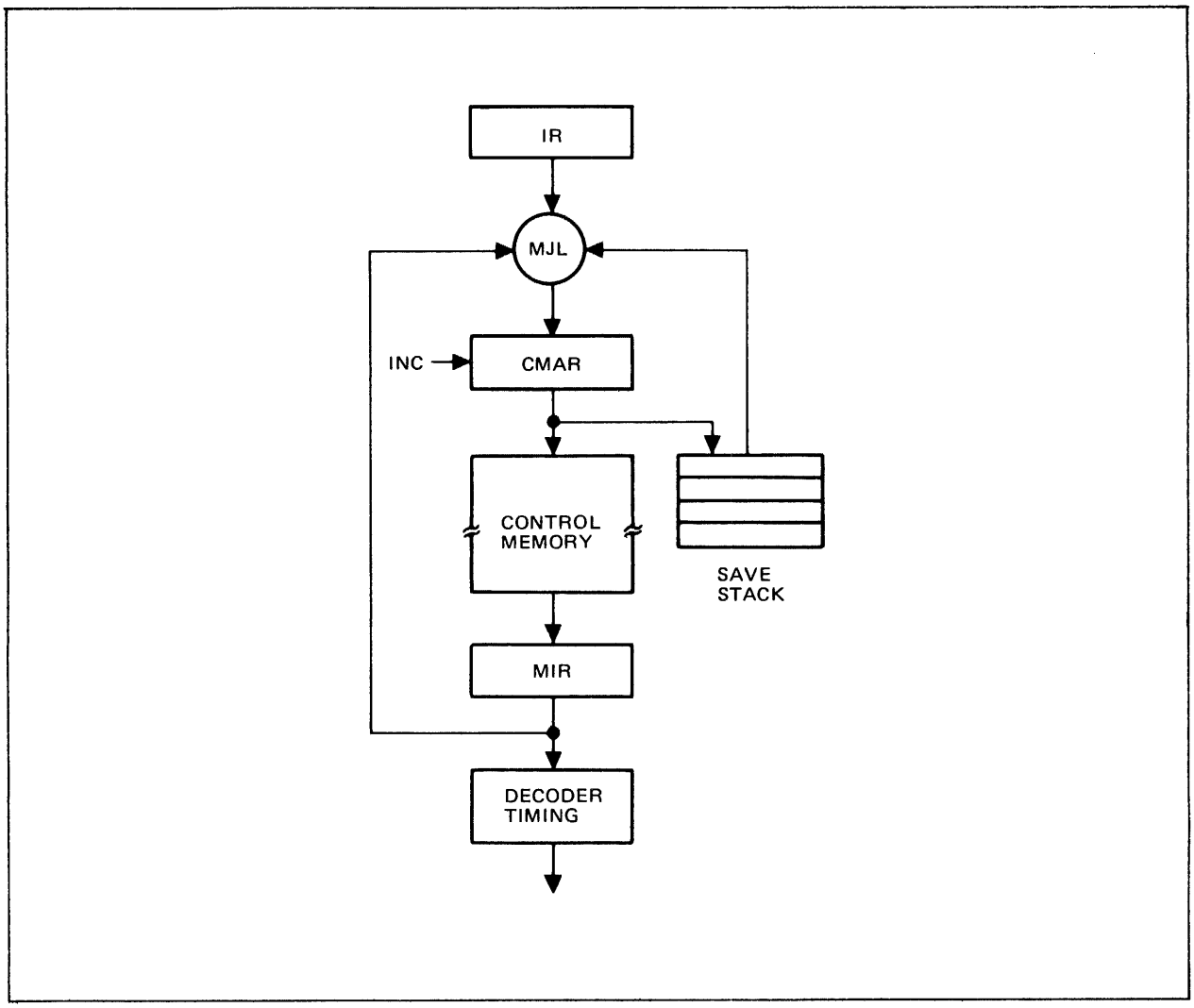


Figure 3-2. Control Processor Functional Block Diagram

3-2. Arithmetic Section

The Arithmetic section of the Control Processor includes the primary hardware required to actually carry out the commands of the microinstructions. Operation of the subsystem may be better understood with reference to figure 3-3 and to the following definitions:

S-BUS: The 16-bit positive-true tri-state TTL S-Bus is the primary bus of the computer and serves as the main data link between most internal computer subsystems.

T-BUS: The 16-bit negative-true T-Bus is the resultant data bus and is local to the Arithmetic section of the CPU.

ALU: The Arithmetic Logic Unit implements all of the arithmetic and logical operations of the CPU under direct control of the Control Processor.

RS: The Rotate/Shift Logic passes the ALU output data onto the T-Bus unless directed to perform a shift or rotate operation.

RAM: The 16-word, 16-bit Random Access Memory contains five specialized registers: P (Program Counter), S (S Pointer), and eleven scratch pad registers (S1, S2, through S11).

L: The 16-bit L-register holds the second operand required when a binary ALU operation is attempted and is loaded in the output of the ALU.

A and B: These are the two main 16-bitaccumulators accessable by assembly language programs.

M: The Memory Address Register holds the 15-bit logical address of any processor memory reference.

IR: The 16-bit Instruction Register is loaded from the main computer data bus and typically holds an assembly language instruction. The lower eight bits of the IR are implemented by an up/down counter for greater flexibility.

LDRS: The loaders consist of up to four read-only-memory (ROM) devices each of which contain a coded assembly language bootstrap program. The ROM data available to the S-Bus is determined by the lower eight bits of the IR which becomes the LDR address.

A typical data manipulation microinstruction calls for the contents of a register to be enabled onto the S-Bus and certain ALU and/or RS operations are further specified for the duration of the microcycle. At the end of the microcycle, a specified destination register is clocked to take the prevailing data off its input lines.

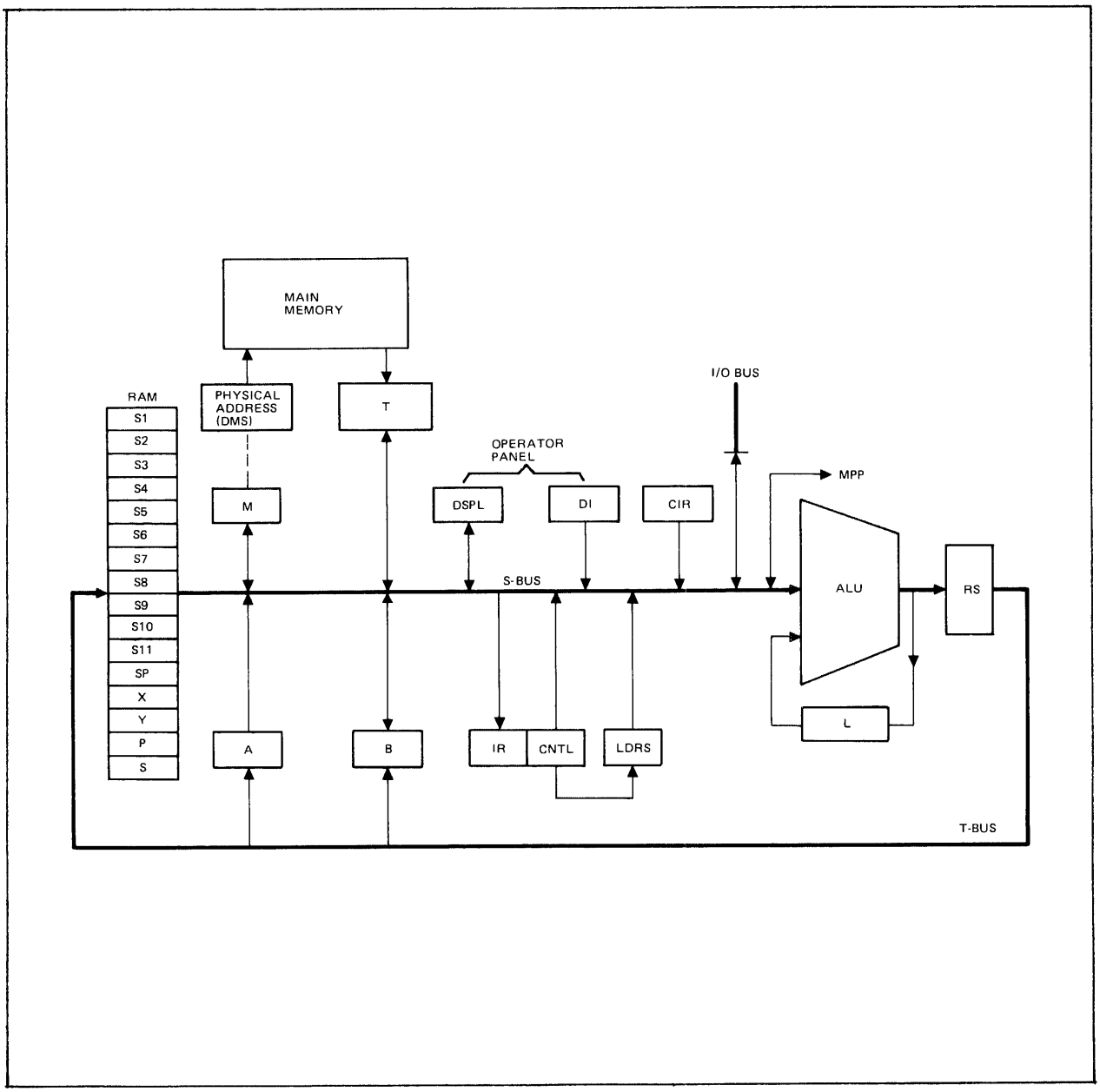


Figure 3-3. Arithmetic,I/O,Memory, and Operator Panel Functional Block Diagram

DETAILED THEORY OF OPERATION	SECTION IV

Although the CPU can be easily diagrammed into functional subsystems, there are substantial logic interconnections in the actual design. Therefore, it is difficult to jump in and begin describing the CPU's operation. Initially, some descriptions, timing and purpose of some signals will be only briefly described. In such cases, the reader's acceptance is the desired result. As the theory proceeds, more precise explanations will follow and eventually the reader should understand most of the signal functions presented earlier.

4-1. TIMING

Timing for HP computers is controlled by the CPU timing logic. Timing systems for the control processor and the I/O system are related, but operate semi-independently from each other until they are actually required to inter-communicate when I/O control activity is encountered.

The control processor timing is illustrated on the left half of sheet 1. Timing is generated from a 28.5 MHz crystal master oscillator circuit (U408) [A2]. The 28.5 MHz buffered output is passed through two NAND gates [A3] to an eight state counter (U383) [B3]. The purpose of the two NAND gates is to permit an external clock signal to be connected to the CPU. The signal line CLKEN- is pulled low and the CLKX- line carries the external oscillator's signal.

U383 contains four flip-flops labelled PA,PB,PC, and PC. PC and PC are logically identical and are ganged to provide high fanout. They are, referred to collectively as the PC flip flop. These four flip-flops define precise hardware clocks that actually guide the CPU through the execution of microinstructions. They are sequenced by the logic at each D input. Assume that both signals SHORT- and PAUSE- are high. Straight forward analysis of the circuit verifies that flip-flops PA,PB, and PC will sequence through the eight states as illustrated by the small diagram to the left of the flip-flops [B2]. The eight P-periods are named P1, P2, P3, E1, E2, E3, P4, and P5. A microcycle is defined to be the time between adjacent forward edges of period P1 (or trailing edge of P5). One microcycle is the time it takes to execute one microinstruction. In this case, one microcycle is equal to eight P-periods, or 280ns.

Any individual P-period, or P-periods may be decoded from the PA,PB, PC flip-flops by combinational logic. Important timing signals are decoded from the Q and Q outputs of the P-flip-flops for use throughout the CPU. Most decoding is straightforward except for the CP12 signal [D4] which will be discussed later. Many signals include as a qualifier FRZ from the FRZ

flip-flop (U403). Signals that have a suffix NF (not freezable) are not freezable.

4-2. I/O TIMING

The I/O system timing is located in the upper left corner of sheet 8. A buffered image of the PC flip-flop (BPC) clocks a mod-5 counter [B,C43] at the end of every microcycle. The counter is implemented by three J-K flip-flops labelled TA,TB, and TC (U326 and U366) and counts as shown in the sketch above the flip-flops. The outputs of the three flip-flops are subsequently decoded into five distinct cycles (also referred to as T-periods) numbered T2, T3, T4, T5, and T6 that control I/O system operation. Five such T-periods are defined as one complete I/O cycle. The mod-5 counter is designed so that lock-up in an illegal state can not occur.

In general, the I/O system performs a specific task only once in one complete I/O cycle. The five T-periods are used for precise implementation of the overall task. This is analagous as to how the control processor guides a microcycle through execution using the more precise P-periods.

The TA,TB, and TC flip-flops are buffered through U306 [A42] for use by both the memory controller and the Dual-Channel Port Controller (DCPC) accessory.

4-3. Freeze

The purpose of the CPU freeze is to selectively disable control processor microinstruction execution for a certain period of time without disabling other computer activities. The freeze is used to synchronize the control processor with the I/O system, to relinguish control to DCPC, and to avoid memory refresh cycles. With the exception of I/O related activities or memory refreshing operations, all computer operations are suspended when the freeze is invoked. Therefore, destinations of the "not freezable" signals are usually the I/O, accessory, and memory systems. A freeze flip-flop FRZFF (U403) [C3] is updated at the end of every microcycle and determines if the control processor is to be frozen during the next microcycle. U364[C,D3] monitors the four conditions that may freeze the CPU. They are as follows:

a. RDORWT·(REFRESH- + DMALO-). If the control processor is preparing to initiate a memory cycle (RDORWT) and either a memory refresh or DCPC data transfer is forthcoming, a control processor freeze occurs. REFRESH- is usually generated by the memory controller at P4 of T5 through T6 to let the CPU know that memory resources will be occupied during T2 and T3 (when the refresh actually occurs). DMALO- is a look-head signal from DCPC that precedes the actual data transfer by two microcycles. It also lets the CPU know that memory will be used by DCPC and that a memory cycle should not be started by the control processor.

b. IAKSP·(T6 + DMALO-). When an interrupting I/O device gains the attention of the CPU, an interrupt acknowledge (IAK) signal is sent to the I/O system, and the device interrupt address (select code) is latched into the Central Interrupt Register (CIR). The 2100 series I/O conventions specify

that IAK must occur during T6. Therefore, if the control processor attempts to generate an IAK signal at a T-period other than T6, a freeze will occur until T6, when IAK may legally occur. Also, if a DCPC cycle has been requested by an I/O device simultaneously with a pending interrupt request of a second lower priority I/O device, the interrupt address at the inputs to the CIR will not equal the select code of the interrupting device, but rather of the device requesting DCPC servicing. Again, the IAK related activities are delayed until after completion of the DCPC cycle.

c. IOGSP·(ENF + DMALO-). The signal IOGSP is generated by the control processor when it recognizes that an I/O instruction is to be executed. Since the control processor is normally executing microinstructions independent of T-period designations and 2100 series I/O conventions specify that certain operations occur synchronous to specific T-periods, a linking of the two timing systems must occur. The control processor therefore freezes until ENF is true. ENF occurs at every T2. Once synchronized, the control processor may execute the I/O instruction at known T-periods.

   The DMALO- condition used to be a required term, but subsequent design changes have eliminated the need for it. It is a redundant term and does no harm.

d. DMACYC-·$\overline{\text{T5}}$-. The signal DMACYC- is low during the complete I/O cycle (five T-periods) that the DCPC is using to transfer data. ANDing this signal with T5- makes this freeze condition a pulse four T-periods long (T6,2,3,4). Therefore, the freeze condition prevails during T2,3,4,5 of the DCPC cycle. The CPU is effectively frozen during the duration of the DCPC cycle in order to provide a clean, interference-free interface between the control processor activities and the DCPC.

4-4. Long/Short Microcycles

The arithmetic section of the CPU is designed to accomodate 175ns microcycles. There are only three circumstances when the computer is not capable of executing at such a fast rate. They are as follows:

a. The control processor may decide to parallel-load the CMAR at the end of P3 of any microcycle in order to execute a branching microinstruction. If a microcycle were only 175ns long, only two P-periods (70ns) would remain in the microcycle. This is not enough time to access control memory to fetch the next microinstruction (about 140ns is required).

b. Certain I/O interfaces may not be able to work with less than 196ns T-periods (the standard enforced by the 2100 computer) during the execution of an I/O instruction. Therefore, during T3, T4, and T5 of every executing I/O instructions, a microcycle longer than 175ns may be required.

c. The Memory Expansion Module (MEM) is unable to complete certain interfacing tasks in 175ns i.e., Q0, Q2, and Q6 MEM microorder commands (see Microprogramming Manual). Therefore, all microcycles are 175ns long unless one of the above conditions is detected.

The purpose of gates U404 and U389 [2B,C] is to detect these special conditions early in every microcycle before PB is high (at P3). If no special conditions are present at P3, the SHORT signal (U405 pin 12) will go low forcing all D inputs to the P-flip-flops high. At the next clock period (end of P3), all three flip-flops are clocked to the P4 state. After P4, counting proceeds to P5 since PB is low. In this case the state counter has counted P1 through P5 for a total of five P-periods or 175ns.

The P-flip-flops utilize an eight state gray code which is a binary representation that is characterized by only a single flip-flop changing at any common clock pulse. This permits spike-free decoding, but moving instantaneously across an odd number of states can potentially cause problems since more than one flip-flop must change states. The P3 to P4 jump in a short microcycle could mean states P2 or E3 could be entered briefly. This hazard, though, is of no concern in the CPU because neither P2 or E3 is specifically decoded or used.

4-5. Pause

The processor or the DCPC accessory may request an operation of the main memory while the memory is completing another task. The memory system informs the CPU that it is busy by lowering the MBUSY signal [IC] which is synchronized with the master CPU clock. There are five microorders that effect memory. They are as follows: READ (OP field), WRTE (OP field), RJ30 (SP field), and TAB (SB and ST fields). These conditions are detected by U406 [3C]. After PB- becomes high (at P3), the PAUSE- line may go low forcing the D inputs to the P-flip-flops to the binary equivalent of P3. The P3 state will be cycled back into the P-flip-flops until MBUSY- goes high when the microcycle is completed. The result is a P3 interval greater than the normal 35ns.

The MBUSY- signal is synchronized by separate dedicated flip-flops U428 and U403 [2C]. The second flip-flop U403 is included to decrease the probability of an oscillatory signal arriving at the counter due to inadequate set-up time at the first flip-flop. MBUSY- is synchronized since it can change at the end of any P3 state and thus cause erroneous sequencing of the state counter. All other qualifiers are valid and stable early in the microcycle. The WRTE in the OP field will pause the control processor. The TST signal from the DCPC is included in the term (DMALO + REFRESH)·(TA·TB-)=>DMALO·T3. If memory is busy during T3 of a DCPC cycle, it is due to either a refresh or a read operation (TEN). In any event, the CPU must be paused until the memory operation is complete. The signal CPUTEN- is the TEN as issued by the CPU.

The signal RDORWT [2D] is masked by both DMALO- and DMACYC·T5-. This is because the RDORWT line may be high will the control processor is frozen during a DCPC cycle. Extra pausing may occur unless this input is suppressed.

The extra pausing is not a problem, but it would lower overall computer performance if DCPC were operating. It may also be noted that CPUTEN- [2C] is suppressed while DCPC has frozen the computer [40B].

There are two other conditions that can pause the control processor at T6. These are REFRESH- + DMALO- [2C]. The purpose of both is the same to insure that no memory cycle is in progress at the end of T6. Recall that REFRESH- occurs at T5 and T6 in anticipation of the refresh cycle itself at T2 and T3. Likewise, DMALO- precedes any DCPC cycle that may itself wish to use memory as early at T2. This condition insures that no memory cycle is in progres and no interference occurs.

4-6. CONTROL PROCESSOR

This section attempts to explain the operation of the control processor. Because the control processor signals are greatly interrelated with other sections, some discussions will be brief and, as in the previous section, complete explanations may be delayed to one of the following three sections.

4-7. Run Flip-Flop

The Run Flip-Flop {RUNFF}(U416)[5,6A] is the most important control flip-flop in the computer. The RUNFF commands the control processor to either process microinstructions that fetch and execute main memory instructions or execute microinstructions that perform operator panel scanning tasks in the HALT mode. The RUNFF is buffered on sheet 6 [32D] to actually illuminate the light emitting diode (RUN) that is viewed on the operator panel.

The RUNFF is set in one of three ways. Pressing the RUN button (RUNB) on the operator panel will set the RUNFF if the run enable (RUNE) signal is high. In the "A" version of the E-Series a key operated swtich is used, and in the "B" versions and the F-Series, a slide switch is used. RUNE is high if the switch is in the OPERATE position and is low if it is in the LOCK position. The RUNFF may also be set if the SRUN microorder is executed in the SP field or if the power fail logic detects an auto restart condition (PWUST). This will be discussed in more detail in a later section covering the Input/Output Section.

The RUNFF is reset when a microinstruction with SHLT in the SP field is executed, when PONB is low (refer to section 4-27, when a HLT instruction is executed (HLTIO-), when the HALT button on the operator panel is pressed and the RUNE signal is high, and when a memory parity error is indicated and switch A1S1 is in the HALT position. the RUNFF is also reset when an I/O interface wishes to call the Remote Program Load (RPL) facility of the computer. This requires directing the control processor to the operator panel microroutines. The interface pulls down the RUN line in the I/O backplane when the RUNFF is set. BRUNFF is high, but the RUN line will be low. The conflict situation is sensed by U354 [5A] and RUNFF is reset. Everytime BRUNFF is set, P3 is used as an added qualifier to account for the propagation delay to the RUN signal through U349. Since the BRUNFF is updated at the end of every microcycle, P3 is a safe qualifier.

The internal logic in the computer is initialized by the PRESET- (PRST-) line [5B]. PRST- is primarily created by the PRESET button on the operator panel, but may appear if PONB is low (see section 4-27). Additionally, a PRESET will occur if the loader-ROMs are enabled (LDREN) during T5 in the halt mode. This obscure condition arises from the need to preset the computer during an RPL operation when an operator is not present. The loader subroutine performs this additional function.

The Parity flip-flop {PARFF}(U376,U396)[6B] drives the operator panel LED indicator and is set when the PE signal is sent by the memory controller. It is cleared by either the PRST- signal or the reset parity error (RSPE-) signal that originates from the memory protect (MP) accessory when it interrupts.

4-8. Timing

CP12 is a 70ns clock pulse during P1 and P2 which is dedicated to the control processor. CP12 is produced by a cross-coupled NOR flip-flop (U381 [4D] on sheet 1) because it has the unique characteristic that it is freezable and appears at the beginning of the microcycle. The operation of the flip-flop can be easily verified. The qualifier ENCP- (enable control processor) is normally low and is high only when the repeat flip-flop {RPTFF} is set, which causes multiple executions of the same microinstruction.

The implementation of CP12 is hazard free except for the case when the FRZ- input changes right after the FRZFF- is clocked. In this case, multiple CP12 pulses may be generated. Future memory controllers (REFRESH) and DMA devices must consider this hazard and allow adequate set-up time for the freeze flip-flop's freeze condition inputs with respect to the signal PC.

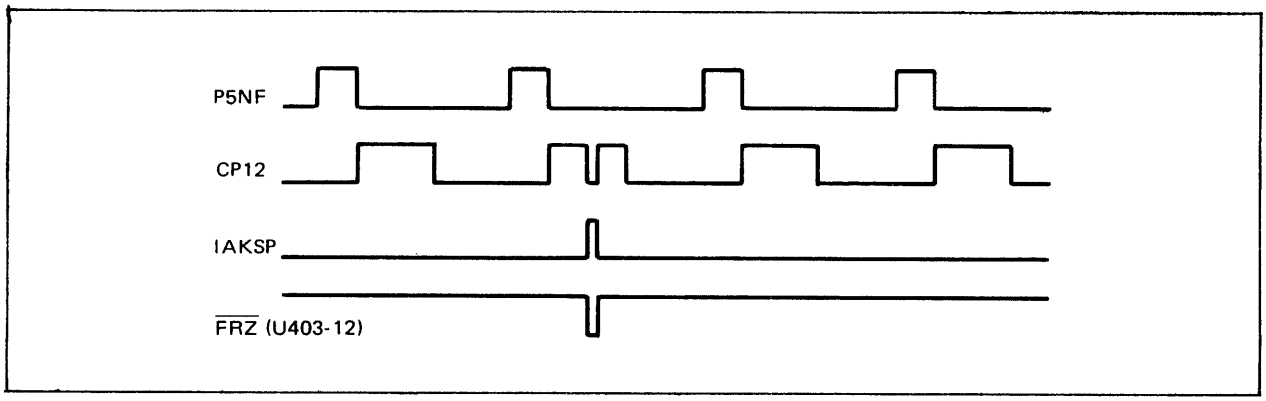


Figure 4-1. Timing Implementation Waveforms

4-9. Control Memory Address Register (CMAR)

Sheet 5 includes the major registers and logic associated with the control processor. The CMAR is implemented by four high speed counters 74S163 (U103,123,143, and 163). The rising edge CMAR clock is CP12+E1E2 [26B]. CP12 is responsible for incrementing the CMAR at the beginning of the microcycle. If a branch is detected, LCMAR (Load CMAR) goes high requesting a long

microcycle (recall from sheet 1) and enabling E1E2. At E1E2 (end of P3) the CMAR again may be clocked.

If the enable inputs (pins 7 and 10) to the CMAR are high, it may count; if they are low, it will not count. Since the enable inputs are high only during P5 and P1 (P51NF) it will easily bracket the beginning of a microcycle and CP12 will increment the CMAR. The load control (pin 9) is low during the middle of every microcycle when a branch is to occur so the parallel load operation is synchronized to the clock at E1E2.

The signal LCMAR is the result of straightforward combinational logic [25,26B,C] that looks at all of the ways the control processor may branch. Recall from the rules of microprogramming that the three OP fields RTN, JSB, and JMP can force a branch if no CNDX is in the SP field (word type 4), or if the condition interrogated by the COND field is met (word type 3). The MET-signal is low if the condition specified by the COND field is met.

JTAB will force a branch if the indirect flip-flop {INDFF}[40C] is not set (i.e., no INCI has preceded the JTAB in the SP field), or if the condition specified is met. RTN in the SP field always forces a branch. A special return (SPRTN) is requested if the micro-orders ASG or SRG2 require branches. The PONB term will be explained later when the power fail system is discussed in the Input/Output Section.

The inputs to the CMAR are the outputs of the nine multiplexers (U84,85,104,105,124,125,144,145, and 164) that are switched by eight SP field micro-orders: RTN,JTAB,CNDX,ION,RJ30,J74,IOG, and NOP. The actual logic implementation is simplified by using the least significant three bits of the SP field of the MIR as the selector inputs to the multiplexers. The multiplexer data inputs can be connected directly to the appropriate source that matches the function called by the SP modifier. A special case is RTN in the OP field which must force the selector inputs to match the code of RTN in the special field. Using the associative bit pattern technique, three other SP field micro-orders (STFL,RPT, and IOFF) may be used in word type 4 microinstructions without modifying and jump target because their least significant three SP bit patterns are the same as NOP. Table 4-1 shows the bit patterns at the selector inputs of the multiplexers.

Table 4-1. Multiplexer Select Input Bit Patterns

Special	MIR 2 1 0	Selector 74LS151	Selector 74LS153	Comments
RTN	0 0 0	111	00	also RTN in OP field
JTAB	0 0 1	110	01	
CNDX	0 1 0	101	10	
ION	0 1 1	100	11	
RJ30	1 0 0	011	11	
J74	1 0 1	010	11	
IOG	1 1 0	001	11	
NOP,RPT, STFL,IOFF	1 1 1	000	11	

Although there are eight different ways to branch through CM, there is substantial similiarity among five of them. ION, RJ30, J74, IOG, and NOP are all unconditional jumps that modify the target address in the microinstruction in the lower four bits. Taking advantage of this, all five codes map to the same selector bits of the 74LS153 and the lower four bits are mapped individually through the 74LS151. The other three modifiers are unique and call their own selected pattern. RTN requests an address from the Save stack, JTAB requests an address from a PROM jump table based upon the IR, and CNDX calls for a nine-bit (512-word) address displacement around the current CMAR address.

4-10. Halt or Interrupt Detection

When either a halt state or a pending I/O interrupt condition prevails in the computer, an interruption of the normal fetch-to-execute flow is made. The CMAR is loaded with the constant 6 when JTAB is executed without a preceding INCI at CM location 1. The special conditions are detected by U126-6 [25,26B]. U166 disables the lower jump table U106 [26A] forcing its output to all ones, and the signal JTAB-·HOIFF- sets the outputs of the multiplexers, except U85 and U104, to zeros. Since the multiplexers are switched to the jump tables, U85 and U104 pass their ones to their outputs consequently loading the constant 6 into the CMAR. The terms PONFF and PONFF- will be discussed with power fail in the paragraphs on the Input/Output Section. The pull-up resistors on LU50 and LU53 are non-functional.

4-11. Save Stack

The save stack (U102, 122,142, and 162)[30A,B] holds microprogramming subroutine addresses. Three levels of micro-subroutine calls are permitted by the control processor, but the stack is actually implemented by a four-word dual-port RAM. Control logic makes the RAM appear as a stack. The fourth register normally holds an all-zeros address that is used when more micro-subroutine returns are executed than branches. Three levels of micro-subroutine calls are available for executing an instruction after the

fetch microinstructions. Initially a read (or "RTN" pointer) addresses an all zeros word and a write (or "JSB" pointer) addresses the first available save word as shown in figure 4-2.

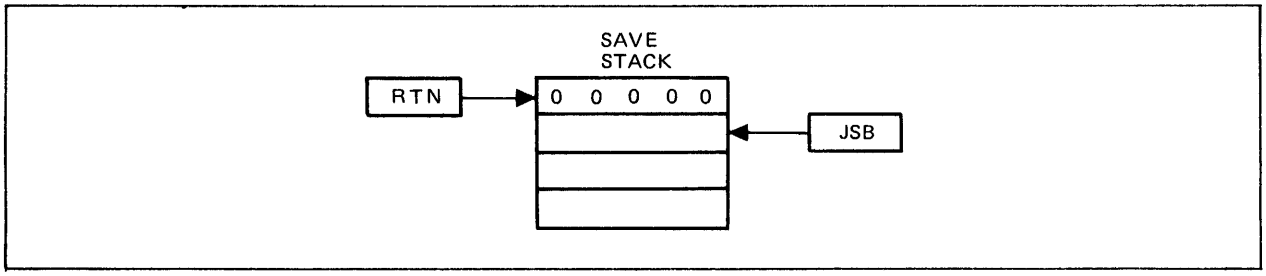


Figure 4-2. Save Stack RTN and JSB Pointers Initial Positions

Any executed return will load the contents of the CMAR at the address pointed to by the RTN pointer. In this case a return to CM location 0 will occur. An executed JSB will write the return address into the location pointed to by the JSB pointer and then advance the pointers. If a JSB occurs from location 200, the pointers will be advanced to this state as shown in figure 4-3.

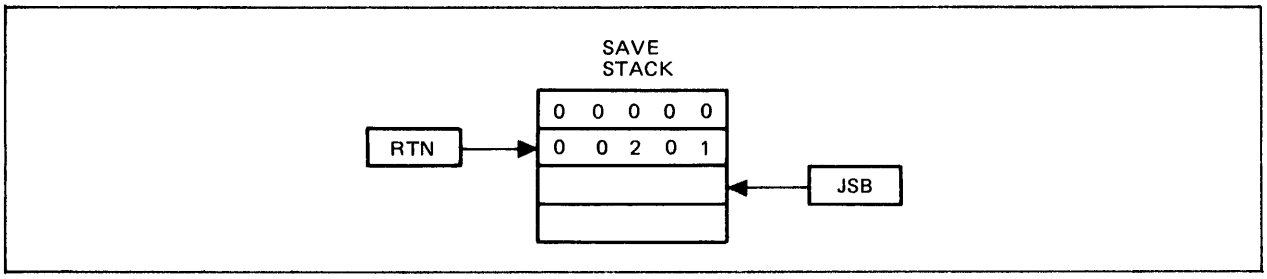


Figure 4-3. Save Stack RTN and JSB Pointers After Execution

An executed RTN will place the contents of the save at the address pointed to by the RTN pointer (201 in this case) into the CMAR and the pointers will move back to the previous state after the fetch. Deeper levels of subroutine operate in a similar manner with the pointers moving back and forth always one away from each other. When one more RTN is executed then a JSB, a branch to zero (FTCH) is made.

Refer at this time to the right side of sheet 5 and examine how the Save Stack is implemented in the hardware.

The RAM includes a write enable, a read enable, two write and two read address lines. Simultaneously reading and writing of two different locations is permitted. This is the key to the selection of the control logic used.

U161 [30B] is a left-right four-bit shift register set up in a circular rotate configuration which actually holds the read address and write address pointers. The two mode controls (pins 2 and 7) manipulate the pointers in response to subroutine branches and returns. S0 is high when a RTN is executed, S1 is high when a JSB is executed, and JTAB forces both S0 and S1

high. Synchronous to P3, and RTN causes a right shift, JSB causes a left shift, and JTAB forces a load.

Three of the four outputs of the shift register become the read and write address pointers to the 74LS670 Save Stack. The read enable is always low since reading of the return address always happens, and, assuming that U182-4 [30C] is high, a write occurs every P12. Therefore, the saved word pointed to by the WRTE pointer is actually written every microcycle with the contents of the CMAR. The CMAR points to one word beyond the address of the currently executing microinstruction. When a JSB is executed, both pointers are advanced beyond their present positions. The last write is left behind and is the valid future return address.

FTCH and JTAB are specific hardware-oriented microorders that are never used by user-microprogrammers. They are infrequently used in the base set. In regard to the Save Stack, the most important usage is at CM locations 0 and 1. First the FTCH- occurs clearing the 74LS194 (U161) shift register and setting both pointers to 00. Next the microinstruction at CM location 1 is executed. During P1 and P2 a save stack write occurs storing the present value of the CMAR into the location pointed to be the JSB pointer (00). Recall that the CMAR is equal to 2 at this time. Since the microinstruction located at CM location 2 includes a JTAB, the CMAR 1 and 2 inputs to the Save Stack are masked by the JTABSP- inputs to the AND gates U166 [30A], thus loading all zeros. After P3, JTAB causes a load of constant 1001 into U161 completing the initialization of the control logic as shown in Table 4-2.

Table 4-2. Save Stack Execution Routine

FUNCTION	74LS94 ABCD	RTN ADDR	JSB ADDR
After FTCH	0000	00	00
After JTAB	1001	00	01
After 1st Subrtn	0011	01	11
After 2nd Subrtn	0110	11	10
After 3rd Subrtn	1100	10	00

Note that the write address (JSB) is the read address (RTN) for the next subroutine level. RAM 00 holds all zeros, RAM 01 holds the first subroutine return address, RAM 11 the second, and RAM 10 the third as shown in figure 4-4.

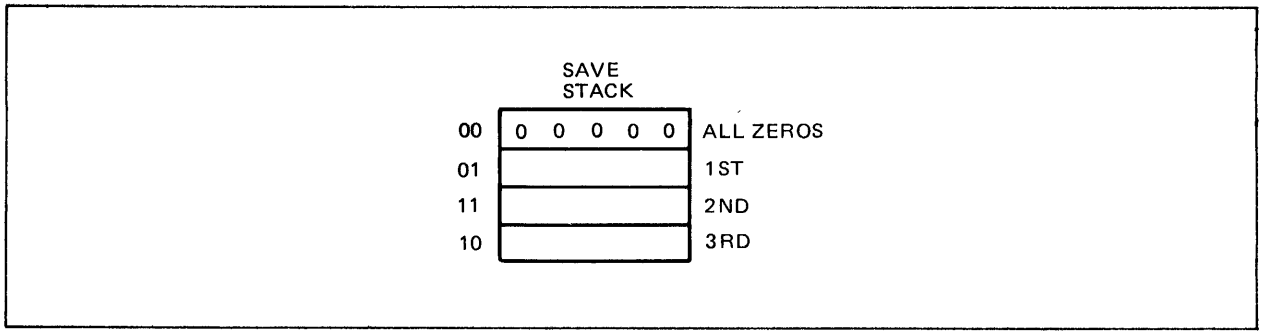


Figure 4-4. RAM Locations

JTAB also occurs at locations 3 and 307 in the base set. The JTAB at 3 in the MRGIND routine again forces zeros into the Save Stack. While JTAB is being executed the CMAR is equal to 4 (advanced by one), but U166 masks this bit before loading it into the stack. The JTAB at 307 in the operator panel routines is not rquired to load zeros onto the stack, but rather to initialize the Save control logic. This application is discussed in greater detail in Section 5 Base Set Microprogram Discussion.

The purpose of U222 [30C] is to detect the third JSB state and disable the write pulse which would modify RAM 00 which is the all zeros word and must remain valid.

4-12. Control Memory

The CMAR drives the base set control memory PROMs (U21, 23, 41, 43, 81, and 83) [26,27,28C,D] on the CPU board and supplies signals to the edge connector J2, which are used by additional firmware routines that may reside on the Firmware Accessory Board (FAB), Writable Control Store (WCS), or User Control Store (UCS). 4K PROMs, arranged 512 words by eight bits, can be used on the CPU board. Three are required for modules 0 and 1 (U21, U41 and U81) and three are required for modules 2 and 3 (U23, U43 and U83). When 4K PROMs are used, only the "4K" jumpers are loaded into U121 [26D]. U121 pins 8-9 and 1-16 are installed, W7 is installed, and U121 pins 6-11 is removed.

The 8K jumper references are for possible future expansion to 8K PROMs.

U66 [25,26C] decodes the base set module address and U46 [25,26C,D] qualifies this with two external, normally high signals called RMX- and ECSEN-. RMX- is low when an external control memory device wishes to override the basic CPU control memory and present its own microinstructions to the control processor. ECSEN is used by an external control memory (FAB, WCS, or UCS) to permit it to be configured as the base modules 0 through 3 and automatically override the CPU control memory. BSEN- [26C] is generated by the CPU to the FAB to override the FAB when module 0-3 are addressed or when RMX- is asserted.

4-13. Micro-Instruction Register (MIR)

The 24 bit microinstruction from control memory appears on the ROM bus and is

clocked into the MIR (U24, 44, and 64) [24,28D] at the beginning of the microcycle by CP12. The outputs of the MIR travel through the CPU. Simultaneous with loading of the MIR, the CMAR is incremented to point to the next microinstruction.

4-14. Decoders

The computer employs a vertical format for its microinstruction. Vertical format microinstructions are often characterized by microorders encoded into several fields of operation. Each is usually decoded to a unique function for which it is responsible. The advantages to using a vertical format include cost (a relatively narrow microinstruction width is required) and ease of microprogramming. In a horizontal format a microinstruction is substantially wider (more than one hundred bits is not uncommon) and little or no decoding is required because bits in the microinstruction may themselves be responsible for a single function. The advantage is speed (no decoding circuits may be required reducing microcycle time).

A 24-bit vertical format microinstruction is used for ease of microprogramming and compatibility with earlier HP microprogrammable minicomputers.

Sheet 4 includes the decoding circuits for the microinstruction fields. The fields not entirely decoded are the SB and ST fields that reference the sixteen registers in the RAM, the ALU field, and the COND field. Understanding the operation of every decoded microorder requires a parallel review of the definition of each microorder.

4-15. Operation Field

U264 and U265 [20A,B] are high speed 74S138 three-to-eight decoders that decode the OP field microorders. IMMOP- [20A] is combined with MIR bit 18 to determine whether the lower byte (LBE-) or upper byte (UBE-) is to be enabled onto the S-bus. The READOP- and WRTEOP- microorders logically set their respective flip-flops (U286)[21A] at the end of the microcycle. The falling edge of the READ- or WRTE- pulse actually initiates the memory cycle. The signal DMAFRZ is low when DCPC requires the S-bus during a data transfer. Its purpose is to disable the decoding of the IMMOP- signal to prevent data from being enabled onto the S-bus.

4-16. Special Field

All thirty-two special field microorders are decoded by U184, U185, U204, and U205 [20C,D]. U185 is a 9334 decoding latch that decodes microorders in a positive-true sense. P5 (at the end at the microcycle) is usually the timing pulse used to actually perform most operations in the computer. Signals such as STFLSP-, CLFLSP-, PRSTSP- and SOVSP- are strobed with P5 to perform their appropriate tasks. FTCHSP- is strobed with P3 and passed to the Memory Protect (MP) accessory to, among other functions, store the value of the M-register into the violation register. Recall from the definition of FTCH that the value of M at the beginning of the microcycle is stored into the MP violation register. Note also that at CM location 0 IRCM appears in the ST

field and FTCH in the SP field. Since IRCM modifies M at P4 of the same microcycle, FTCH must be sent to MP before P4, so P3 was selected. INCISP and SHLTSP are strobed with P3.

MPCKSP requests the MP accessory to check the value of the M register against the fence register for possible violation. It is qualified by DMACYC·$\overline{T5}$- because MP actually checks the M-bus against its fence register. While the M-bus is normally driven by the M register it actually holds a DCPC memory address during a transfer cycle. Since the control processor is frozen during a DCPC transfer and DCPC must never cause a MP violation, MPCK is suppressed at this time.

CCTR- [21C] enables the CNTR register to count up or down and is the equivalent of DCNTSP- + ICNTSP- + RPTFF-. The FRZFF- qualifier is no longer necessary and causes no problems.

4-17. Store Field

The sixteen SB and ST field micro-orders that are not associated with the sixteen registers contained in the RAM are decoded by U224,225,244, and 245. The RAM addressing logic is discussed in Section 4 Arithmetic/Logic Section. All registers on the CPU board are loaded at the end of P5 (end of the microcycle) the IR,CNTR, and M which are loaded from the S-bus at P4.

The ST field is decoded by U224 and U244 [22B]. The decoding is disabled by the combinational logic at pin 4. The first term JMPOP + JSBOP + CNDXSP is a detection of word type 3 or 4 microinstructions. Such a microinstruction does not operate on data and hence does not require usage of the S-bus. However, MIR bits 10 through 14 enable a register onto the S-bus and MIR bits 5 through 9 decode a register to be stored, if not disabled. The second qualifier (U222-1) [22B] is an implementation of the definition of JTAB. JTAB is to suppress the operation of the ST field if an indirect MRG instruction is in the IR or if a JMP, JSB, ISZ, STA, or STB is in the IR.

A third qualifier inhibits the ST field if a halt or interrupt condition is pending as JTAB is decoded. JTAB will force the CMAR to CM location 6 as previously explained. The updating of the register stored in the ST field is suspended to provide a clean interface to the HORI routine.

Most decoded outputs from the ST field are straightforward so only the more complex need to be discussed. MRST is an OR condition of all three micro-orders that store into the M register. PGUPDT (Page Update) decodes the two micro-orders that may load the upper five bits of the M register. ZPG (zero page) detects the condition that clear the upper five bits of the M register. When IRCM is decoded, the instruction is on the S-bus, and a low SB10 indicates a zero page operation.

The A and B addressable flip-flops (ABFF) [38A] can be clocked at the end of every microcycle. Addresses 0 and 1 are detected on the M-bus and the flip-flops are set accordingly. When DCPC uses the M-bus the ABFFST signal is suppressed. JTAB also inhibits updating the addressable flip-flops (see

Microprogramming manual). PRST- in the special field (PRSTSP-) also holds off ABFFST and causes RABFF- [21D]. This allows access to memory locations 0 and 1.

The purpose of 100 is to enable the S-bus onto the I/O-bus which is performed by the signal EIOBO (enable I/O-bus output) [24C]. The discussions of the other tems of EIOBO is deferred to a later section.

4-18. S-Bus Field

The SB field is decoded by U225 and U245 [22C,D]. Decoding is suppressed if DMAFRZ- is low (the DCPC requests control of the S-bus or if IMM- (IMMOP- Rev C) has been decoded in the OP field. Recall IMMOP implies a word type 2 wherein an eight bit byte is to be enabled onto the S-bus rather than to a register.

IOI (U266) [22D] is typically used to execute I/O input instructions. The literal definition of IOI is to gate the I/O bus onto the S-bus. There are, however, certain cases in this context when the I/O bus must not be enabled onto the S-bus. All I/O interfaces reside on the I/O bus, but accessories with select codes less than ten reside on the S-bus. To enable the I/O bus onto the S-bus when an I/O input instruction is requesting data from, say, DCPC would be extremely serious. Therefore, the gating of the I/O bus is inhibited if select codes one through seven {SC1-7} are detected. Select code 0 does enable the I/O bus but this is just to maintain compatibility with earlier computers when executing LIA 0 or LIB 0 instructions. Therefore, IOIEN- is qualified with SC1-7 (U242-8) [23C] and enables both the lower byte of the I/O-bus {ELIOB} and the upper byte of the I/O bus {EUIOB}.

ECIR- is lowered enabling the central interrupt register (U170) [17D] onto the lower byte of the S-bus when CIR is in the S-bus field. ECIR- also lowers EUIOB- forcing the undriven I/O bus onto the upper byte of the S-bus. The result on the S-bus is the CIR with leading zeros.

The other terms that enable the I/O-bus onto the S-bus will be discussed later.

4-19. Conditional Field

The control processor jump conditional logic is illustrated on sheet 6. MIR bits 15 through 19 compose the COND field and are selectors to a high speed multiplexing tree (U316, U335, U336, U355, and U356) [32,33A,B,C] that selects one of thirty-two possible jump conditions. MIR 14 is the reverse jump sense (RJS) bit and is used in the last stage (U335) to complement the condition sensed. The signal output of the tree is the signal MET- which goes into the CMAR load logic on sheet 5.

Eight state and data flip-flops are buffered by U296 [32D] before entering the jump tree. U296 is clocked at the end of every microcycle that executes a word type 1 or 2 microinstruction. The information stored in these flip-flops is preserved during word type 3 of 4 microinstructions.

Although TBO- cannot be tested it is required for the SRG2 branch test. The RUNFF and HOI signals are buffered to synchronize those tests to the beginning of the microcycle. Recall that the RUNFF (and hence HOI) may be asynchronously set from the operator panel RUN button.

For the branch logic to function properly it is assumed that all conditions to be tested are valid well before the ultimate parallel-loading of the CMAR. An unstable LCMAR line [26B] at the end of P3 may cause problems with CMAR loading or the P-flip-flops to change state.

4-20. Arithmetic/Logic Section

Portions of the Arithmetic/Logic Section are located among sheets 2,3,6, and 7. Sheets 2 and 3 are the most important and contain all of the data registers and the four 16-bit data busses in the CPU. The tri-state bi-directional positive-true S-bus is the main data bus upon which the SB field of the microinstruction enables a register for the duration of the microcycle. The S-bus drives the ALU which in turn drives the RS and the ground-true T-bus. The CTL-compatible bi-directional positive-true I/O bus can be gated from or onto the S-bus. The tri-state positive-true M-bus is shared by the CPU's M-register, the DCPC's memory address registers and may also be enabled back onto the S-bus.

4-21. RAM

There are sixteen registers that are physically located in the 16-word by 16-bit RAM (U114,115,134, and 135) [8B,C] on the left side of sheet 2. The registers contained in the RAM are the P (program counter), S, X, Y, SP, and eleven scratch registers.

The 74S189 RAM is a difficult device to control because it is a single-port device. Simultaneously writing and reading at different locations is not permitted. The control logic circuits above and below the RAMs determine how they are to operate. The microinstruction control bits of the SB and ST fields are multiplexed by U194 [8B] into the address inputs of the RAM. The Write Enable- (WE-) [8D] line is low during P5 of any microcycle that the ST field calls for a RAM register to be loaded. The Chip Select- (CS-) input is low when a RAM register is to be read onto the S-bus or stored from the T-bus.

Recall from the microprogramming manual that MIR14 is high if a RAM register is to be enabled onto the S-bus. MIR 9 is high if a RAM register is to be loaded at the end of the microcycle. The exception is PNM in the ST field that stores the M register off the S-bus and the P register off the T-bus during the same microcycle. This condition is detected by U193-6 [7D] by generating a composite RAMST signal.

There are four possible permutations of RAM registers between the SB and ST fields. They are as follows:

a. Consider the case of a microinstruction that has a RAM register neither in the SB nor ST field. In this case both MIR14 and RAMST are low, so both

CS- and WE- are high and the RAM remains completely inactive during the microcycle.

b. The case of a microinstruction that calls for a RAM register in the SB field, but not in the ST field. MIR14 is high so CS- is low during the entire microcycle. This is qualified by both IMMOP- and DMAFRZ- for the same reasons the SB field decoders (U225 and U245 [22C,D] on sheet 4) are disabled. RAS- is high because RAMST is low, so the SB field address is passed through the RAM address multiplexer U194 [8B] for the duration of the microcycle.

c. Consider the case of a microinstruction that includes a RAM register in the ST field, but not in the SB field. MIR14 is low, but RAMST is high. CS- and WE- will be low during P5 and RAS- will be low during P51. The RAM therefore remains inactive until P5, when the write operation occurs. When both CS- and WE- are low the RAM outputs remain in the disabled state insuring that the S-bus is not inadvertently driven. Timing considerations for this and the final example are very critical (see figure 4-5). Assume perfect timing standards at the output of the P flip-flops. RAS- begins switching the address multiplexer one gate delay away from the P flip-flops. The address is switched to the ST field for both P5 and P1 to completely bracket the write operation. WE- is the delayed RAS- signal gated with BPC. CS- is also P5 in duration (approximately three gate delays (10-15ns) away from the timing flip-flops). The delay element on RAS before U314 (WE-,CS-) [8D] insures adequate address setup time at RAM's (74S189). The 470 ohm resistor on WE- helps insure that CS- will not be asserted after WE-, this would be destructive.

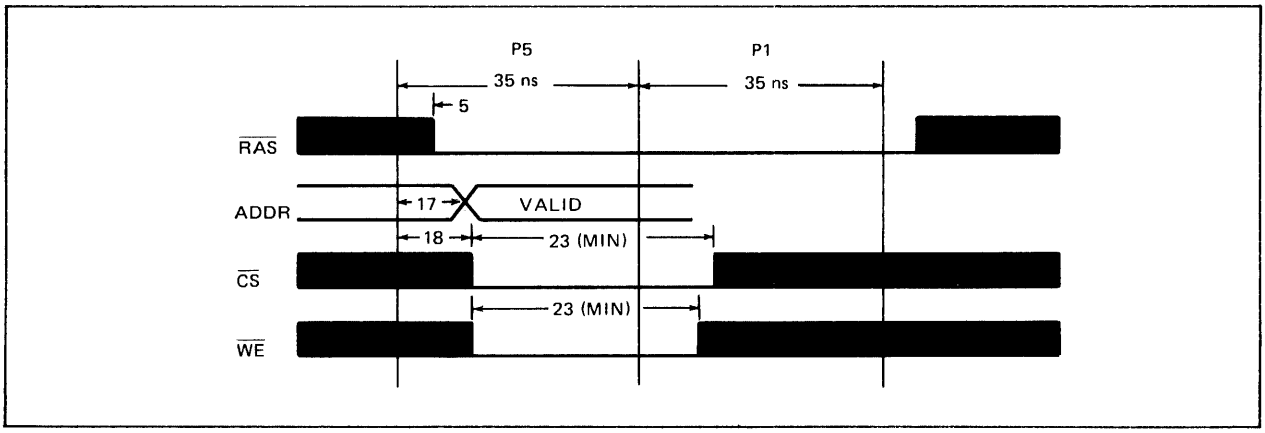


Figure 4-5. RAM Timing Diagram

d. The last case to consider is when a RAM register is to be enabled onto the S-bus and a different RAM register is to be stored. Both MIR14 and RAMST will be high so CS- will be low during the entire microcycle. RAS- and WE- function as in the previous example. The unique problem is one of data integrity. Recall that when WE- is low during P5, the RAM enters the tri-state disabled state. It may be possible that the undriven S-bus

could invalidate the T-bus data before the RAM write is complete. To insure this does not happen, an S-bus latching scheme was devised that uses the I/O-bus to hold the S-bus data.

When both MIR14 and RAMST are high, the S-bus is gated out to the I/O-bus. For the last half of the microcycle when both the S-bus and I/O-bus are stable, the I/O-bus is gated back onto the S-bus forming a latch. The latching operation keeps the S-bus valid through P5 permitting safe completion of the microcycle. These operations are initiated by U201 [23C] on sheet 4. The switching function is disabled when the control processor is frozen primarily DCPC transfers data between the S-bus and I/O-bus and such latching would be destructive.

This design idea may be viewed by some with caution since latched busses may become noise catchers. In actual practice, it is problem free since noise on either bus at the end of any microcycle is non-existent.

4-22. A and B Accumulators

The A-register (U36 and U96) [10A] and the B-register (U76 and U136) [8,9A] shown at the top of sheet 2 are implemented by eight-bit left-right shift registers (74198). Both ground-true registers may be enabled onto the S-bus through the inverting tri-state multiplexers (U154, 155, 174, and 175) [8,9,10B].

The registers are placed onto the S-bus when ABEN- (A or B enable) is low. Register selection is accomplished by the BRS (B register select) line. These two control signals are generated on sheet 7, ABEN- [40A] and BRS [39B]. ABEN- is low if A, B, or CAB is in the SB field of the microinstruction, or if TAB is in the SB field, and either the A or B addressable flip-flops are set. BRS is high when the B register is actually desired, if B is in the SB field, if CAB is in the S-bus and IR bit 11 is high, or if TAB is in the SB field and the B addressable flip-flop is set.

The registers are controlled by two mode control inputs S0 and S1. These permit loading and shifting of each register. The appropriate load or shift operation always occurs at the end of P5. The purpose of PONB is to clear the A and B registers on an initial power-up.

The A-register mode controls are generated on sheet 7. The signal ASTORE- [40B] is low forcing both AS0 and AS1 [39,40C] high when the A register is to be loaded. AS0 is also high (right shift) when MPY or an extended right shift is called for in the OP field. AS1 is also high (left shift) when DIV or an extended left shift is called for in the OP field.

The signal BSTORE [40B] is low forcing both BS0 and BS1 high when the B register is to be loaded. BS0 is also high (right shift) when NRM right shift is decoded. BS1 is high (left shift) when a NRM left shift is decoded. Recall that all 32-bit shift micro-orders (ARS, CRS and LGS) shift the A register internally within the shift register. The B register is shifted in the RS via the S-bus and T-bus because it is difficult to alter the sign bit

of B within the register itself. The NRM micro-order (which can shift up to 48 data bits) operates on the B register in an explicit manner and the shift takes place within the register. If a carryout (COUT-) does not occur in a DIV microinstruction then the planned store of the B register is inhibited in preference to a left shift. U370-3 detects this condition. This condition is detected by U370-3 [39B,C].

4-23. Arithmetic/Logic Unit, Rotate Shifter, and L Register

The ALU is made of four high speed 74S181 ALU circuits (U32, U72, U92, and U132) [11B,C] and one carry look-ahead generator (U113) [11B]. The ALU is driven directly by MIR bits 15 through 19. The function of each ALU field bit pattern parallels directly the definition of the 74S181 devices. There are three special conditions that are decoded by U193 and U333 [10,11A,B]. When U333-8 is low the four function select inputs of the ALU are forced low. This subsequently sets the ALU to pass the inputs if the M input (MIR19) is low, or to complement the inputs if M is high.

Recall that MPY in the OP field is usually accompanied by an ADD in the ALU field. If the lease significant bit of the A register is zero (AR0), the ALU must be overridden to a PASS state, which is what happens since MIR19 for an ADD is low.

If the ASG microorder is in the special field and IR bit 2 (IR2) is low, then the ALU is forced to PASS since INC is usually in the ALU field and MIR19 is low.

A word type 2 microinstruction (IMMOP-) also forces U333-8 [10B] low. Recall that the PASS/COMPLEMENT bit of such an instruction is MIR19. When MIR19 is high, a complement occurs and when MIR19 is low, a pass occurs.

The outputs from the ALU are applied to the inputs of the L register (U30 and U50) [10D] and also to the 74S04 inverters. This was done for three reasons. First, it was decided that the T-bus would be ground-true in anticipation of the inverting 74S189 RAMs. Second, complemented outputs could be easily checked for all logical zeros output by U93 and U133 [11,12A] for the ALZ test condition. Third, in anticipation of the complex sign bit, rotate, and shift operations that must be merged between the ALU and RS, high speed AND-OR-INVERT gates (left side of sheet 7) could be used for this function. Total delay between the ALU and RS is limited to 6ns.

The RS logic is made of eight dual four-to-one multiplexers (U34, U35, U54, U55, U74, U75, U94, and U95) [12A-D]. They function as a high speed shifter, capable of shifting left one bit, left four bits, right one bit, or straight pass through in 9ns. Boundary bit conditions (sign bits and least significant bits) are more complex functions and are created on the left side of sheet 7.

4-24. Instruction Register

The IR circuit is located on sheet 3 [13-15,B,C]. The upper eight bits are implemented by two 74S175 quad D-type flip-flops (U213 and U233) with

complementary outputs. The lower eight bits are also called the CNTR and are implemented by two up/down counters 74LS191 (U215 and U235). Since the CNTR normally counts down (for RPT and DCNT conditions), the ICNTSP- signal is used as a mode control for incrementing the counter. The CNTR can be passed onto the S-bus if CNTR is in the SB field (U276) [15B,C].

The loaders are 1K PROMs (U196, 216, 255, and 256) [16A,B] arranged 256 words by four bits wide. The function of the CNTR bits is to drive the loaders as address inputs since eight bits may address 256 words. All loaders are addressed in parallel, but only one will have enabled outputs depending upon the upper two bits of the IR (pins 13 and 14). The selected loader will be enabled and complemented onto the S-bus through U275 [16C] if LDR is in the SB field.

4-25. M-Register

The M-register is a special two stage register, implemented by two octal latches 74S373 with tri-state outputs (U70 and U152) [17D] that directly drive the M-bus. A separate PAGE register is loaded from the S-bus by the PGUPDT signal and cleared by ZPG-. The PAGE operations are completed by P4 of the microcycle, and since the MRST signal is high during P4, all M register activity is completed transparently in the same microcycle. The 200pf capacitor on MRST [24A] was added to slow down the rising edge at the G input (pin 11) of the M registers to solve an early device problem on the early 74S373's. The M register does not drive the M-bus when the DCPC wishes to access main memory by lowering the DMAEN signal forcing pin 1 high. The M register or more properly the M-bus may be enabled onto the S-bus when M is in the Store field. The microprogrammer always sees the M register because when the DCPC seizes the M-bus, an image of the valid CPU M register is latched into U90 and U150 [18D] by the DMACYC-·$\overline{T5}$- signal.

U133, U156 and U176 [17,18A,B] of sheet 3 detect memory addresses 0 and 1 for use by the A and B addressable flip-flops.

4-26. Miscellaneous

Word type 2 immediate byte operations are performed by U110 and U130 [18D] of sheet 3. The byte constant that resides in the MIR is enabled as the lower byte (LBE-) to SB0 through SB7, or as the upper byte (UBE-) to SB8 through SB15 for Rev C CPU7's only.

There are several operations that utilize the S-bus while moving data from one subsystm in the computer to another, but do not drive the entire S-bus. Word type one DES and DSPI drive SB0, 6, 7, 8, 9, 10, 14, 15 and SB0 through SB7 respectively. The used bits are left to float high (true) during these operations. In operations which are sensitive to timing considerations (storing the current S-bus data), the unused S-bus bits are driven high (true) via the circuitry on sheet 3 of the schematics. Word type two immediate byte operations are performed by U89, 109, 129, and 130 [13-15A]. The byte constant that resides in the MIR is enabled as the lower byte (LBE-) to SB0 through SB7 (U109, U110) or as the upper byte (UBE-) to SB8 through SB15

(U129, U130). LBE- and UBE- determine whether MIR data or "ones" are gated on to the S-bus. Since LBE- and UBE- are both inactive during Word Type two CNTR and LDR, the appropriate unused bits of those particular operations are driven high (true) immediately to avoid any hysteresis problems from the upcoming store operations. U110, 129, and 130 are all enabled during LDR while U129 and U130 are only enabled during CNTR and U109 is enabled by IMMOP-.

The RPL configuration switches (U173 [8D] on sheet 2) are enabled to their appropriate S-bus bits by U172 when DES is coded in the S-bus field. the irregular S-bus bit selections are made to facilitate the actual base set firmware processing to approximate bit assignments of the S register when the IBL button is activated. The use of PARFF- (parity error flip-flop) as a qualifier for the descripter block (and S, ALU Bit 0) prevents a remote program load when a halt is encountered due to a parity error.

The special ASG and SRG skip logic shown on sheet 6 [35C] is a key link between the Arithmetic/Logic section and the control processor. Recall that the signal SPRTN- (special return) is low when neither the ASG nor SRG instruction in the IR will perform a program skip. While the SRG has only a single skip condition, the ASG has many skip tests. The combinational logic is a straightforward implementation of the rather irregular definition of the instruction groups.

The SRG shift and rotate logic appears on sheet 6 [34-36A,B]. The decoding scheme here is a direct implementation of the various shift combinations available with the SRG.

The first and second IR shift bits are multiplexed into the decoding logic by U253. The TBS0 and TBS1 are the selectors for the RS multiplexers. The outputs LWE (link with E), CSGN- (clear sign), ROT1 (rotate one), and SH1 (shift one) are combined with the outputs of the ALU to perform the sign and boundary bit handling appropriate of the SRG. U294 [35B] decodes the special non-obvious SRG instructions that manipulate the Extend register (see rule 4 under section 3-18 in the Operating and Reference manual). This logic was included to guarantee compatibility with earlier computers.

The four ways that the extend (E) register (U394 [42A] on sheet 7) is set are detected by the gate U393 74LS54[41A]. If the special ERxNE condition is detected, ALU0 is stored into the E register. If ELxNE is detected, ALU15 is stored into the E register. If ENVEOP has been coded into the OP field, a carryout (COUT) [40A] from the ALU will set the E register. ASG in the SP field will set, clear, or complement E depending on IR6 and IR7. However, if a COUT occurs simultaneously with ASG, E is set regardless of IR6 and IR7. Recall from the base set microprogram that a COUT may occur as a result of an increment of the A or B accumulators simultaneous with the IR6 and IR7 operations on E. Because of the sequencing rule of the ASG instruction, the COUT takes precedence. U375-8 [40-42B] detects the CLE SRG instruction and clears the E register.

Setting and clearing the overflow flip-flop {OVFF} (U394 [42B] on sheet 7) follows directly from the definitions of several microprogramming

microinstructions (SOV, COV, ENV, ENVE and ARS) and I/O instructions (STF 1 and CLF 1) [40-42B] and the MPP signal STOV-.

4-27. Input/Output Section

The I/O section logic circuitry is contained on sheets 8 and 9 with the I/O bus shown on sheet 2.

I/O-bus gating has been fairly well discussed earlier in this section. The I/O-bus is CTL-compatible and must be driven by emitter-followlwer current sourcing devices such as the 8T13/75121 AND-OR gates. The I/O-bus to S-bus gating is accomplished by two octal tri-state devices.

4-28. I/O Control Signals

The I/O control signals are generated on the upper left corner of sheet 8 and the right side of sheet 9. I/O timing has already been discussed in section 4-2. the column of emitter-follower devices to the right of the timing logic ar interconnected with IR bits and timing signals to implement the 2100 and HP 1000 I/O instruction set. This is summarized by figure 4-6. The critical flip-flop is U366 the IOG enable flip-flop {IOGENFF} [43C]. The flip-flop is set when a programmed I/O instruction is to be executed. Recall that the control processor freezes until T2{ENF} when IOG is in the SP field. When ENF is decoded and no DCPC cycle is pending, the control processor unfreezes and the IOGENFF is set (U365-6). DMALO- is used as the term in the freeze logic and DMACYC- is used as a qualifier for the IOGENFF logic, but at T2 both signals are in the same state.

The IOGENFF is used as a qualifier for all I/O control signals on sheet 8 except IAK (which is interrupt driven), ENF, SIR, and T3 (which are always present I/O timingsignals). All I/O control signals are executed at T3, T4 or T5 during the I/O cycle so the IOGENFF is always reset at the end of T5. Should the MP accessory detect an illegal I/O instruction about to be executed, it will immediately lower the MPV- line resetting the IOGENFF and disabling the execution of the I/O instruction. MPV- is also a qualifier for IOG to eliminate the race in resetting the IOGENFF.

In summary, it is a microinstruction that contains IOG in the SP field that signals the I/O control logic to the fact that an I/O instruction is to be executed. Then the I/O hardware simply looks at IR bits 6, 7, 8, 9, and 11 to determine what actual I/O instruction is to be executed during the three actual T-periods when the IOGNEFF is set.

Two special signals decoded on sheet 8 are IAK [45A] and HLTIO- [45D]. The generation of IAK is initiated by a microinstruction that has an IAK in the SP field. Recall that the control processor freezes until T6 or al long as a DCPC cycle is pending. When T6 decodes and a DCPC cycle is not forthcoming, the control processor unfreezes and during the last half of the microcycle IAK is sent up the backplane.

HLTIO- is a special decode of the HLT instruction which resets the RUNFF. Recall that a HLT instruction is classified as an I/O group instruction.

There are two special initializing control signals on sheet 9. Power-on or Preset I/O {POPIO} [54B] is generated every T5 when the RUNFF is low, during power-on, or when the PRESET button on the operator panel is pressed. Most I/O interfaces set their respective flags at this time.

Control Reset {CRS} [54C] is generated under the same conditions as POPIO in addition to the execution of a CLC0 instruction. Most I/O interfaces clear their respective control flip-flops at this time.

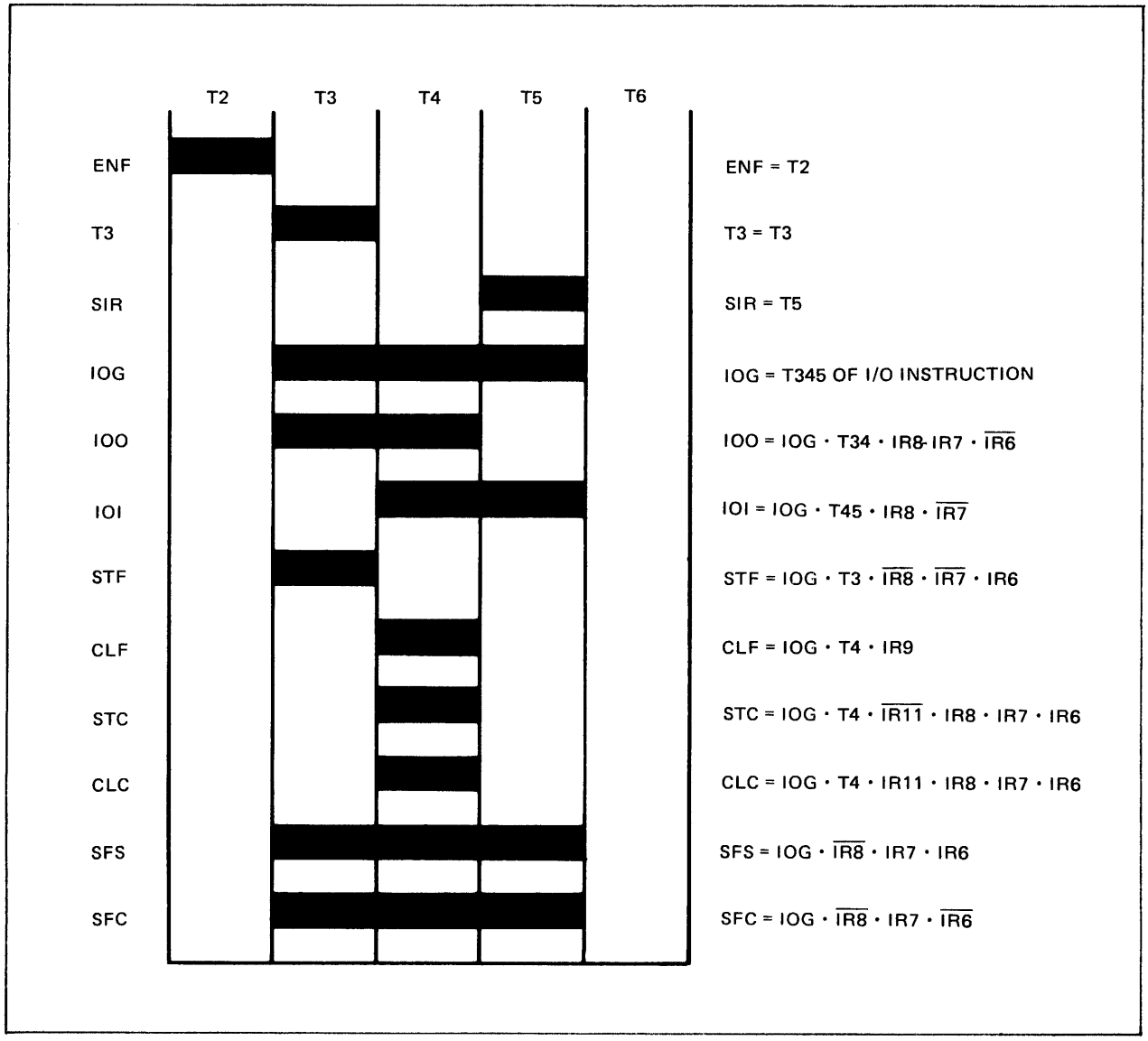


Figure 4-6. I/O Control Signals

4-29. I/O Select Code Logic

The I/O select code logic on the left side of sheet 9 is always enabled whether or not an I/O instruction is being executed. The generation of the select code logic is only important during either the execution of an I/O instruction or DCPC data transfer. The lower six bits of the IR are buffered by a tri-state buffer (U281) onto the select code bus (SCB). Recall that the lower six bits of an I/O instruction are the select code of the I/O device. The tri-state bus is received by the DCPC, MP and I/O extender accessories. DCPC may also disable the CPU from driving the SCB by DMACYC and enable the select code of the target I/O device with which it is transferring. To insure I/O system compatibility the binary SCB must be translated to an octal select code recognizable by U301 (74LS138) and buffered by U321 (74S240) to become the eight select code least (SCL) signals to the backplane. I/O interfaces expect the SCL lines to be CTL-compatible current sources, but since each SCL line must drive at most two I/O interfaces in the mainframe, a high current driver, such as a 8T13/75121, is not required. The 74S240 sources 15mA and is a good substitute. Certain select codes used internally by the CPU are decoded by U322.

The upper three bits of the SCB are decoded by U282 and U302 to produce the select code most (SCM) signals. Only two SCM lines need to be decoded since at most only select codes 10 through 25 reside in the computer mainframe. Full current sourcing buffers are provided since more than minimum loading is expected.

4-30. POWER FAIL/AUTO RESTART

The Power Fail/Auto Restart Logic (PF/ARS) is located on sheet 8. It is essentially a four-state machine defined by the two upper state flip-flops contained in the 74LS175 (U362) [46D]. Feedback through the 74LS153 multiplexer (U342) provides next state information. There are two different power supplies used in the E-Series computers. Since both supplies react in a somewhat different manner, both will be described here. The "B" model power supply is used on all F-Series computers.

a. "A" Model Power Supply Section IX. The power supply sends three control signals that contain information that permits the CPU to determine the state of the ac and dc voltages in the computer. The three signals are called PON (Power On), PWU (Power Up), and MLOST- (Memory Lost). When ac line (mains) power is first applied to the power supply, both PON and PWU are low until all dc voltages are stabilized within their tolerance specifications. At this point PON and PWU go high (usually PWU follows PON by approximately 50-100ns). The MLOST- signal should be high, unless memory was lost in a power down sequence. Then MLOST- will be low and go high with PON and PWU. When the power supply detects an ac line (mains) failure (ac invalid), PWU goes low after a delay of from 10ms to one or two seconds depending on the load on the power supply. This generates a power fail interrupt. DC power is valid for at least 500us after which PON goes low ending the CPU processing activites. The signals are diagrammed in figure 4-7.

b. "B" Model Power Supply Section IX. The power supply sends three control signals that contain information that permits the CPU to determine the state of the ac and dc voltages in the computer. The three signals are called PON (Power On), PWU (Power Up), and MLOST- (Memory Lost). When ac line (mains) power is first applied to the computer both PON and PWU are low. When the ac line (mains) power reaches a valid level PWU goes high, but PON remains low until all dc voltages are stabilized within their tolerance specifications. At this point PON goes high, the CPU checks the status of the MLOST- signal. If MLOST- is high, this means that the memory was not lost during a power down sequence. Therefore, the CPU will not perform a clear memory routine. If MLOST- is low, the CPU will perform a clear memory routine. MLOST- must remain low for at least 50us after PON goes high on a power-up sequence so that the CPU has enough time to check the status of the MLOST- signal. When the power supply detects an ac line (mains) failure (ac invalid), it causes the PWU signal to go low after approximately an 8 ms delay. With PWU low and PON high, this means that the ac line (mains) input is invalid, but the dc voltages are still valid. After approximately 500us PON goes low, thus allowing enough time for the CPU to perform a power fail routine. The power fail routine is halted when PON goes low, because approximately 50us later, the dc voltages are considered invalid. The MLOST- signal is a "don't care" condition in a power-down sequence as shown in figure 4-7.

4-31. Power-Up

After the DC voltages are valid in the CPU, but while Pon is low, the CPU is held in a partially suspended state. Logic in the timing section is active, but no real operations are carried out. A low PONB (buffered PON) forces a perpetual LCMAR signal (see sheet 5) in the control processor, and clears all power fail state machine flip-flops (U362 on sheet 8). PONFF- is low and PONFF is high which insures that all nine multiplexers on sheet 5 are disabled. Since the CMAR is always being parallel loaded, the microinstruction at CM location 0 is the only microcycle executed. The control processor is essentially suspended. (Some references refer to PONFF- as PF1FF).

PONB is also responsible for several other initialization operations. PONB resets the RUNFF (sheet 1) and generates a PRST condition which initializes various internal flip-flops. Most importantly, though, PRST generates both a POPIO and CRS signals to all I/O interfaces thus disabling all I/O activity.

Before PONB goes high and before PONFF and PWUFF are high (the next T6), the control processor repeatedly executes microinstructions 0 and 1. Since the RUNFF is reset when the T6 clock arrives, the control processor executes microinstructions at CM locations 0, 1, 6, 7, and then those of the operator panel micro-routines. The hardwired power fail state machine concurrently scans several inputs to determine if the power is coming up or down, and if it should initiate an auto restart procedure.

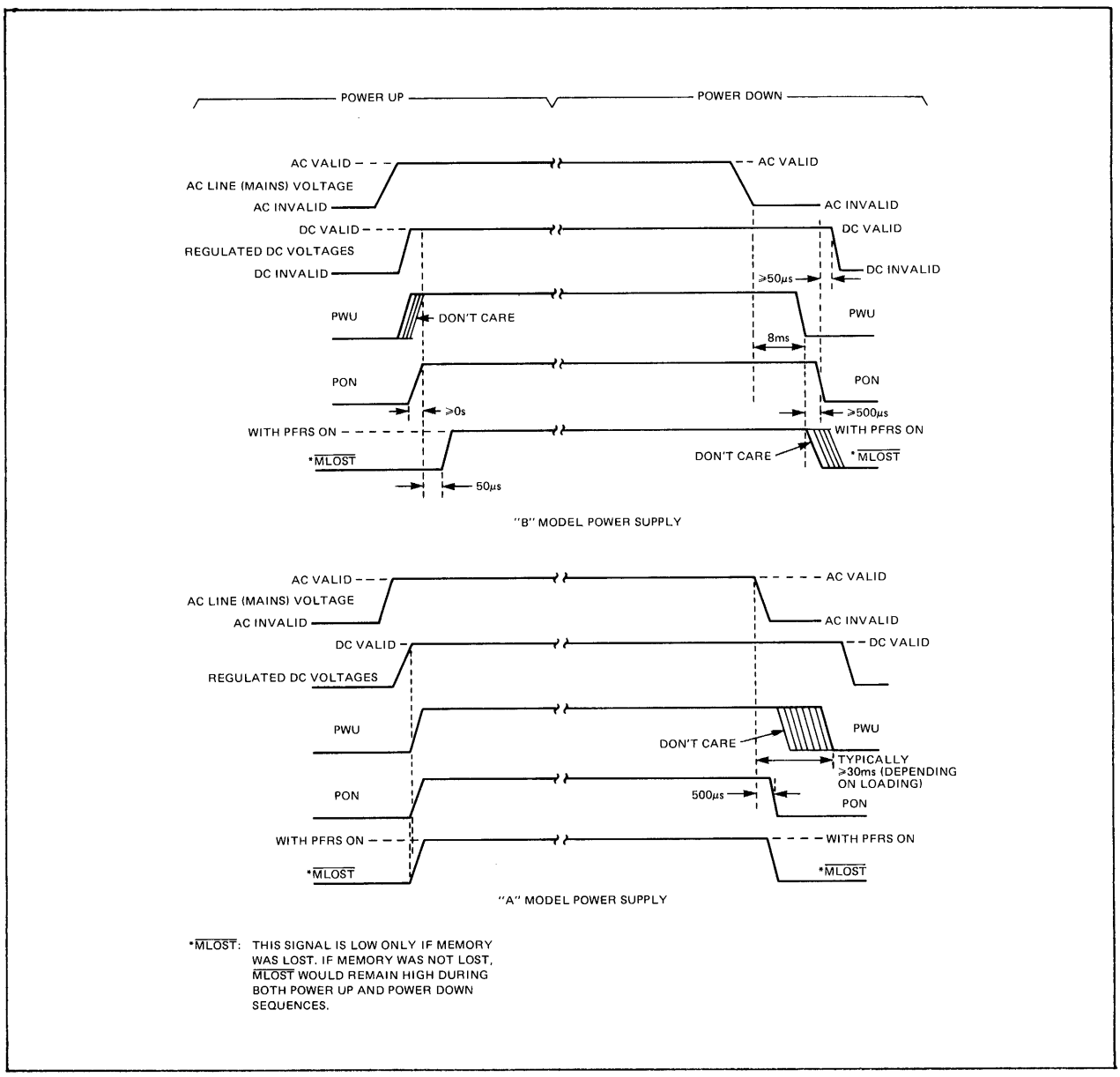


Figure 4-7. Power Fail Logic Signals

There are really only two outputs (and their complements) from the power fail logic: U362-2 [46D] is the first state flip-flop and is the interrupt request flip-flop. It generates an interrupt request and presents its select code via U303-13 [48C] and U327-10 [46C], clears the PRH5 flip-flop to disable all I/O interrupts and sets the operator panel POWER FAIL LED indicator. U362-7 is the second state flip-flop and the direction flip-flop. It may be sensed by a SFC4 instruction to determine if the interrupt is associated with power-up or power-down conditions. The operation of the power fail state machine is diagrammed in figure 4-8.

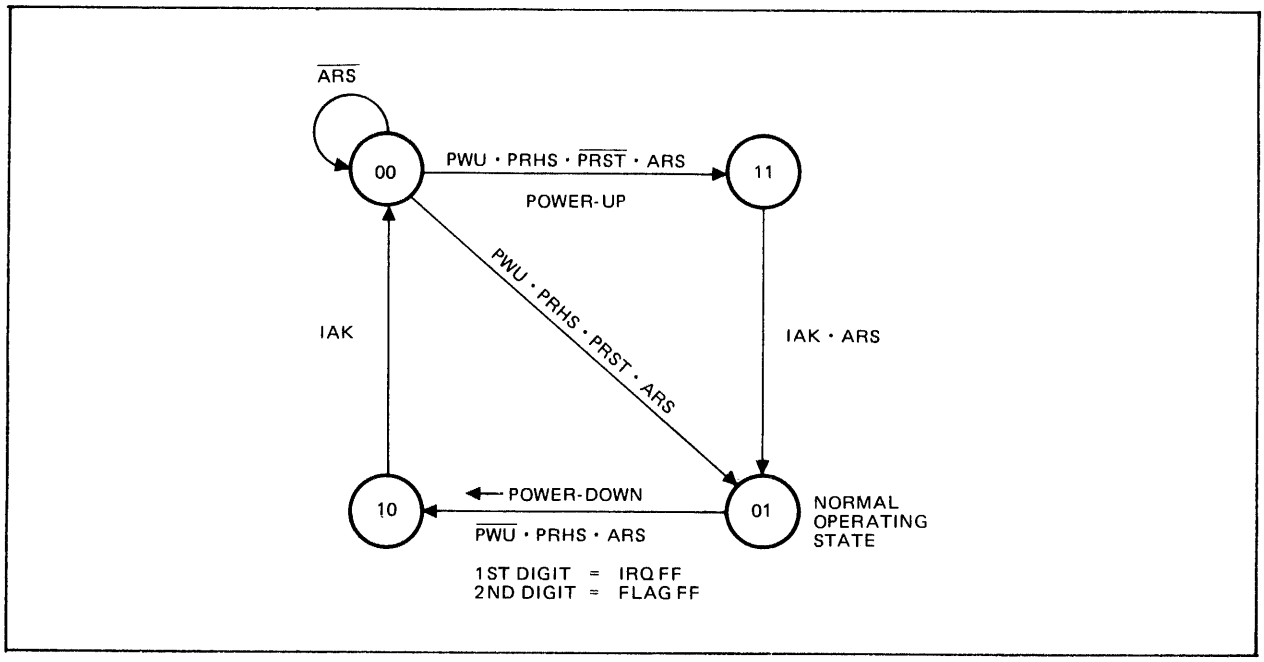


Figure 4-8. Power Fail State Machine Operation

Consider the auto rstart (ARS) position of switch A1S2 and assume that the operator panel PRESET button is not pressed. After PON goes high, PWUFF PRH5 (U341-11) [46D] is high as is PRST (U384-15) [47B], so state 11 is entered. This simultaneously generates an interrupt request, sets the RUNFF {PWUST-}, and waits for an IAK from U361-6 [45D]. A SFC4 instruction will not skip. By the time of the execution of the microinstruction at line 272 the test for the run state is guaranteed. In RUN, the routine prepares the processor to execute the first instruction and branches to CM location 1. The JTAB takes the control processor to location 6 because an interrupt is pending and subsequently an IAK is generated taking the state machine to the normal operations state 01. The auto restart procedure is now complete in that the computer is running and handling an interrupt from main memory location 4.

If the PRESET button on the operator panel is pressed as power is restored to the CPU, the PRST- signal will be low after both PON and PWU go high. From the 00 state the state machine enters normal operations state 01 since U341-8 [46D] is inhibited. By circumventing the 11 state, the PWUST signal and the interrupt request are avoided.

Once in the normal operating 01 state, the machine waits for a power failure indicated by a low PWU. With PRH5 high, U341-3 [45D] goes high placing the machine in the 10 state creating an interrupt request. When the computer recognizes the interrupt request with an IAK, the state machine comes to rest back at the 00 state. Note that the direction flip-flop is now low, so an executed SFC4 test will be met.

Consider the not auto restart (ARS-) position of switch A1S2 [45C]. The definition of this position is that the computer will halt on a power-up and

no interrupts will occur on either power-up or down. The ARS- position disables the next state multiplexer U342 [46D] and keeps the power fail machine in the 00 state (see figure 4-8). No PWUST or interrupt will occur.

Moving the A1S2 switch while the computer power is applied may cause unexpected power fail interrupts to occur.

The interrupt systm {INTSYS} flip-flop [47C] is a software controlled flip-flop that is set by a STF 0 and cleared by a CLF 0 instruction. PRST- is also reset condition to insure that the flip-flop will be cleared on power-up or by pressing the PRESET button on the operator panel. For the INTSYSFF to be recognized, it must be set for any interrupt except power fail or parity error.

The not Priority High 5 {PRH5} flip-flop (U324) [47B] is set when a power fail interrupt is in progress and is used to mask all other I/o interrupts. Executing a STC 4 or CLC 4 instruction or pressing the operator panel PRESET button restores interrupt continuity.

The not Interrupt Enable flip-flop {INTENFF-} (U304) [47A] is a control processor interrupt masking flip-flop. It is not normally used by the user microprogrammer, but is critical to the proper operation of several base set instructions. The flip-flop may temporarily mask all interupts except parity and memory protect. The mask is enabled if the microorder IOFF is executed in the SP field. It is disabled if either ION or JTAB in the SP field is executed. JMP,I, JSB,I, and I/O control instructions must hold off all interrupts (except parity and memory protect) after the execution of the instruction itself. The IOFF associated with each instruction masks any interrupt that could be recognized by the JTAB of the instruction following the masking instruction. JTAB then resets the INTENFF- to allow interrupts to be recognized following the next instruction.

The not Normal Interrupt flip-flop {NRMINTFF-} (U304) [47B] IS cleared when an interrupt is detected. An I/O device interface requests an interrupt and reports its select code by driving its octally coded IRQ and FLG lines. All available IRQ and FLG lines in the computer mainframe are received by an octal-to-binary encoder made from a series of open-collector inverters (U327, 328, and 329) [49B,C]. The outputs of the inverters represent a ground-true binary equivalent of the select code, or interrupt address (IA) of the interrupting device. Interrupts from interfaces located in I/O extenders appear directly via J2 on the IA lines. The IA lines are passed through the U270 inverters [17,18D] on sheet 3 into the inputs of the CIR (U170) [17,18D].

Any pending interrupt is detected by U325-6 [46C]. INTX is low if any extender has an interrupt. A minimum number of lines have to be tested since no interrupt can occur with a select code less than four.

The interrupt request is qualified with TC and DMALO- before it clears the NRMINTFF-. Recall that both the IRQ and FLG lines are T56 in duration on most I/O interfaces so the masking with TC (T56) is redundant. This had to be added when the HP 91200 video interface was tested to keep its IRQ and FLG

lines high for a complete I/O cycle beginning and ending at T5. Typically IAK would reset the NRMINTFF- at T6 clearing out the interrupt request, but the video card's IRQ and FLG lines would reset the flip-flop again causing unmentionable problems. The masking with DMALO- is to prevent requesting an interrupt when an I/O device under DCPC transfer has set its flag to request a cycle and consequently set its IRQ and FLG lines. In this case it does not wish to generate an interrupt, but only desires a DCPC transfer. This is the same precaution noted earlier in the freeze logic on sheet 1.

The NRMINTFF- is set by IAK or IOFF. The use of IAK is conceptually obvious since this means that the interrupt has been acknowledged by the control processor and the CIR contains the select code of the interrupting device. IOFF used to be a required set of the NRMINTFF- in a certain set of infrequent circumstances, but now is no longer needed, however, it does no harm.

The Interrupt Flag {INTFLG-} [48C] indicates to the control processor that an interrupt is pending. There are four combinations of interrupts that may lower INTFLG-. The NRMINTFF- is already explicitly qualified with the INTSYSTFF and needs only to be qualified with the INTENFF- mask. IRQ4 is the power fail interrupt and does not require the interrupt system to be set, but is masked by the INTENFF-. FLG5- is for the parity error interrupt and is qualified by PRH5- on the MP accessory. It does not require that the interrupt system have power applied. MPV is the MP fence, I/O, and DMS violation. It must be qualified by both PRH5- and INTSYSFF-. This special logic permits interrupt recognition immediately following the instruction that caused the violation. Note that the MP accessory would have eventually raised its IRQ5 line and interrupted via the NRMINTFF-, but this could be up to two instructions beyond the actual violating instruction.

The Halt or Interrupt {HOI} (U344) [48C] signal is the logical OR of an interrupt pending and a low RUNFF.

The interrupt priority logic is located on the lower right corner of sheet 8. The concept and use of PRH, PRL, and IEN in the general case for I/O interfaces must be thoroughly understood prior to the following brief discussion.

The first unit that may interrupt after power fail is the MP accessory. IEN5 and PRH5 go to the accessory. The IEN5 signal also goes to the operator panel {INTL} to indicate status of the interrupt system. The MP accessory qualifies any parity error interrupt (via FLG5-) with PRH5 to insure that no power fail interrupt is in progress. MP interrupts via MPV, FLG5-, and IRQ5 if a fence, I/O, or DMS violation occurs only if PRH5 and IEN5if a fence, I/O, are both high. In either MP interrupt case FLG5- is low. The DCPC accessory qualifies its interupt with PRH6. PRH6 is generated from the INTSYSFF (the interrupt system must be on), PRH5 (no power fail can be in progress) and FLG5- (no MP interrupt must be pending). An interrupt by DCPC is signaled via IRQ6 or IRQ7, and PRL7 is low.

I/O device interfaces between select codes 10 and 17 are able to interrupt only if the composite IEN10 and PRH10 signals [47,48A] are high. This is high

if INTSYSFF, PRH5, FLG5-, and PRL7 are all high. If any interrupt between select codes 10 and 17 is pending, PRL17 is low.

Likewise, when IEN20 is high only I/O device interfaces with select codes of at least 20 may interrupt. IEN20 (IEN10) is qualified by PRL17. The additional qualifiers FLG5- and INTSYSFF are redundant, but are included because they may save delay in disabling the interrupt system chain should either condition go low.

The I/O flag skip logic is a straightforward detection of all flag test instructions. External accessories and I/O interfaces with select codes five or greater are tested via the CTL-compatible SKF line. Internal CPU tests are specifically decoded. They include select code 0 (the INTSYSFF), select code 1 (the OVERFF0), and select code 4 (direction of the power failure). Only SFC 4 is decoded since SFS 4 is defined similar to a NOP instruction. The SKF signal is passed to the control processor branch logic on sheet 6 and may be tested by SKPF in the COND field.

4-32. Dual-Channel Port Controller (DCPC)

The reader is certainly aware from the many references thus far in this Theory that there is considerable interaction between the DCPC and the CPU. The fundamentals of the DCPC deserve an overview in this document, but the HP 12897B Theory of Operation in Section III should be consulted at this time.

This is only a brief description of selected DCPC signals initiated by the DCPC and received by the CPU. All signals become true at P4NF of the specified period, and then become false at P4NF of the succeeding period. See Figure 4-9 for DCPC Timing.

DMALO- DMA lock-out precedes the actual data transfer to insure that no interfering processor operations will be in progress when the transfer begins. It is used primarily to freeze or pause the control processor if a memory operation, I/O instruction, or interrupt acknowledge is pending. It is enabled by the leading edge of T4P4 and disabled by the next leading edge of T4P4.

DMACYC- DMA cycle parallels the actual data transfer and primarily freezes the control processor during T2, 3,4,5 insuring no interfering activities. It also inhibits the setting of the IOGENFF and disables the CPU from driving the SCB in duration. It is enabled by the leading edge of T5P4 and disabled by the next leading edge of T5P4.

DMAEN- DMA Enable prevents the CPU from driving the M-bus. DCPC gates its target memory address onto the M-bus at this time. It is enabled by the leading edge of T6P4 and disabled by the following T4P4.

DMAFRZ- DMA Freeze prevents the CPU from driving the S-bus. DCPC needs to use the S-bus at T3, but T2 is included to insure that the S-bus is completely disabled at the beginning of T3.

DCPC output transfer begins with a read from memory. The resultant data in the memory T-register is enabled onto the S-bus, the S-bus is enabled onto the I/O-bus, and the I/O interface buffer is loaded from the I/O-bus. Figure 4-9 diagrams an output operation.

READ- Memory begins a read operation on the falling edge of this signal.

TEN- The T-Register enable gates the read memory data onto the S-bus.

DMAIOO- DMA I/O output gates the S-bus onto the I/O bus.

DMALCH- DMA latch latches the transfer data around the 8T13 I/O-bus driver on the CPU. This provides a valid I/O-bus through the cycle even after the S-bus is changed.

IOO The I/O output signal is sent to the device interface to load the data on the I/O-bus into the output buffer. The store operation is to occur on the trailing edge of IOO.

A DCPC input transfer is begun with the I/O device interface enabling its input buffer onto the I/O-bus. The I/O-bus is gated onto the S-bus, the memory data register (T) is latched from the S-bus, and finally a memory write cycle is begun. Figure 4-9 diagrams an input operation.

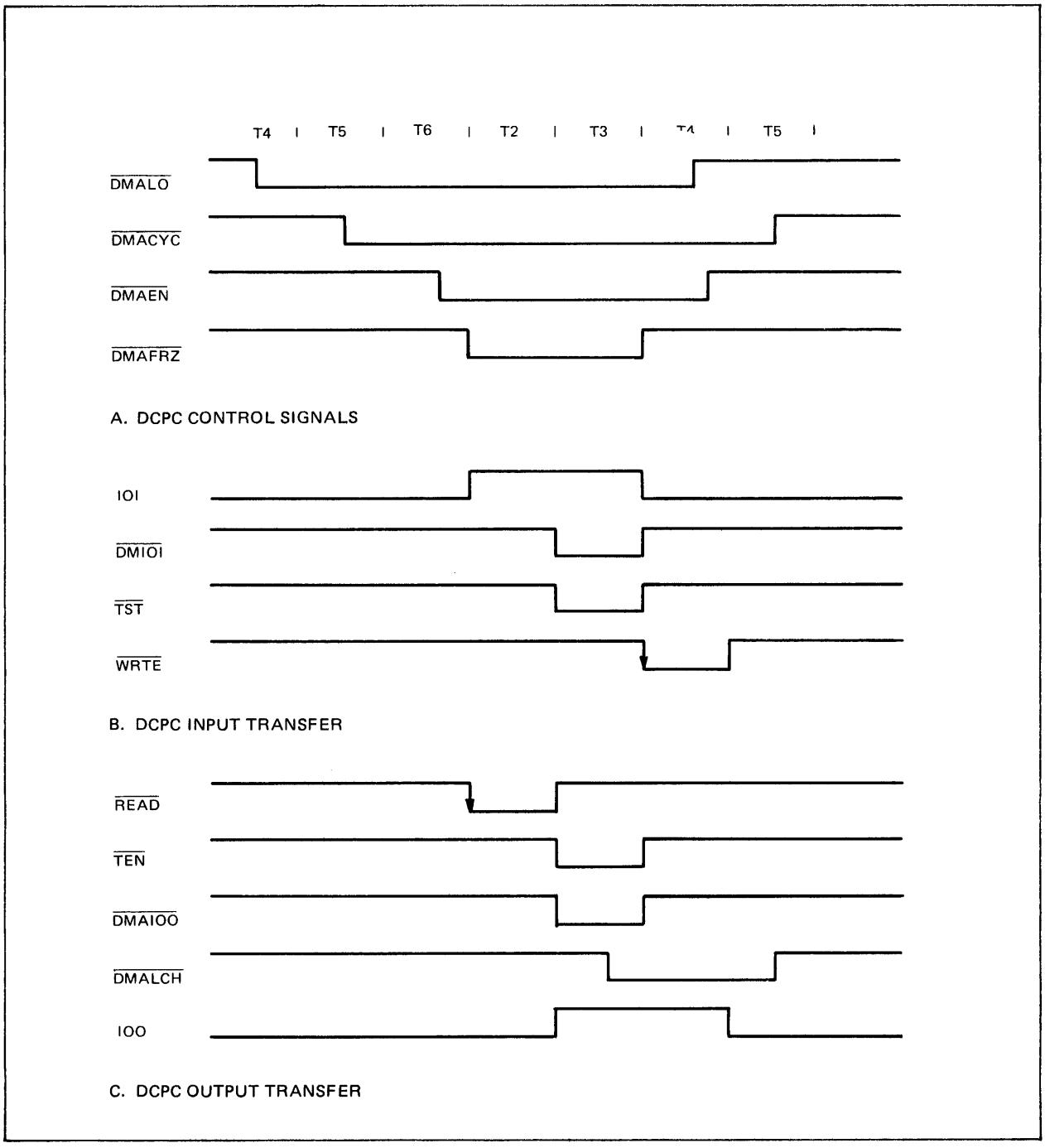


Figure 4-9. DCPC Timing

IOI — The I/O input signal requests the I/O interface to enable its buffered input data onto the I/O-bus.

DMAIOI- — The DMA I/O input signal gates the I/O-bus onto the S-bus.

TST- The memory controller loads its T-Register from the S-bus at P4.

WRTE- The memory begins a write cycle on the falling edge of this signal.

4-33. Operator Panel

The E and F-Series use the same operator panel (5061-1343). The only difference is in the utilization of the Microprogrammable Processor Port (MPP). As shown on the system block diagram, located in the front of this Section, the MPP in the E-Series is available for use with user custom designed hardware; the F-Series uses the MPP to interconnect the 12740A Hardware Floating Point Processor with the CPU through the operator panel. The operator panel schematics are numbered 5061-1343-51 and -52 and are to be found in Section VIII, Panel Assemblies.

Refer to the schematic 5061-1343-51 at this time. The operator panel has two registers called the Display Register (U18-21) and the Display Indicator Register (U25). The 16-bit Display Register may be enabled or stored by the control processor via the S-bus or modified by the operator with the panel buttons. The register is implemented by sixteen cross-coupled NAND gate latches (U18-21) because they are easily toggled by the selected switch buttons. The octal drivers that gate the S-bus into the register are capable of forcing only logic ones into the register, but are not cable of clearing the latches. Therefore, the control processor issues a clear display (DSPCL-) command one P-period before the latches are actually stored (DSPST-) with the desired data (see CPU schematic sheet 4 [24B]).

The Display Indicator Register (U25), is eight bits wide and modifiable by only the control processor. Only the least significant six bits of the register are displayed (CR23-28), although the two hidden bits are passed across the S-bus. The Display Indicator is a ground-true register and a low level causes the appropriate LED to illuminate. Also, bit 0 of the S-bus corresponds to the A-Register LED and bit 5 corresponds to the S-Register LED.

Six state indicators (CR1,2, and 19-22) in the lower left corner of the operator panel schematic monitor specific CPU flip-flops.

All other control buttons (except HALT) are targeted for the conditional jump logic on CPU schematic sheet 6, but not before they generate and are qualified with a STROBE signal. The STROBE signal serves several purposes. First, it is a convenient OR for all buttons which permits the control processor to easily test for a pressed button.

Second, it triggers an active debounce circuit to mask the button signal until all noise associated with the switch has settled. When no button is pressed the transistor Q1 is saturated permitting little charge storage on the load capacitor C7. When a button is pressed, Q1 turns off allowing the capacitor to slowly charge. After about 150 ms the schmitt-trigger buffer (U23F) switches to set the strobe flip-flop (U22), thus enabling appropriate button signal to the CPU. The active debouncer timing is illustrated in figure 4-10.

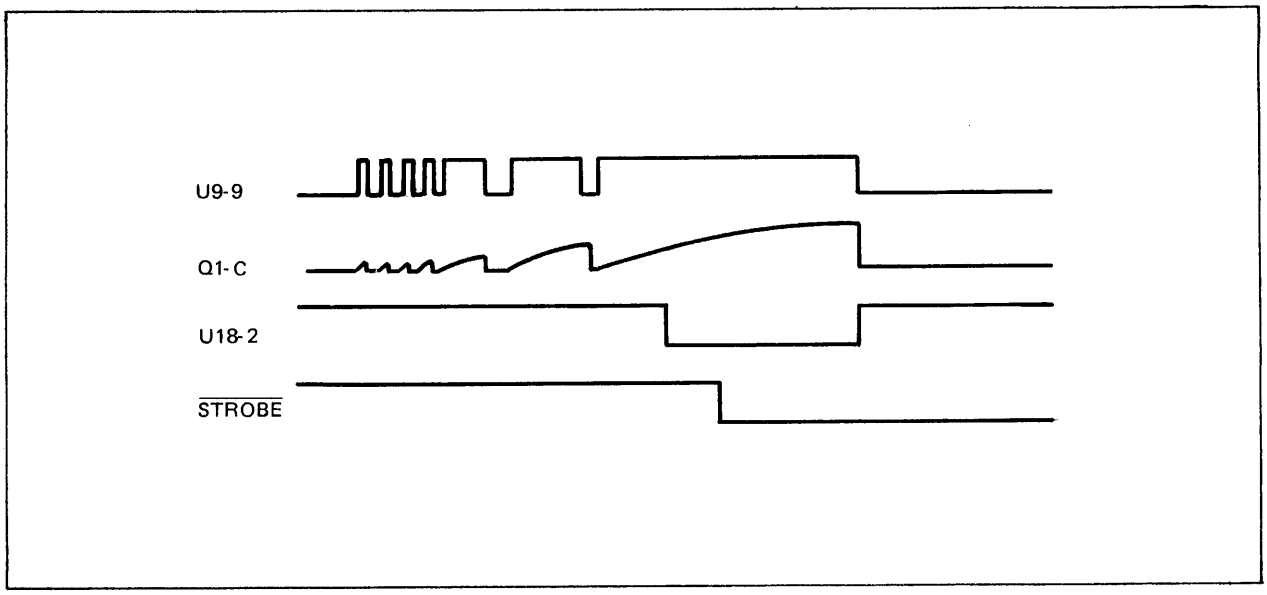


Figure 4-10. Switch Debounce Timing

Third, it synchronizes the STROBE signal to P5 by the flip-flop (U22). This synchronizes all the other button signals which satisfy the interface requirements for the jump conditional logic on CPU schematic sheet 6. The multiplexing tree depends upon all conditions being stable early in the microcycle.

4-34. Firmware Accessory Board (FAB)

The FAB resides beneath the main CPU and attaches via four standoffs which also deliver power to the board. It provides space for optional and user written firmware. Configuration and installation information may be found in the "M/E/F-Series Firmware Installation and Reference Manual", the schematic and assembly diagrams are located in Section VI of this document. Note that in F-Series Computera a customer may elect to have a 12791A FEM board in lieu of the FAB.

The control store priority scheme is shown in Figure 4-11.

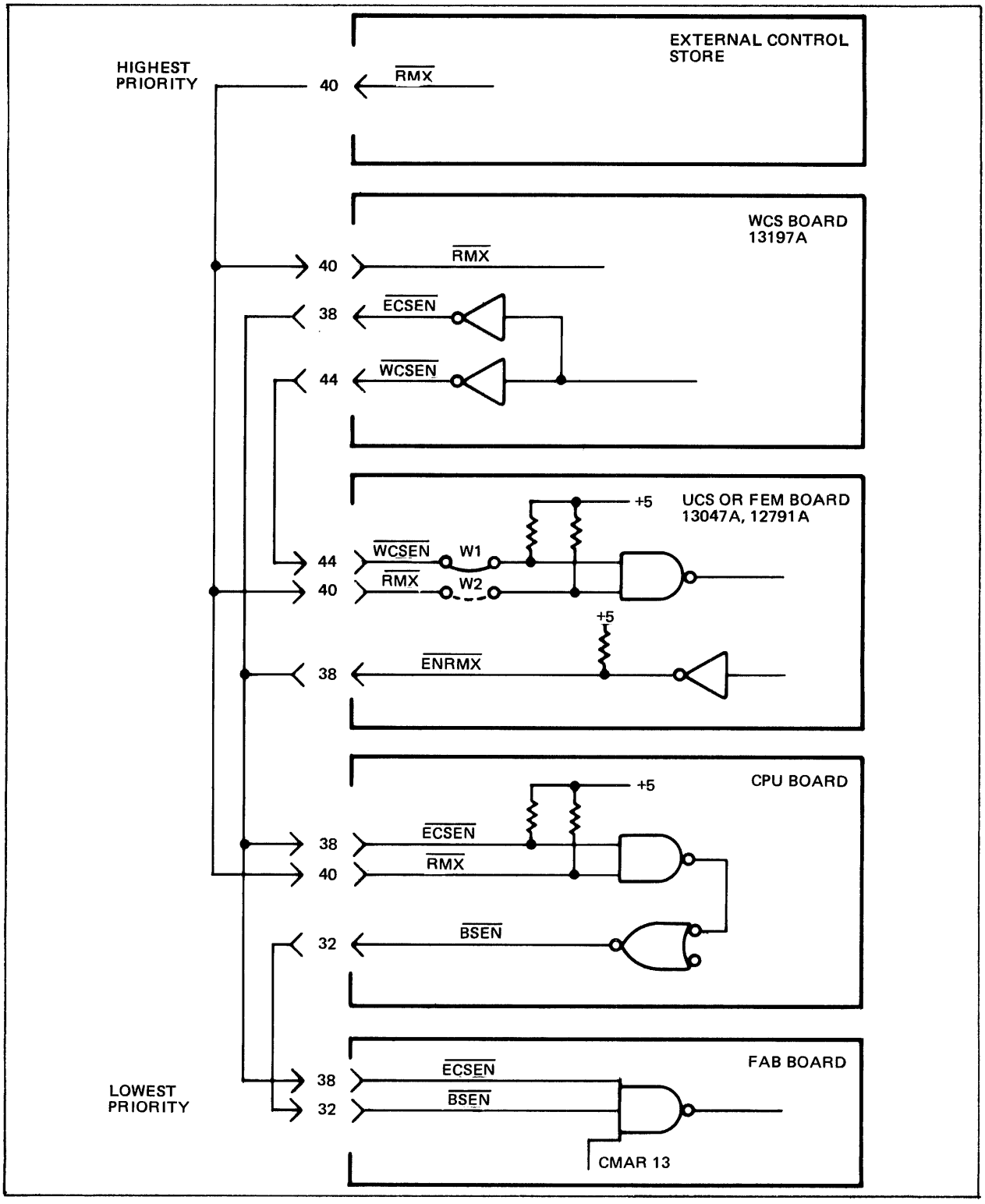


Figure 4-11. Control Store Priority

The FAB will be disabled for all higher priority and enabled sources of control store. This is implemented by U510 (FAB) utilizing the signals BSEN- and ECSEN-. CMAR13 is selectable to enable the FAB in the upper or lower 8K of control store address space. The data out of the FAB (ROM0-to ROM2?) is buffered to provide higher drive current and a cleaner interface to the ROM bus.

Address decoding is a straightforward implementation of the configuration table in the reference manual. When a set of PROMs or ROMs are addressed, power is applied to the Vcc pin of the ROMs via the circuit shown in figure 4-12.

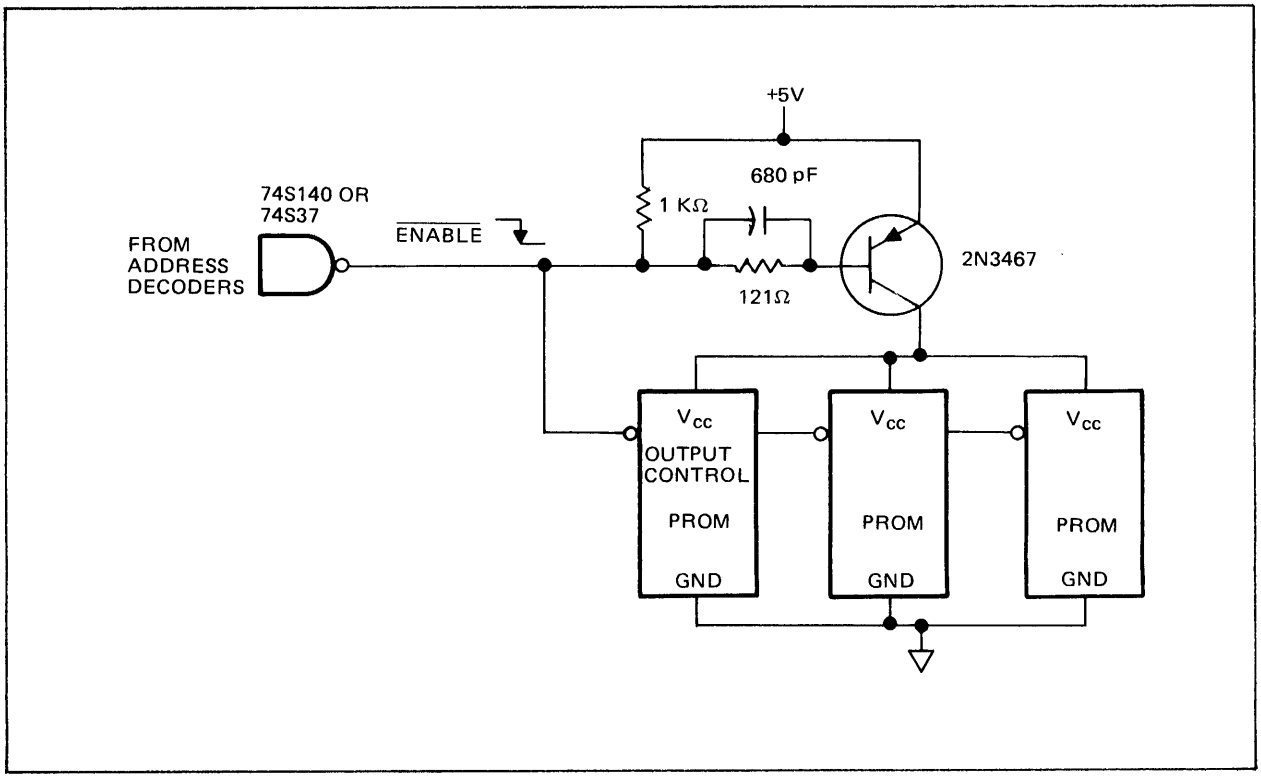


Figure 4-12. Address Decoding

The sink current of the 74S140 and 74S37 (60mA) is sufficient to saturate the PNP transistor, which has a Vce (sat) of approximately 0.2V.

The base capacitor is added to speed up the switching transitions (instantaneous base current is much higher than the steady state DC). The 1k pullup resistor aids in turning-off the transistor.

Power switching is necessary due to the limited cooling and airflow available at the bottom corner of the CPU. Certain 1k PROMs, however, fail to power switch in this configuration and are not recommended for use (though they work functionally). The failure to switch is attributed to a parasitic transistor in the output control of the PROM (pin 13). At de-select time, it is capable

of sinking over lmA, and causes the switching transistor to operate in the linear region.

BASE SET MICROPROGRAM DISCUSSION	SECTION V

The purpose of this section is to review the less obvious operational features of the base set microprograms. The discussion will focus on the more intricate microprogramming subleties of the first two modules. The Extended Instruction Group (EIG) and Floating Point (FP) microprograms will not be considered, but are relatively straight forward. The EIG is easy to analyze, but the FP is more difficult due to the complexity of the algorithms and boundary conditions.

5-1. Fetch and Indirect

The FETCH microroutine has been explained in the microprogramming manual so by this time it is not likely that the reader has had any difficulty in understanding it. This microprogram discussion, however, will begin here because it is such an important routine.

When the FETCH micro-routine is entered, the M-Register contains the address of the instruction, the P-Register points to the location just beyond the instruction, and a read of memory is in progress. The result of the read is, of course, the instruction itself so the purpose of CM 00 is to complete the fetch of the instruction.

Understandably, TAB is in the SB field and IRCM is in the ST field. The IR receives the instruction and the M-Register is conditionally modified depending upon SB10. Assume that the instruction is an MRG instruction. Bit 10 is the operand page bit and the lower ten bits form the displacement. The lower ten bits are unconditionally stored into the M-Register. SB10 clears the upper five bits of M if it is low (zero page), but if it is high (current page) they are not modified. A read is begun concurrently with the creation of the target operand address.

The M-Register would have been modified by the same procedures as above even if an MRG instruction had not been in the IR. In this case the contents of the M-Register would be useless and the subsequent read would be discarded. Since MRG instructions are typically 60% to 70% of all instruction distributions in application programs this operand anticipation technique is statistically a powerful performance aid.

Assuming neither a halt nor interrupt condition is pending, the JTAB at CM 1 forces a branch through the PROM jump table. Usually the M-Register is updated to point to the next instruction (assuming this is a single word instruction) and P is advanced two passed the current instruction. Instructions that disable the ST field operation are those instructions that require preservation of the operand prepared in M. These are limited to ISZ,

JMP, JSB, STA, and STB. All of these use the target address in M for modification or as a jump target. These instructions deliberately update both P and M in their execution routines.

All indirect MRG indirections also disable the modification of P and M. The routine MRGIND is the jump target for all indirect MRG instructions except JMP, I, and JSB, I. Exit from MRGIND is to the actual MRG execution routine and is accomplished after all indirect levels have been resolved.

The microinstruction at CM 2 takes the address read from memory and passes it into the M-Register and another read is begun. INCI serves no real purpose beyond priming the following JTAB to be interpreted as a special conditional branch.

The JTAB branch will occur only if the AL15 flag is not set (i.e., all indirect levels have been resolved). The P and M store activity will not occur unless the branch occurs to (see U290-8 [21B] on sheet 4). As long as indirection is not complete, the microinstruction sequence 2,3, and 4 is repeatedly executed without P or M being updated. When the indirection has finally been resolved the branch through the jump tables occurs and the update of P and M is permitted (except if one of the five MRG instructions is in the IR). This provides the same interface to the execution routines as if MRGIND has not been entered.

The primary uncertainty to the reader is probably how the control processor detects the low AL15 condition. For the jump to occur LCMAR ([26B] sheet 5) must be high, but since INDFF (U26-7) [25B] is high then the MET- signal must be low (U26-6) [25C]. Note that the bit assignment for the ALU field micro-order "INC" (01111) is the same as the COND field micro-order "AL15." These two fields share the same MIR bits. Also, note that MIR14 is high because the code for P in the SB field (11110) includes bit 14 high. Therefore, the multiplexing jump logic [33B] on sheet 6 has been set up for an "AL15 RJS" test. This is the only type of microinstruction in the base set that is micro-assembled and executed as a word type 1 microinstruction, but is partially processed as a word type 3 microinstruction.

A second subtle situation characterized by the special JTAB concerns the jump target itself. Recall that most indirect MRG instructions map through the jump table to CM location 2 (the MRGIND micro-routine). The JTAB at CM 3 would likely carry the control processor back to CM location 2 since the IR has not been modified. In fact, when INCI precedes the JTAB, the INDFF forces the IR15 address bit low to the jump table PROMs (U166-11) [25A] (on sheet 5) as if a direct MRG instruction is actually now residing in the IR. The branch then successfully carries control to the proper MRG execution routine.

5-2. Executing MRG Instructions

The execution of the MRG instructions is performed mainly between CM locations 15 and 52. All such micro-routines are straightforward except perhaps JMP,I and JSB,I. Both of these instructions share the property that no interrupts (except MP or parity) will be acknowledged immediately following the

instruction. Both of these instructions use IOFF to set the INTENFF mask. If more than one indirect level is detected, the universal indirect operand micro-subroutine INDIRECT is called.

The INDIRECT subroutine is used by many standard base set instructions and HP supported firmware accessories. It may also be easily used by user-generated firmware. It is always called with a read of an operand address which may or may not have its indirect bit set. The routine eventually returns when indirection is resolved with a read in progress. The purpose of INCI is to increment the indirect level counter on the MP accessory (if installed). After three levels of indirection, MP will lower the MPINTON line (U304-1) [42B] (see sheet 8) thus disabling the INTENFF mask. This INCI feature is only for those instructions that use the INTENFF as a mask to hold off interrupts (i.e., JMP, I and JSB, I).

5-3. Executing ASG Instructions

The ASG instructions are carried out in a straightforward manner unless the powerful ASG microinstruction is not understood. The first microinstruction of the execution routines performs the primary accumulator operation, sets up the bit skip test, and begins a read of the next instruction. The next microinstruction essentially performs the rest of the instruction. ASG operates on the extend register (CCE, CLE, and CME), forces the ALU to a PASS if IR2 is low (disabling the increment if neither INA nor INB is desired), and simultaneously looks at all possible skip conditions and requests a pop of the save stack (return to CM 0) if the ASG skip conditions are not met. This would complete the instruction, but if the skip conditions are met the return will not occur and an increment of P and M must happen for the read of the next instruction to begin. The ENVE enables setting of the overflow and extend registers if the increment of the accumulators is allowed to proceed. Because the L-Register is cleared by FTCH the overflow checks may proceed properly. Recall from the definition of the ASG instructions that there is a specific priority to the combining of many functions in one instruction (see Table 3-4 of the Operating and Reference Manual). It may not be immediately obvious that all functions are handled in their appropriate order. It is apparent that the primary accumulator operations occur first (CC*, CL* and CM*) followed by the skip tests at the end of P3 of the ASG microcycle, and then the extend register operations (clocked at P5) and accumulator increment operations. The relative sequencing is followed except in the case of the hardware INA/INB operation (P5) which follows the skip test (P3). To accomodate this case and bring all skip tests together, the SZA/SZB test actually looks at IR2 (the increment bit) to anticipate if the increment is enabled and appropriately check the result from the output of the ALU in the previous microinstruction. If the increment is not to occur, zeros are checked. If the increment is to occur, all ones are checked at U234-6 [33C] (sheet 6).

As noted above, a return is called for if the skip conditions are not met. SPRTN forces a load of the CMAR, but there must be some guarantee that the multiplexers are selected by the lower three bits of the SP field so ASG has been assigned the same bit pattern of RTN in the SP field (000).

5-4. Executing SRG Instructions

The SRG instructions are executed in a similar manner to the ASG instructions. SRG1 activates the first shift and may clear the E-Register if CLE is coded. SRG2 activates the second shift and may also force a return to FETCH if the one SRG skip test (SLA, SLB) is not met. Like ASG in the SP field, SRG2 has ben assigned a SP bit pattern with the least significant bits all zero.

5-5. Executing IOG Instructions

Control processor synchronization is required to execute IOG instructions. Microinstruction 77 is the first to be executed following the JTAB. This freezes the CPU until T2 when the IOGENFF is set. During the next three microcycles all I/O activity must be completed. A word type 4 with IOG as the jump modifier takes the control processor to locations 00, 104, 110, or 114 depending on the instruction in the IR.

The first two routines perform input operations and depend upon the I/O system standard that input data is guaranteed to be valid on the S-bus at T5. The output transfer routine insures that the I/O bus data is valid during both T4 and T5. Every I/O interface latches its data at the end of T4 so it is wise to bracket the latching operation to eliminate potential skewing problems between control processor and I/O system timing.

All I/O control instructions (CLC, STC, CLF, STF, SFC, SFS, and HLT) are executed via the CONTROL routines. The IOFF sets the interrupt mask (INTENFF) to prevent interrupts from being ackowledged following the I/O instruction just as JMP,I and JSB,I do. A skip test is made to detect if any skip conditions exists to appropriately increment the P-Register.

The purpose of CM locations 116 and 117 is discussed in paragraph 5-9.

5-6. Executing EAU Instructions

All EAU shifts are reached from JTAB via either JTBL1000 or JTBL1010 and EM1000 or EM1010. Recall that the lower eight bits of the IR is a counter and that the lower four bits contain the number of shifts to be executed. The repeat flip-flop is set and the counter counts down as each 32-bit shift is executed.

DLD, DST, DIV, and MPY are microprogrammed directly from their definitions. MPY is programmed very tightly to augment its speed. The only obscure point here is the storage of the multiplier's opposite sign bit into the overflow register (micro-instruction 142). The L-Register was cleared by FTCH so if the sign bit of A was zero, bit 15 from the ALU would be one and the overflow would set. If the sign bit of A was one, and an overflow can not occur. Recall that the lower four bits of a DIV or MPY instruction are all zero, so the appropriate DIV or MPY step is repeated sixteen times.

5-7. Special Instructions

There are three special instructions that are new in the HP 1000 E and F-Series computer. HP makes no guarantee as to the operation and offers no support for the TIMER or EXECUTE Instructions, but does support the DIAGNOSTIC (Self-Test) Instruction.

The TIMER instruction (octal 100060) is a four microinstruction routine beginning at CM location 230. TIMER increments the B-Register until B is either zero or a halt or an interrupt occurs. The purpose of this instruction is to generate a specific time interval. In the past, software timing routines were created that would execute in a precise unit of time for every computer. In the E and F-Series, however, additional timing uncertainties such as different speed memories and DMS enabled or not greatly effect computer performance. The TIMER instruction increments B once every 630ns independent of any other factors except of course for DCPC activity. Fetch overhead may effect the time interval for small numbers of B increments, but for large numbers the overhead is not appreciable.

The EXECUTE instruction is called by an octal 100120 followed by an address DEF of a second instruction which is to be "executed". Control is passed back to the instruction following the DEF unless the instruction "executed" itself was a branch (for example a JSB or JMP) or if the instruction "executed" is a multiple word instruction or skips. In the latter case, the instruction will not work correctly, since the DEFs for the multiword instruction are fetched from memory after the execute DEF instead of after the target instruction opcode. The EXECUTE instruction should be useful in tracing simple instructions.

The DIAG instruction (self-test) begins at location 234 and allows a user to call the firmware diagnostic by an octal 100000. These diagnostics are destructive to memory so they may not be executed while the computer is running. They may only be run manually by the INSTR STEP button on the operator panel. The instructions loop if the RUNE line is low (LOCK position of the LOCK/OPERATE switch on the F and E-Series "B" Models and LOCK position of the key operated switch on the "A" Models).

It should be noted that Self-Test 1 tests the CPU. A failure is accompanied by all the display register lights, all the display lights, and the overflow light being illuminated. If a failure occurs changing the CPU should solve the problem. Self-Test 2 is a Semiconductor Memory Test that checks up to 32k words of memory and is performed only when the IBL button is pressed. A failure is usually accompanied by the parity light, all the display register lights, and all the display indicator lights will be illuminated, but the overflow light will be extinguished. The A-Register will contain the valid data and the B-Register will contain the invalid data. The M-Register contains the failing memory address.

Self-Test 3 tests all the memory that is installed in the computer, if dynamic mapping (DMS) is installed. If DMS is not installed only 32k words (or less

depending on the amount of memory in the computer) of memory are tested. If no DMS is installed in the computer the same method of error reporting is used in Self-Test 3 that is used in Self-Test 2. If DMS is installed and more than 32k words of memory are in the computer, error reporting is the same as in Self-Test 2, except for the failing memory address. Since the memory addresses can go up to 1M (a 20-bit address), the M-Register cannot display the actual address because it can only display a maximum of 15 bits. With the aid of figure 5-1 we will now show how the actual 20-bit address can be obtained. As shown in figure 5-1, M-Register bits 0-9 are the final register bits 0-9. To botain the final address bits 10-19 the M-Register bits 10-14 must be loaded into the "m" register bits 0-4. Then the "t" register bits 0-9 are read as the final address bits 10-19. Once the final 20-bit address is obtained bits 15-19 will tell in what 32k word space the failure occurred and bits 0-14 will tell the address within that 32k word space. An example using figure 5-2 will help in understanding this method of finding the failing memory board. If the first 32k word space had a failure, bits 15-19 would all be low, therefore, the failure is between MODs 0 through 7. Bits 12, 13, and 14 tell which of the eight MODs failed. Bits 0 through 11 tell the address within the individual MOD. If the third 32k word space has a failure, bits 15-19 would equal 00010. This means that the failure occurred somewhere between MODs 20 through 27. Bits 12, 13 and 14 tell which MOD, and bits 0 through 11 tell the address within the MOD. Self-Tests 1 and 2 are executed when the front panel IBL/TEST switch is pressed with the computer halted. On a cold power-up Self-Tests 1 and 3 are executed once. By setting bit 15 in the "A" register or by setting the "A" register to 100000, clearing the "P" register, setting the "S" register to desired memory data pattern, pressing PRESET, then INSTR the STEP will execute Self-Tests 1 and 3 once, if the front panel key operated switch is in the OPERATE position on the "A" Models or the LOCK/OPERATE switch is in the OPERATE position on the "B" Models. If the switch is in the LOCK position, the tests will loop until the switch is set to the OPERATE position or until a failure occurs.

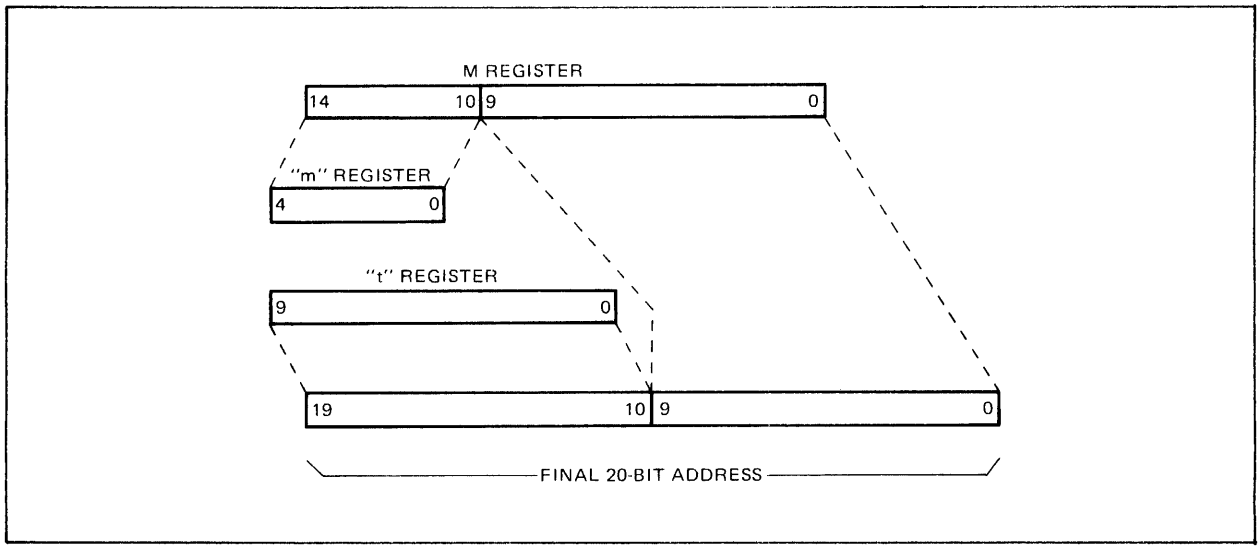


Figure 5-1. Final 20-Bit Memory Address

32k WORD SPACE NO. (IN OCTAL)	BITS 15-19	MEMORY MODS
	19 15	
5th	00100	40,41,42,43 44,45,46,47
4th	00011	30,31,32,33 34,35,36,37
3rd	00010	20,21,22,23 24,25,26,27
2nd	00001	10,11,12,13 14,15,16,17
1st	00000	0,1,2,3 4,5,6,7

Figure 5-2. Memory Failure Locations

5-8. Selected Operator Panel Routines

SCAN: SCAN is usually entered when an operator panel button is activated and calls the appropriate subroutine which performs the function. CM locations 325 through 374 execute the more straightforward functions.

DSPICODE: This subroutine (CM locations 475 to 512) stores a unique number into the lower four bits of the IR which corresponds to the pointer in the Display Indicator register. As examples, code 0000 is returned if the S-pointer is up, 0001 is returned if P is up, and 1011 is returned if X is up.

STORE: This performs the STORE button function by first determining what target register is to be loaded by calling the DSPICODE subroutine. A JSB RJ30 branch to a table beginning at STORES (CM375) begins to execute the appropriate function. All routines between CM400 and 436 are fairly straightforward except perhaps STCPUS (Store CPU Status) (CM 420) which must scan across the display register data to set up the extend, overflow, and interrupt system flip-flops.

WAIT: This is the button test and wait loop for the operator panel microprogram. The display register is first updated to reflect the state of the computer as specified by the display indicator. The indicator is coded by the DSPICODE subroutine (CM location 476) and then a branch to UPDATES (CM location 440) actually updates the display register number to the STORE subroutine. UPDCPUS (Update CPU Status) (CM 456) requires building up the

status from various registers in the CPU. CM locations 306 and 307 initialize the save stack. The JTAB is a conditional JTAB (because INCI precedes it), but the condition tested (ALO) is never true because all zeros were passed through the ALU in the previous microinstruction. See paragraph 4-11 Save Stack for further information.

LOADER: The initial binary loader routine (CM516) is segmented into functional routines. MEMSIZE determines the memory size from the top down, SELCODE checks the desired select code, LOOP builds the 16-bit instruction from the loader PROMs and configures appropriate I/O instructions, and STWORD writes the instruction into main memory. The IAK in CM 540 synchronizes the control processor to T6. This guarantees that at least the LDR in the S-bus field at CM 551 occurs at T5 on the first word, which presets the CPU I/O. This is essential for RPL.

TEST32K: (Self-Test 2) This is a non-destructive test of the enabled memory space (up to 32k words). Each memory word is read complemented, written back, read again, complemented, verified, and rewritten. This routine starts a CM 604.

CPTEST: (Self-Test 1) This tests the ALU, RS, scratch registers, flag, IMM logic, and ALU tests with an alternating ones pattern and a call to REGTEST. This routine starts at CM 617.

RIPP1MW: (Self-Test 3) This routine (CM 661) performs a destructive bit-ripple test in all physical memory installed in the computer. A subroutine RIPP32K is repeatedly called to test each 32k word space. Tests are included for "REAL" memory locations 0 and 1 for each 32k.

5-9. Run/Halt Microprogram Linkage

The linkage between the run microprograms and the halt microprograms include some subtle timing and logic situations.

Consider first the transition from the halt to the run mode. The control processor executes microinstructions in the idle loop at CM 310 until a button is pressed. In either case the SCAN routine and the RUN routine ar executed. RUN places the S-pointer into the display indicator, begins the fetch of the instruction at location P, and jumps from CM location 333 to 1. The computer is now running and all registers are properly oriented. The purpose of the instruction pass through S1 at CM locations 331 and 332 will become apparent when the INSTR STEP function is discussed later.

The transistion from run to halt may take one of two forms. Any halt condition (except a HLT instruction) or interrupt condition is detected at CM location 1 when JTAB branches to location 6. Since the P-Register is advanced one location beyond the present instruction, it must be decremented. Then the

decision as to whether a halt or interrupt condition has really been detected is made at CM location 7.

The halt microprogram is entered at CM 265 if the RUNFF is low. The M-Register on the E-Series does not necessarily match the expected value of earlier HP minicomputers when a computer halt or INSTR STEP button is pressed. As a result of microinstruction 0, M has no resemblance to the address of the last instruction executed. To satisfy most users of the E-Series computer most of the time, M (except for one situation) always follows the P-Register by one upon entry to the halt microprogram. Since FTCH at location 0 cleared the CPU flag, microinstructions 266 and 267 load M. Microinstructions 270 through 276 implement the control functions diagrammed in figure 2-4 of the Operating and Reference manual.

If the RUNFF was high, the control processor synchronizes to T6 to acknowledge the interrupt, load the CIR, and read the interrupt location memory address (trap cell). The instruction fetch is completed with the jump to microinstruction 1. The M-Register is saved in S1 and the CPU flag is set. The importance of this is discussed next.

Consider now the case of a HLT instruction not executed in a trap cell location. A halt instruction resets the RUNFF at CM location 114 via HLTIO (sheets 1 and 8). A simple RTN microinstruction could have been coded in location 116 and the subsequent branch back to fetch would have eventually been routed to location 6 and the halt would have been detected as above. The purpose of microinstructions 116 and 117 is to circumvent microinstructions 0 and 1 if a HLT instruction is executed. The FTCH (which clears the CPU flag) must be avoided. This is to take care of the case when an interrupt has been detected as in the previous paragraph, but a HLT instruction is executed via microinstruction 77, 114, 115, 116 and 117. Now CM 6, 7, and 265 are executed. Since the CPU flag was set by the interrupt routine and no FTCH has been executed to clear the flag, microinstruction 266 is not executed and M, instead of being loaded with the P-Register minus one, is loaded with S1. Recall S1 was loaded with the interrupt address at CM location 12. Therefore, in the unique case of a HLT instruction in the interrupt trap cell, the P-Register points to the point of program suspension before the interrupt and M points to the interrupt address itself.

The INSTR STEP button on the operator panel requests the control processor to execute the next instruction. To do this the run microprogram is entered, but the RUNFF is reset so that control will be returned to the halt microprogram after the instruction is executed. The INSTR STEP button is sensed and a branch to NSTP is made. If the front panel was in the special register display mode (flag is true), the mode is switched at CM 326 back to the standard register display since the flag will be cleared by FTCH. The RUNFF is set at CM 327, but is reset at CM 334. The instruction in S1 is passed through the ALU at CM334 before the jump to microinstruction 1 is executed. Both of these are subtle points crucial to the proper execution of the instruction step function.

If a halt or interupt condition (HOIFF) is detected by JTAB immediately after

the execution of microinstruction 335 then a branch to CM6 occurs followed by the microinstruction at CM7 and then control is returned to the halt microprogram. The instuction will never be executed. By setting the RUNFF at CM327, the HOIFF is not true. by resetting the RUNFF at CM334, the HOIFF becomes true, but it is not actually true instantaneously. The RUNFF and the BRUNFF [32D] (sheet 6) are reset at P5 of microcycle 334, but the HOIFF is not set until the end of the microcycle 1 since the buffereing flip-flops of sheet 6 are not updated during word type 3 or 4 microinstructions.

Therefore, microinstruction 1 will not detect a set HOIFF and a successful branch will be achieved to begin proper execution of the instruction. When microinstruction 1 is executed after the completion of the instruction, HOIFF will be true and control will be relinquished to the halt microprogram.

The JTAB function expects that the instruction has just been passed through the ALU so that it may look at the AL15FF [32D] (sheet 6) to accurately enable or disable the ST field function (see U290-8 [21B] on sheet 4). S1 contains the instruction to be executed and sets up the flags at microinstruction 334.

EXAMPLES OF HARDWARE/FIRMWARE INTERACTION — SECTION VI

This section includes four instruction examples that are analyzed one microinstruction at a time. Each example includes an accompanying figure that diagrams selected hardware signals on the CPU. The diagrams use idealized wave forms found only in engineering texts without resemblance to the real world. Decoding, set-up times, propagation delay, reflection, and other properties must be anticipated when viewing these signals with an oscilloscope. These examples are designed only to give a flavor to the hardware/firmware interaction of the CPU.

6-1. ADA Instruction

An ADA 200 instruction is located in address 1000. The contents of A is 300 and location 200 is 50. The hardware is shown in Figure 6-1.

Microinstruction 0 : Loads the instruction into the IR and forms the page zero operand in M. Begins a read of the operand at the end of the microcycle.

Microinstruction 1 : Branch through the PROM jump table. Increment the P and M registers. The I/O bus must be used as a data latch.

Microinstruction 17: Loads the L-Register with the operand and begins a read of the next instruction.

Microinstruction 20: Sums the A register with the latch register and stores the result in the A register. Enables the overflow and extend register logic and returns to the fetch routine.

6-2. STB,I Instruction

A STB, I 100, C instruction is located in address 2000. The contents of B is 177777 and location 2100 is 3300. The hardware is shown in Figure 6-2.

Microinstruction 0 : Loads the instruction into the IR and forms the current page operand as M. Begins a read of the operand at the end of the microcycle.

Microinstruction 1 : Branch through the PROM jump table, but disables the ST field operation because the instruction is both an MRG indirect and a STB.

Microinstruction 2 : Load M with the target operand and begin a read.

Microinstruction 3 : Branch through the PROM jump table again (because AL15FF is low), but IR bit 15 is masked low to the PROMs. The ST field is disabled again.

Microinstruction 135: Load the memory T register with B, request an address check by the MP accessory, and begin a write cycle.

Microinstruction 136: Update P and M and begin the fetch of the next instruction.

Microinstruction 137: Return to the fetch routine.

6-3. ASG Instruction

A CMA, INA, SZA, instruction is located in address 500. The contents of A is 70. The instruction is shown in Figure 6-3.

Microinstruction 0 : Loads the instruction into the IR and forms an operand address in M. Because this is an ASG instruction, the contents of M are useless.

Microinstruction 1 : Branch through the PROM jump table. Update the P and M registers to the next instruction.

Microinstruction 67: The A register is complemented and a read of the next instruction is begun.

Microinstruction 70: The skip logic tests for a skip condition. U274-4, 5, 6 ([34C] sheet 6) will all be high forcing SPRTN low indicating no instruction skip will occur. A return to the fetch routine is executed.

6-4. SRG Instruction

A ELB, SLB instruction is located in address 500. The B-Register contains 177777 and the extend register is clear. The hardware is show on figure 6-4. Microinstructions 0 and 1 operate as in the ASG example.

Microinstruction 73: The B accumulator and the extend register are rotated through the RS logic. The TBO flag ([36A] sheet 6) is set.

Microinstruction 74: No shift activity occurs, but U295-3 is low ([34C] sheet 6) so SPRTN stays high.

Microinstruction 75: The P and M registers are updated to indicate that an instruction skip is to take place. A read of the next instruction is begun.

Microinstruction 76: Return to the fetch routine.

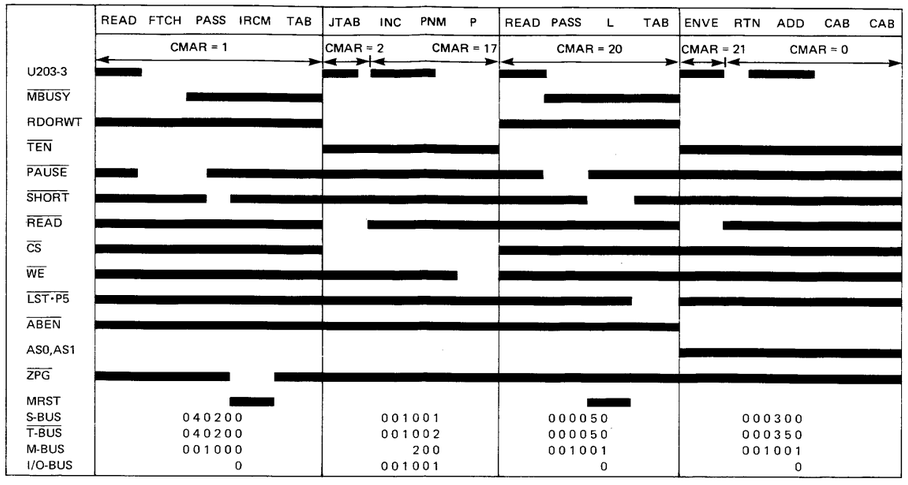


Figure 6-1 ADA Instruction Implementation

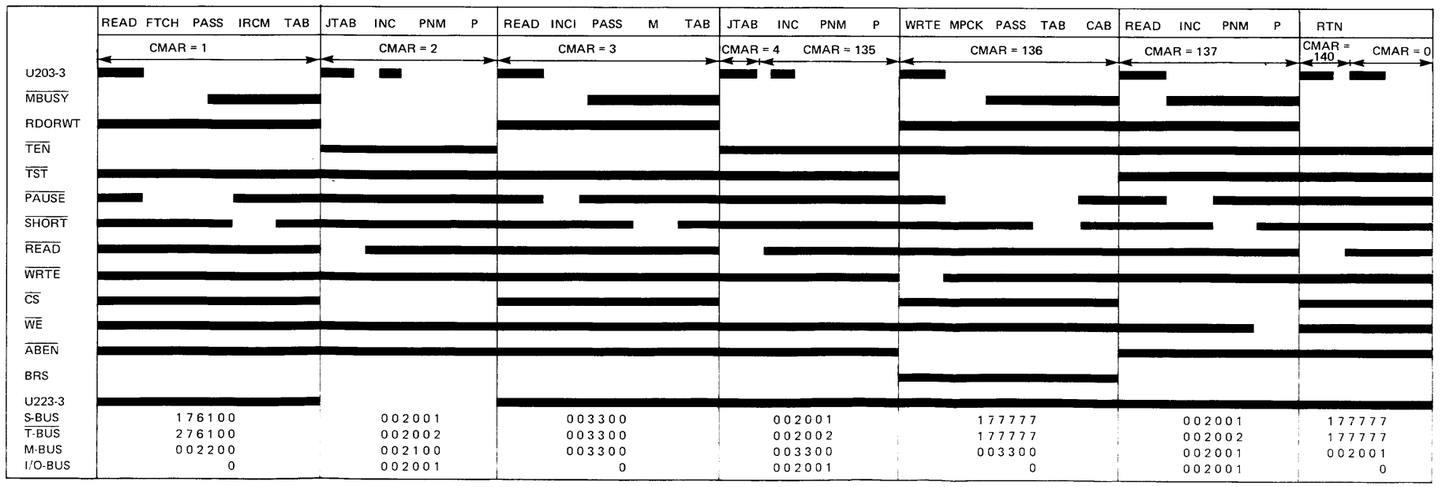


Figure 6-2 STB,I Instruction Implementation

NOTES

1. The wire jumper (W2) installed between XA5-40 and +5V is intended to be installed on Rev C and newer E-Series CPU's when the redesigned memory backplane is installed (which forces the memory controller into one of the bottom 2 slots). The desired effect is to provide continuity between the +5V planes on the CPU and backplane and thus reduce a source of noise and signal degradation encountered in the past. If a Rev C CPU is used with the older HP 1000 backplanes, this jumper should be removed.

2. The plug-in jumper above connector J1 is provided to allow the signal NSFP to be set to "NON-standard" regardless of whether or not a standard front panel is installed. This allows the execution of user written front panel microcode (at CM 5000) to be entered (see CM 437, 757) without removal of the standard hardware front panel (which may still be needed as a run mode device, e.g., LIA1, OTA1, HALT, etc.). The user program microcode can detect the mode in which it is entered (i.e., at 5000). If the CPU flag is clear, a normal entry from the HALT routine is indicated. If the flag is set, a CPU failure as indicated by the firmware diagnostic is indicated by the firmware diagnostic is indicated, and the A, B, and S registers and overflow bit indicate the failing mode (see reference manual).

3. On the Rev C CPU board, a diode bus is installed to clamp the S-Bus at a specified voltage. This is needed, in conjunction with reworked memory backplanes, to attempt to minimize the voltage swing (and therefore, energy transmission) of the S-Bus. Under certain conditions (typically following a memory refresh freeze) when the memory controller captures the M-Bus for a READ request, a 16-bit "1" to "0" transition on the S-Bus causes the M-Bus to glitch, causing a read of an erroneous location. The effect on a program is clearly dramatic and destructive.

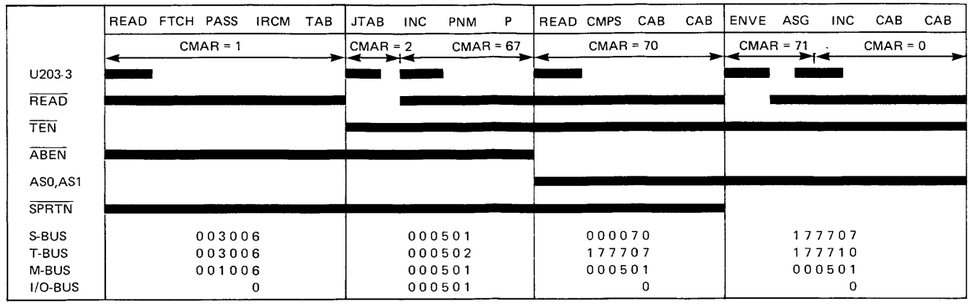


Figure 6-3 ASG Instruction Implementation

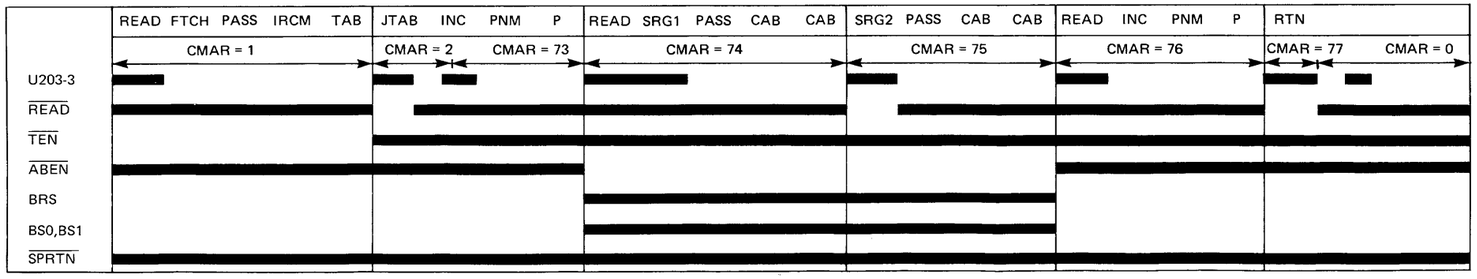


Figure 6-4 SRG Instruction Implementation

SELECTED ABBREVIATIONS

APPENDIX A

A - A Register
ALU - Arithmetic/Logic Unit
ARS - Auto Restart
ASG - Alter-Skip Group

B - B Register

CIR - Central Interrupt Register
CM - Control Memory
CMAR - Control Memory Address Register
CNTR - Counter
CPU - Central Processing Unit
CTL - Complementary Transistor Logic

DCPC - Dual-Channel Port Controller
DMA - Direct Memory Access
DMS - Dynamic Mapping System

EAU - Extended Arithmetic Unit
EIG - Extended Instruction Group

FAB - Firmware Accessory Board
FF - Flip-Flop
FP - Floating Point

IBL - Initial Binary Loader
I/O - Input/Output
IR - Instruction Register

L - L Register
LDR - Loader
LED - Light-Emitting Diode

M - M Register
MB - M-Bus

MEB - Memory Expansion Bus
MIR - Microinstruction Register
MP - Memory Protect
MRG - Memory Reference Group
ms - Milliseconds

ns - Nanoseconds

P - Program Counter
PCA - Printed Circuit Assembly
PE - Parity Error
PF - Power Fail
PROM - Programmable Read-Only Memory

RAM - Random Access Memory
ROM - Read-Only Memory
RPL - Remote Program Load
RS - Rotate/Shift Logic

S - S Register
SB - S-Bus
SP - Stack Pointer
SRG - Shift-Rotate Group

T - Transfer Register
TB - T-Bus

us - Microseconds
UIG - User Instruction Group

X - X Register

Y - Y Register

CPU BOARD	APPENDIX B

Components: Approximately 235 standard TTL integrated circuits and 40 standard passive components are used in the CPU. In addition, custom programmable read-only memory components are used for control memory, jump tables, and loaders. The approximate breakdown of packages used is as follows:

116 - 14 pin integrated circuits
 95 - 16 pin integrated circuits
 18 - 20 pin integrated circuits
  8 - 24 pin integrated circuits

 16 - Resistor array packages
  2 - Capacitor array packages

Plus base set control memory, optional loaders, switches, jumpers and discrete components.

Power: +5 volts: 9.5 A rms
       -2 volts: 450 mA typical

PC Board: The large eight-layer PC board was selected primarily for packaging compatibility between the M/E/F-Series Computers. Layer 1 is the component (top) side. Layers 3 and 6 are the ground and voltage planes, respectively. The voltage layers provide good noise immunity, high distributive capacitance, and low inductance. Component designators, connector pinouts, and other valuable information have been silkscreened onto the top and bottom sides of the board for easy loading, debugging, and identification.

IC CROSS REFERENCE TABLE

APPENDIX C

This appendix is useful for locating actual integrated circuits in the schematics. The component number is in the left-hand column, followed by the commercial part type, the HP internal part number, and the schematic sheet number(s) on which the component is located. Base set control memory and PROM jump tables are merely noted as "PROM" and no part number is given. All HP part numbers are assumed to be 1820- unless otherwise specified.

NUMBER	COMMERCIAL DESCRIPTION	HP PART NUMBER	SCHEMATIC SHEETS
21	PROM		5
23	PROM		5
24	74S374	-1677	5
25	74S32	-1449	5
26	74S08	-1367	5
30	74273	-1461	2
32	74S181	-0999	2
33	74S04	-0683	2
34	74S153	-0998	2
35	74S153	-0998	2
36	74198	-1032	2
41	PROM		5
43	PROM		5
44	74S374	-1677	5
45	74S30	-1323	5
46	74S10	-0685	5
50	74273	-1461	2
53	74S04	-0683	2
54	74S153	-0998	2
55	74S153	-0998	2
64	74S374	-1677	5
65	74S00	-0681	4,5
66	74S260	-1275	5
70	74S373	-1676	3
72	74S181	-0999	2
73	74S04	-0683	2,3
74	74S153	-0998	2
75	74S153	-0998	2
76	74198	-1032	2
81	PROM		5
83	PROM		3

NUMBER	COMMERCIAL DESCRIPTION	HP PART NUMBER	SCHEMATIC SHEETS
84	75LS151	-1217	5
85	74LS151	-1217	5
86	74LS00	-1197	5
89	74S11	-0686	3
90	74S373	-1676	3
92	74S181	-0999	2
93	74S133	-1130	2
94	74S153	-0998	2
95	74S153	-0998	2
96	74198	-1032	2
101	82S42 (OPT.)		5
102	74LS670	-1447	5
103	74S163	-1453	5
104	74LS151	-1217	5
105	74LS151	-1217	5
106	Jump Table		5
109	74S257	-1301	3
110	74S257	-1301	3
113	74S182	-1305	2
114	74S189	1816-0724	2
115	74S189	1816-0724	2
121	Jumpers		5
122	72LS670	-1447	5
123	74S163	-1453	5
124	74LS153	-1244	5
125	74LS153	-1244	5
126	74S00	-0681	5
129	74S257	-1301	3
130	74S257	-1301	2
132	74S181	-0999	2
133	74S260	-1275	2,3
134	74S189	1816-0724	2
135	74S189	1816-0724	2
136	74198	-1032	2
141	74S20 (OPT.)	-0688	4
142	74LS670	-1447	5
143	74S163	-1453	5
144	74LS153	-1244	5
145	74LS153	-1244	5
146	Jump Table		5
150	74S373	-1676	3
152	74S373	-1676	3
153	74S174	-1076	3
154	74S258	-1309	2
155	74S258	-1309	2
156	74S260	-1275	3

NUMBER	COMMERCIAL DESCRIPTION	HP PART NUMBER	SCHEMATIC SHEETS
161	74LS194	-1276	5
162	74LS670	-1447	5
163	74S163	-1453	5
164	74LS153	-1244	5
165	74S04	-0683	1,4
166	74S08	-13671	5
170	74S373	-1676	3
172	74S373	-1676	2
173	SWITCH	3101-1983	2
174	74S258	-1309	2
175	74S258	-1309	2
176	74S20	-0688	3
181	74S04	-0683	4
182	74S10	-0685	4,5
183	74LS04	-1199	4,7
184	74S138	-1240	4
185	9334	-0833	4
186	74S00	-0681	4
190	74S373	-1676	3
192	74S373	-1676	3
193	74S00	-0681	2
194	74S158	-1015	2
195	74LS14	-1416	3
196	LDR (OPT.)		3
201	74S20	-0688	4
202	74S08	-1367	4
203	74S32	-1449	4,5
204	74S138	-1240	4
205	74S138	-1240	4
206	74S20	-0688	4,7
210	8T13	-1080	3
211	8T13	-1080	3
212	8T13	-1080	3
213	74S175	-1191	3
214	74LS86	-1211	6,7
215	74LS191	-1278	3
216	LDR (OPT.)		3
221	74S40	-0690	4
222	74S10	-0685	4,5
223	74S00	-0681	4
224	74S138	-1240	4
225	74S138	-1240	4
226	74S32	-1449	4
231	8T13	-1080	3
232	8T13	-1080	3
233	74S175	-1191	3

NUMBER	COMMERCIAL DESCRIPTION	HP PART NUMBER	SCHEMATIC SHEETS
234	74S51	-1158	6
235	74LS191	-1278	3
241	74S11	-0686	4
242	74S32	-1449	4
243	74S00	-0681	4,8
244	74S138	-1240	4
245	74S138	-1240	4
246	74S20	-0688	1,7
250	8T13	-1080	3
251	8T13	-1080	3
252	8T13	-1080	3
253	74LS157	-1470	6
254	74S151	-1319	6
255	LDR	1816-0420	3
256	LDR	5081-2361	3
261	74S08	-1367	1,4
262	74S00	-0681	1,4
263	74S32	-1449	4,8
264	74S138	-1240	4
265	74S138	-1240	4
266	74LS00	-1197	4
270	74LS14	-1416	3
271	74S04	-0683	2,3,7
272	74LS00	-1197	6
273	74S138	-1240	6
274	74S64	-0691	6
275	8098	-1255	3
276	74S373	-1676	3
281	8095	-1254	9
282	74LS14	-1416	1,9
283	74S00	-0681	1,4
284	74LS04	-1199	4
285	74S32	-1449	4
286	74S74	-0693	4
290	74S00	-0681	2,4,7
291	74S10	-0685	2,6
292	7428	-1184	2
293	74LS10	-1202	4
294	74LS21	-1205	6
295	75S00	-0681	6
296	74273	-1461	6
301	74LS138	-1216	9
302	8T13	-1080	9
303	74LS54	-1285	8
304	74109	-1116	8
305	74S11	-0686	1,8

NUMBER	COMMERCIAL DESCRIPTION	HP PART NUMBER	SCHEMATIC SHEETS
306	74S04	-0683	4,8
310	74LS00	-1197	7
311	74S260	-1275	7
312	11 NSEC DELAY LINE		2
313	74LS20	-1204	4
314	74S51	-1158	2
315	74S08	-1367	2,6,7
316	74S151	-1319	6
321	74S240	-1633	9
322	74LS08	-1201	8
323	74LS54	-1285	8
324	74LS112	-1212	8
325	74LS13	-1415	8
326	74LS112	-1212	8
327	74S05	-0684	4,8
328	74S05	-0684	8
329	74S05	-0684	8
330	74S22	-0689	7
331	74S00	-0681	2,7
332	74S74	-0693	7
333	74S11	-0686	2,7
334	74S04	-0683	6
335	74S151	-1319	6
336	74S151	-1319	6
341	74LS08	-1201	8
342	74LS153	-1244	8
343	74LS54	-1285	8
344	74LS00	-1197	8
345	74LS08	-1201	8
346	8T13	-1080	8
347	8T13	-1080	8
348	8T13	-1080	8
349	8T13	-1080	8
350	74S74	-0693	7
351	74S32	-1449	1,7
352	74LS32	-1208	7
353	74S32	-1449	7
354	74S10	-0685	1,7
355	74S151	-1319	6
356	74S151	-1319	6
361	74LS04	-1199	8,9
362	74LS175	-1195	8
363	74LS132	-1425	8
364	74S64	-0691	1
365	74LS11	-1203	8
366	74LS112	-1212	8

NUMBER	COMMERCIAL DESCRIPTION	HP PART NUMBER	SCHEMATIC SHEETS
367	8T13	-1080	8
368	8T13	-1080	8
369	8T13	-1080	8
370	74S08	-1367	7
371	74LS00	-1197	7
372	74LS11	-1203	7
373	74S04	-0683	7
374	74LS00	-1197	7
375	74LS10	-1202	7
376	74LS13	-1415	1
381	74S260	-1275	1
382	74S20	-0688	1
383	74S175	-2095	1
384	74S51	-1158	1
385	74S132	-1307	1
386	8T13	-1080	8
387	8T13	-1080	9
388	74S00	-0681	1,5
389	74S132	-1307	1
390	74S04	-0683	4,7
391	74S65	-0692	7
392	74S64	-0691	7
393	74LS54	-1285	7
394	74LS109	-1282	7
395	74LS30	-1207	1
396	74LS132	-1425	1
401	74S04	-0683	1
402	74S10	-0685	1
403	74S74	-0693	1
404	74S51	-1158	1
405	74S10	-0685	1
406	74S20	-0688	1
407	SPARE		
408	CUSTOM OSCILLATOR	1813-0081	1
409	74S10	-0685	1
410	74LS51	-1210	7
411	74S65	-0692	7
412	74S64	-0691	7
413	74S54	-1285	7
414	74S64	-0691	7
415	74LS00	-1197	1
416	74109	-1116	1,7
422	SPARE		
428	74S74	-0693	1,2
433	SPARE		

E AND F-SERIES BASE SET LISTING

APPENDIX D

This appendix contains the listing of the E and F-Series computer base set microprogram.

NOTE

The jump table for the F-Series base set is in Appendix E.

```
0001                    MICMXE,L,C,T
0002                    ****************************************************************
0003                    * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1979.  ALL RIGHTS      *
0004                    * RESERVED.  NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,       *
0005                    * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT*
0006                    * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.        *
0007                    ****************************************************************
0008                    *
0009                    *
0010                    $NOLIST
0056                    $LIST
0057                    *
0058                    *
0059                    *         HP 1000 E-SERIES BASE SET MICROCODE
0060                    *         MODULES 0,1
0061                    *         -----------------------------------
0062                    *         1980-11-21-0900
0063                    *
0064                              ORG                 0B
0065                    *
0066 00000  230 000633  FETCH     READ FTCH PASS IRCM TAB       IR := T/A/B; M := OP ADR; READ
0067 00001  007 174701            JTAB INC  PNM  P          JMP THRU LUT--M=P CNDL;P=P+1 CNDL
0068                    *
0069 00002  230 000674  MRGIND    READ INCI PASS M    TAB       M := T/A/B; READ
0070 00003  007 174701            JTAB INC  PNM  P          JMP LUT CNDL--M=P CNDL;P=P+1 CNDL
0071 00004  323 140102            JMP  CNDX HOI  RJS  MRGIND    TEST FOR HALT OR INTERRUPT
0072 00005  336 040102            JMP  CNDX NSNG RJS  MRGIND    TEST FOR INSTRUCTION STEP
0073                    *
0074 00006  000 075707  HORI                  DEC  P    P         P := P-1
0075 00007  323 053402            JMP  CNDX RUN  RJS  HALT      TEST FOR HALT
0076 00010  010 036771            IAK                           LOAD CIR; ACKNOWLEDGE INTERRUPT
0077 00011  230 024677            READ IOFF PASS M    CIR       M := CIR; READ TRAP CELL
0078 00012  010 033017            STFL PASS S1   M          S1 := M
0079 00013  230 000607            READ      PASS IRCM TAB       IR := T/A/B; M := OP ADR; READ
0080 00014  320 000047            JMP                 FETCH+1
```

```
0082                        *
0083                        *       MEMORY REFERENCE GROUP
0084                        *       ----------------------
0085                        *
0086 00015  230 000507  AND       READ       PASS L    TAB       L := T/A/B; READ
0087 00016  372 006147            RTN        AND  A    A         A := A AND T/A/B
0088                        *
0089 00017  230 000507  AD*        READ       PASS L    TAB       L := T/A/B; READ
0090 00020  263 002040            ENVE RTN   ADD  CAB  CAB       A/B := A/B + T/A/B
0091                        *
0092 00021  230 000507  CP*        READ       PASS L    TAB       L := T/A/B; READ
0093 00022  014 102747            XOR        XOR  CAB  CAB       COMPARE
0094 00023  360 000042            RTN  CNDX  ALZ                   TEST IF EQUAL
0095 00024  227 174707            READ       INC  PNM  P         M := P; P := P+1; READ
0096 00025  370 036747            RTN
0097                        *
0098 00026  230 000507  IOR       READ       PASS L    TAB       L := T/A/B; READ
0099 00027  370 106147            RTN        IOR  A    A         A := A IOR T/A/B
0100                        *
0101 00030  007 101007  ISZ                  INC  S1   TAB       S1 := T/A/B + 1
0102 00031  210 040036            WRTE MPCK  PASS TAB  S1        T/A/B := S1; WRITE
0103 00032  320 041602            JMP  CNDX  ALZ  RJS  *+2       TEST IF ZERO
0104 00033  007 175707                       INC  P    P         P := P+1
0105 00034  227 174707            READ       INC  PNM  P         M := P; P := P+1; READ
0106 00035  370 036747            RTN
0107                        *
0108 00036  230 000677  JMP,I     READ IOFF  PASS M    TAB       M := T/A/B; READ
0109 00037  307 112442            JSB  CNDX  AL15       INDIRECT  TEST FOR MORE INDIRECTS
0110 00040  367 133736  JMP       RTN  MPCK  INC  P     M         P := M+1
0111                        *
0112 00041  230 000677  JSB,I     READ IOFF  PASS M     TAB       M := T/A/B; READ
0113 00042  307 112442            JSB  CNDX  AL15       INDIRECT  TEST FOR MORE INDIRECTS
0114 00043  210 074036  JSB       WRTE MPCK  PASS TAB   P         T/A/B := P; WRITE
0115 00044  007 133716                  CLFL  INC  P     M         P := M+1
0116 00045  227 174707            READ       INC  PNM  P         M := P; P := P+1; READ
0117 00046  370 036747            RTN
0118                        *
0119 00047  230 000047  LD*        READ       PASS CAB  TAB       A/B := T/A/B; READ
0120 00050  370 036747            RTN
0121                        *
0122 00051  230 000507  XOR       READ       PASS L    TAB       L := T/A/B; READ
0123 00052  374 106147            RTN        XOR  A    A         A := A XOR T/A/B
```

```
0125                    *
0126                    *         ALTER-SKIP GROUP
0127                    *         ----------------
0128                    *
0129 00053  230 002047  ASGNO*    READ       PASS CAB  CAB        READ
0130 00054  267 102070            ENVE ASG   INC  CAB  CAB        A/B=A/B + 1 CNDL;RTN CNDL;E CNDL
0131 00055  227 174707            READ       INC  PNM  P          M := P; P := P+1; READ
0132 00056  370 036747            RTN
0133                    *
0134 00057  231 136047  ASGCC*    READ       ONE  CAB             A/B := ONES; READ
0135 00060  267 102070            ENVE ASG   INC  CAB  CAB        A/B=A/B + 1 CNDL;RTN CNDL;E CNDL
0136 00061  227 174707            READ       INC  PNM  P          M := P; P := P+1; READ
0137 00062  370 036747            RTN
0138                    *
0139 00063  226 036047  ASGCL*    READ       ZERO CAB             A/B := ZEROS; READ
0140 00064  267 102070            ENVE ASG   INC  CAB  CAB        A/B=A/B + 1 CNDL;RTN CNDL;E CNDL
0141 00065  227 174707            READ       INC  PNM  P          M := P; P := P+1; READ
0142 00066  370 036747            RTN
0143                    *
0144 00067  237 102047  ASGCM*    READ       CMPS CAB  CAB        A/B := CMP A/B
0145 00070  267 102070            ENVE ASG   INC  CAB  CAB        A/B=A/B + 1 CNDL;RTN CNDL;E CNDL
0146 00071  227 174707            READ       INC  PNM  P          M := P; P := P+1; READ
0147 00072  370 036747            RTN
0148                    *
0149                    *         SHIFT-ROTATE GROUP
0150                    *         ------------------
0151                    *
0152 00073  230 002061  SRG       READ SRG1  PASS CAB  CAB        FIRST SHIFT; CLEAR E CNDL; READ
0153 00074  010 002060                 SRG2  PASS CAB  CAB        SECOND SHIFT; RTN CNDL
0154 00075  227 174707  RET       READ       INC  PNM  P          M := P; P := P+1; READ
0155 00076  370 036747            RTN
```

```
0157                    *
0158                    *       INPUT-OUTPUT GROUP
0159                    *       ------------------
0160                    *
0161 00077  320 004006  IOG     JMP  IOG              MI*       T2: SYNCHRONIZE AND JUMP
0162                    *
0163 00100  230 002507  MI*     READ       PASS L     CAB       T3: L := A/B; READ
0164 00101  010 112747                     IOR        IOI       T4:
0165 00102  370 112044          RTN  RJ30  IOR  CAB   IOI       T5: A/B := A/B IOR I/O
0166 00103  320 040005  MAC0    JMP  J74              MACTABL0
0167                    *
0168 00104  230 012747  LI*     READ       PASS       IOI       T3: READ
0169 00105  010 012747                     PASS       IOI       T4:
0170 00106  370 012044          RTN  RJ30  PASS CAB   IOI       T5: A/B := I/O
0171 00107  320 037005  MAC1    JMP  J74              MACTABL1
0172                    *
0173 00110  230 002747  OT*     READ       PASS       CAB       T3: READ
0174 00111  010 002247                     PASS IOO   CAB       T4: I/O := A/B
0175 00112  370 002247          RTN        PASS IOO   CAB       T5: I/O := A/B
0176 00113  320 012005  JTBL1000 JMP J74               EM1000
0177                    *
0178 00114  237 134537  CONTROL  READ IOFF CMPS L     DES       T3: READ, SAVE OLD DES
0179 00115  336 103642          JMP  CNDX  SKPF       RET       T4: TEST FOR SKIP FLAG
0180 00116  363 003642          RTN  CNDX  RUN                  T5: TEST FOR HALT
0181 00117  320 017447          JMP                   HALTIO    T6: GO TO HALT INSTRUCTION
```

```
0183                    *
0184                    *         EXTENDED ARITHMETIC GROUP
0185                    *         -------------------------
0186                    *
0187 00120  230 036747  DLD       READ                                READ
0188 00121  300 012447            JSB                       INDIRECT
0189 00122  007 133007                      INC  S1    M          S1 := M+1
0190 00123  010 000147                      PASS A     TAB        A := T/A/B
0191 00124  230 040647            READ      PASS M     S1         M := S1; READ
0192 00125  010 000207                      PASS B     TAB        B := T/A/B:= P+1
0193 00126  227 174707            READ      INC  PNM   P          M := P/ P := P+1  READ
0194 00127  363 041024  ADDR1S1   RTN  R1    ADD  S1    S1         S1 SET FOR UPDCPUS
0195                    *
0196 00130  230 036747  DST       READ                                READ
0197 00131  300 012447            JSB                       INDIRECT
0198 00132  210 006036            WRTE MPCK PASS TAB   A          T/A/B := A; WRITE
0199 00133  007 133007                      INC  S1    M          S1 := M + 1
0200 00134  010 040647                      PASS M     S1         M := S1
0201 00135  210 002036  ST*       WRTE MPCK PASS TAB   CAB        T/A/B=A/B;WRITE;ENTRY FOR STA,STB
0202 00136  227 174707            READ      INC  PNM   P          M := P; P := P+1; READ
0203 00137  357 173040  CLFS2     IMM  RTN  CMHI S2    375B       S2=CLF0 FOR STCPUS
0204                    *
0205 00140  010 036753  MPY            COV
0206 00141  300 012447            JSB                       INDIRECT
0207 00142  257 107007            ENV       CMPS S1    A          S1=MULTIPLICAND; NSIGN IN OVFL
0208 00143  010 000507                      PASS L     TAB        LOAD L WITH MULTIPLIER
0209 00144  006 036227                 RPT  ZERO B               CLEAR B. INITIATE REPEAT STEP
0210 00145  163 010224            MPY  R1   ADD  B     B          REPEAT MULTIPLY STEP 16 TIMES
0211 00146  315 146502            JSB  CNDX OVFL RJS   *+4        SUBTRACT IF MULTIPLICAND NEGATIVE
0212 00147  227 174713            READ COV  INC  PNM   P          M := P; P := P+1; READ
0213 00150  362 141702            RTN  CNDX L15  RJS              TEST FOR POSITIVE MULTIPLIER
0214 00151  237 140507            READ      CMPS L     S1         PLACE MULTIPLICAND IN L
0215 00152  364 110207            RTN       SUB  B     B          SUBTRACT FOR NEGATIVE MULTIPLIER
```

```
0217 00153  237 110516  DIV     READ CLFL CMPS L    B         L := NDIVIDENDHI; READ
0218 00154  300 012447          JSB               INDIRECT
0219 00155  017 101007                CMPS S1   TAB       S1 := NDIVISOR
0220 00156  014 141047                XOR  S2   S1        EXPECTED QUOTIENT SIGN IN S2
0221 00157  322 107202          JMP  CNDX L15       DIVS    TEST FOR NEGATIVE DIVIDEND
0222 00160  017 110217               STFL CMPS B    B
0223 00161  017 106147                CMPS A    A         MAKE
0224 00162  007 106147                INC  A    A         DIVIDEND
0225 00163  301 010342          JSB  CNDX COUT      RMDR+2  POSITIVE
0226 00164  017 140507  DIVS          CMPS L    S1
0227 00165  327 147342          JMP  CNDX AL15 RJS  *+2     TEST FOR POSITIVE DIVISOR
0228 00166  007 140507                INC  L    S1        L := ABSOLUTE VALUE OF DIVISOR
0229 00167  004 110754               SOV  SUB        B
0230 00170  327 143642          JMP  CNDX AL15 RJS  RET     TEST FOR DIVISOR TOO SMALL
0231 00171  070 010222          LGS  L1   PASS B    B       PRESHIFT THE DIVIDEND
0232 00172  007 174727               RPT  INC  PNM  P       M := P; P := P+1
0233 00173  124 110222          DIV  L1   SUB  B    B       REPEAT DIVIDE STEP 16 TIMES
0234 00174  010 010224               R1   PASS B    B       REMAINDER := B/2
0235 00175  237 107013          READ COV  CMPS S1   A       S1 := NQUOTIENT
0236 00176  320 110242          JMP  CNDX ONES      RMDR    TEST FOR ZERO QUOTIENT
0237 00177  010 042507                PASS L    S2
0238 00200  327 150102          JMP  CNDX AL15 RJS  *+2     TEST FOR EXPECTED QUOTIENT SIGN
0239 00201  007 140147                INC  A    S1        COMPLEMENT QUOTIENT
0240 00202  234 106747          READ      XOR       A       COMPARE QUOTIENT
0241 00203  327 150242          JMP  CNDX AL15 RJS  RMDR     WITH EXPECTED SIGN
0242 00204  230 036754          READ SOV
0243 00205  374 040742  RMDR    RTN  CNDX FLAG RJS          TEST EXPECTED SIGN OF REMAINDER
0244 00206  237 110207          READ      CMPS B    B       BEGIN 2'S COMPLEMENT OF REMAINDER
0245 00207  367 110207          RTN       INC  B    B       COMPLETE TWOS COMPLEMENT
```

```
0247 00210  010 036753  ASL           COV
0248 00211  010 036767                RPT
0249 00212  030 010222          ARS   L1    PASS B    B          ARITHMETIC LEFT SHIFT
0250 00213  223 041000  ADDS1   READ  RTN   ADD  S1   S1         READ, S1 SET FOR UPDCPUS
0251                    *
0252 00214  030 010224  ASR     ARS   R1    PASS B    B          ARITHMETIC SHIFT RIGHT
0253 00215  230 075313          READ  COV   PASS S7   P          READ, S7 SET FOR SELF TEST
0254 00216  370 077207          RTN         PASS S5   S          S5 SET FOR SELF TEST
0255                    *
0256 00217  070 010222  LSL     LGS   L1    PASS B    B          LOGICAL LEFT SHIFT
0257 00220  230 036740          READ  RTN                        READ
0258                    *
0259 00221  070 010224  LSR     LGS   R1    PASS B    B          LOGICAL RIGHT SHIFT
0260 00222  230 036740          READ  RTN                        READ
0261                    *
0262 00223  050 010222  RRL     CRS   L1    PASS B    B          ROTATE LEFT
0263 00224  230 036740          READ  RTN                        READ
0264                    *
0265 00225  050 010224  RRR     CRS   R1    PASS B    B          ROTATE RIGHT
0266 00226  230 036740          READ  RTN                        READ
0267                    *
0268 00227  320 013005  JTBL1010 JMP  J74           EM1010
0269                    *
```

```
0271 00230  323 112742  TIMER    JMP  CNDX HOI       INDIRECT+6 TEST FOR HALT OR INTERRUPT
0272 00231  007 110207                INC  B    B          INCREMENT B
0273 00232  320 051402           JMP  CNDX ALZ  RJS  *-2       TEST FOR ZERO
0274 00233  230 036740           READ RTN
0275                     *
0276 00234  017 117007  MODE           CMPS S1   DSPI       S1 := COMPLEMENTED INDICATOR BITS
0277 00235  334 011742           JMP  CNDX FLAG       *+2
0278 00236  370 040357           RTN  STFL PASS DSPI S1       REVERSE INDICATOR BITS;COMP. FLAG
0279 00237  370 040356           RTN  CLFL PASS DSPI S1       REVERSE INDICATOR BITS;COMP. FLAG
0280                     *
0281                     *        EAU/MACTABLE 1 000 000 0
0282                     *        ------------
0283                     *
0284 00240  320 013257  EM1000   JMP  STFL           DIAG       00 00
0285 00241  320 010407           JMP                 ASL        00 01
0286 00242  320 010767           JMP  RPT            LSL        00 10
0287 00243  320 011407           JMP                 TIMER      00 11
0288 00244  320 011167           JMP  RPT            RRL        01 00
0289 00245  300 061107  FPDIAG   JSB                 RETNFP     01 01 MOD 3 TEST POINT
0290 00246  220 075000  DECP     READ RTN  DEC  S1   P          01 10, S1 SET FOR HALT ROUTINE
0291 00247  220 045100  DECDMS   READ RTN  DEC  S3   S3         01 11, DECREMENT DMS MAP POINTER
0292 00250  320 006004           JMP  RJ30           MPY        10 00
0293                     *
0294                     *        UNIVERSAL INDIRECT OPERAND ROUTINE
0295                     *        ----------------------------------
0296                     *
0297 00251  230 000674  INDIRECT READ INCI PASS M    TAB        M := T/A/B; READ
0298 00252  367 140002           RTN  CNDX AL15 RJS             TEST FOR MORE INDIRECTS
0299 00253  230 036774           READ INCI
0300 00254  323 152442           JMP  CNDX HOI  RJS  INDIRECT   TEST FOR HALT OR INTERRUPT
0301 00255  336 052442           JMP  CNDX NSNG RJS  INDIRECT   TEST FOR INSTRUCTION STEP
0302 00256  337 100302           JMP  CNDX MRG       HORI       TEST FOR JMP,I OR JSB,I
0303 00257  000 075707                DEC  P    P          DECREMENT P
0304                     *
0305                     *        EAU/MACTABLE 1 000 001 0
0306                     *        ------------
0307                     *
0308 00260  320 000307  EM1010   JMP                 HORI       HALT OR INTERRUPT PENDING
0309 00261  320 010627           JMP  RPT            ASR        00 01
0310 00262  320 011067           JMP  RPT            LSR        00 10
0311 00263  227 145100  INCDMS   READ RTN  INC  S3   S3         00 11, INCREMENT DMS MAP POINTER
0312 00264  320 011267           JMP  RPT            RRR        01 00
0313                     *
0314 00265  323 011542  DIAG     JMP  CNDX RUN       TIMER+3    ABORT TEST IF IN RUN MODE
0315 00266  300 032747           JSB                 MEMLOST+1  TEST CPU, 1 MEGAWORD MEMORY
0316 00267  325 053302           JMP  CNDX RUNE RJS  *-1        LOOP IF SWITCH IS (LOCK+RUNE)
```

```
0318                    *
0319                    *       FRONT PANEL ROUTINES
0320                    *       --------------------
0321                    *
0322 00270  350 000507  HALT    IMM       CMLO L    000B      INITIALIZE NON SUSPEND INDICATOR
0323 00271  314 052302  HALTSUSP JSB  CNDX FLAG RJS  DECP      TEST FOR HALT IN TRAP CELL
0324 00272  010 040647                    PASS M    S1
0325 00273  305 172702          JSB  CNDX NMLS RJS  MEMLOST   TEST FOR COLD POWER UP
0326 00274  017 135756               CLFL CMPS S    DES       S := DESCRIPTOR BLOCK
0327 00275  336 054142          JMP  CNDX NSNG RJS  NORPL     CHECK FOR INSTRUCTION STEP
0328 00276  322 114142          JMP  CNDX L15       NORPL     CHECK FOR SUSPEND STATE
0329 00277  323 015542          JMP  CNDX RUN       RUN       TEST FOR AUTO-RESTART
0330 00300  327 154142          JMP  CNDX AL15 RJS  *+3       TEST FOR SWITCH 15
0331 00301  327 024702          JMP  CNDX NSFP      RPL       TEST FOR NO FRONT PANEL
0332 00302  325 064702          JMP  CNDX RUNE RJS  RPL       TEST LOCK POSTION OF POWER SWITCH
0333 00303  327 021702  NORPL   JMP  CNDX NSFP      USER      USER FRONT PANEL MODULE
0334 00304  010 015747                    PASS S    DSPL      S := DSPL REGISTER
0335 00305  006 037107                    ZERO S3             CLEAR DMS MAP POINTER
0336 00306  336 054402          JMP  CNDX NSNG RJS  WAIT      TEST FOR INSTRUCTION STEP
0337 00307  343 156347          IMM       LOW  DSPI 367B      MAKE DSPL INDICATOR=T-REGISTER
0338                    *
0339 00310  300 023547  WAIT    JSB                 DSPICODE  BINARY ENCODE OF DSPL INDICATOR
0340 00311  300 022004          JSB  RJ30           UPDATES   UPDATE DSPL REGISTER
0341 00312  006 036774  PAUSE        INCI ZERO                INITIALIZE SAVE STACK
0342 00313  001 136741               JTAB DBLS
0343 00314  330 154602          JMP  CNDX NSTB RJS  *         WAIT FOR BUTTON TO BE RELEASED
0344 00315  323 024502          JMP  CNDX RUN       SUSPEND   TEST FOR SUSPENDED STATE
0345 00316  330 114702  IDLE    JMP  CNDX NSTB      *         WAIT FOR BUTTON TO BE PRESSED
0346                    *
0347 00317  300 015047  JSBSCAN JSB                 SCAN      GO TO SCAN SUBROUTINE
0348 00320  320 014407          JMP                 WAIT
```

```
0350 00321  007 133047  SCAN                INC  S2   M          S2 := M+1
0351 00322  332 156202            JMP  CNDX NLT  RJS  LEFT       LEFT
0352 00323  332 056702            JMP  CNDX NRT  RJS  RIGHT      RIGHT
0353 00324  331 065142            JMP  CNDX NINC RJS  INCM       INC M
0354 00325  331 165042            JMP  CNDX NDEC RJS  DECM       DEC M
0355 00326  330 065402            JMP  CNDX NLDR RJS  LOADER     IBL/TEST
0356 00327  333 065242            JMP  CNDX NSTR RJS  STORE      STORE
0357 00330  333 151602            JMP  CNDX NMDE RJS  MODE       MODE
0358 00331  336 055602            JMP  CNDX NSNG RJS  INSTP      INSTRUCTION STEP
0359 00332  323 054502            JMP  CNDX RUN  RJS  PAUSE      PRESET
0360                    *
0361 00333  343 076356  RUN       IMM  CLFL LOW  DSPI 337B       MAKE DSPL INDICATOR=S-REGISTER
0362 00334  314 011602  INSTP     JSB  CNDX FLAG      MODE       TEST FOR INVERSE VIDEO
0363 00335  227 174710            READ SRUN INC  PNM  P          M := P; P := P+1; READ
0364 00336  010 076307                      PASS DSPL S          PLACE S IN DSPL REGISTER
0365 00337  010 001007                      PASS S1   TAB        S1 := T/A/B
0366 00340  230 040633            READ FTCH PASS IRCM S1         IR = S1;M = OPERAND ADDRESS;READ
0367 00341  336 000042            JMP  CNDX NSNG      FETCH+1    TEST FOR NOT SINGLE INSTRUCTION
0368 00342  010 040775                 SHLT PASS      S1
0369 00343  320 000047            JMP                 FETCH+1    COMPLETE FETCH
```

```
0371                        *
0372 00344  010 017024  LEFT        R1   PASS S1   DSPI      SHIFT DSPL INDICATOR
0373 00345  334 016442          JMP CNDX FLAG     *+4       TEST FOR REVERSE DISPLAY MODE
0374 00346  321 156402          JMP CNDX AL0  RJS *+2       TEST FOR WRAP-AROUND
0375 00347  370 040347          RTN      PASS DSPI S1
0376 00350  343 076340          IMM RTN  LOW  DSPI 337B     PLACE S-POINTER IN DSPL INDICATOR
0377 00351  321 156542          JMP CNDX AL0  RJS *+2       TEST FOR WRAP-AROUND
0378 00352  340 100340          IMM RTN  LOW  DSPI 040B     PLACE S-POINTER IN DSPL INDICATOR
0379 00353  340 176507          IMM      LOW  L    077B     L := 77
0380 00354  012 041007                   AND  S1   S1       MASK DSPL INDICATOR
0381 00355  370 040347          RTN      PASS DSPI S1
0382                        *
0383 00356  334 017202  RIGHT   JMP CNDX FLAG     *+6       TEST FOR REVERSE DISPLAY MODE
0384 00357  342 176517          IMM STFL LOW  L    277B     L := 177677
0385 00360  150 017022          LWF L1   PASS S1   DSPI     SHIFT DSPL INDICATOR LEFT 1 PLACE
0386 00361  010 140356               CLFL IOR  DSPI S1
0387 00362  360 101002          RTN CNDX ONES               TEST FOR NO WRAP-AROUND
0388 00363  343 174340          IMM RTN  LOW  DSPI 376B     PLACE A-POINTER IN DSPL INDICATOR
0389 00364  010 017022               L1   PASS S1   DSPI
0390 00365  352 176507          IMM      CMLO L    277B     L := 100
0391 00366  012 040347                   AND  DSPI S1       MASK DSPL INDICATOR
0392 00367  360 001002          RTN CNDX ALZ                TEST FOR NO WRAP-AROUND
0393 00370  340 002340          IMM RTN  LOW  DSPI 001B     PLACE X-POINTER IN DSPL INDICATOR
0394                        *
0395 00371  325 000302  HALTIO  JMP CNDX RUNE     HORI      HALT IF RPL NOT ENABLED
0396 00372  012 034747                   AND      DES       CHECK FOR A CHANGE IN DES(15)
0397 00373  327 140302          JMP CNDX AL15 RJS HORI      HALT IF RPL DISABLED OR 106,107XXX
0398 00374  000 075710  SUSPINIT     SRUN DEC  P    P        RESET P FOR FRONT PANEL ROUTINE
0399 00375  320 013447          JMP                HALTSUSP  ENTER SUSPENDED STATE
0400                        *
0401 00376  010 061775  EXITSUSP      SHLT PASS S    S9       HALT CPU, RESTORE S
0402 00377  320 014747          JMP                JSBSCAN   GO TO NORMAL FP. ROUTINES
```

```
0404                                ALGN
0405                                ORG                 400B
0406 00400  370 015747  STORES      RTN       PASS S    DSPL      S := DSPL REGISTER
0407 00401  370 015707  STOREP      RTN       PASS P    DSPL      P := DSPL REGISTER
0408 00402  320 021547              JMP                 STORET
0409 00403  370 014647  STOREM      RTN       PASS M    DSPL      M := DSPL REGISTER
0410 00404  370 014207  STOREB      RTN       PASS B    DSPL      B := DSPL REGISTER
0411 00405  370 014147  STOREA      RTN       PASS A    DSPL      A := DSPL REGISTER
0412 00406  320 021007  STOREST     JMP                 STCPUS
0413 00407  320 020607  STOREF      JMP                 STFENCE
0414 00410  370 014452  STOREMM     RTN  MESP PASS MEU  DSPL      DMS MAP DATA := DSPL REGISTER
0415 00411  370 015107  STOREMN     RTN       PASS S3   DSPL      DMS MAP NUMBER := DSPL REGISTER
0416 00412  370 015647  STOREY      RTN       PASS Y    DSPL      Y := DSPL REGISTER
0417 00413  370 015607  STOREX      RTN       PASS X    DSPL      X := DSPL REGISTER
0418 00414  344 016507  STFENCE     IMM       HIGH L    007B      L := 003777
0419 00415  012 015007              AND  S1   DSPL      MASK DSPL REGISTER
0420 00416  010 022447              PASS MEU  MEU
0421 00417  370 040447              RTN       PASS MEU  S1        STORE INTO DMS FENCE
0422 00420  010 033153  STCPUS          COV  PASS S4   M         SAVE M
0423 00421  010 015022               L1  PASS S1   DSPL
0424 00422  327 161202              JMP  CNDX AL15 RJS  *+2       TEST FOR DSPL 14
0425 00423  010 036754               SOV
0426 00424  342 000607              IMM       LOW  IRCM 200B    SET UP ELB INSTRUCTION
0427 00425  010 041021               SRG1 PASS S1   S1        STORE DSPL 14 INTO EXTEND
0428 00426  352 177047              IMM       CMLO S2   277B    S2 := STF 0 INSTRUCTION
0429 00427  010 040747                    PASS      S1
0430 00430  307 145742              JSB  CNDX AL15 RJS  CLFS2     TEST FOR INTERRUPT SYSTEM
0431 00431  010 042606               IOG  PASS IRCM S2
0432 00432  370 046647              RTN       PASS M    S4        RESTORE M
0433 00433  210 014007  STORET      WRTE      PASS TAB  DSPL      T := DSPL REGISTER
0434 00434  010 042647                    PASS M    S2        INCREMENT M
0435 00435  320 014507              JMP                 PAUSE     DO NOT UPDATE DSPL
0436                    *
0437 00436  325 040007  USER        JMP                 25000B    JUMP TO USER FRONT PANEL ROUTINE
```

```
0439                                    ORG                 440B
0440 00440  370 076307  UPDATES  RTN        PASS DSPL S          DSPL REGISTER := S
0441 00441  370 074307  UPDATEP  RTN        PASS DSPL P          DSPL REGISTER := P
0442 00442  370 000307  UPDATET  RTN        PASS DSPL TAB        DSPL REGISTER := T
0443 00443  370 032307  UPDATEM  RTN        PASS DSPL M          DSPL REGISTER := M
0444 00444  370 010307  UPDATEB  RTN        PASS DSPL B          DSPL REGISTER := B
0445 00445  370 006307  UPDATEA  RTN        PASS DSPL A          DSPL REGISTER := A
0446 00446  320 022707  UPDATEST JMP                  UPDCPUS
0447 00447  320 022607  UPDATEF  JMP                  UPDFENCE
0448 00450  370 022312  UPDATEMM RTN   MESP PASS DSPL MEU        DSPL REGISTER := DMS MAP DATA
0449 00451  370 044307  UPDATEMN RTN        PASS DSPL S3         DSPL REGISTER := DMS MAP NUMBER
0450 00452  370 072307  UPDATEY  RTN        PASS DSPL Y          DSPL REGISTER := Y
0451 00453  370 070307  UPDATEX  RTN        PASS DSPL X          DSPL REGISTER := X
0452 00454  010 022447  UPDFENCE            PASS MEU  MEU
0453 00455  370 022307           RTN        PASS DSPL MEU        DSPL REG=DMS STATUS/FENCE REG
0454 00456  355 176507  UPDCPUS  IMM        CMHI L    177B       L := 100000
0455 00457  010 033047                      PASS S2   M          SAVE M
0456 00460  350 177007           IMM        CMLO S1   077B       S1 := 000300  SFS 0
0457 00461  010 040606                IOG   PASS IRCM S1         IR := SFS 0
0458 00462  006 037007                      ZERO S1              INITIALIZE CPU STATUS WORD
0459 00463  316 105342           JSB  CNDX SKPF       ADDR1S1    TEST FOR INTERRUPT SYSTEM ON
0460 00464  314 110542           JSB  CNDX E          ADDS1      TEST FOR EXTEND SET
0461 00465  010 041024                R1   PASS S1   S1
0462 00466  315 110542           JSB  CNDX OVFL       ADDS1      TEST FOR OVERFLOW SET
0463 00467  010 024507                      PASS L    CIR        L := CIR
0464 00470  010 141007                      IOR  S1   S1         MERGE IN CIR
0465 00471  010 042647                      PASS M    S2         RESTORE M
0466 00472  370 040307           RTN        PASS DSPL S1         DSPL := E,O,I, AND CIR
0467                        *
0468 00473  343 164547  DSPICODE IMM        LOW  CNTR 372B       CNTR := 000372
0469 00474  010 077407                      PASS S9   S          SAVE S IN CASE OF SUSPEND STATE
0470 00475  012 137347                      PASL S8              S8 := DES(OLD)
0471 00476  017 117023                L4    CMPS S1   DSPI       S1 := NDSPI SHIFTED LEFT FOUR
0472 00477  334 064142           JMP  CNDX FLAG RJS  *+4         TEST FOR NO REVERSE DISPLAY MODE
0473 00500  340 000547           IMM        LOW  CNTR 000B       CNTR := 000
0474 00501  344 000507           IMM        HIGH L    000B       L := 000377
0475 00502  013 017023                L4    XNOR S1   DSPI
0476 00503  001 141026                ICNT  DBLS S1   S1         LEFT SHIFT S1; INCREMENT COUNTER
0477 00504  327 164142           JMP  CNDX AL15 RJS  *-1         TEST FOR INDICATOR BIT
0478 00505  352 000507           IMM        CMLO L    200B       L := 177
0479 00506  012 045107                      AND  S3   S3         MASK DMS MAP POINTER
0480 00507  357 076507           IMM        CMHI L    337B       L := 020000
0481 00510  230 145007           READ       IOR  S1   S3         MERGE DMS CONTROL BIT
0482 00511  370 040447           RTN        PASS MEU  S1         LOAD DMS MAP ADDRESS REGISTER
```

```
0484                    *
0485 00512  330 157702  SUSPEND  JMP  CNDX NSTB RJS  EXITSUSP  ABORT IF A BUTTON IS PUSHED
0486 00513  010 057747                  PASS S    S8         S := DES(OLD), ALLOW STATUS UPDATE
0487 00514  323 024502            JMP  CNDX RUN     SUSPEND   LOOP IF STILL IN SUSPEND STATE
0488 00515  325 014402            JMP  CNDX RUNE    WAIT      AVOID RPL IF NOT IN LOCK POSITION
0489 00516  347 000447  RPL       IMM       HIGH MEU 300B      DISABLE DMS MAPS
0490 00517  300 025407            JSB                 LOADER    GO TO LOADER SUBROUTINE
0491 00520  320 015547            JMP                 RUN       REMOTE PROGRAM LOAD
0492                    *
0493 00521  334 012342  DECM      JMP  CNDX FLAG      DECDMS    TEST FOR REVERSE DISPLAY MODE
0494 00522  000 033047                  DEC  S2   M              DECREMENT M
0495 00523  334 013142  INCM      JMP  CNDX FLAG      INCDMS    TEST FOR REVERSE DISPLAY MODE
0496 00524  370 042647            RTN       PASS M    S2
0497                    *
0498 00525  300 023547  STORE     JSB                 DSPICODE  BINARY ENCODE OF DSPL INDICATOR
0499 00526  300 020004            JSB  RJ30            STORES    STORE DSPL REGISTER
0500 00527  320 014407            JMP                 WAIT
```

```
0502                        *
0503                        *         INITIAL BINARY LOADER
0504                        *         --------------------
0505                        *
0506 00530   345 177014  LOADER   IMM  SOV  HIGH S1   177B      S1 := 077777
0507 00531   010 076607                    PASS IRCM S         IR = S TO SET UP LOADER SELECTION
0508 00532   343 000507  MEMSIZE  IMM       LOW  L    300B      L  := 177700
0509 00533   012 040707                    AND  PNM  S1        M  := S1; P := S1 AND L
0510 00534   367 101002            RTN  CNDX AL15                TEST FOR NO READ/WRITE CAPABILITY
0511 00535   210 074007            WRTE      PASS TAB  P         WRITE INTO MEMORY
0512 00536   357 136507            IMM       CMHI L    357B      L  := 010000
0513 00537   224 141007            READ      SUB  S1   S1        READ BACK FROM MEMORY
0514 00540   010 074507                      PASS L    P         L  := WRITTEN DATA
0515 00541   014 100747                      XOR       TAB       COMPARE
0516 00542   320 065502            JMP  CNDX ALZ  RJS  MEMSIZE   TEST FOR PRESENT MEMORY
0517 00543   010 075007                      PASS S1   P         S1 := P
0518                        *
0519 00544   350 007063  SELCODE  IMM  L4   CMLO S2   003B      S2 := 007700
0520 00545   010 042507                      PASS L    S2
0521 00546   340 014547            IMM       LOW  CNTR 006B      COUNTER := 6
0522 00547   012 077067                 RPT  AND  S2   S         MASK SELECT CODE
0523 00550   010 043064                 R1   PASS S2   S2        SHIFT SELECT CODE 6 PLACES RIGHT
0524 00551   353 156507            IMM       CMLO L    367B      L  := 000010
0525 00552   004 143071                 IAK  SUB  S2   S2        S2 := SELECT CODE -10,SYNC TO T6
0526 00553   367 101042            RTN  CNDX AL15                TEST FOR SELECT CODE LESS THAN 10
0527                        *
0528 00554   010 040647  LOOP                PASS M    S1
0529 00555   010 031023                 L4   PASS S1   LDR
0530 00556   010 040526                 ICNT PASS L    S1        THE FIRST PART OF THIS LOOP
0531 00557   012 031023                 L4   AND  S1   LDR       ROUTINE PACKS EACH FOUR BIT
0532 00560   010 040526                 ICNT PASS L    S1        SEGMENT FROM THE SPECIFIED
0533 00561   012 031023                 L4   AND  S1   LDR       LOADER ROM INTO A 16-BIT WORD
0534 00562   010 040526                 ICNT PASS L    S1
0535 00563   015 131013                 COV  NAND S1   LDR       T5 ON FIRST PASS.LDR ->> PRESET
```

```
0537 00564  354 026526           IMM  ICNT CMHI L    013B      L := 172000
0538 00565  012 041107                     AND  S3   S1
0539 00566  345 166507           IMM       HIGH L    173B      L := 075777
0540 00567  013 044747                     XNOR      S3
0541 00570  320 070102           JMP  CNDX ALZ  RJS  STWORD   TEST FOR I/O INSTRUCTION
0542 00571  350 077122           IMM  L1   CMLO S3   037B      S3 := 000700
0543 00572  010 044507                     PASS L    S3
0544 00573  012 040747                     AND       S1
0545 00574  320 030102           JMP  CNDX ALZ       STWORD   TEST FOR HALT INSTRUCTION
0546 00575  353 016507           IMM       CMLO L    307B      L := 000070
0547 00576  012 040747                     AND       S1
0548 00577  320 030102           JMP  CNDX ALZ       STWORD   TEST FOR SELECT CODE LESS THAN 10
0549 00600  010 042507                     PASS L    S2
0550 00601  003 041007                     ADD  S1   S1        PATCH IN CONFIGURING SELECT CODE
0551 00602  210 040007  STWORD  WRTE       PASS TAB  S1        WRITE WORD INTO MEMORY
0552 00603  007 133007                     INC  S1   M
0553 00604  326 166602           JMP  CNDX CNT8 RJS  LOOP      TEST FOR LOADER COMPLETION
0554 00605  017 175007                     CMPS S1   P         TWOS COMPLEMENT LAST AVAILABLE
0555 00606  007 141007                     INC  S1   S1        WORD OF PROGRAM MEMORY AND
0556 00607  210 040007           WRTE      PASS TAB  S1        STORE INTO LAST LOADER ADDRESS
0557 00610  000 033007                     DEC  S1   M
0558 00611  230 040647           READ      PASS M    S1
0559 00612  010 042507                     PASS L    S2        PATCH SELECT CODE INTO
0560 00613  003 001007                     ADD  S1   TAB       PORT CONTROLLER WORD 1
0561 00614  210 040007           WRTE      PASS TAB  S1        STORE PORT CONTROLLER  WORD 1
0562 00615  300 031447           JSB                 CPTEST    PERFORM QUICK PROCESSOR TEST
```

```
0564                        *
0565                        *         FIRMWARE DIAGNOSTICS
0566                        *         --------------------
0567                        *
0568 00616  220 033007  TEST32K  READ       DEC  S1   M          S1 := M - 1; READ MEMORY WORD
0569 00617  360 000642           RTN  CNDX  ALZ                  CHECK FOR TEST COMPLETION
0570 00620  017 101047                      CMPS S2   TAB        S2 := COMPLEMENTED DATA
0571 00621  210 042007           WRTE       PASS TAB  S2         T/A/B := COMPLEMENTED DATA; WRITE
0572 00622  230 042507           READ       PASS L    S2         L := COMPLEMENTED DATA
0573 00623  017 143047                      CMPS S2   S2         S2 := ORIGINAL DATA
0574 00624  014 100747                      XOR       TAB        COMPARE
0575 00625  320 076502           JMP  CNDX  ALZ  RJS  FAILURE    TEST FOR MEMORY FAILURE
0576 00626  210 042007           WRTE       PASS TAB  S2         T/A/B=ORIG. DATA;RESTORE MEMORY
0577 00627  010 040647                      PASS M    S1
0578 00630  320 030707           JMP                 TEST32K
0579                        *
0580 00631  343 053122  CPTEST   IMM  L1    LOW  S3   325B       S3 := 177652
0581 00632  346 124507           IMM        HIGH L    252B       L  := 125377
0582 00633  012 045416                CLFL AND  S9   S3         S9 := 125252
0583 00634  300 031707           JSB                 REGTEST
0584 00635  010 063417                CLFL PASS S9   S10        S9 := 052525
0585                        *
0586 00636  017 161454  REGTEST      SOV  CMPS S10  S9         S10 = NS9   THIS ROUTINE LOADS
0587 00637  150 063162           LWF  L1   PASS S4   S10        S4 = NS10   WITH ONE OF TWO
0588 00640  150 047224           LWF  R1   PASS S5   S4         S5 = NS4   COMPLEMENTARY DATA
0589 00641  017 151263                L4   CMPS S6   S5         S6 = NS5   PATTERNS.  REGISTERS
0590 00642  157 153322           LWF  L1   CMPS S7   S6         S7 =  S6   WITH 1 BIT DIFFER.
0591 00643  150 055364           LWF  R1   PASS S8   S7         S8 = NS7   IN ADDRESS ARE FILLED
0592 00644  017 157507                      CMPS S11  S8         S11= NS8   THE ROTATE/SHIFT AND
0593 00645  010 050507                      PASS L    S5         L := OTHER TEST PATTERN
0594 00646  014 157256                CLFL XOR  S6   S8         XOR SAME PATTERN, S6 := 000000
0595 00647  320 076602           JMP  CNDX  ALZ  RJS  FAILURE+2 TEST FOR NON-ZEROS
0596 00650  013 064747                      XNOR      S11        XNOR DIFFERENT PATTERN
0597 00651  320 076602           JMP  CNDX  ALZ  RJS  FAILURE+2 TEST FOR NON-ZEROS
0598 00652  003 054747                      ADD       S7         ADD UNLIKE PATTERNS
0599 00653  321 036602           JMP  CNDX  COUT     FAILURE+2 TEST FOR CARRY OUT
0600 00654  320 110642           JMP  CNDX  ONES     ASR+1      TEST FOR NON-ONES
0601 00655  320 036607           JMP                 FAILURE+2
```

```
0603 00656   006 037747  MEMLOST            ZERO S              CLEAR DISPLAY ON POWER UP
0604 00657   300 031447           JSB                  CPTEST    TEST CENTRAL PROCESSOR
0605                     *
0606 00660   010 052313  RIPP1MW      COV   PASS DSPL S6        CLEAR DISPLAY REGISTER
0607 00661   340 100547  DMSLOAD IMM       LOW  CNTR 040B      COUNTER := 40
0608 00662   357 077047          IMM       CMHI S2   337B      S2 := 020000
0609 00663   345 004447          IMM       HIGH MEU  102B      ENABLE SYSTEM MAP
0610 00664   010 042447                    PASS MEU  S2        CLEAR DMS ADDRESS REGISTER
0611 00665   010 052452                MESP PASS MEU  S6        LOAD MAP
0612 00666   007 153265                DCNT INC  S6   S6        INCREMENT MAP ADDRESS
0613 00667   326 173242          JMP   CNDX CNT8 RJS  *-2        TEST FOR ALL MAPS LOADED
0614 00670   353 077747          IMM       CMLO S    337B      PASS LOADER INVALID SC.,SET IR
0615 00671   300 025407          JSB                 LOADER    FIND HOW MUCH MEMORY AVAILABLE
0616 00672   010 015761                SRG1 PASS S    DSPL      RESTORE S. CLEAR EXTEND
0617 00673   010 033107                     PASS S3   M         S3 := TOP OF ENABLED MEMORY
0618 00674   322 134402          JMP   CNDX L15       TESTDMS   TEST FOR PRESENT MEMORY
0619 00675   353 175047          IMM       CMLO S2   376B      BACKGROUND PATTERN := 000001
0620 00676   343 154147          IMM       LOW  A    366B      TEST PATTERN := 177766
0621 00677   300 035107          JSB                 RIPP32K   TEST #1
0622 00700   355 177047          IMM       CMHI S2   177B      BACKGROUND PATTERN := 100000
0623 00701   353 154147          IMM       CMLO A    366B      TEST PATTERN := 000011
0624 00702   300 035107          JSB                 RIPP32K   TEST #2
0625 00703   343 177047          IMM       LOW  S2   377B      BACKGROUND PATTERN := 177777
0626 00704   355 176147          IMM       CMHI A    177B      TEST PATTERN := 100000
0627 00705   300 035107          JSB                 RIPP32K   TEST #3
0628 00706   010 051047                    PASS S2   S5        BACKGROUND PATTERN := S5 (DSPL)
0629 00707   300 035107          JSB                 RIPP32K   TEST #4
0630 00710   010 022447  TESTDMS           PASS MEU  MEU
0631 00711   010 022747                    PASS      MEU       ENABLE MEM STATUS REGISTER
0632 00712   320 135002          JMP   CNDX ONES      *+6       TEST IF DMS IS PRESENT
0633 00713   007 115747                       INC  S    DSPL    S := DISPLAY REGISTER
0634 00714   353 076507          IMM       CMLO L    337B      L := 40
0635 00715   014 176307                    XOR  DSPL S         DISPLAY REGISTER := S
0636 00716   320 073042          JMP   CNDX ALZ  RJS  DMSLOAD   TEST FOR ALL MEMORY TESTED
0637 00717   345 000447          IMM       HIGH MEU  100B      DISABLE DMS MAPS
0638 00720   010 051775                SHLT PASS S    S5        RESTORE S
0639 00721   360 055707          RTN       DEC  P    S7        RESTORE P AND EXIT
```

```
0641 00722  000 044735  RIPP32K       SHLT DEC  PNM  S3
0642 00723  210 042007           WRTE       PASS TAB  S2         T/A/B := BACKGROUND PATTERN
0643 00724  000 074707                      DEC  PNM  P          M := P; P := P-1
0644 00725  327 175142           JMP  CNDX AL15 RJS  *-2         TEST FOR COMPLETE 32K
0645 00726  352 177147           IMM       CMLO S4   277B       S4 := 000100
0646 00727  010 046715                PRST PASS PNM  S4         P := S4; M := S4
0647 00730  210 006007  RIPLOOP  WRTE       PASS TAB  A          T/A/B := TEST PATTERN; WRITE
0648 00731  352 174507           IMM       CMLO L    276B       L := 000101
0649 00732  223 075707           READ      ADD  P    P          P := P + 101
0650 00733  010 006515                PRST PASS L    A          L := TEST PATTERN
0651 00734  014 100747                     XOR       TAB        COMPARE
0652 00735  320 076542           JMP  CNDX ALZ  RJS  FAILURE+1  TEST FOR SUCCESSFUL COMPARE
0653 00736  010 044515                PRST PASS L    S3         L := TOP OF ENABLED MEMORY
0654 00737  210 042007           WRTE       PASS TAB  S2         T/A/B= BACKGROUND PATTERN RESTORE
0655 00740  004 174647                     SUB  M    P          TEST FOR NON-EXISTENT MEMORY
0656 00741  321 075402           JMP  CNDX COUT RJS  RIPLOOP    TEST FOR RIPPLE PASS COMPLETE
0657 00742  000 047147                     DEC  S4   S4         DECREMENT 32K COUNTER
0658 00743  327 175342           JMP  CNDX AL15 RJS  RIPLOOP-1  TEST FOR ENTIRE 32K TESTED
0659 00744  010 042507                     PASS L    S2         L := BACKGROUND PATTERN
0660 00745  010 044707                     PASS PNM  S3         P := TOP OF ENABLED MEMORY
0661 00746  220 074707  BKGNDCK   READ      DEC  PNM  P          M := P; P := P-1; READ
0662 00747  367 101702           RTN  CNDX AL15                 TEST FOR ENTIRE 32K READ
0663 00750  014 100747                     XOR       TAB        TEST AGAINST EXPECTED BACKGROUND
0664 00751  320 036302           JMP  CNDX ALZ       BKGNDCK    TEST FOR EXPECTED BACKGROUND
0665                     *
0666 00752  010 042147  FAILURE           PASS A    S2         A := EXPECTED DATA
0667 00753  010 000213                COV  PASS B    TAB        B := ACTUAL DATA
0668 00754  340 000375           IMM  SHLT LOW  DSPI 000B       SET ALL DISPLAY INDICATOR BITS
0669 00755  343 176307           IMM       LOW  DSPL 377B       SET ALL DISPLAY REGISTER BITS
0670 00756  327 054502           JMP  CNDX NSFP RJS  PAUSE      SUSPEND TEST
0671 00757  320 021717           JMP  STFL           USER       FLAG --> TEST REPORTED ERROR
0672                     *
```

```
0674                                ORG                 760B
0675                          *
0676                          *     PRIMARY MAPPING TABLE
0677                          *     --------------------
0678                          *
0679 00760  324 161007  MACTABL1 JMP                 23420B    HP RESERVED
0680 00761  324 161007           JMP                 23420B    HP RESERVED
0681 00762  325 140004           JMP  RJ30           27000B
0682 00763  324 160004           JMP  RJ30           23400B    HP RESERVED
0683 00764  325 160004           JMP  RJ30           27400B
0684 00765  326 000004           JMP  RJ30           30000B
0685 00766  326 020004           JMP  RJ30           30400B
0686 00767  326 040004           JMP  RJ30           31000B
0687 00770  327 000004           JMP  RJ30           34000B
0688 00771  327 020004           JMP  RJ30           34400B
0689 00772  327 040004           JMP  RJ30           35000B
0690 00773  327 060004           JMP  RJ30           35400B
0691 00774  324 000004           JMP  RJ30           20000B    DYNAMIC MAPPING SYSTEM
0692 00775  324 001004           JMP  RJ30           20020B    DYNAMIC MAPPING SYSTEM
0693 00776  320 041004           JMP  RJ30           EIG       EXTENDED INSTRUCTION GROUP
0694 00777  320 042004           JMP  RJ30           EIG+20B   EXTENDED INSTRUCTION GROUP
0695                          *
0696 01000  320 061557  MACTABL0 JMP  STFL           FAD       FLOATING POINT ADD
0697 01001  320 061547           JMP                 FSB       FLOATING POINT SUBTRACT
0698 01002  320 065107           JMP                 FMP       FLOATING POINT MULTIPLY
0699 01003  320 067047           JMP                 FDV       FLOATING POINT DIVIDE
0700 01004  320 060257           JMP  STFL           FIX       FLOATING POINT TO INTEGER
0701 01005  320 060007           JMP                 FLT       INTEGER TO FLOATING POINT
0702 01006  327 100004           JMP  RJ30           36000B
0703 01007  327 140004           JMP  RJ30           37000B
0704 01010  324 040004           JMP  RJ30           21000B    FAST FORTRAN
0705 01011  324 060004           JMP  RJ30           21400B    FAST FORTRAN
0706 01012  324 100004           JMP  RJ30           22000B    EMA
0707 01013  324 120004           JMP  RJ30           22400B    HP RESERVED
0708 01014  324 140004           JMP  RJ30           23000B    DS1000
0709 01015  325 000004           JMP  RJ30           24000B    HP RESERVED
0710 01016  325 100004           JMP  RJ30           26000B    HP RESERVED
0711 01017  325 120004           JMP  RJ30           26400B    HP RESERVED
```

```
0713                        *
0714                        *         EXTENDED INSTRUCTION GROUP
0715                        *         -------------------------
0716                        *
0717 01020   320 043007  EIG        JMP                 S*X        SAX/SBX
0718 01021   370 003607            RTN       PASS X    CAB        CAX/CBX
0719 01022   320 043307            JMP                 L*X        LAX/LBX
0720 01023   320 043647            JMP                 STX        STX
0721 01024   370 070047            RTN       PASS CAB  X          CXA/CBX
0722 01025   320 044007            JMP                 LDX        LDX
0723 01026   320 044147            JMP                 ADX        ADX
0724 01027   320 044347            JMP                 X*X        XAX/XBX
0725 01030   320 045007            JMP                 S*Y        SAY/SBY
0726 01031   370 003647            RTN       PASS Y    CAB        CAY/CBY
0727 01032   320 045307            JMP                 L*Y        LAY/LBY
0728 01033   320 045607            JMP                 STY        STY
0729 01034   370 072047            RTN       PASS CAB  Y          CYA/CYB
0730 01035   320 045747            JMP                 LDY        LDY
0731 01036   320 046107            JMP                 ADY        ADY
0732 01037   320 046307            JMP                 X*Y        XAY/XBY
0733 01040   320 044507            JMP                 ISX        ISX
0734 01041   320 044647            JMP                 DSX        DSX
0735 01042   320 046747            JMP                 JLY        JLY
0736 01043   320 054107            JMP                 LBT        LBT
0737 01044   320 054407            JMP                 SBT        SBT
0738 01045   320 051507            JMP                 MBT        MBT
0739 01046   320 052147            JMP                 CBT        CBT
0740 01047   320 053107            JMP                 SFB        SFB
0741 01050   320 046447            JMP                 ISY        ISY
0742 01051   320 046607            JMP                 DSY        DSY
0743 01052   320 047147            JMP                 JPY        JPY
0744 01053   320 056707            JMP                 BITS       SBS
0745 01054   320 056707            JMP                 BITS       CBS
0746 01055   320 056707            JMP                 BITS       TBS
0747 01056   320 047407            JMP                 CMW        CMW
0748 01057   320 050747            JMP                 MVW        MVW
```

```
0750                    *
0751                    *       INDEX REGISTER GROUP
0752                    *       --------------------
0753                    *
0754 01060  300 012447  S*X     JSB                     INDIRECT
0755 01061  010 070507                  PASS L  X       L := X
0756 01062  003 033007                  ADD  S1 M
0757 01063  010 040647                  PASS M  S1      M := X + T/A/B
0758 01064  210 002036          WRTE MPCK PASS TAB CAB  T/A/B := A/B; WRITE
0759 01065  320 043547          JMP                     RETURN
0760                    *
0761 01066  300 012447  L*X     JSB                     INDIRECT
0762 01067  010 070507                  PASS L  X       L := X
0763 01070  003 033007                  ADD  S1 M
0764 01071  230 040647          READ    PASS M  S1      M := X + T/A/B; READ
0765 01072  010 000047                  PASS CAB TAB    A/B := T/A/B
0766 01073  227 174707  RETURN  READ    INC  PNM P      M := P; P := P+1; READ
0767 01074  370 036747          RTN
0768                    *
0769 01075  300 012447  STX     JSB                     INDIRECT
0770 01076  210 070036          WRTE MPCK PASS TAB X    T/A/B := X; WRITE
0771 01077  320 043547          JMP                     RETURN
0772                    *
0773 01100  300 012447  LDX     JSB                     INDIRECT
0774 01101  010 001607                  PASS X  TAB     X := T/A/B
0775 01102  227 174700          READ RTN INC  PNM P     M := P; P := P+1; READ
0776                    *
0777 01103  300 012447  ADX     JSB                     INDIRECT
0778 01104  010 070507                  PASS L  X       L := X
0779 01105  263 001607          ENVF    ADD  X  TAB     X := X + T/A/B
0780 01106  227 174700          READ RTN INC  PNM P     M := P; P := P+1; READ
0781                    *
0782 01107  230 002507  X*X     READ    PASS L  CAB     L := A/B
0783 01110  010 070047                  PASS CAB X      A/B := X
0784 01111  372 137607          RTN     PASL X          X := L
0785                    *
0786 01112  227 171607  ISX     READ    INC  X  X       INCREMENT X; READ
0787 01113  360 041602          RTN  CNDX ALZ  RJS      TEST FOR ZERO
0788 01114  227 174700          READ RTN INC  PNM P     M := P; P := P+1; READ
0789                    *
0790 01115  220 071607  DSX     READ    DEC  X  X       DECREMENT X; READ
0791 01116  360 041602          RTN  CNDX ALZ  RJS      TEST FOR ZERO
0792 01117  227 174700          READ RTN INC  PNM P     M := P; P := P+1; READ
```

```
0794                        *
0795 01120  300 012447  S*Y     JSB                     INDIRECT
0796 01121  010 072507                  PASS L    Y     L := Y
0797 01122  003 033007                  ADD  S1   M
0798 01123  010 040647                  PASS M    S1    M := Y + T/A/B
0799 01124  210 002036          WRTE MPCK PASS TAB  CAB   T/A/B := A/B; WRITE
0800 01125  320 043547          JMP                 RETURN
0801                        *
0802 01126  300 012447  L*Y     JSB                     INDIRECT
0803 01127  010 072507                  PASS L    Y     L := Y
0804 01130  003 033007                  ADD  S1   M
0805 01131  230 040647          READ    PASS M    S1    M := Y + T/A/B; READ
0806 01132  010 000047                  PASS CAB  TAB   A/B := T/A/B
0807 01133  227 174700          READ RTN  INC  PNM  P     M := P; P := P+1; READ
0808                        *
0809 01134  300 012447  STY     JSB                     INDIRECT
0810 01135  210 072036          WRTE MPCK PASS TAB  Y     T/A/B := Y; WRITE
0811 01136  320 043547          JMP                 RETURN
0812                        *
0813 01137  300 012447  LDY     JSB                     INDIRECT
0814 01140  010 001647                  PASS Y    TAB   Y := T/A/B
0815 01141  227 174700          READ RTN  INC  PNM  P     M := P; P := P+1; READ
0816                        *
0817 01142  300 012447  ADY     JSB                     INDIRECT
0818 01143  010 072507                  PASS L    Y     L := Y
0819 01144  263 001647          ENVE    ADD  Y    TAB   Y := Y + T/A/B
0820 01145  227 174700          READ RTN  INC  PNM  P     M := P; P := P+1; READ
0821                        *
0822 01146  230 002507  X*Y     READ    PASS L    CAB   L := A/B
0823 01147  010 072047                  PASS CAB  Y     A/B := Y
0824 01150  372 137647          RTN     PASL Y          Y := L
0825                        *
0826 01151  227 173647  ISY     READ    INC  Y    Y     INCREMENT Y; READ
0827 01152  360 041642          RTN  CNDX ALZ  RJS        TEST FOR ZERO
0828 01153  227 174700          READ RTN  INC  PNM  P     M := P; P := P+1; READ
0829                        *
0830 01154  220 073647  DSY     READ    DEC  Y    Y     DECREMENT Y; READ
0831 01155  360 041642          RTN  CNDX ALZ  RJS        TEST FOR ZERO
0832 01156  227 174700          READ RTN  INC  PNM  P     M := P; P := P+1; READ
```

```
0834                        *
0835                        *       JUMP INSTRUCTIONS
0836                        *       -----------------
0837                        *
0838 01157  300 012447  JLY     JSB                 INDIRECT
0839 01160  010 075647                  PASS Y      P           Y := P
0840 01161  010 033707                  PASS P      M           P := M
0841 01162  320 047247          JMP                 JPY+2       DO MP CHECK, COMPLETE JLY
0842                        *
0843 01163  010 072507  JPY             PASS L      Y           L := Y
0844 01164  003 001707                  ADD  P      TAB         P := Y + T/A/B
0845 01165  344 120607          IMM     HIGH IRCM   050B        PREPARE MP FOR 0,1 PROTECTION
0846 01166  227 174707          READ    INC  PNM    P           M := P; P := P+1, READ
0847 01167  370 036776          RTN  MPCK                       CHECK JUMP TARGETS
```

```
0849                        *
0850                        *         WORD MANIPULATION INSTRUCTIONS
0851                        *         ------------------------------
0852                        *
0853 01170  300 056007  CMW       JSB                       INITIAL
0854 01171  230 006647  LCMW      READ        PASS M        A          M := WORD ADDRESS OF ARRAY 1
0855 01172  010 010647                        PASS M        B          M := WORD ADDRESS OF ARRAY 2
0856 01173  230 000507              READ      PASS L        TAB        L := ARRAY 1 WORD
0857 01174  007 110207                        INC  B        B          BUMP ARRAY 2 ADDRESS
0858 01175  014 101007                        XOR  S1       TAB
0859 01176  327 110342              JMP  CNDX AL15          NOTEQ      TEST FOR SIMILAR SIGN BITS
0860 01177  014 141007                        XOR  S1       S1
0861 01200  004 140747                        SUB           S1
0862 01201  320 050442              JMP  CNDX ALZ  RJS      NOTEQ+2    TEST FOR WORD COMPARE
0863 01202  007 106147                        INC  A        A          BUMP ARRAY 1 ADDRESS
0864 01203  000 045107                        DEC  S3       S3         INCREMENT WORD COUNT
0865 01204  320 003542              JMP  CNDX ALZ           RETURN     TEST FOR COMPLETE COMPARE
0866 01205  335 007442              JMP  CNDX NINT          LCMW       TEST FOR INTERRUPT PENDING
0867 01206  320 056507              JMP                     INTPEND
0868 01207  322 110542  NOTEQ     JMP  CNDX L15           *+4        TEST FOR WORD 1 NEGATIVE
0869 01210  320 050507              JMP                     *+2        AVOID COUT CHECK FOR XOR
0870 01211  321 010542              JMP  CNDX COUT          *+2        TEST FOR WORD 1 LESS THAN WORD 2
0871 01212  007 175707                        INC  P        P          BUMP P
0872 01213  007 175707                        INC  P        P          BUMP P
0873 01214  000 044507                        DEC  L        S3         L := RESIDUAL STRING COUNT
0874 01215  003 010207                        ADD  B        B          UPDATE B PAST STRING
0875 01216  227 174700              READ RTN  INC  PNM      P          M := P; P := P+1; READ
0876                        *
0877 01217  300 056007  MVW       JSB                       INITIAL
0878 01220  230 006647  LMVW      READ        PASS M        A          M  := SOURCE ADDRESS; READ
0879 01221  007 106147                        INC  A        A          BUMP SOURCE ADDRESS COUNTER
0880 01222  010 010647                        PASS M        B          M  := DESTINATION ADDRESS
0881 01223  010 001047                        PASS S2       TAB        S2 := SOURCE WORD
0882 01224  210 042036              WRTE MPCK PASS TAB      S2         STORE SOURCE WORD TO DESTINATION
0883 01225  007 110207                        INC  B        B          BUMP DESTINATION COUNTER
0884 01226  000 045107                        DEC  S3       S3         DECREMENT WORD COUNTER
0885 01227  320 003542              JMP  CNDX ALZ           RETURN     TEST FOR COMPLETE MOVE
0886 01230  335 011002              JMP  CNDX NINT          LMVW       TEST FOR PENDING INTERRUPT
0887 01231  320 056507              JMP                     INTPEND
```

```
0889                        *
0890                        *       BYTE MANIPULATION INSTRUCTIONS
0891                        *       ------------------------------
0892                        *
0893 01232  300 056007  MBT     JSB                     INITIAL
0894 01233  150 007064  LMBT    LWF  R1    PASS S2      A          S2 := FROM WORD ADDRESS
0895 01234  300 055507          JSB                     LDBYTE     JUMP TO BYTE LOADING SUBROUTINE
0896 01235  300 054647          JSB                     STBYTE     JUMP TO BYTE STORING SUBROUTINE
0897 01236  007 106147                  INC  A          A          BUMP FROM ADDRESS
0898 01237  000 045107                  DEC  S3         S3         DECREMENT BYTE COUNT
0899 01240  320 003542          JMP  CNDX ALZ           RETURN     TEST FOR COMPLETE MOVE
0900 01241  335 011542          JMP  CNDX NINT          LMBT       TEST FOR INTERRUPT PENDING
0901 01242  320 056507          JMP                     INTPEND
0902                        *
0903 01243  300 056007  CBT     JSB                     INITIAL
0904 01244  150 007064  LCBT    LWF  R1    PASS S2      A          S2 := WORD ADDRESS
0905 01245  300 055507          JSB                     LDBYTE     JUMP TO BYTE LOADING SUBROUTINE
0906 01246  010 041207                  PASS S5         S1         S5 := BYTE 1
0907 01247  150 011064          LWF  R1    PASS S2      B          S2 := WORD ADDRESS
0908 01250  300 055507          JSB                     LDBYTE     JUMP TO BYTE LOADING SUBROUTINE
0909 01251  007 110207                  INC  B          B          BUMP STRING 2 ADDRESS
0910 01252  010 050507                  PASS L          S5         L := BYTE 1
0911 01253  004 140747                  SUB             S1         SUBTRACT: BYTE 2 - BYTE 1
0912 01254  320 050442          JMP  CNDX ALZ  RJS       NOTEQ+2    TEST FOR BYTE COMPARE
0913 01255  007 106147                  INC  A          A          BUMP STRING 1 ADDRESS
0914 01256  000 045107                  DEC  S3         S3         DECREMENT BYTE COUNT
0915 01257  320 003542          JMP  CNDX ALZ           RETURN     TEST FOR COMPLETE COMPARE
0916 01260  335 012202          JMP  CNDX NINT          LCBT       TEST FOR INTERRUPT PENDING
0917 01261  320 056507          JMP                     INTPEND
0918                        *
0919 01262  344 000507  SFB     IMM         HIGH L      000B       L  := 377B
0920 01263  010 033207                  PASS S5         M          SAVE M
0921 01264  012 007107                  AND  S3         A          S3 := TEST BYTE
0922 01265  014 007163              L4  SANL S4         A
0923 01266  010 047163              L4  PASS S4         S4         S4 := TERMINATION BYTE
0924 01267  150 011064  LSFB    LWF  R1    PASS S2      B          S2 := WORD ADDRESS
0925 01270  300 055507          JSB                     LDBYTE     JUMP TO BYTE LOADING SUBROUTINE
0926 01271  010 040507                  PASS L          S1         L  := RIGHT JUSTIFIED BYTE
0927 01272  014 144747                  XOR             S3         COMPARE TO TEST BYTE
0928 01273  320 014602          JMP  CNDX ALZ           SBT+4      TEST FOR TEST BYTE MATCH
0929 01274  007 110207                  INC  B          B          BUMP STRING ADDRESS
0930 01275  014 146747                  XOR             S4         COMPARE TO TERMINATION BYTE
0931 01276  320 003542          JMP  CNDX ALZ           RETURN     TEST FOR TERMINATION BYTE MATCH
0932 01277  335 013342          JMP  CNDX NINT          LSFB       TEST FOR INTERRUPT PENDING
0933 01300  000 075707                  DEC  P          P          DECREMENT P
0934 01301  320 000307          JMP                     HORI       INTERRUPT PENDING
0935                        *
```

```
0937 01302  150 011064  LBT     LWF  R1   PASS S2   B        S2 := WORD ADDRESS
0938 01303  010 033107                    PASS S3   M        SAVE M
0939 01304  300 055507          JSB                LDBYTE   JUMP TO BYTE LOADING SUBROUTINE
0940 01305  230 044647          READ       PASS M   S3       RESTORE M AND READ
0941 01306  007 110207                     INC  B   B        BUMP BYTE ADDRESS
0942 01307  370 040147          RTN        PASS A   S1       A  := RIGHT JUSTIFIED BYTE
0943                        *
0944 01310  344 000507  SBT     IMM        HIGH L   000B     L  := 000377
0945 01311  010 033207                     PASS S5  M
0946 01312  012 007007                     AND  S1  A        S1 := RIGHT JUSTIFIED BYTE; READ
0947 01313  300 054647          JSB                STBYTE   JUMP TO BYTE STORING SUBROUTINE
0948 01314  230 050640          READ RTN   PASS M   S5       RESTORE M AND READ
```

```
0950                        *
0951                        *         COMMON SUBROUTINES
0952                        *         ------------------
0953                        *
0954 01315  150 011064  STBYTE   LWF  R1    PASS S2    B          S2 := WORD ADDRESS
0955 01316  230 042647          READ       PASS M     S2         M  := WORD ADDRESS; READ
0956 01317  334 055202          JMP  CNDX  FLAG RJS   *+5        TEST FOR HIGH ORDER BYTE
0957 01320  014 000507                     SANL L     TAB        L  := BYTE TO BE PRESERVED
0958 01321  010 141007                     IOR  S1    S1         S1 := WORD WITH MERGED BYTES
0959 01322  210 040036          WRTE MPCK  PASS TAB   S1         STORE WORD INTO MEMORY
0960 01323  367 110207          RTN        INC  B     B          BUMP B
0961 01324  012 000507                     AND  L     TAB        L  := BYTE TO BE PRESERVED
0962 01325  010 041023               L4    PASS S1    S1
0963 01326  010 041023               L4    PASS S1    S1
0964 01327  010 141007                     IOR  S1    S1         S1 := WORD WITH MERGED BYAES
0965 01330  210 040036          WRTE MPCK  PASS TAB   S1         STORE WORD IN MEMORY
0966 01331  367 110207          RTN        INC  B     B          BUMP B
0967                        *
0968 01332  230 042647  LDBYTE   READ       PASS M     S2         READ
0969 01333  344 000507          IMM        HIGH L     000B       L  := 000377
0970 01334  334 055702          JMP  CNDX  FLAG RJS   *+2        TEST FOR HIGH ORDER BYTE
0971 01335  372 001007          RTN        AND  S1    TAB        S1 := RIGHT JUSTIFIED BYTE
0972 01336  014 001023               L4    SANL S1    TAB
0973 01337  370 041023          RTN  L4    PASS S1    S1         S1 := RIGHT JUSTIFIED BYTE
0974                        *
0975 01340  230 036747  INITIAL  READ                             READ
0976 01341  300 012447          JSB                    INDIRECT
0977 01342  007 174707                     INC  PNM   P          M := P; P := P+1
0978 01343  230 001107          READ       PASS S3    TAB        S3 := INITIAL COUNT; READ
0979 01344  320 017502          JMP  CNDX  ALZ        GOFETCH    TEST FOR ZERO WORD COUNT
0980 01345  010 000507                     PASS L     TAB
0981 01346  360 000002          RTN  CNDX  ALZ                    TEST FOR RESIDUAL COUNT
0982 01347  006 037047                     ZERO S2
0983 01350  210 042036          WRTE MPCK  PASS TAB   S2         CLEAR WORD 3
0984 01351  372 137107          RTN        PASL S3                S3 := ACTUAL COUNT
0985                        *
0986 01352  000 075707  INTPEND            DEC  P     P          DECREMENT P
0987 01353  000 074707                     DEC  PNM   P          DECREMENT P
0988 01354  210 044007          WRTE       PASS TAB   S3         STORE PRESENT WORD COUNT
0989 01355  320 000307          JMP                   HORI       INTERRUPT PENDING
```

```
0991                    *
0992                    *         BIT MANIPULATION INSTRUCTIONS
0993                    *         -----------------------------
0994                    *
0995 01356  300 012447  BITS      JSB                   INDIRECT
0996 01357  010 000507                      PASS L      TAB         L := MASK
0997 01360  230 074647            READ      PASS M      P           READ WORD TO BE MODIFIED
0998 01361  300 012447            JSB                   INDIRECT
0999 01362  300 057603            JSB  ION              CBS
1000 01363  210 040036            WRTE MPCK PASS TAB    S1          STORE WORD BACK INTO MEMORY
1001 01364  007 175707                      INC  P      P           P  := P + 1
1002 01365  227 174700            READ RTN  INC  PNM    P
1003                    *
1004 01366  007 175707  FTBS                INC  P      P           P  := P + 1
1005 01367  007 174707                      INC  PNM    P           M  := P; P := P + 1
1006 01370  234 140747            READ      XOR         S1
1007 01371  320 017542            JMP  CNDX ALZ         *+2
1008 01372  227 174707  GOFETCH   READ      INC  PNM    P           M := P; P := P+1; READ
1009 01373  320 000007            JMP                   FETCH
1010                    *
1011 01374  374 001007  CBS       RTN       SANL S1     TAB         S1 := WORD WITH BITS CLEARED
1012 01375  012 001007  TBS                 AND  S1     TAB         S1 := WORD WITH BITS CLEARED
1013 01376  320 057307            JMP                   FTBS        FINISH TBS
1014 01377  370 101007  SBS       RTN       IOR  S1     TAB         S1 := WORD WITH BITS SET
```

```
1016                            ORG                 1400B
1017                    ********************************************************************************
1018                    *
1019                    *       21XE MICRO-CODE
1020                    *       MODULE 03: FLOATING POINT INSTRUCTIONS
1021                    *
1022                    *       REV 1976-04-26-1800        EAS
1023                    ********************************************************************************
1024                    *
1025                    *
1026                    ********************************************************************************
1027                    FLOAT   EQU                 *
1028 01400  010 006213 FLT           COV  PASS B    A
1029 01401  006 036147                  ZERO A              CLEAR LSB'S TO SHIFT INTO B
1030 01402  353 141147            IMM   CMLO S4   %360      SET EXPONENT FOR MAX INTEGER
1031 01403  000 075707                  DEC  P     P         BECAUSE PACK BUMPS IT
1032 01404  320 073007            JMP               PACK
```

```
1034                        *
1035                        *         ON ENTRY-- A,B = FLOATING POINT NUMBER
1036                        *                    FLAG = 1
1037                        *
1038                        *         ON EXIT    A = INTEGER    B = CHANGED(USUALLY = A,THOUGH)
1039                        *
1040                        *         USES A,B,S1,S2
1041                        *
1042 01405  340 000513  FIX       IMM  COV  LOW  L    %000       L := 1 111 111 100 000 000
1043 01406  153 111024           LWF  R1   NSOL S1   B          S1 := - EXP - 1
1044 01407  321 120442           JMP  CNDX ALO       FIXOK1     RETURN ZERO IF EXP < 0
1045 01410  226 036140           READ RTN  ZERO A
1046 01411  007 141007  FIXOK1              INC  S1   S1         S1 := -EXP
1047                        *
1048 01412  012 011047                      AND  S2   B          B := LSB'S
1049 01413  010 006207                      PASS B    A          B := MSB'S
1050 01414  010 042147                      PASS A    S2         A := LO BITS
1051 01415  353 140507           IMM        CMLO L    %360       L := 15
1052 01416  003 041014                SOV   ADD  S1   S1         CALCULATE 17 - EXP
1053 01417  320 021242           JMP  CNDX ALZ       RTRNINTG   NO SHIFTING IF EXP = 17
1054 01420  327 161142           JMP  CNDX AL15 RJS  FIXOK2     OVERFLOW IF EXP > 17
1055 01421  011 136164                R1   ONE  A              SET A TO MAX INTEGER
1056 01422  230 036740  RETNFP   READ RTN                       START INSTRUCTION READ; EXIT
1057                        *
1058 01423  010 040567  FIXOK2       RPT  PASS CNTR S1         COUNTER := #SHIFTS; SET REPEAT FF
1059 01424  030 010224           ARS  R1   PASS B    B          DO THE SHIFTS
1060                        *
1061                        RTRNINTG EQU                *
1062 01425  010 006513                SOV  PASS L    A          L := LSB'S FROM SHIFT
1063 01426  010 010147                     PASS A    B          A := INTEGER
1064 01427  327 161102           JMP  CNDX AL15 RJS  RETNFP     WE ARE DONE IF A POSITIVE INTEGER
1065 01430  010 142747                     IOR       S2         ELSE CHECK FOR ROUND NECESSARY
1066 01431  320 021102           JMP  CNDX ALZ       RETNFP     RETURN IF NO BITS HANGING
1067                        *
1068 01432  227 106140           READ RTN  INC  A    A          ELSE ROUND UP AND RETURN
```

```
1070                    *     F A D  /  F S B -- FLOATING POINT ADD / SUBTRACT
1071                    *     -----     -----    -------- ----- ---   --------
1072                    *        ON ENTRY--A,B = FIRST OPERAND
1073                    *                  P = POINTER TO ADDRESS OF SECOND OPERAND
1074                    *                  FLAG = 1 MEANS ADD    =0 MEANS SUBTRACT
1075                    *        ON EXIT-- A,B = (FIRST OPERAND) +(-) (SECOND OPERAND)
1076                    *
1077                    *        USES REGISTERS S1,S2,S3,S4,S5,S6
1078                    *
1079                    FSB      EQU                 *        ITS THE SAME AS FAD
1080 01433  230 036747 FAD      READ
1081                    *
1082 01434  300 012447          JSB                 INDIRECT  GO CLEAR INDIRECTS IF NECESSARY
1083 01435  300 071607          JSB                 UNPACK    GO UNPACK THE NUMBERS
1084 01436  010 042747                    PASS      S2        IS OP2 = 0?
1085 01437  320 062042          JMP  CNDX ALZ  RJS  *+2       SKIP IF NOT
1086 01440  342 001147          IMM       LOW  S4   %200      EXP(D) := -200
1087                    *
1088 01441  010 010747                    PASS      B         IS OP1 = 0?
1089 01442  320 062202          JMP  CNDX ALZ  RJS  *+2       SKIP IF NOT
1090 01443  342 001207          IMM       LOW  S5   %200      EXP(C) := -200
1091                    *
1092 01444  334 022742          JMP  CNDX FLAG      DIFR      SKIP AHEAD IF DOING AN ADD
1093 01445  017 143047                    CMPS S2   S2        -ELSE NEGATE OP2
1094 01446  017 145107                    CMPS S3   S3
1095 01447  007 145107                    INC  S3   S3
1096 01450  321 062742          JMP  CNDX COUT RJS  DIFR      IF NO CARRY OUT, GO PROCEED
1097 01451  007 143047                    INC  S2   S2        -BUMP MSB'S
1098 01452  327 162742          JMP  CNDX AL15 RJS  DIFR      IF POSITIVE, GO PROCEED
1099 01453  001 142747                    DBLS      S2        -WAS IT  100...0?
1100 01454  320 062742          JMP  CNDX ALZ  RJS  DIFR
1101                    *
1102 01455  010 043064               R1   PASS S2   S2        YES, MAKE IT 010...0
1103 01456  007 147147                    INC  S4   S4        AND ADJUST EXPONENT
```

```
1105                        *
1106                        *     FIND DIFFERENCE IN EXPONENTS--SMALLER EXPONENT GETS RIGHT-SHIFTED.
1107                        *
1108 01457  010 046507  DIFR              PASS L    S4        L := EXP(D)
1109 01460  004 151007                    SUB  S1   S5        S1 := EXP(C) - EXP(D)
1110 01461  320 024102          JMP  CNDX ALZ       ADD2      IF 0,DO THE ADDITION (NO SHIFTS)
1111                        *
1112 01462  327 163302          JMP  CNDX AL15 RJS  RVRS
1113                        *
1114                        *     WE NEED TO SHIFT THE SECOND NUMBER.
1115                        *
1116 01463  017 141007                    CMPS S1   S1
1117 01464  007 141007                    INC  S1   S1        S1 := POSITIVE DIFFERENCE
1118 01465  320 063607          JMP                 SWAMPCHK  GO CHECK IF ONE OF THEM>THE OTHER
1119                        *
1120                        *     SWAP THE NUMBERS--WRONG ONE IS IN B,A
1121                        *
1122 01466  010 042207  RVRS              PASS B    S2        B := S2
1123 01467  010 053047                    PASS S2   S6
1124 01470  010 006507                    PASS L    A
1125 01471  010 044147                    PASS A    S3        A := S3
1126 01472  012 137107                    PASL S3
1127 01473  010 051147                    PASS S4   S5        S4 := LARGER EXPONENT
1128                        *
1129                        *     CHECK FOR  ABS(EXP2 - EXP1) > 30 --IF SO, ADDING WILL DO NOTHING
1130                        *
1131 01474  343 116507  SWAMPCHK IMM       LOW  L    %347
1132 01475  003 040547                    ADD  CNTR S1        -- TEST (S1 - 30(B8))
1133 01476  327 172702          JMP  CNDX AL15 RJS  TOOBIG    BUG OUT EARLY
1134 01477  010 036767  ALIGN       RPT
1135 01500  030 010224          ARS  R1   PASS B    B         ALIGN THE OPERAND FOR ADDING
1136 01501  326 163742          JMP  CNDX CNT8 RJS  ALIGN     IF NOT DONE LOOP
1137 01502  150 042522  ADD2    LWF  L1   PASS L    S2        SET UP TO ADD THE HI BITS
1138 01503  243 010207          ENV       ADD \B    B         ADD THE HIGH BITS
1139 01504  010 044507                    PASS L    S3        PREPARE FOR ADDING THE LO BITS
1140 01505  003 006147                    ADD  A    A         ADD THE LO BITS
1141 01506  321 064442          JMP  CNDX COUT RJS  NOCARY    TEST CARRY OUT FROM LO BITS
1142 01507  345 176507          IMM       HIGH L    %177
1143 01510  247 110207          ENV       INC  B    B         BUMP B IF CARRY OUT OF LO BITS
1144 01511  335 173002  NOCARY  JMP  CNDX OVFL RJS  PACK      IF NO OVERFLOW GO PACK IT UP
1145 01512  334 064702          JMP  CNDX FLAG RJS  OFLOW     IF SIGN POSITIVE HANDLE ODD CASE
1146 01513  345 176507          IMM       HIGH L    %177      SET UP L FOR OVF TEST
1147 01514  014 110747                    XOR       B
1148 01515  320 133002          JMP  CNDX ONES      PACK      IF UNIQUE CASE GO PACK IT UP
1149                        *
1150 01516  150 010224  OFLOW    LWF  R1   PASS B    B         FULL WORD SHIFT;USE FLAG FOR SIGN
1151 01517  150 006164          LWF  R1   PASS A    A
1152 01520  007 147147                    INC  S4   S4        BUMP THE EXPONENT
1153 01521  320 073007          JMP                 PACK      GO PACK IT UP
```

```
1155                    *            F M P -- FLOATING POINT MULTIPLY
1156                    *            -----    -------- ----- --------
1157                    *        ON ENTRY--A,B = C
1158                    *                  P = POINTER TO ADDRESS OF D
1159                    *
1160                    *        ON EXIT--A,B = RESULT
1161                    *
1162                    *        USES REGISTERS A,B,S1,S2,S3,S4,S5,S6
1163                    *
1164 01522  230 036747  FMP       READ
1165 01523  300 012447           JSB                INDIRECT  GO CLEAR INDIRECTS IF NECESSARY
1166 01524  300 071607           JSB                UNPACK    GO UNPACK THE NUMBERS
1167                    *
1168                    *        FORM EXP(C)+EXP(D)+1 IN S4; SAVE AS THE EXPONENT OF THE RESULT
1169                    *
1170 01525  007 150507                    INC  L     S5
1171 01526  003 047147                    ADD  S4    S4         S4 = EXP(C) + EXP(D) + 1
1172                    *
1173                    *       CALCULATE   MSB(D)*(LSB(C)/2)
1174                    *
1175 01527  010 006164              R1    PASS A     A          A = LSB(C)/2
1176 01530  010 042507                    PASS L     S2         L = MSB(D)
1177 01531  300 077047           JSB                MPYX       MSB(D)*(LSB(C)/2)
1178 01532  010 007207                    PASS S5    A          S5 = LSB(TEMP)
1179 01533  010 044164              R1    PASS A     S3         A = LSB(D)/2
1180 01534  010 011107                    PASS S3    B          S3 := MSB(TEMP)
1181                    *
1182                    *        CALCULATE MSB(C)*( LSB(D)/2 )
1183                    *
1184 01535  010 052507                    PASS L     S6         L = MSB(C)
1185 01536  300 077047           JSB                MPYX       MSB(C)*(LSB(D)/2)
1186                    *
1187                    *        ADD RESULTS TO TEMP1
1188                    *
1189 01537  010 006507                    PASS L     A          L = LSB(RESULT)
1190 01540  003 050747                    ADD        S5
1191 01541  321 066142           JMP  CNDX COUT RJS  *+2        TEST FOR CARRY OUT AND SKIP
1192                    *
1193 01542  007 110207                    INC  B     B          ADD IN THE CARRY BIT
1194 01543  010 010507                    PASS L     B          L = MSB(RESULT)
1195 01544  003 045107                    ADD  S3    S3         S3 = MSB(RESULT)
1196                    *
```

```
1198                        *         CALCULATE MSB(C)*MSB(D)
1199                        *
1200 01545  010 052507                      PASS L     S6          L = MSB(C)
1201 01546  010 042147                      PASS A     S2          A = MSB(D)
1202 01547  300 077047               JSB                MPYX        MSB(C)*MSB(D)
1203 01550  010 006164  FMPY7             R1   PASS A     A           A := LSB(RESULT)/2
1204 01551  010 044513                  COV  PASS L     S3
1205 01552  243 006162               ENV  L1   ADD  A     A           A := (LSB(RESULT)/2+TEMP1)*2
1206 01553  327 173002               JMP  CNDX AL15 RJS   PACK
1207 01554  335 126742               JMP  CNDX OVFL       FMPY8
1208 01555  000 010207                      DEC  B     B           BORROW FROM MSB'S
1209 01556  320 073007               JMP                PACK        GO PACK IT UP
1210                        *
1211 01557  007 110207  FMPY8             INC  B     B           CARRY TO MSB'S
1212 01560  320 073007               JMP                PACK        GO PACK IT UP
```

```
1214                    *             F D V -- FLOATING POINT DIVIDE
1215                    *             -----    -------- ----- ------
1216                    *
1217                    *        ON ENTRY-- A,B = C
1218                    *                   P = POINTER TO ADDRESS OF D
1219                    *
1220                    *        ON EXIT--  A,B = RESULT
1221                    *
1222                    *        USES REGISTERS A,B,S1,S2,S3,S4,S5,S6
1223                    *
1224 01561  230 036747  FDV      READ
1225 01562  300 012447           JSB                 INDIRECT  GO CLEAR INDIRECTS IF NECESSARY
1226 01563  300 071607           JSB                 UNPACK    GO UNPACK THE NUMBERS
1227                    *
1228                    *    GET SET TO FORM FIRST QUOTIENT OF THE APPROXIMATION (Q0).
1229                    *
1230 01564  017 143253                COV  CMPS S6   S2        S6 := NOT(MSB(D))
1231 01565  320 135542           JMP  CNDX ONES      OVERFLOW  CHECK FOR DIVIDE BY ZERO!
1232 01566  327 127402           JMP  CNDX AL15      *+2
1233 01567  007 153054                SOV  INC  S2   S6        S2 := ABS(MSB(D)); OVF := SIGN
1234 01570  000 046507                     DEC  L    S4        L := EXP(D) - 1
1235 01571  004 151147                     SUB  S4   S5        S4 := EXP(C)-EXP(D); CNTR := 1'S
1236 01572  030 010224           ARS  R1   PASS B    B         PRESHIFT TO AVOID OVERFLOW
1237 01573  300 075707           JSB                 DIVX
1238 01574  010 007207                     PASS S5   A         S5 := Q0
1239 01575  010 010747                     PASS      B
1240 01576  321 170002           JMP  CNDX AL0  RJS  *+2
1241 01577  000 010207                     DEC  B    B           FIRST LEFT SHIFT FOR NEXT
1242 01600  006 036147                     ZERO A
1243 01601  300 075707           JSB                 DIVX
1244 01602  010 007247                     PASS S6   A
1245 01603  010 044224 FDIV71         R1   PASS B    S3        B := LSB(D)/4
1246 01604  010 010224                R1   PASS B    B
1247 01605  006 036147                     ZERO A
1248 01606  300 075707           JSB                 DIVX
1249 01607  017 106147                     CMPS A    A
1250 01610  007 106147                     INC  A    A
1251                    *
1252 01611  010 050507                     PASS L    S5        L  := Q0; COUNTER := ALL ONES
1253 01612  300 077047           JSB                 MPYX
1254                    *
```

```
1256 01613  010 011047  FDIV81          PASS S2   B          S2 := MSB(-Q0*Q2)
1257 01614  006 036207                  ZERO B               B := 0
1258 01615  010 052747                  PASS      S6         IF Q1
1259 01616  327 171002          JMP  CNDX AL15 RJS  *+2        NEGATIVE,
1260 01617  011 136207                  ONE  B               B := ONES
1261 01620  010 042747                  PASS      S2         IF (-Q0*Q2)
1262 01621  327 171142          JMP  CNDX AL15 RJS  *+2        NEGATIVE,
1263 01622  000 010207                  DEC  B    B            B := B + (ALL ONES)
1264 01623  001 143062               L1 DBLS S2   S2         REORIENT PRODUCT (*4)
1265 01624  010 042507                  PASS L    S2
1266 01625  003 052147                  ADD  A    S6           ADD TO Q1
1267 01626  321 071402          JMP  CNDX COUT RJS  *+2      IF THERE WAS A CARRY OUT,
1268 01627  007 110207                  INC  B    B            ADD IT TO THE HIGH BITS.
1269                    *
1270 01630  070 010222          LGS  L1  PASS B    B
1271 01631  010 050507                  PASS L    S5
1272 01632  003 010207                  ADD  B    B          ADD Q0 TO MSB
1273 01633  320 073007          JMP                PACK       GO PACK IT UP
```

```
1275                        *
1276                        *     UNPACK THE NUMBERS:    B  := MSB(C)          S2 := MSB(D)
1277                        *                            A  := LSB(C)          S3 := LSB(D)
1278                        *                            S5 := EXP(C)          S4 := EXP(D)
1279                        *
1280 01634  010 001047     UNPACK              PASS S2    TAB        S2 := MSB(D)
1281                        *
1282 01635  007 133107                         INC  S3    M          S3 := ADDRESS OF LSB(D) + EXP(D)
1283 01636  230 044647               READ      PASS M     S3         READ THE WORD
1284 01637  344 000507               IMM       HIGH L     %0         L := 0 000 000 011 111 111
1285 01640  010 007247                         PASS S6    A          S6 := MSB(C)
1286 01641  010 001107                         PASS S3    TAB        S3 := LSB(D) + EXP(D)
1287 01642  012 045153                    COV  AND  S4    S3         S4 := EXP(D)
1288 01643  014 045107                         SANL S3    S3         S3 := LSB(D)
1289 01644  014 010147                         SANL A     B          A := LSB(C)
1290 01645  012 011224                    R1   AND  S5    B          S5 := UNPACKED EXP(C)
1291 01646  321 172442               JMP  CNDX ALO  RJS   *+3        TEST EXP SIGN AND SKIP IF +
1292 01647  342 000507               IMM       LOW  L     %200       L := %177600
1293 01650  003 051207                         ADD  S5    S5         S5 := S5 + %177600
1294 01651  010 052207                         PASS B     S6         B := MSB(C)
1295 01652  010 047164                    R1   PASS S4    S4         UNPACK EXP(D)
1296 01653  361 141142               RTN  CNDX ALO  RJS              TEST EXP SIGN AND EXIT IF +
1297 01654  342 000507               IMM       LOW  L     %200       L := %177600
1298 01655  003 047140                    RTN  ADD  S4    S4         S4 := S4 + %177600
1299                        *
```

```
1301                    *     PACK THE NUMBER
1302                    *
1303                    *          IT IS ASSUMED THAT THE MANTISSA IS UNNORMALIZED AND
1304                    *          CONTAINED IN THE ACCUMULATORS AND THE EXPONENT IN S4
1305                    *
1306                    *
1307 01656  010 042207  TOOBIG              PASS B     S2          ENTER HERE IF SWAMP CHECK IN FAD FAILED
1308 01657  010 044147                      PASS A     S3          LOAD THE ACC WITH THE LARGER NUM
1309 01660  010 006513  PACK           COV  PASS L     A
1310 01661  010 110747                      IOR        B           A/B = 0?
1311 01662  320 035642            JMP  CNDX A&Z       RETNFP2     -RETURN IF SO
1312 01663  343 176547            IMM       LOW  CNTR %377        INIT CNTR FOR 1'S COMP COUNTING
1313 01664  001 110507  NORMLIZ             DBLS L     B           L := LEFT SHIFT B BY ONE BIT
1314 01665  014 110747                      XOR        B           SET UP FOR NORMALIZED TEST
1315 01666  327 133502            JMP  CNDX AL15       ADJEXP      IF NORMALIZED THEN GO AJUST EXP
1316 01667  010 036767                 RPT
1317 01670  106 036762            NRM  L1   ZERO                   NORMALIZE A 32 BIT OPERAND
1318 01671  320 073207            JMP                 NORMLIZ     GO LOOP
1319 01672  007 126507  ADJEXP             INC  L     CNTR        L := -(NUMBER OF SHIFTS REQUIRED)
1320 01673  003 047147                     ADD  S4    S4          S4 := CORRECTED EXPONENT
1321 01674  351 176507  ROUND    IMM       CMLO L     %177        L := +200
1322 01675  010 010747                     PASS       B
1323 01676  327 174002            JMP  CNDX AL15 RJS   *+2         CHECK SIGN OF B--ADJUST ROUND-OFF
1324 01677  003 036507                     ADD  L                    TO  177  (DECREMENT LATCH)
1325 01700  003 006153                 COV  ADD  A     A           ADD 200 (OR 177 IF +) TO LSB'S
1326 01701  321 074602            JMP  CNDX COUT RJS   ADSBXPNT    -ANY CARRY OUT FROM LSB'S?
1327                    *   NOTE --  BIT 15 OF THE LATCH MUST (!!) BE ZERO AT THIS POINT.
1328 01702  247 110207            ENV       INC  B     B           ADD CARRY TO MSB'S,
1329 01703  335 174342            JMP  CNDX OVFL RJS   ADSBNOOV      CHECK FOR OVERFLOW
1330 01704  010 010224                 R1   PASS B     B           B := 0100...
1331 01705  007 147153                 COV  INC  S4    S4          EXP := EXP + 1
1332 01706  320 074607            JMP                 ADSBXPNT
1333 01707  001 110507  ADSBNOOV           DBLS L     B
1334 01710  014 110747                     XOR        B
1335 01711  327 134602            JMP  CNDX AL15       ADSBXPNT    CHECK FOR B=11...
1336 01712  070 010222            LGS  L1   PASS B     B           RE-NORMALIZE
1337 01713  000 047147                     DEC  S4    S4
```

```
1339 01714   342 000514  ADSBXPNT IMM   SOV   LOW   L     %200       GET OVF SET FOR ERROR; L := -200
1340 01715   004 146747                      SUB                    TEST (EXP + 200)
1341 01716   327 135402             JMP     CNDX  AL15        UNDERFLO   -IF NEGATIVE, UNDERFLOW
1342 01717   003 046747                      ADD                    TEST (EXP - 200)
1343 01720   327 175542             JMP     CNDX  AL15  RJS   OVERFLOW   -IF POSITIVE, OVERFLOW
1344 01721   150 046762             LWF     L1    PASS        S4         FLAG := EXPONENT SIGN
1345 01722   154 047162             LWF     L1    SANL  S4    S4
1346 01723   340 000507             IMM             LOW   L     0          L := %177400
1347 01724   012 006507                      AND   L     A                  L := LSB'S
1348 01725   227 174707             READ            INC   PNM   P          START NEXT INSTRUCTION FETCH
1349 01726   010 010153                   COV   PASS  A     B            A := MSB'S
1350 01727   010 146200                   RTN   IOR   B     S4           B := LSB'S OR EXPONENT
1351                     *
1352 01730   006 036207  UNDERFLO                ZERO  B                  UNDERFLOW; A,B:=0; OVF := 1
1353 01731   006 036147  RZERO                   ZERO  A
1354 01732   227 174700             READ    RTN   INC   PNM   P          START READ AND EXIT
1355                     *
1356 01733   343 174214  OVERFLOW IMM   SOV   LOW   B     %376       OVERFLOW; A,B := MOST + NUMBER
1357 01734   011 136164  OVER32K        R1    ONE   A
1358 01735   227 174700  RETNFP2  READ  RTN   INC   PNM   P          START READ; EXIT
```

```
1360                        *      MULTIPLY AND DIVIDE UTILITIES FOR FLOATING POINT USE ONLY
1361                        *
1362                        *
1363                        DIVX       EQU                    *
1364 01736  150 010762                LWF  L1   PASS         B          B < 0?  FLAG := SIGN
1365 01737  327 176242                JMP  CNDX AL15 RJS     READY
1366 01740  017 110207                     CMPS F    B       B          DOUBLE-WORD NEGATE
1367 01741  017 106147                     CMPS A    A       A
1368 01742  007 106147                     INC  A    A       A
1369 01743  321 076242                JMP  CNDX COUT RJS     READY
1370 01744  007 110207                     INC  B    B       B          ADD IN THE CARRY
1371 01745  010 042527  READY              RPT  PASS L       S2         GET THE DIVISOR
1372 01746  124 110222                DIV  L1   SUB  S    B       B          DO THE DIVIDE STEP 16 TIMES.
1373                        *
1374 01747  010 010224                     R1   PASS B    B       B          -FORM POSITIVE REMAINDER
1375 01750  334 076542                JMP  CNDX FLAG RJS     *+3
1376 01751  017 110207                     CMPS F    B       B
1377 01752  007 110207                     INC  B    B       B                        ADD ONE
1378 01753  335 136742                JMP  CNDX OVFL         DIVXFTST
1379 01754  374 076742                RTN  CNDX FLAG RJS
1380 01755  017 106147  COMPLEMT           CMPS A    A       A
1381 01756  007 106140                RTN  INC  A    A       A
1382 01757  334 076642  DIVXFTST     JMP  CNDX FLAG RJS     COMPLEMT
1383 01760  370 036747                RTN
1384                        *
1385 01761  010 007007  MPYX                PASS S1   A
1386 01762  006 036227                RPT  ZERO B
1387 01763  163 010224                MPY  R1   ADD  B    B
1388 01764  010 040747                     PASS            S1
1389 01765  307 137402                JSB  CNDX AL15         SUBB
1390 01766  362 177402                RTN  CNDX L15  RJS
1391 01767  010 040507                     PASS L    S1
1392 01770  364 110207  SUBB           RTN       SUB  S    B
1393                        *
1394                                   ORG                    %1777
1395 01777  320 073007                JMP                    PACK       EXTERNAL ENTRY FOR PACK
```

```
1397                                ORG                 2000B
1398                          *
1399                          *     ROM JUMP TABLE
1400                          *     --------------
1401                          *
1402 02000  000 000073              DEF                 SRG         0
1403 02001  000 000073              DEF                 SRG         1
1404 02002  000 000073              DEF                 SRG         2
1405 02003  000 000073              DEF                 SRG         3
1406 02004  000 000053              DEF                 ASGNO*      4
1407 02005  000 000063              DEF                 ASGCL*      5
1408 02006  000 000067              DEF                 ASGCM*      6
1409 02007  000 000057              DEF                 ASGCC*      7
1410 02010  000 000073              DEF                 SRG        10
1411 02011  000 000073              DEF                 SRG        11
1412 02012  000 000073              DEF                 SRG        12
1413 02013  000 000073              DEF                 SRG        13
1414 02014  000 000053              DEF                 ASGNO*     14
1415 02015  000 000063              DEF                 ASGCL*     15
1416 02016  000 000067              DEF                 ASGCM*     16
1417 02017  000 000057              DEF                 ASGCC*     17
1418 02020  000 000015              DEF                 AND        20
1419 02021  000 000015              DEF                 AND        21
1420 02022  000 000015              DEF                 AND        22
1421 02023  000 000015              DEF                 AND        23
1422 02024  000 000015              DEF                 AND        24
1423 02025  000 000015              DEF                 AND        25
1424 02026  000 000015              DEF                 AND        26
1425 02027  000 000015              DEF                 AND        27
1426 02030  000 000043              DEF                 JSB        30
1427 02031  000 000043              DEF                 JSB        31
1428 02032  000 000043              DEF                 JSB        32
1429 02033  000 000043              DEF                 JSB        33
1430 02034  000 000043              DEF                 JSB        34
1431 02035  000 000043              DEF                 JSB        35
1432 02036  000 000043              DEF                 JSB        36
1433 02037  000 000043              DEF                 JSB        37
```

```
1435 02040  000 000051          DEF             XOR     40
1436 02041  000 000051          DEF             XOR     41
1437 02042  000 000051          DEF             XOR     42
1438 02043  000 000051          DEF             XOR     43
1439 02044  000 000051          DEF             XOR     44
1440 02045  000 000051          DEF             XOR     45
1441 02046  000 000051          DEF             XOR     46
1442 02047  000 000051          DEF             XOR     47
1443 02050  000 000040          DEF             JMP     50
1444 02051  000 000040          DEF             JMP     51
1445 02052  000 000040          DEF             JMP     52
1446 02053  000 000040          DEF             JMP     53
1447 02054  000 000040          DEF             JMP     54
1448 02055  000 000040          DEF             JMP     55
1449 02056  000 000040          DEF             JMP     56
1450 02057  000 000040          DEF             JMP     57
1451 02060  000 000026          DEF             IOR     60
1452 02061  000 000026          DEF             IOR     61
1453 02062  000 000026          DEF             IOR     62
1454 02063  000 000026          DEF             IOR     63
1455 02064  000 000026          DEF             IOR     64
1456 02065  000 000026          DEF             IOR     65
1457 02066  000 000026          DEF             IOR     66
1458 02067  000 000026          DEF             IOR     67
1459 02070  000 000030          DEF             ISZ     70
1460 02071  000 000030          DEF             ISZ     71
1461 02072  000 000030          DEF             ISZ     72
1462 02073  000 000030          DEF             ISZ     73
1463 02074  000 000030          DEF             ISZ     74
1464 02075  000 000030          DEF             ISZ     75
1465 02076  000 000030          DEF             ISZ     76
1466 02077  000 000030          DEF             ISZ     77
1467 02100  000 000017          DEF             AD*     100
1468 02101  000 000017          DEF             AD*     101
1469 02102  000 000017          DEF             AD*     102
1470 02103  000 000017          DEF             AD*     103
1471 02104  000 000017          DEF             AD*     104
1472 02105  000 000017          DEF             AD*     105
1473 02106  000 000017          DEF             AD*     106
1474 02107  000 000017          DEF             AD*     107
```

```
1476 02110  000 000017          DEF             AD*     110
1477 02111  000 000017          DEF             AD*     111
1478 02112  000 000017          DEF             AD*     112
1479 02113  000 000017          DEF             AD*     113
1480 02114  000 000017          DEF             AD*     114
1481 02115  000 000017          DEF             AD*     115
1482 02116  000 000017          DEF             AD*     116
1483 02117  000 000017          DEF             AD*     117
1484 02120  000 000021          DEF             CP*     120
1485 02121  000 000021          DEF             CP*     121
1486 02122  000 000021          DEF             CP*     122
1487 02123  000 000021          DEF             CP*     123
1488 02124  000 000021          DEF             CP*     124
1489 02125  000 000021          DEF             CP*     125
1490 02126  000 000021          DEF             CP*     126
1491 02127  000 000021          DEF             CP*     127
1492 02130  000 000021          DEF             CP*     130
1493 02131  000 000021          DEF             CP*     131
1494 02132  000 000021          DEF             CP*     132
1495 02133  000 000021          DEF             CP*     133
1496 02134  000 000021          DEF             CP*     134
1497 02135  000 000021          DEF             CP*     135
1498 02136  000 000021          DEF             CP*     136
1499 02137  000 000021          DEF             CP*     137
1500 02140  000 000047          DEF             LD*     140
1501 02141  000 000047          DEF             LD*     141
1502 02142  000 000047          DEF             LD*     142
1503 02143  000 000047          DEF             LD*     143
1504 02144  000 000047          DEF             LD*     144
1505 02145  000 000047          DEF             LD*     145
1506 02146  000 000047          DEF             LD*     146
1507 02147  000 000047          DEF             LD*     147
1508 02150  000 000047          DEF             LD*     150
1509 02151  000 000047          DEF             LD*     151
1510 02152  000 000047          DEF             LD*     152
1511 02153  000 000047          DEF             LD*     153
1512 02154  000 000047          DEF             LD*     154
1513 02155  000 000047          DEF             LD*     155
1514 02156  000 000047          DEF             LD*     156
1515 02157  000 000047          DEF             LD*     157
```

```
1517 02160  000 000135          DEF             ST*         160
1518 02161  000 000135          DEF             ST*         161
1519 02162  000 000135          DEF             ST*         162
1520 02163  000 000135          DEF             ST*         163
1521 02164  000 000135          DEF             ST*         164
1522 02165  000 000135          DEF             ST*         165
1523 02166  000 000135          DEF             ST*         166
1524 02167  000 000135          DEF             ST*         167
1525 02170  000 000135          DEF             ST*         170
1526 02171  000 000135          DEF             ST*         171
1527 02172  000 000135          DEF             ST* @       172
1528 02173  000 000135          DEF             ST*         173
1529 02174  000 000135          DEF             ST*         174
1530 02175  000 000135          DEF             ST*         175
1531 02176  000 000135          DEF             ST*         176
1532 02177  000 000135          DEF             ST*         177
1533 02200  000 000113          DEF             JTBL1000    200 ASL,LSL,RRL,MPY
1534 02201  000 000153          DEF             DIV         201
1535 02202  000 000227          DEF             JTBL1010    202 ASR,LSR,RRR
1536 02203  000 000107          DEF             MAC1        203
1537 02204  000 000077          DEF             IOG         204 HLT,STF,SFC,SFS
1538 02205  000 000077          DEF             IOG         205 MIA,LIA,OTA,STC
1539 02206  000 000077          DEF             IOG         206 HLT,CLF
1540 02207  000 000077          DEF             IOG         207 MIA,LIA,OTA,STC
1541 02210  000 000120          DEF             DLD         210
1542 02211  000 000130          DEF             DST         211
1543 02212  000 000103          DEF             MAC0        212
1544 02213  000 000107          DEF             MAC1        213
1545 02214  000 000077          DEF             IOG         214 HLT,STF,SFC,SFS
1546 02215  000 000077          DEF             IOG         215 MIB,LIB,OTB,CLC
1547 02216  000 000077          DEF             IOG         216 HLT,CLF
1548 02217  000 000077          DEF             IOG         217 MIB,LIB,OTB,CLC
1549 02220  000 000002          DEF             MRGIND      220
1550 02221  000 000002          DEF             MRGIND      221
1551 02222  000 000002          DEF             MRGIND      222
1552 02223  000 000002          DEF             MRGIND      223
1553 02224  000 000002          DEF             MRGIND      224
1554 02225  000 000002          DEF             MRGIND      225
1555 02226  000 000002          DEF             MRGIND      226
1556 02227  000 000002          DEF             MRGIND      227
```

```
1558 02230  000 000041          DEF             JSB,I     230
1559 02231  000 000041          DEF             JSB,I     231
1560 02232  000 000041          DEF             JSB,I     232
1561 02233  000 000041          DEF             JSB,I     233
1562 02234  000 000041          DEF             JSB,I     234
1563 02235  000 000041          DEF             JSB,I     235
1564 02236  000 000041          DEF             JSB,I     236
1565 02237  000 000041          DEF             JSB,I     237
1566 02240  000 000002          DEF             MRGIND    240
1567 02241  000 000002          DEF             MRGIND    241
1568 02242  000 000002          DEF             MRGIND    242
1569 02243  000 000002          DEF             MRGIND    243
1570 02244  000 000002          DEF             MRGIND    244
1571 02245  000 000002          DEF             MRGIND    245
1572 02246  000 000002          DEF             MRGIND    246
1573 02247  000 000002          DEF             MRGIND    247
1574 02250  000 000036          DEF             JMP,I     250
1575 02251  000 000036          DEF             JMP,I     251
1576 02252  000 000036          DEF             JMP,I     252
1577 02253  000 000036          DEF             JMP,I     253
1578 02254  000 000036          DEF             JMP,I     254
1579 02255  000 000036          DEF             JMP,I     255
1580 02256  000 000036          DEF             JMP,I     256
1581 02257  000 000036          DEF             JMP,I     257
1582 02260  000 000002          DEF             MRGIND
1583 02261  000 000002          DEF             MRGIND
1584 02262  000 000002          DEF             MRGIND
1585 02263  000 000002          DEF             MRGIND
1586 02264  000 000002          DEF             MRGIND
1587 02265  000 000002          DEF             MRGIND
1588 02266  000 000002          DEF             MRGIND
1589 02267  000 000002          DEF             MRGIND
1590 02270  000 000002          DEF             MRGIND
1591 02271  000 000002          DEF             MRGIND
1592 02272  000 000002          DEF             MRGIND
1593 02273  000 000002          DEF             MRGIND
1594 02274  000 000002          DEF             MRGIND
1595 02275  000 000002          DEF             MRGIND
1596 02276  000 000002          DEF             MRGIND
1597 02277  000 000002          DEF             MRGIND
```

```
1599 02300  000 000002          DEF             MRGIND
1600 02301  000 000002          DEF             MRGIND
1601 02302  000 000002          DEF             MRGIND
1602 02303  000 000002          DEF             MRGIND
1603 02304  000 000002          DEF             MRGIND
1604 02305  000 000002          DEF             MRGIND
1605 02306  000 000002          DEF             MRGIND
1606 02307  000 000002          DEF             MRGIND
1607 02310  000 000002          DEF             MRGIND
1608 02311  000 000002          DEF             MRGIND
1609 02312  000 000002          DEF             MRGIND
1610 02313  000 000002          DEF             MRGIND
1611 02314  000 000002          DEF             MRGIND
1612 02315  000 000002          DEF             MRGIND
1613 02316  000 000002          DEF             MRGIND
1614 02317  000 000002          DEF             MRGIND
1615 02320  000 000002          DEF             MRGIND
1616 02321  000 000002          DEF             MRGIND
1617 02322  000 000002          DEF             MRGIND
1618 02323  000 000002          DEF             MRGIND
1619 02324  000 000002          DEF             MRGIND
1620 02325  000 000002          DEF             MRGIND
1621 02326  000 000002          DEF             MRGIND
1622 02327  000 000002          DEF             MRGIND
1623 02330  000 000002          DEF             MRGIND
1624 02331  000 000002          DEF             MRGIND
1625 02332  000 000002          DEF             MRGIND
1626 02333  000 000002          DEF             MRGIND
1627 02334  000 000002          DEF             MRGIND
1628 02335  000 000002          DEF             MRGIND
1629 02336  000 000002          DEF             MRGIND
1630 02337  000 000002          DEF             MRGIND
1631 02340  000 000002          DEF             MRGIND
1632 02341  000 000002          DEF             MRGIND
1633 02342  000 000002          DEF             MRGIND
1634 02343  000 000002          DEF             MRGIND
1635 02344  000 000002          DEF             MRGIND
1636 02345  000 000002          DEF             MRGIND
1637 02346  000 000002          DEF             MRGIND
1638 02347  000 000002          DEF             MRGIND
```

```
1640 02350  000 000002          DEF             MRGIND
1641 02351  000 000002          DEF             MRGIND
1642 02352  000 000002          DEF             MRGIND
1643 02353  000 000002          DEF             MRGIND
1644 02354  000 000002          DEF             MRGIND
1645 02355  000 000002          DEF             MRGIND
1646 02356  000 000002          DEF             MRGIND
1647 02357  000 000002          DEF             MRGIND
1648 02360  000 000002          DEF             MRGIND
1649 02361  000 000002          DEF             MRGIND
1650 02362  000 000002          DEF             MRGIND
1651 02363  000 000002          DEF             MRGIND
1652 02364  000 000002          DEF             MRGIND
1653 02365  000 000002          DEF             MRGIND
1654 02366  000 000002          DEF             MRGIND
1655 02367  000 000002          DEF             MRGIND
1656 02370  000 000002          DEF             MRGIND
1657 02371  000 000002          DEF             MRGIND
1658 02372  000 000002          DEF             MRGIND
1659 02373  000 000002          DEF             MRGIND
1660 02374  000 000002          DEF             MRGIND
1661 02375  000 000002          DEF             MRGIND
1662 02376  000 000002          DEF             MRGIND
1663 02377  000 000002          DEF             MRGIND
1664                            END
```

END OF PASS 2: NO ERRORS

```
PAGE 0002 MICRO XREF  REV.2013 801031

SYMBOLS=0207  REFERENCES=0517  SOURCE LINES=1664

AD*        0089  1467  1468  1469  1470  1471  1472  1473  1474  1476
           1477  1478  1479  1480  1481  1482  1483

ADD2       1137  1110

ADDR1S1    0194  0459

ADDS1      0250  0460  0462

ADJEXP     1319  1315

ADSBNOOV   1333  1329

ADSBXPNT   1339  1326  1332  1335

ADX        0777  0723

ADY        0817  0731

ALIGN      1134  1136

AND        0086  1418  1419  1420  1421  1422  1423  1424  1425

ASGCC*     0134  1409  1417

ASGCL*     0139  1407  1415

ASGCM*     0144  1408  1416

ASGNO*     0129  1406  1414

ASL        0247  0285

ASR        0252  0309  0600

BITS       0995  0744  0745  0746

BKGNDCK    0661  0664

CBS        1011  0999

CBT        0903  0739

CLFS2      0203  0430

CMW        0853  0747

COMPLEMT   1380  1382

CONTROL    0178  **NOT REFERENCED**

CP*        0092  1484  1485  1486  1487  1488  1489  1490  1491  1492
           1493  1494  1495  1496  1497  1498  1499

CPTEST     0580  0562  0604
```

```
DECDMS     0291  0493
DECM       0493  0354
DECP       0290  0323
DIAG       0314  0284
DIFR       1108  1092  1096  1098  1100
DIV        0217  1534
DIVS       0226  0221
DIVX       1363  1237  1243  1248
DIVXFTST   1382  1378
DLD        0187  1541
DMSLOAD    0607  0636
DSPICODE   0468  0339  0498
DST        0196  1542
DSX        0790  0734
DSY        0830  0742
EIG        0717  0693  0694
EM1000     0284  0176
EM1010     0306  0268
EXITSUSP   0401  0485
FAD        1080  0696
FAILURE    0666  0575  0595  0597  0599  0601  0652
FDIV71     1245  **NOT REFERENCED**
FDIV81     1256  **NOT REFERENCED**
FDV        1224  0699
FETCH      0066  0080  0367  0369  1009
FIX        1042  0700
FIXOK1     1046  1044
FIXOK2     1058  1054
FLOAT      1027  **NOT REFERENCED**
FLT        1028  0701
```

```
FMP         1164  0698
FMPY7       1203  **NOT REFERENCED**
FMPY8       1211  1207
FPDIAG      0289  **NOT REFERENCED**
FSB         1079  0697
FTBS        1004  1013
GOFETCH     1008  0979
HALT        0322  0075
HALTIO      0395  0181
HALTSUSP    0323  0399
HORI        0074  0302  0308  0395  0397  0934  0989
IDLE        0345  **NOT REFERENCED**
INCDMS      0311  0495
INCM        0495  0353
INDIRECT    0297  0109  0113  0188  0197  0206  0218  0271  0300  0301
            0754  0761  0769  0773  0777  0795  0802  0809  0813  0817
            0838  0976  0995  0998  1082  1165  1225
INITIAL     0975  0853  0877  0893  0903
INSTP       0362  0356
INTPEND     0986  0867  0887  0901  0917
IOG         0161  1537  1538  1539  1540  1545  1546  1547  1548
IOR         0098  1451  1452  1453  1454  1455  1456  1457  1458
ISX         0786  0733
ISY         0826  0741
ISZ         0101  1459  1460  1461  1462  1463  1464  1465  1466
JLY         0838  0735
JMP         0110  1443  1444  1445  1446  1447  1448  1449  1450
JMP,I       0108  1574  1575  1576  1577  1578  1579  1580  1581
JPY         0843  0743  0841
JSB         0114  1426  1427  1428  1429  1430  1431  1432  1433
JSB,I       0112  1558  1559  1560  1561  1562  1563  1564  1565
```

```
JSBSCAN   0347  0402

JTBL1000  0176  1533

JTBL1010  0268  1535

L*X       0761  0719

L*Y       0802  0727

LBT       0937  0736

LCBT      0904  0916

LCMW      0854  0866

LD*       0119  1500  1501  1502  1503  1504  1505  1506  1507  1508  -
          1509  1510  1511  1512  1513  1514  1515

LDBYTE    0968  0895  0905  0908  0925  0939

LDX       0773  0722

LDY       0813  0730

LEFT      0372  0351

LI*       0168  **NOT REFERENCED**

LMBT      0894  0900

LMVW      0878  0886

LOADER    0506  0355  0490  0615

LOOP      0528  0553

LSFB      0924  0932

LSL       0256  0286

LSR       0259  0310

MAC0      0166  1543

MAC1      0171  1536  1544

MACTABL0  0696  0166

MACTABL1  0679  0171

MBT       0893  0738

MEMLOST   0603  0315  0325

MEMSIZE   0508  0516

MI*       0163  0161

MODE      0276  0357  0362
```

```
MPY         0205  0292

MPYX        1385  1177  1185  1202  1253

MRGIND      0069  0071  0072  1549  1550  1551  1552  1553  1554  1555
            1556  1566  1567  1568  1569  1570  1571  1572  1573  1582
            1583  1584  1585  1586  1587  1588  1589  1590  1591  1592
            1593  1594  1595  1596  1597  1599  1600  1601  1602  1603
            1604  1605  1606  1607  1608  1609  1610  1611  1612  1613
            1614  1615  1616  1617  1618  1619  1620  1621  1622  1623
            1624  1625  1626  1627  1628  1629  1630  1631  1632  1633
            1634  1635  1636  1637  1638  1640  1641  1642  1643  1644
            1645  1646  1647  1648  1649  1650  1651  1652  1653  1654
            1655  1656  1657  1658  1659  1660  1661  1662  1663

MVW         0877  0748

NOCARY      1144  1141

NORMLIZ     1313  1318

NORPL       0333  0327  0328

NOTEQ       0868  0859  0862  0912

OFLOW       1150  1145

OT*         0173  **NOT REFERENCED**

OVER32K     1357  **NOT REFERENCED**

OVERFLOW    1356  1231  1343

PACK        1309  1032  1144  1148  1153  1206  1209  1212  1273  1395

PAUSE       0341  0359  0435  0670

READY       1371  1365  1369

REGTEST     0586  0583

RET         0154  0179  0230

RETNFP      1056  0289  1064  1066

RETNFP2     1358  1311

RETURN      0766  0759  0771  0800  0811  0865  0885  0899  0915  0931

RIGHT       0383  0352

RIPLOOP     0647  0656  0658

RIPP1MW     0606  **NOT REFERENCED**

RIPP32K     0641  0621  0624  0627  0629

RMDR        0243  0225  0236  0241

ROUND       1321  **NOT REFERENCED**
```

```
RPL         0489  0331  0332

RRL         0262  0288

RRR         0265  0312

RTRNINTG    1061  1053

RUN         0361  0329  0491

RVRS        1122  1112

RZERO       1353  **NOT REFERENCED**

S*X         0754  0717

S*Y         0795  0725

SBS         1014  **NOT REFERENCED**

SBT         0944  0737  0928

SCAN        0350  0347

SELCODE     0519  **NOT REFERENCED**

SFB         0919  0740

SRG         0152  1402  1403  1404  1405  1410  1411  1412  1413

ST*         0201  1517  1518  1519  1520  1521  1522  1523  1524  1525
            1526  1527  1528  1529  1530  1531  1532

STBYTE      0954  0896  0947

STCPUS      0422  0412

STFENCE     0418  0413

STORE       0498  0356

STOREA      0411  **NOT REFERENCED**

STOREB      0410  **NOT REFERENCED**

STOREF      0413  **NOT REFERENCED**

STOREM      0409  **NOT REFERENCED**

STOREMM     0414  **NOT REFERENCED**

STOREMN     0415  **NOT REFERENCED**

STOREP      0407  **NOT REFERENCED**

STORES      0406  0499

STOREST     0412  **NOT REFERENCED**

STORET      0433  0408
```

```
STOREX    0417  **NOT REFERENCED**
STOREY    0416  **NOT REFERENCED**
STWORD    0551  0541  0545  0548
STX       0769  0720
STY       0809  0728
SUBB      1392  1389
SUSPEND   0485  0344  0487
SUSPINIT  0398  **NOT REFERENCED**
SWAMPCHK  1131  1118
TBS       1012  **NOT REFERENCED**
TEST32K   0568  0578
TESTDMS   0630  0618
TIMER     0271  0267  0314
TOOBIG    1307  1133
UNDERFLO  1352  1341
UNPACK    1280  1083  1166  1226
UPDATEA   0445  **NOT REFERENCED**
UPDATEB   0444  **NOT REFERENCED**
UPDATEF   0447  **NOT REFERENCED**
UPDATEM   0443  **NOT REFERENCED**
UPDATEMM  0448  **NOT REFERENCED**
UPDATEMN  0449  **NOT REFERENCED**
UPDATEP   0441  **NOT REFERENCED**
UPDATES   0440  0340
UPDATEST  0446  **NOT REFERENCED**
UPDATET   0442  **NOT REFERENCED**
UPDATEX   0451  **NOT REFERENCED**
UPDATEY   0450  **NOT REFERENCED**
UPDCPUS   0454  0446
UPDFENCE  0452  0447

USER      0437  0333  0671
WAIT      0339  0336  0348  0488  0500
X*X       0782  0724
X*Y       0822  0732
XOR       0122  1435  1436  1437  1438  1439  1440  1441  1442
```

JUMP TABLE FOR F-SERIES BASE SET

APPENDIX E

PAGE 0026 RTE MICRO-ASSEMBLER REV.2040 800521

```
0758                                ORG                 760B
0759                        *
0760                        *       PRIMARY MAPPING TABLE
0761                        *       --------------------
0762                        *
0763 00760  320 100004  MACTABL1 JMP  RJ30          2000B     10X400B, HP RESERVED
0764 00761  322 100004           JMP  RJ30          12000B    10X420B, HP RESERVED
0765 00762  325 140004           JMP  RJ30          27000B    10X440B, USER RESERVED
0766 00763  321 100004           JMP  RJ30          6000B     10X460B, VIS
0767 00764  325 160004           JMP  RJ30          27400B    10X500B, USER RESERVED
0768 00765  326 000004           JMP  RJ30          30000B    10X520B, USER RESERVED
0769 00766  326 020004           JMP  RJ30          30400B    10X540B, USER RESERVED
0770 00767  326 040004           JMP  RJ30          31000B    10X560B, USER RESERVED
0771 00770  327 000004           JMP  RJ30          34000B    10X600B, USER RESERVED
0772 00771  327 020004           JMP  RJ30          34400B    10X620B, USER RESERVED
0773 00772  327 040004           JMP  RJ30          35000B    10X640B, USER RESERVED
0774 00773  327 060004           JMP  RJ30          35400B    10X660B, USER RESERVED
0775 00774  324 000004           JMP  RJ30          20000B    10X700B, DMS
0776 00775  324 001004           JMP  RJ30          20020B    10X720B, DMS
0777 00776  320 041004           JMP  RJ30          EIG       10X740B, EIG
0778 00777  320 042004           JMP  RJ30          EIG+20B   10X760B, EIG
0779                        *
0780                        *
0781                        *
0782                        *
0783                        *
0784                        *
```

PAGE 0028 RTE MICRO-ASSEMBLER REV.2040 800521

```
0807                        *
0808                                ORG                 1000B     BEGINNING OF MODULE 2
0809                        *
0810 01000  320 060004  MACTABL0 JMP  RJ30          ASMD2345  105000B, FLOATING POINT ADD
0811 01001  320 060004           JMP  RJ30          ASMD2345  105020B, FLOATING POINT SUBTRACT
0812 01002  320 060004           JMP  RJ30          ASMD2345  105040B, FLOATING POINT MULTIPLY
0813 01003  320 060004           JMP  RJ30          ASMD2345  105060B, FLOATING POINT DIVIDE
0814 01004  343 130507           IMM       LOW  L   354B      105100B, L= COMPL OF 24B
0815 01005  320 061004           JMP  RJ30          XTSD2345  105120B, FLOATING POINT TO INTEGER
0816                        *
0817 01006  327 100004           JMP  RJ30          36000B    105140B, USER RESERVED
0818 01007  327 140004           JMP  RJ30          37000B    105160B, USER RESERVED
0819 01010  324 040004           JMP  RJ30          21000B    105200B, FFP
0820 01011  324 060004           JMP  RJ30          21400B    105220B, FFP
0821 01012  324 100004           JMP  RJ30          22000B    105240B, EMA
0822 01013  321 000004           JMP  RJ30          4000B     105260B, HP RESERVED
0823 01014  324 140004           JMP  RJ30          23000B    105300B, DS1000
0824 01015  325 000004           JMP  RJ30          24000B    105320B, SIS
0825 01016  322 000004           JMP  RJ30          10000B    105340B, HP RESERVED
0826 01017  322 040004           JMP  RJ30          11000B    105360B, HP RESERVED
0827                        *NOLIST
```

```
0001                    MICMXE,L,C,T
0002                    ************************************************************
0003                    * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1979.  ALL RIGHTS     *
0004                    * RESERVED.  NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,      *
0005                    * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT*
0006                    * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.       *
0007                    ************************************************************
0008                    *
0009                    *
0010                    *  ENHANCED SCIENTIFIC INSTRUCTION SET MICROCODE
0011                    *  FOR HP1000 F-SERIES COMPUTERS
0012                    *
0013                    *  JULY 31, 1979 CC,BG,CRG
0014                    *
0015                    *  NOTE:  ESISREVN SHOULD BE DECREMENTED ON EVERY PROM CHANGE
0016                    *            I.E. INCREMENT X REG. RTN (X=COMPL OF ESISREVN- SEE SELFTEST)
0017                    *
0018                    ************************************************************
0019                    *
0020                    $NOLIST
0025                    $LIST
0026                    FETCH    EQU                00000B
0027                    HORI     EQU                00006B
0028                    ESISREVN EQU                00373B     1'S COMPLEMENT OF 4=REVN
0029                             ORG                24000B
0030 24000  325 012047          JMP                TAN
0031 24001  325 040007          JMP                SQRT
0032 24002  325 023047          JMP                ALOG
0033 24003  325 027647          JMP                ATAN
0034 24004  325 015647          JMP                COS
0035 24005  325 015747          JMP                SIN
0036 24006  325 043447          JMP                EXP
0037 24007  325 027207          JMP                ALOGT
0038 24010  325 047147          JMP                TANH
0039 24011  325 052747          JMP                DPOLY
0040 24012  325 056547          JMP                /CMRT
0041 24013  325 065707          JMP                /ATLG
0042 24014  325 070207          JMP                .FPWR
0043 24015  325 070047          JMP                .TPWR
0044 24016  230 036740          READ RTN
0045 24017  325 035347          JMP                SELFTEST
```

```
0047                    ************************************************************
0048                    *
0049                    *  SUBROUTINE FLUN
0050                    *
0051                    *  ENTER:  B = LOW PART OF FLOATING POINT NUMBER
0052                    *  RETURN: A = EXPONENT  B = LOW MANTISSA
0053                    *
0054 24020  340 000516  FLUN     IMM  CLFL LOW  L    000B       L = 177400B
0055 24021  154 010164          LWF  R1   SANL A    B          GET EXPONENT IN A
0056 24022  012 010207               AND  B    B    B          PUT MANTISSA IN B
0057 24023  374 040202          RTN  CNDX FLAG RJS              RETURN IF EXP POSITIVE
0058 24024  342 000507          IMM       LOW  L    200B       L = 177600B
0059 24025  370 106147          RTN       IOR  A    A          RETURN, EXTEND SIGN BIT
0060                    *
0061                    ************************************************************
0062                    *
0063                    *  SUBROUTINE PWR2
0064                    *
0065                    *  ENTER:  FLOATING POINT NUMBER IN BOX
0066                    *          INTEGER IN S11
0067                    *  RETURN: (A B) = X*2**S11, P = P + 1
0068                    *
0069 24026  306 043242  PWR2     JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0070 24027  010 020172               MPP1 PASS A    MPPB       GET A FROM BOX
0071 24030  320 002642          JMP  CNDX ALZ       PDONE      EXIT IF X = 0
0072 24031  010 020232               MPP1 PASS B    MPPB       GET B FROM BOX
0073 24032  340 000516          IMM  CLFL LOW  L    000B       L  = 177400B
0074 24033  154 011024          LWF  R1   SANL S1   B          S1 = EXPONENT
0075 24034  334 041742          JMP  CNDX FLAG RJS  PNEXT1     JUMP IF EXP POSTIVE
0076 24035  342 000507          IMM       LOW  L    200B       L  = 177600B
0077 24036  010 141007               IOR  S1   S1   S1         EXTEND EXP SIGN BIT
0078 24037  010 040507  PNEXT1         PASS L    S1   S1         PUT EXPONENT IN L
0079 24040  143 064762          LWF  L1   ADD       S11        SET NEW EXP SIGN BIT
0080 24041  143 065022          LWF  L1   ADD  S1   S11        S1 = NEW EXP
0081 24042  340 000507          IMM       LOW  L    000B       L  = 177400B
0082                    *
0083                    *  CHECK FOR UNDER/OVERFLOW
0084                    *
0085 24043  012 040753               COV  AND       S1
0086 24044  320 002402          JMP  CNDX ALZ       PNEXT2     JUMP IF NO OVERFLOW
0087 24045  011 040747                    SONL      S1
0088 24046  320 102402          JMP  CNDX ONES      PNEXT2     JUMP IF NO UNDERFLOW
0089 24047  370 036754          RTN  SOV                       OVER/UNDERFLOW RETURN
0090                    *
0091 24050  014 041007  PNEXT2          SANL S1   S1   S1         S1 = 000EXP
0092 24051  012 010507               AND  L    B    S1         L  = MAN000
0093 24052  010 140207               IOR  B    S1   S1         B  = MANEXP
0094 24053  227 174707          READ      INC  PNM  P          START READ
0095 24054  370 036747          RTN                            RETURN
0096                    *
0097 24055  010 020232  PDONE         MPP1 PASS B    MPPB       GET B
0098 24056  227 174707          READ      INC  PNM  P          START READ
0099 24057  370 036747          RTN                            RETURN
0100                    *
0101                    ************************************************************
```

```
0103                    ************************************************************
0104                    *
0105                    *  SUBROUTINE FMPY
0106                    *
0107                    *  STARTS BOX MULTIPLY ON ACCUMULATOR
0108                    *  AND REGISTERS (S2 S3)
0109                    *
0110                    *
0111 24060  306 043242  FMPY     JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0112 24061  340 110607           IMM       LOW  IRCM 044B      BOX MPY OPCODE
0113 24062  010 036751                MPP2                      START BOX
0114 24063  010 042432                MPP1 PASS MPPB S2         SEND OP1 = (S2 S3)
0115 24064  370 044432           RTN  MPP1 PASS MPPB S3         RETURN
0116                    *
0117                    *
0118                    ************************************************************
0119                    *
0120                    *
0121                    *  SUBROUTINE WAIT1
0122                    *
0123                    *  WAITS FOR BOX TO COMPLETE EXECUTION
0124                    *  LOOPS FOR COUNT OF 32, THEN ABORTS
0125                    *  ON ABORT:  A SET TO ALL ONES
0126                    *             B RESTORED
0127                    *             GENERATE MP INT
0128                    *
0129                    *
0130 24065  340 100547  WAIT1    IMM       LOW  CNTR 040B      CNTR=32
0131 24066  323 104102           JMP  CNDX HOI       INTRT1    CHECK FOR INTERRUPTS
0132 24067  366 004102  LOOP1    RTN  CNDX MPP                 RETURN WHEN DONE
0133 24070  010 036765           DCNT                          DECREMENT CNTR
0134 24071  366 000742           RTN  CNDX MPP                 RETURN WHEN DONE
0135 24072  326 143342           JMP  CNDX CNT8 RJS  LOOP1     ELSE LOOP1 32 TIMES
0136 24073  011 136154  BAILOUT   '  SOV  ONE  A               SET A = 177777B
0137 24074  355 165047           IMM       CMHI S2   172B      SET IRCM = MIA 00
0138 24075  010 042607                     PASS IRCM S2
0139 24076  010 052206                MOG  PASS B    S6         RESTORE B, MP INT
0140 24077  000 075707                     DEC  P    P          SET P = ERROR ADDR
0141 24100  227 174707           READ     INC  PNM  P          START READ
0142 24101  320 000007           JMP                 FETCH     RETURN
0143                    *
0144                    ************************************************************
0145                    *
0146                    *  INTERRUPT ROUTINE
0147                    *
0148                    *  RESTORES A,B,P AND RETURNS
0149                    *
0150                    *
0151 24102  336 043342  INTRT1   JMP  CNDX NSNG RJS  LOOP1     RTN IF SINGLE STEP
0152 24103  000 075732           MPP1 DEC  P    P          RESET BOX, SET ADDR
0153 24104  010 050147                PASS A    S5         RESTORE A
0154 24105  010 052207                PASS B    S6         RESTORE B
0155 24106  320 000307           JMP                 HORI
0156                    *
0157                    ************************************************************
```

```
0159                    ************************************************************
0160                    *
0161                    *  SUBROUTINE FSUB2
0162                    *
0163                    *  STARTS BOX SUB ON ACC AND (S2 S3)
0164                    *
0165 24107  306 043242  FSUB2    JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0166 24110  340 060607           IMM       LOW  IRCM 030B       BOX SUB OPCODE
0167 24111  010 036751                MPP2                      START BOX
0168 24112  010 042432                MPP1 PASS MPPB S2         SEND OP1 = (S2 S3)
0169 24113  370 044432           RTN  MPP1 PASS MPPB S3         RETURN
0170                    *
0171                    ************************************************************
0172                    *
0173                    *  SUBROUTINE FADD
0174                    *
0175                    *  STARTS BOX ADD ON ACC AND (S2 S3)
0176                    *
0177 24114  306 043242  FADD     JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0178 24115  340 020607           IMM       LOW  IRCM 010B       BOX ADD OPCODE
0179 24116  010 036751                MPP2                      START BOX
0180 24117  010 042432                MPP1 PASS MPPB S2         SEND OP1 = (S2 S3)
0181 24120  370 044432           RTN  MPP1 PASS MPPB S3         RETURN
0182                    *
0183                    ************************************************************
0184                    *
0185                    *  SUBROUTINE FDIV7
0186                    *
0187                    *  STARTS BOX DIV ON (S7 S8) AND ACC
0188                    *
0189 24121  306 043242  FDIV7    JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0190 24122  340 150607           IMM       LOW  IRCM 064B       BOX DIV OPCODE
0191 24123  010 036751                MPP2                      START BOX
0192 24124  010 054432                MPP1 PASS MPPB S7         SEND OP1 = (S7 S8)
0193 24125  370 056432           RTN  MPP1 PASS MPPB S8         RETURN
0194                    *
0195                    ************************************************************
0196                    *
0197                    *  SUBROUTINE XSQ
0198                    *
0199                    *  SAVES ACC IN (S7 S8) AND STARTS ACC*ACC
0200                    *
0201 24126  306 043242  XSQ      JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0202 24127  340 100607           IMM       LOW  IRCM 040B       BOX MPY OPCODE
0203 24130  010 021332                MPP1 PASS S7   MPPB       SAVE IN (S7 S8)
0204 24131  010 021372                MPP1 PASS S8   MPPB
0205 24132  010 036751                MPP2                      START BOX
0206 24133  010 054432                MPP1 PASS MPPB S7         SEND OP1 = (S7 S8)
0207 24134  010 056432                MPP1 PASS MPPB S8
0208 24135  010 054432                MPP1 PASS MPPB S7         SEND OP2 = (S7 S8)
0209 24136  370 056432           RTN  MPP1 PASS MPPB S8         RETURN
0210                    *
0211                    ************************************************************
```

```
0213                    ************************************************************
0214                    *
0215                    *  SUBROUTINE FADA2
0216                    *
0217                    *  STARTS BOX ADD ON (A B) AND (S2 S3)
0218                    *
0219 24137  306 043242  FADA2    JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0220 24140  340 000607           IMM       LOW  IRCM 0006      BOX ADD OPCODE
0221 24141  010 036751  BAB23         MPP2                     START BOX
0222 24142  010 006432               MPP1 PASS MPPB A          SEND OP1 = (A B)
0223 24143  010 010432               MPP1 PASS MPPB B
0224 24144  010 042432               MPP1 PASS MPPB S2         SEND OP2 = (S2 S3)
0225 24145  370 044432           RTN  MPP1 PASS MPPB S3         RETURN
0226                    *
0227                    ************************************************************
0228                    *
0229                    *  SUBROUTINE FDAVC2
0230                    *
0231                    *  STARTS BOX DIV ON ACC AND (S2 S3)
0232                    *
0233 24146  306 043242  FDVAC2   JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0234 24147  340 150607           IMM       LOW  IRCM 064B      BOX DIV OPCODE
0235 24150  010 036751               MPP2                     START BOX
0236 24151  010 042432               MPP1 PASS MPPB S2         SEND OP1 = (S2 S3)
0237 24152  370 044432           RTN  MPP1 PASS MPPB S3         RETURN
0238                    *
0239                    ************************************************************
0240                    *
0241                    *  SUBROUTINE VMPY
0242                    *
0243                    *  STARTS BOX MPY ON ACC AND (S9 S10)
0244                    *
0245 24153  306 043242  VMPY     JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0246 24154  340 110607           IMM       LOW  IRCM 044B      BOX MPY OPCODE
0247 24155  010 036751               MPP2                     START BOX
0248 24156  010 060432               MPP1 PASS MPPB S9         SEND OP1 = (S9 S10)
0249 24157  370 062432           RTN  MPP1 PASS MPPB S10        RETURN
0250                    *
0251                    ************************************************************
0252                    *
0253                    *  SUBROUTINE FSUB
0254                    *
0255                    *  STARTS SUB ON (S2 S3) AND ACC
0256                    *
0257 24160  006 037047  NEGATE             ZERO S2              SET (S2 S3) = 0
0258 24161  006 037107                    ZERO S3
0259                    *
0260                    *
0261 24162  306 043242  FSUB     JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0262 24163  340 050607           IMM       LOW  IRCM 024B      BOX SUB OPCODE
0263 24164  010 036751               MPP2                     START BOX
0264 24165  010 042432               MPP1 PASS MPPB S2         SEND OP1 = (S2 S3)
0265 24166  370 044432           RTN  MPP1 PASS MPPB S3
0266                    *
0267                    ************************************************************
```

```
0269                    ************************************************************
0270                    *
0271                    *  SUBROUTINE REDUCE
0272                    *
0273 24167  345 043047  FOPI     IMM        HIGH S2   121B      SET (S2 S3 S4) TO 4/PI
0274 24170  341 170507             IMM      LOW  L    174B
0275 24171  012 043047                      AND  S2   S2
0276 24172  347 003107             IMM      HIGH S3   301B
0277 24173  342 156507             IMM      LOW  L    267B
0278 24174  012 045107                      AND  S3   S3
0279 24175  344 117147             IMM      HIGH S4   047B
0280 24176  340 004507             IMM      LOW  L    002B
0281 24177  012 047147                      AND  S4   S4
0282                    *
0283                    *
0284 24200  340 102607  REDUCE   IMM        LOW  IRCM 041B      BOX MPY OPCODE
0285 24201  340 000511             IMM MPP2 LOW  L    000B      SET L = 177400B
0286 24202  010 042432                 MPP1 PASS MPPB S2        SEND OP1 = (S2 S3 S4)
0287 24203  010 044432                 MPP1 PASS MPPB S3
0288 24204  010 046432                 MPP1 PASS MPPB S4
0289 24205  010 006432                 MPP1 PASS MPPB A         SEND OP2 = (A B)
0290 24206  012 011307                      AND  S7   B         SET S7 = MAN(B)
0291 24207  010 054432                 MPP1 PASS MPPB S7
0292 24210  014 011347                      SANL S8   B         SET S8 = EXPO(B)
0293 24211  010 056432                 MPP1 PASS MPPB S8
0294 24212  305 003247             JSB                WAIT1    WAIT FOR BOX
0295 24213  010 021332                 MPP1 PASS S7   MPPB      SAVE IN (S7 S8 S9)
0296 24214  010 021372                 MPP1 PASS S8   MPPB
0297 24215  010 021432                 MPP1 PASS S9   MPPB
0298 24216  341 022607             IMM      LOW  IRCM 111B      BOX FIX OPCODE
0299 24217  353 172511             IMM MPP2 CMLO L    375B      SET L = 000002B
0300 24220  305 003247             JSB                WAIT1    WAIT FOR BOX
0301 24221  010 020172                 MPP1 PASS A    MPPB      SAVE INTEGER IN A
0302 24222  150 006164             LWF R1   PASS A    A         SET FLAG = BIT0
0303 24223  010 006162                 L1   PASS A    A         SET BIT0 = 0
0304 24224  334 051402             JMP CNDX FLAG RJS  RNXT
0305 24225  010 006747                      PASS      A         TEST A
0306 24226  327 111402             JMP CNDX AL15      RNXT      JUMP IF A < 0
0307 24227  243 006147             ENV      ADD  A    A         SET A = A+2
0308 24230  341 042607  RNXT     IMM        LOW  IRCM 121B      BOX FLOAT OPCODE
0309 24231  010 036751                 MPP2                     START BOX
0310 24232  010 006432                 MPP1 PASS MPPB A         SEND OPA = A
0311 24233  305 003247             JSB                WAIT1    WAIT FOR BOX
0312 24234  340 052607             IMM      LOW  IRCM 025B      BOX SUBTRACT OPCODE
0313 24235  010 036751                 MPP2                     START BOX
0314 24236  010 054432                 MPP1 PASS MPPB S7        SEND OP1 = (S7 S8 S9)
0315 24237  010 056432                 MPP1 PASS MPPB S8
0316 24240  370 060432             RTN  MPP1 PASS MPPB S9
0317                    *
0318                    *
0319                    ************************************************************
```

```
0321                    ************************************************************
0322                    *
0323                    * TANGENT ROUTINE
0324                    *
0325                    *
0326 24241  010 007213  TAN          COV  PASS S5   A          SAVE A,B FOR
0327 24242  010 011247               PASS S6   B             INTRT ROUTINE
0328 24243  305 007347         JSB                FOPI       X = 4X/PI, REDUCE
0329 24244  335 115242         JMP  CNDX OVFL     TANERR     REDUCE ERROR
0330 24245  010 006164              R1   PASS A    A          SET FLAG = BIT1(N)
0331 24246  150 006764         LWF  R1   PASS     A
0332 24247  305 005307         JSB                XSQ        GET X, SQUARE
0333 24250  346 177047         IMM       HIGH S2  277B       SET (S2 S3) = C4 =
0334 24251  343 146507         IMM       LOW  L   363B          -4.0030956
0335 24252  012 043047              AND  S2   S2
0336 24253  345 045107         IMM       HIGH S3  122B
0337 24254  340 014507         IMM       LOW  L   006B
0338 24255  012 045107              AND  S3   S3
0339 24256  306 043242         JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0340 24257  010 021432              MPP1 PASS S9   MPPB       SAVE IN XSQ (S9 S10)
0341 24260  010 021472              MPP1 PASS S10  MPPB
0342 24261  305 004647         JSB                FADD+1     Z = Z + C4
0343 24262  346 141047         IMM       HIGH S2  260B       SET (S2 S3) = C3 =
0344 24263  340 016507         IMM       LOW  L   007B          -1279.5424
0345 24264  012 043047              AND  S2   S2
0346 24265  345 045107         IMM       HIGH S3  122B
0347 24266  340 054507         IMM       LOW  L   026B
0348 24267  012 045107              AND  S3   S3
0349 24270  305 006307         JSB                FDVAC2     Z = C3/Z
0350 24271  010 061047              PASS S2   S9       SET (S2 S3) = XSQ
0351 24272  010 063107              PASS S3   S10
0352 24273  305 004607         JSB                FADD       Z = Z + XSQ
0353 24274  345 003047         IMM       HIGH S2  101B       SET (S2 S3) = C2 =
0354 24275  341 150507         IMM       LOW  L   164B          .0019974806
0355 24276  012 043047              AND  S2   S2
0356 24277  354 053107         IMM       CMHI S3  025B
0357 24300  353 142507         IMM       CMLO L   361B
0358 24301  017 045107              NOR  S3   S3
0359 24302  305 003007         JSB                FMPY       Z = C2 * Z
0360 24303  345 027047         IMM       HIGH S2  113B       SET (S2 S3) = C1 =
0361 24304  340 164507         IMM       LOW  L   072B          .146926953
0362 24305  012 043047              AND  S2   S2
0363 24306  354 005107         IMM       CMHI S3  002B
0364 24307  353 172507         IMM       CMLO L   375B
0365 24310  017 045107              NOR  S3   S3
0366 24311  305 004607         JSB                FADD       Z = Z + C1
0367 24312  010 055047              PASS S2   S7       SET (S2 S3) = X
0368 24313  010 057107              PASS S3   S8
0369 24314  305 003007         JSB                FMPY       Z = X * Z
0370 24315  334 055042         JMP  CNDX FLAG RJS  EXIT1
```

```
0372 24316  355 177047            IMM       CMHI S2   177B      SET (S2 S3) = -1.0
0373 24317  006 037107                      ZERO S3
0374 24320  305 006307            JSB                 FDVAC2    Z = -1/Z
0375 24321  305 003247  EXIT1     JSB                 WAIT1     WAIT FOR BOX
0376 24322  227 174707            READ      INC  PNM  P         START READ
0377 24323  010 020172                 MPP1 PASS A    MPPB      PUT ANSWER IN (A B)
0378 24324  370 020232            RTN  MPP1 PASS B    MPPB      RETURN
0379                     *
0380                     *
0381 24325  340 162514  TANERR    IMM  SOV  LOW  L    071B      SET (A B) TO
0382 24326  344 140147            IMM       HIGH A    060B         ASCII 0?OR
0383 24327  012 006147                      AND  A    A
0384 24330  341 044507            IMM       LOW  L    122B
0385 24331  345 036207            IMM       HIGH B    117B
0386 24332  000 075732  ERRET          MPP1 DEC  P    P         SET P = ERROR ADDR
0387 24333  227 174707            READ      INC  PNM  P         START READ
0388 24334  372 010207            RTN       AND  B    B         ERROR RETURN
0389                     *
0390                     *
0391                     ************************************************************
```

```
0393                    ************************************************************
0394                    *
0395                    *
0396                    *      SINE ROUTINE
0397                    *
0398                    *
0399 24335  353 173007  COS       IMM        CMLO S1    375B       SET J=2
0400 24336  325 016007             JMP                   SIN+1
0401                    *
0402                    *
0403 24337  006 037007  SIN                   ZERO S1               SET J=0
0404 24340  010 007213                  COV   PASS S5    A          SAVE A,B FOR
0405 24341  010 011247                        PASS S6    B             INTRT ROUTINE
0406 24342  305 007347             JSB                   FOPI       X = 4X/PI, REDUCE
0407 24343  335 122542             JMP   CNDX OVFL       SINERR     REDUCE ERROR
0408 24344  010 040507                        PASS L     S1         SET L=J
0409 24345  003 006164                  R1    ADD  A     A          N = (N+J)/2
0410 24346  150 007024             LWF   R1    PASS S1    A          SET FLAG = BIT1(N)
0411 24347  305 005307             JSB                   XSQ        GET X, SQUARE
0412 24350  334 060242             JMP   CNDX FLAG RJS   SINAG      SIN OR COS ?
0413                    *
0414 24351  343 127107  COSAG     IMM        LOW  S3    353B       SET (S2 S3) = CC4 =
0415 24352  344 012507             IMM        HIGH L     005B          -.00031957
0416 24353  012 045107                        AND  S3    S3
0417 24354  346 131047             IMM        HIGH S2    254B
0418 24355  340 164507             IMM        LOW  L     072B
0419 24356  012 043047                        AND  S2    S2
0420 24357  306 043242             JSB   CNDX MPP  RJS   WAIT1      WAIT FOR BOX
0421 24360  010 021432                  MPP1  PASS S9    MPPB       SAVE VSQR IN (S9 S10)
0422 24361  010 021472                  MPP1  PASS S10   MPPB
0423 24362  305 003047             JSB                   FMPY+1     Z = Z*CC4
0424 24363  345 000507             IMM        HIGH L     100B       SET (S2 S3) = CC3 =
0425 24364  343 133047             IMM        LOW  S2    355B          .015851077
0426 24365  012 043047                        AND  S2    S2
0427 24366  354 036507             IMM        CMHI L     017B
0428 24367  353 157107             IMM        CMLO S3    367B
0429 24370  017 045107                        NOR  S3    S3
0430 24371  305 004607             JSB                   FADD       Z = Z+CC3
0431 24372  305 006547             JSB                   VMPY       Z = Z*VSQR
0432 24373  356 142507             IMM        CMHI L     261B       SET (S2 S3) = CC2 =
0433 24374  350 027047             IMM        CMLO S2    013B          -.30842483
0434 24375  017 043047                        NOR  S2    S2
0435 24376  344 045107             IMM        HIGH S3    022B
0436 24377  305 004607             JSB                   FADD       Z = Z+CC2
0437 24400  305 006547             JSB                   VMPY       Z = Z*VSQR
0438 24401  356 177047             IMM        CMHI S2    277B       SET (S2 S3) = CC1 =
0439 24402  353 173107             IMM        CMLO S3    375B          1.0
0440 24403  305 004607             JSB                   FADD       Z = Z+CC1
0441 24404  325 022347             JMP                   CKSGN      TEST SIGN OF ANSWER
```

```
0443 24405  342 067047  SINAG    IMM        LOW  S2   233B      SET (S2 S3) = C4 =
0444 24406  346 150507            IMM        HIGH L    264B         -.000035950439
0445 24407  012 043047                       AND  S2   S2
0446 24410  355 002507            IMM        CMHI L    101B
0447 24411  353 113107            IMM        CMLO S3   345B
0448 24412  017 045107                       NOR  S3   S3
0449 24413  306 043242            JSB  CNDX MPP  RJS  WAIT1     WAIT FOR BOX
0450 24414  010 021432                 MPP1 PASS S9   MPPB      SAVE VSQR IN (S9 S10)
0451 24415  010 021472                 MPP1 PASS S10  MPPB
0452 24416  305 003047            JSB                FMPY+1    Z = VSQR*C4
0453 24417  345 042507            IMM        HIGH L    121B      SET (S2 S3) = C3 =
0454 24420  342 057047            IMM        LOW  S2   227B         .002490001
0455 24421  012 043047                       AND  S2   S2
0456 24422  356 110507            IMM        CMHI L    244B
0457 24423  353 143107            IMM        CMLO S3   361B
0458 24424  017 045107                       NOR  S3   S3
0459 24425  305 004607            JSB                FADD      Z = Z+C3
0460 24426  305 006547            JSB                VMPY      Z = Z*VSQR
0461 24427  346 132507            IMM        HIGH L    255B      SET (S2 S3) = C2 =
0462 24430  341 043047            IMM        LOW  S2   121B         -.0807454325
0463 24431  012 043047                       AND  S2   S2
0464 24432  354 044507            IMM        CMHI L    022B
0465 24433  353 167107            IMM        CMLO S3   373B
0466 24434  017 045107                       NOR  S3   S3
0467 24435  305 004607            JSB                FADD      Z = Z+C2
0468 24436  305 006547            JSB                VMPY      Z = Z*VSQR
0469 24437  355 110507            IMM        CMHI L    144B      SET (S2 S3) = C1 =
0470 24440  352 017047            IMM        CMLO S2   207B         .78539816
0471 24441  017 043047                       NOR  S2   S2
0472 24442  354 045107            IMM        CMHI S3   022B
0473 24443  305 004607            JSB                FADD      Z = Z+C1
0474 24444  010 055047                       PASS S2   S7        SET (S2 S3) = V
0475 24445  010 057107                       PASS S3   S8
0476 24446  305 003007            JSB                FMPY      Z = Z * V
0477                    *
0478 24447  010 040747  CKSGN              PASS       S1        TEST SIGN
0479 24450  321 155042            JMP  CNDX ALO  RJS  EXIT1     EXIT IF BIT2(N) = 1
0480 24451  305 007007            JSB                NEGATE    Z = -Z
0481 24452  325 015047            JMP                EXIT1     EXIT ROUTINE
0482                    *
0483                    *
0484 24453  344 140514  SINERR    IMM  SOV  HIGH L    060B      SET (A B) TO
0485 24454  340 152147            IMM        LOW  A    065B         ASCII 05OR
0486 24455  012 006147                       AND  A    A
0487 24456  345 036507            IMM        HIGH L    117B
0488 24457  341 044207            IMM        LOW  B    122B
0489 24460  325 015507            JMP                ERRET     ERROR RETURN
0490                    *
0491                    *
0492                    ************************************************************
```

```
0494                    ************************************************************
0495                    *
0496                    *      NATURAL LOGARITHM ROUTINE
0497                    *
0498 24461  010 011253  ALOG           COV  PASS S6   B          SAVE A,B FOR
0499 24462  010 007207                     PASS S5   A              INTRT ROUTINE
0500 24463  320 026702           JMP  CNDX ALZ       LOGERR     ERROR IF X = 0
0501 24464  327 126702           JMP  CNDX AL15      LOGERR     ERROR IF X < 0
0502 24465  305 001007           JSB                 FLUN       UNPACK MANT AND EXP
0503 24466  357 062507           IMM       CMHI L    331B       TEST FOR BITS 14-7
0504 24467  003 050747                     ADD       S5            OF X > 264B
0505 24470  327 123602           JMP  CNDX AL15      LGNXT      JUMP IF GREATER
0506 24471  000 006147                     DEC  A    A          DECREMENT EXPO(X)
0507 24472  007 110207                     INC  B    B          SET EXPO(MAN(X)) = 1
0508 24473  007 110207                     INC  B    B
0509 24474  341 040607  LGNXT    IMM       LOW  IRCM 120B       BOX FLOAT OPCODE
0510 24475  356 177051           IMM  MPP2 CMHI S2   277B       SET (S2 S3) = 1.0
0511 24476  010 006432                MPP1 PASS MPPB A          SEND OP1 = EXPO(X)
0512 24477  010 050147                     PASS A    S5         SET (A B) = MAN(X)
0513 24500  353 173107           IMM       CMLO S3   375B
0514 24501  306 043242           JSB  CNDX MPP  RJS  WAIT1      WAIT FOR BOX
0515 24502  010 021032                MPP1 PASS S1   MPPB       SAVE CHAR IN (S1 S11)
0516 24503  010 021532                MPP1 PASS S11  MPPB
0517 24504  340 040607           IMM       LOW  IRCM 020B       BOX SUB OPCODE
0518 24505  305 006047           JSB                 BAB23      Z = MAN(X) - 1.0
0519 24506  305 003247           JSB                 WAIT1      WAIT FOR BOX
0520 24507  010 021332                MPP1 PASS S7   MPPB       SAVE IN Y (S7 S8)
0521 24510  010 021372                MPP1 PASS S8   MPPB
0522 24511  305 006007           JSB                 FADA2+1    Z = MAN(X) + 1.0
0523 24512  305 005047           JSB                 FDIV7      Z = Z/Y
0524 24513  305 003247           JSB                 WAIT1      WAIT FOR BOX
0525 24514  010 021432                MPP1 PASS S9   MPPB       SAVE IN W (S9 S10)
0526 24515  010 021472                MPP1 PASS S10  MPPB
0527 24516  340 130607           IMM       LOW  IRCM 054B       BOX Z*Z OPCODE
0528 24517  345 125051           IMM  MPP2 HIGH S2   152B       START BOX
0529 24520  340 020507           IMM       LOW  L    010B       SET (S2 S3) = C =
0530 24521  012 043047                     AND  S2   S2            1.6567626301
0531 24522  345 115107           IMM       HIGH S3   146B
0532 24523  340 004507           IMM       LOW  L    002B
0533 24524  012 045107                     AND  S3   S3
0534 24525  305 004347           JSB                 FSUB2      Z = Z-C
0535 24526  346 126507           IMM       HIGH L    253B       SET (S2 S3) = MB =
0536 24527  342 015047           IMM       LOW  S2   206B          -2.6398577035
0537 24530  012 043047                     AND  S2   S2
0538 24531  345 023107           IMM       HIGH S3   111B
0539 24532  340 010507           IMM       LOW  L    004B
0540 24533  012 045107                     AND  S3   S3
0541 24534  305 006307           JSB                 FDVAC2     Z = MB/Z
0542 24535  345 044507           IMM       HIGH L    122B       SET (S2 S3) = A =
0543 24536  342 141047           IMM       LOW  S2   260B          1.2920070987
0544 24537  012 043047                     AND  S2   S2
0545 24540  344 177107           IMM       HIGH S3   077B
0546 24541  340 004507           IMM       LOW  L    002B
0547 24542  012 045107                     AND  S3   S3
```

```
0549 24543  305 004607          JSB                 FADD      Z = Z+A
0550 24544  305 006547          JSB                 VMPY      Z = Z*W
0551 24545  010 041047                    PASS S2   S1        SET (S2 S3) = CHAR
0552 24546  010 065107                    PASS S3   S11
0553 24547  305 004607          JSB                 FADD      Z = Z+CHAR
0554 24550  355 060507          IMM       CMHI L    130B      SET (S2 S3) = LE2 =
0555 24551  352 163047          IMM       CMLO S2   271B         .6931471806
0556 24552  017 043047                    NOR  S2   S2
0557 24553  357 147116          IMM  CLFL CMHI S3   363B
0558 24554  305 003007          JSB                 FMPY      Z = Z*LE2
0559 24555  325 015047          JMP                 EXIT1     EXIT ROUTINE
0560                    *
0561                    *
0562 24556  344 140514 LOGERR    IMM  SOV  HIGH L    060B      SET (A B) TO
0563 24557  340 144157          IMM  STFL LOW  A    062B         ASCII 02UN
0564 24560  012 006147                    AND  A    A
0565 24561  345 052507          IMM       HIGH L    125B
0566 24562  341 034207          IMM       LOW  B    116B
0567 24563  325 015507          JMP                 ERRET     ERROR RETURN
0568                    *
0569                    *
0570                    ****************************************************************
0571                    *
0572                    *
0573                    *  COMMON LOG ROUTINE
0574                    *
0575 24564  305 023047 ALOGT     JSB                 ALOG      COMPUTE LN(X)
0576 24565  374 023042          RTN  CNDX FLAG                ERROR RETURN
0577                    *
0578 24566  000 075707                    DEC  P    P         SET RETURN ADDRESS
0579 24567  355 136507          IMM       CMHI L    157B      SET (S2 S3) = LOG(E) =
0580 24570  350 133047          IMM       CMLO S2   055B         .43429228
0581 24571  017 043047                    NOR  S2   S2
0582 24572  347 131107          IMM       HIGH S3   354B
0583 24573  305 003047          JSB                 FMPY+1
0584 24574  325 015047          JMP                 EXIT1     EXIT ROUTINE
0585                    *
0586                    *
0587                    ****************************************************************
```

```
0589                    ************************************************************
0590                    *
0591                    *      ARCTANGENT ROUTINE
0592                    *
0593                    *
0594 24575  340 000607 ATAN      IMM        LOW  IRCM 000B       BOX ADD OPCODE
0595 24576  150 010764           LWF  R1    PASS      B          SET FLAG = BIT0(B)
0596 24577  010 011253                COV   PASS S6   B          SAVE A,B FOR
0597 24600  010 007207                      PASS S5   A             INTRT ROUTINE
0598 24601  320 035202           JMP  CNDX ALZ       ZERO1      RETURN ZERO IF X = 0
0599 24602  334 030242           JMP  CNDX FLAG      POS        JUMP IF ABS(X) < .5
0600 24603  327 170242           JMP  CNDX AL15 RJS  POS        JUMP IF X > 0
0601 24604  340 040607           IMM        LOW  IRCM 020B       BOX SUBTRACT OPCODE
0602 24605  006 037051 POS              MPP2 ZERO S2             SET S2 = 0
0603 24606  010 042432                 MPP1 PASS MPPB S2        SEND OP1 = 0
0604 24607  010 042432                 MPP1 PASS MPPB S2
0605 24610  010 006432                 MPP1 PASS MPPB A         SEND OP2 = (A B)
0606 24611  010 010432                 MPP1 PASS MPPB B
0607 24612  334 032042           JMP  CNDX FLAG      ATAN3      JUMP IF ABS(X) < .5
0608 24613  345 111007           IMM        HIGH S1   144B       SET (S1 S11) = PI/4
0609 24614  342 016507           IMM        LOW  L    207B
0610 24615  012 041007                      AND  S1   S1
0611 24616  354 045507           IMM        CMHI S11  022B
0612 24617  356 177047           IMM        CMHI S2   277B       SET (S2 S3) = 1.0
0613 24620  353 173107           IMM        CMLO S3   375B
0614 24621  350 006507           IMM        CMLO L    003B       SET L = 000374B
0615 24622  012 010747                      AND       B          TEST EXPO(X)
0616 24623  320 031402           JMP  CNDX ALZ       ATAN1      JUMP IF ABS(X) < 2
0617 24624  007 165507                      INC  S11  S11        SET (S1 S11) = PI/2
0618 24625  007 165507                      INC  S11  S11
0619 24626  305 006307           JSB                  FDVAC2     X = 1/X
0620 24627  325 032047           JMP                  ATAN3      CONTINUE
0621 24630  306 043242 ATAN1    JSB  CNDX MPP  RJS    WAIT1      WAIT FOR BOX
0622 24631  010 020172                 MPP1 PASS A     MPPB       SAVE X IN (A B)
0623 24632  010 020232                 MPP1 PASS B     MPPB
0624 24633  305 007147           JSB                  FSUB+1     Z = 1 - X
0625 24634  305 003247           JSB                  WAIT1      WAIT FOR BOX
0626 24635  010 021332                 MPP1 PASS S7    MPPB       SAVE (1-X) IN (S7 S8)
0627 24636  010 021372                 MPP1 PASS S8    MPPB
0628 24637  305 006007           JSB                  FADA2+1    Z = 1 + X
0629 24640  305 005047           JSB                  FDIV7      Z = (1-X)/(1+X)
0630 24641  305 003247 ATAN3    JSB                  WAIT1      WAIT FOR BOX
0631 24642  010 021332                 MPP1 PASS S7    MPPB       SAVE NEW X IN (S7 S8)
0632 24643  010 021372                 MPP1 PASS S8    MPPB
0633 24644  340 130613           IMM  COV  LOW  IRCM 054B       BOX Z*Z OPCODE
0634 24645  345 001051           IMM  MPP2 HIGH S2   100B       SET (S2 S3) = C4 =
0635 24646  342 136507           IMM        LOW  L    257B          2.0214656
0636 24647  012 043047                      AND  S2   S2
0637 24650  347 063107           IMM        HIGH S3   331B
0638 24651  340 010507           IMM        LOW  L    004B
0639 24652  012 045107                      AND  S3   S3
```

```
0641 24653  306 043242          JSB  CNDX MPP  RJS  WAIT1     WAIT FOR BOX
0642 24654  010 021432               MPP1 PASS S9   MPPB      SAVE XSQ IN (S9 S10)
0643 24655  010 021472               MPP1 PASS S10  MPPB
0644 24656  305 004647          JSB                FADD+1    Z = Z + C4
0645 24657  346 151047          IMM       HIGH S2  264B      SET (S2 S3) = C3 =
0646 24660  340 144507          IMM       LOW  L   062B         -4.7376165
0647 24661  012 043047                    AND  S2  S2
0648 24662  346 157107          IMM       HIGH S3  267B
0649 24663  340 014507          IMM       LOW  L   006B
0650 24664  012 045107                    AND  S3  S3
0651 24665  305 006307          JSB                FDVAC2    Z = C3/Z
0652 24666  010 061047                    PASS S2  S9        SET (S2 S3) = XSQ
0653 24667  010 063107                    PASS S3  S10
0654 24670  305 004607          JSB                FADD      Z = Z + XSQ
0655 24671  345 037047          IMM       HIGH S2  117B      SET (S2 S3) = C2 =
0656 24672  340 016507          IMM       LOW  L   007B         .154357652
0657 24673  012 043047                    AND  S2  S2
0658 24674  357 157107          IMM       CMHI S3  367B
0659 24675  353 172507          IMM       CMLO L   375B
0660 24676  017 045107                    NOR  S3  S3
0661 24677  305 003007          JSB                FMPY      Z = C2 * Z
0662 24700  345 057047          IMM       HIGH S2  127B      SET (S2 S3) = C1 =
0663 24701  340 116507          IMM       LOW  L   047B         1.3617611
0664 24702  012 043047                    AND  S2  S2
0665 24703  344 061107          IMM       HIGH S3  030B
0666 24704  340 004507          IMM       LOW  L   002B
0667 24705  012 045107                    AND  S3  S3
0668 24706  305 004607          JSB                FADD      Z = C1 + Z
0669 24707  305 005047          JSB                FDIV7     Z = X/Z
0670 24710  334 034742          JMP  CNDX FLAG       EXIT2     EXIT IF ABS(X) < .5
0671 24711  010 041047                    PASS S2  S1        SET (S2 S3) = PI/N
0672 24712  010 065107                    PASS S3  S11
0673 24713  305 004347          JSB                FSUB2     Z = Z - PI/N
0674 24714  010 050747                    PASS     S5        TEST MAN(X)
0675 24715  327 134742          JMP  CNDX AL15       EXIT2     EXIT IF X < 0
0676 24716  305 007007          JSB                NEGATE    Z = - Z
0677                      *
0678 24717  305 003247  EXIT2   JSB                WAIT1     WAIT FOR BOX
0679 24720  000 075707                    DEC  P   P         SET RETURN ADDR
0680 24721  227 174707          READ      INC  PNM P         START READ
0681 24722  010 020172               MPP1 PASS A    MPPB      SAVE ANS IN (A B)
0682 24723  370 020232          RTN  MPP1 PASS B    MPPB      RETURN
0683                      *
0684                      *
0685 24724  000 075707  ZERO1              DEC  P   P         SET RETURN ADDR
0686 24725  227 174707          READ      INC  PNM P         START READ
0687 24726  370 036747          RTN                          RETURN
0688                      *
0689                      *
0690                      ****************************************************************
```

```
0692                    ************************************************************
0693                    *
0694                    *
0695                    *  SELF-TEST ROUTINE
0696                    *
0697                    *
0698 24727  343 076347  SELFTEST IMM        LOW  DSPI 337B       TURN ON "S" DSPI LED
0699 24730  355 167747           IMM        CMHI S    173B       SET S = 102001B
0700 24731  007 177747                      INC  S    S
0701 24732  010 076307                      PASS DSPL S          SEND TO PANEL
0702 24733  356 176147           IMM        CMHI A    277B       SET (A B) = 4.0
0703 24734  353 162207           IMM        CMLO B    371B
0704 24735  305 040007           JSB                  SQRT       CALCULATE SQRT(4)
0705 24736  007 177747                      INC  S    S          SET S = 102002B
0706 24737  335 136402           JMP  CNDX OVFL       DISPLAY    JUMP IF OVERFLOW
0707 24740  356 176507           IMM        CMHI L    277B       CHECK ANSWER
0708 24741  014 106747                      XOR       A
0709 24742  320 076402           JMP  CNDX ALZ  RJS   DISPLAY    JUMP IF A WRONG
0710 24743  353 166507           IMM        CMLO L    373B
0711 24744  014 110747                      XOR       B
0712 24745  320 076402           JMP  CNDX ALZ  RJS   DISPLAY    JUMP IF B WRONG
0713 24746  353 000507           IMM        CMLO L    300B
0714 24747  010 177747                      IOR  S    S          SET S = 102077B
0715 24750  353 167604  DISPLAY  IMM  RJ30 CMLO X    ESISREVN   REVISION #.
0716 24751  370 076307           RTN        PASS DSPL S          SEND TO PANEL, RETURN
0717                    *
0718                    *
0719                    ************************************************************
```

```
0721                                ORG                 25000B
0722                          *
0723                          ************************************************************
0724                          *
0725                          *     SQUARE ROOT ROUTINE
0726                          *
0727 25000  010 011253  SQRT          COV   PASS S6   B          SAVE A,B FOR
0728 25001  010 007207                      PASS S5   A             INTRT ROUTINE
0729 25002  320 012442          JMP   CNDX ALZ        EXIT3      EXIT IF X = 0
0730 25003  327 103142          JMP   CNDX AL15       SQRERR     ERROR IF X < 0
0731 25004  305 001007          JSB                   FLUN       UNPACK EXP AND MAN
0732 25005  150 006762          LWF   L1   PASS       A          ARITHMETIC SHIFT
0733 25006  150 007524          LWF   R1   PASS S11  A             A-REG RIGHT
0734 25007  321 101202          JMP   CNDX AL0        ODD        JUMP IF EXP ODD
0735                          *
0736 25010  010 050147  EVEN                PASS A    S5         RESTORE A
0737 25011  345 027047          IMM         HIGH S2   113B       SET (S2 S3) = A2 =
0738 25012  342 024507          IMM         LOW  L    212B          .5901621
0739 25013  012 043047                      AND  S2   S2
0740 25014  356 041107          IMM         CMHI S3   220B
0741 25015  340 100607          IMM         LOW  IRCM 040B       BOX MPY OPCODE
0742 25016  305 006047          JSB                   BAB23      Z = F*A2
0743 25017  345 125047          IMM         HIGH S2   152B       SET (S2 S3) = B2 =
0744 25020  343 050507          IMM         LOW  L    324B          .4173076
0745 25021  012 043047                      AND  S2   S2
0746 25022  346 131107          IMM         HIGH S3   254B
0747 25023  325 042047          JMP                   BOTH       CONTINUE
0748                          *
0749 25024  010 050147  ODD                 PASS A    S5         RESTORE A
0750 25025  345 125047          IMM         HIGH S2   152B       SET (S2 S3) = A1 =
0751 25026  343 050507          IMM         LOW  L    324B          .8346152
0752 25027  012 043047                      AND  S2   S2
0753 25030  355 045107          IMM         CMHI S3   122B
0754 25031  340 100607          IMM         LOW  IRCM 040B       BOX MPY OPCODE
0755 25032  305 006047          JSB                   BAB23      Z = F*A1
0756 25033  345 027047          IMM         HIGH S2   113B       SET (S2 S3) = B1 =
0757 25034  342 024507          IMM         LOW  L    212B          .5901621
0758 25035  012 043047                      AND  S2   S2
0759 25036  356 041107          IMM         CMHI S3   220B
0760 25037  007 110207                      INC  B    B          SET F = 2 * F
0761 25040  007 110207                      INC  B    B
0762                          *
0763 25041  305 004607  BOTH    JSB                   FADD       Z = Z+B
0764 25042  010 007307                      PASS S7   A          SET (S7 S8) = F =
0765 25043  010 011347                      PASS S8   B             MAN(X)
0766 25044  305 003247          JSB                   WAIT1      WAIT FOR BOX
0767 25045  010 021072                MPP1  PASS S2   MPPB       SAVE IN P (S2 S3)
0768 25046  010 021132                MPP1  PASS S3   MPPB
0769 25047  305 005107          JSB                   FDIV7+1    Z = F/P
0770 25050  353 172507          IMM         CMLO L    375B       SET L = 2
0771 25051  003 057347                      ADD  S8   S8         SET F = 4 * F
0772 25052  003 057347                      ADD  S8   S8
```

```
0774 25053  305 004607          JSB                     FADD      Z = Z+P
0775 25054  305 003247          JSB                     WAIT1     WAIT FOR BOX
0776 25055  010 021072               MPP1 PASS S2       MPPB      SAVE IN P (S2 S3)
0777 25056  010 021132               MPP1 PASS S3       MPPB
0778 25057  305 005107          JSB                     FDIV7+1   Z = F/P
0779 25060  305 004607          JSB                     FADD      Z = Z+P
0780 25061  004 165507                    SUB   S11      S11       DEC EXPONENT BY 2
0781 25062  325 001307          JMP                     PWR2      Z = Z*2**S11, RETURN
0782                      *
0783                      *
0784 25063  344 140514  SQRERR  IMM  SOV   HIGH  L       060B      SET (A B) TO
0785 25064  340 146147          IMM        LOW   A       063B         ASCII 03UN
0786 25065  012 006147                     AND   A       A
0787 25066  345 052507          IMM        HIGH  L       125B
0788 25067  341 034207          IMM        LOW   B       116B
0789 25070  325 015507          JMP                     ERRET     ERROR RETURN
0790                      *
0791                      *
0792                      ************************************************************
```

```
0794                    ************************************************************
0795                    *
0796                    *     EXPONENTIATION ROUTINE
0797                    *
0798 25071  010 007213 EXP          COV  PASS S5   A        SAVE A,B
0799 25072  010 011247               PASS S6   B           FOR INTRT ROUTINE
0800 25073  345 070507         IMM       HIGH L    134B    SET (S2 S3 S4) =
0801 25074  341 053047         IMM       LOW  S2   125B       2/LE2
0802 25075  012 043047               AND  S2   S2
0803 25076  344 073107         IMM       HIGH S3   035B
0804 25077  342 050507         IMM       LOW  L    224B
0805 25100  012 045107               AND  S3   S3
0806 25101  346 135147         IMM       HIGH S4   256B
0807 25102  340 010507         IMM       LOW  L    004B
0808 25103  012 047147               AND  S4   S4
0809 25104  305 010007         JSB                 REDUCE
0810 25105  150 006762         LWF  L1   PASS      A       ARITH RT SHIFT
0811 25106  150 006164         LWF  R1   PASS A    A
0812 25107  007 107507               INC  S11  A           INC, SAVE IN S11
0813                    *
0814                    *  PERFORM BOUNDS CHECKS
0815                    *
0816 25110  010 050747               PASS      S5        TEST X
0817 25111  327 144702         JMP  CNDX AL15 RJS POSCHK   JUMP IF X > 0
0818                    *
0819 25112  351 176507 NEGCHK  IMM       CMLO L    177B    SET L = 128
0820 25113  003 064747               ADD       S11       INT < -128 ?
0821 25114  327 106502         JMP  CNDX AL15      ZERO2   YES, ANS = 0
0822 25115  325 045107         JMP                 EXPNXT  NO, CONTINUE
0823                    *
0824 25116  335 106642 POSCHK  JMP  CNDX OVFL      EXPERR  REDUCE ERROR
0825 25117  342 000507         IMM       LOW  L    200B    SET L = -128
0826 25120  003 006747               ADD       A         INT > 128?
0827 25121  327 146642         JMP  CNDX AL15 RJS EXPERR   YES, ERROR
0828                    *
0829 25122  305 005307 EXPNXT  JSB                 XSQ     GET X, SQUARE
0830 25123  356 023047         IMM       CMHI S2   211B    SET (S2 S3) = C2 =
0831 25124  355 143107         IMM       CMHI S3   161B       .05761803
0832 25125  353 162507         IMM       CMLO L    371B
0833 25126  017 045107               NOR  S3   S3
0834 25127  305 003007         JSB                 FMPY    Z = C2 * Z
0835 25130  010 055047               PASS S2   S7        SET (S2 S3) = F
0836 25131  010 057107               PASS S3   S8
0837 25132  305 004347         JSB                 FSUB2   Z = Z - F
0838 25133  345 071047         IMM       HIGH S2   134B    SET (S2 S3) = C1 =
0839 25134  341 052507         IMM       LOW  L    125B       5.7708162
0840 25135  012 043047               AND  S2   S2
0841 25136  345 007107         IMM       HIGH S3   103B
0842 25137  340 014507         IMM       LOW  L    006B
0843 25140  012 045107               AND  S3   S3
0844 25141  305 004607         JSB                 FADD    Z = C1 + Z
```

```
0846 25142  305 005047          JSB                     FDIV7     Z = F/Z
0847 25143  356 177047          IMM       CMHI S2       277B      SET (S2 S3) = .5
0848 25144  006 037107                    ZERO S3
0849 25145  305 004607          JSB                     FADD      Z = Z + .5
0850 25146  305 001307          JSB                     PWR2      Z = Z*2**S11
0851 25147  375 141302          RTN   CNDX OVFL RJS               RETURN IF PWR2 OK
0852 25150  010 050747                PASS              S5        TEST X
0853 25151  327 146642          JMP   CNDX AL15 RJS     EXPERR    ERROR IF X > 0
0854                    *
0855                    *
0856 25152  227 174732  ZERO2   READ  MPP1 INC  PNM     P         START READ
0857 25153  006 036153                COV  ZERO A                 SET ANS = 0
0858 25154  366 036207          RTN        ZERO B                 RETURN
0859                    *
0860                    *
0861 25155  344 140514  EXPERR  IMM   SOV  HIGH L      060B      SET (A B) TO
0862 25156  340 156147          IMM        LOW  A      067B         ASCII 07OF
0863 25157  012 006147                     AND  A      A
0864 25160  345 036507          IMM        HIGH L      117B
0865 25161  341 014207          IMM        LOW  B      106B
0866 25162  325 015507          JMP                    ERRET     ERROR RETURN
0867                    *
0868                    *
0869                    ************************************************************
```

```
0871                    ************************************************************
0872                    *
0873                    *  TANH ROUTINE
0874                    *
0875 25163  010 007213  TANH          COV   PASS S5    A          SAVE A,B FOR
0876 25164  010 011247                    PASS S6    B             INTRT ROUTINE
0877                    *
0878                    *  PERFORM BOUNDS CHECKS
0879                    *
0880 25165  321 110342          JMP   CNDX ALO        TANH1      JUMP IF ABS(X) < .5
0881 25166  350 016507          IMM         CMLO L    007B       SET L = 000370B
0882 25167  012 010747                      AND       B
0883 25170  320 052202          JMP   CNDX ALZ  RJS   TANH2      JUMP IF ABS(X) > 8
0884                    *
0885 25171  007 110207                      INC   B     B          SET X = 2*X
0886 25172  007 110207                      INC   B     B
0887 25173  305 043547          JSB                     EXP+2      GET EXP(2*X)
0888 25174  000 075707                      DEC   P     P          SET RETURN ADDRESS
0889 25175  356 177047          IMM         CMHI  S2    277B       SET (S2 S3) = 1.0
0890 25176  353 173107          IMM         CMLO  S3    375B
0891 25177  340 040607          IMM         LOW   IRCM  020B       BOX SUB OPCODE
0892 25200  305 006047          JSB                     BAB23      Z = EXP(2*X) - 1
0893 25201  305 003247          JSB                     WAIT1      WAIT FOR BOX
0894 25202  010 021332                MPP1  PASS  S7    MPPB       SAVE IN (S7 S8)
0895 25203  010 021372                MPP1  PASS  S8    MPPB
0896 25204  305 006007          JSB                     FADA2+1    Z = EXP(2*X) + 1
0897 25205  305 005047          JSB                     FDIV7      Z = TANH(X)
0898 25206  325 034747          JMP                     EXIT2      EXIT ROUTINE
0899                    *
0900 25207  340 100607  TANH1   IMM         LOW   IRCM  040B       BOX MPY OPCODE
0901 25210  010 007047                    PASS  S2    A          SET (S2 S3) = X
0902 25211  010 011107                    PASS  S3    B
0903 25212  305 006047          JSB                     BAB23      Z = X * X
0904 25213  345 041047          IMM         HIGH  S2    120B       SET (S2 S3) = C3 =
0905 25214  340 112507          IMM         LOW   L     045B          2.5045337
0906 25215  012 043047                      AND   S2    S2
0907 25216  344 111107          IMM         HIGH  S3    044B
0908 25217  340 010507          IMM         LOW   L     004B
0909 25220  012 045107                      AND   S3    S3
0910 25221  305 004607          JSB                     FADD       Z = Z + C3
0911 25222  345 005307          IMM         HIGH  S7    102B       SET (S7 S8) = C2 =
0912 25223  343 116507          IMM         LOW   L     347B          2.0907609
0913 25224  012 055307                      AND   S7    S7
0914 25225  346 007347          IMM         HIGH  S8    203B
0915 25226  340 010507          IMM         LOW   L     004B
0916 25227  012 057347                      AND   S8    S8
0917 25230  305 005047          JSB                     FDIV7      Z = C2 / Z
```

```
0919 25231  345 051047           IMM       HIGH S2   124B      SET (S2 S3) = C1 =
0920 25232  342 054507           IMM       LOW  L    226B
0921 25233  012 043047                     AND  S2   S2
0922 25234  355 035107           IMM       CMHI S3   116B
0923 25235  353 172507           IMM       CMLO L    375B
0924 25236  017 045107                     NOR  S3   S3
0925 25237  305 004607           JSB                 FADD      Z = C1 + Z
0926 25240  010 007047                     PASS S2   A         SET (S2 S3) = X
0927 25241  010 011107                     PASS S3   B
0928 25242  305 003007           JSB                 FMPY      Z = Z * X
0929 25243  325 034747           JMP                 EXIT2     EXIT ROUTINE
0930                    *
0931                    *
0932 25244  000 075707  TANH2              DEC  P    P         SET RETURN ADDRESS
0933 25245  010 006747                     PASS      A         TEST X
0934 25246  327 112542           JMP  CNDX AL15      NEG       JUMP IF X < 0
0935 25247  356 176147           IMM       CMHI A    277B      SET (A B) = 1.0
0936 25250  353 172207           IMM       CMLO B    375B
0937 25251  227 174707  EXIT3    READ      INC  PNM  P         START READ
0938 25252  370 036747           RTN                           RETURN
0939                    *
0940                    *
0941 25253  355 176147  NEG      IMM       CMHI A    177B      SET (A B) = -1.0
0942 25254  006 036207                     ZERO B
0943 25255  227 174707           READ      INC  PNM  P         START READ
0944 25256  370 036747           RTN                           RETURN
0945                    *
0946                    *
0947                    ****************************************************************
```

```
0949                    *   DPOLY EVALUATES A RATIONAL FORM USING
0950                    *   TRIPLE-PRECISION OPERATIONS.
0951                    *
0952                    *   CALLING SEQUENCE:
0953                    *
0954                    *        OPCODE
0955                    *        DEF *+6    OR    OCT FLAGS+100000B
0956                    *        DEF <RESULT>
0957                    *        DEF <ARG>
0958                    *        DEF <COEFS>
0959                    *        DEF <M>            ORDER OF NUMERATOR
0960                    *        DEF <N>            ORDER OF DENOMINATOR
0961                    *
0962                    *   THE CONSTANTS SHOULD BE ARRANGED FROM HIGHEST TO
0963                    *   LOWEST ORDER, NUMERATOR THEN DENOMINATOR.  THE
0964                    *   FIRST (I.E. HIGHEST-ORDER) DENOMINATOR COEFFICIENT
0965                    *   IS ASSUMED TO BE 1.0 AND MUST NOT BE IN THE LIST.
0966                    *
0967                    *   THERE ARE M+1 NUMERATOR COEFFICIENTS AND N
0968                    *   DENOMINATOR COEFFICIENTS.  IF N=0, TRNL IS A
0969                    *   POLYNOMIAL EVALUATOR.  M MUST BE GREATER THAN ZERO.
0970                    *
0971                    *   THE FLAGS ARE INTERPRETED AS FOLLOWS:
0972                    *
0973                    *       BIT 0:  IF 0, MULTIPLY NUM BY X.
0974                    *       BIT 14: IF 1, SUBTRACT NUM FROM DENOM.  (N>0)
0975                    *       BIT 15: IF 1, EVALUATE IN X**2 & ENABLE FLAGS.
0976                    *
0977                    ********************************************************
0978                    *
0979                    *        ENTRY.  SET A = FLAGS, INITIALIZE.
0980                    *
0981 25257  010 000157 DPOLY          STFL PASS A     TAB        GET FLAG WORD.
0982 25260  327 113102           JMP  CNDX AL15       TRNL1      IF ENABLED.
0983 25261  344 000156           IMM  CLFL HIGH A     0          ELSE SET DEFAULT.
0984 25262  000 033613 TRNL1          COV  DEC  X     M          X=ADDR INST
0985 25263  007 175547                INC  SP    P     SP=INST+3
0986 25264  353 166507           IMM        CMLO L     373B       SET L = 4
0987 25265  003 067647                     ADD  Y     SP         SET Y = INST+7
0988 25266  305 077007           JSB                   RDPARAM    READ ARG
0989 25267  305 073447           JSB                   RD14
0990 25270  010 001147                     PASS S4    TAB
0991 25271  334 054242           JMP  CNDX FLAG RJS   TRNL1A     IF NO X**2
0992 25272  305 074347           JSB                   BOXIN14C   SQUARE X
0993 25273  305 074547           JSB                   BOXIN14A
0994 25274  010 041347                     PASS S8    S1         COPY X
0995 25275  010 043407                     PASS S9    S2
0996 25276  010 045447                     PASS S10   S3
0997 25277  010 047507                     PASS S11   S4
0998 25300  305 077007           JSB                   RDPARAM    READ P[M]
0999 25301  305 074007           JSB                   RD57
1000 25302  305 075707           JSB                   BOXOUT14   SAVE XSQ
1001 25303  340 114607           IMM        LOW  IRCM  046B       FOR XSQ*P[M]
1002 25304  325 054407           JMP                   TRNL1B
```

```
1004                    *              COMPUTE NUMERATOR.
1005                    *
1006 25305  305 074347  TRNL1A   JSB                    BOXIN14C  SEND X TO BOX
1007 25306  305 077007           JSB                    RDPARAM   READ P[M]
1008 25307  305 074007           JSB                    RD57
1009 25310  305 075047  TRNL1B   JSB                    BOXIN57A  P[M]
1010 25311  305 077007           JSB                    RDPARAM   READ M
1011 25312  010 001747                    PASS S         TAB       S=M
1012 25313  305 056147           JSB                    LOOPADD   FINISH NUMERATOR
1013 25314  227 174707           READ       INC  PNM    P         READ Q[N-1]
1014 25315  305 074007           JSB                    RD57
1015                    *
1016                    *         CONDITIONALLY MULTIPLY BY X.
1017                    *
1018 25316  340 114647           IMM        LOW  M      046B      FOR X*NUMERATOR
1019 25317  010 006747                    PASS        A         TEST BIT 0 FLAG.
1020 25320  301 175342           JSB  CNDX AL0  RJS    BOXIN811  IF A<0>=0, MPY
1021 25321  305 077007           JSB                    RDPARAM   READ N
1022 25322  010 001747                    PASS S         TAB       S=N.
1023 25323  320 015702           JMP  CNDX ALZ          NZERO     IF N=0.
1024 25324  305 076147           JSB                    BXOUT811  SAVE NUMERATOR.
1025                    *
1026                    *         EVALUATE DENOMINATOR AND DIVIDE.
1027                    *
1028 25325  340 004647           IMM        LOW  M      002B      FOR X+Q[N-1]
1029 25326  305 074407           JSB                    BOXIN14   X
1030 25327  305 075047           JSB                    BOXIN57A  DO X+Q[N-1]
1031 25330  305 056407           JSB                    LOOP3     FINISH DENOMINATOR
1032 25331  340 064647           IMM        LOW  M      032B      FOR DENOM-NUM
1033 25332  001 106747                    DBLS        A         TEST BIT 14 FLAG.
1034 25333  307 135342           JSB  CNDX AL15         BOXIN811  IF A<14>, DO IT.
1035 25334  340 154647           IMM        LOW  M      066B      FOR NUM/DENOM
1036 25335  305 075347           JSB                    BOXIN811  DIVIDE
1037                    *
1038                    *         STORE & EXIT.
1039                    *
1040 25336  007 171547  NZERO               INC  SP     X         SET SP=INST+2
1041 25337  007 167556               CLFL INC  SP     SP
1042 25340  325 067047           JMP                    EXITB     WRITE & EXIT
1043                    *
1044                    *         MAIN LOOP:  ACC _ CONST + X * ACC
1045                    *
1046 25341  340 114647  LOOP     IMM        LOW  M      046B      FOR X*ACC
1047 25342  305 074407           JSB                    BOXIN14   DO IT
1048 25343  227 174707  LOOPADD  READ       INC  PNM    P         READ NEXT COEF
1049 25344  305 074007           JSB                    RD57
1050 25345  323 156342           JMP  CNDX HOI  RJS    LOOP2     INT CHECK
1051 25346  336 037342           JMP  CNDX NSNG         INT
1052 25347  305 074747  LOOP2    JSB                    BOXIN57   CONST+X*ACC
1053 25350  000 077747  LOOP3              DEC  S      S         LOOP COUNTER
1054 25351  320 056042           JMP  CNDX ALZ  RJS    LOOP
1055 25352  370 036747           PTN
```

```
1057                    *       /CMPT MULTIPLES A 4-WORD NUMBER
1058                    *       BY A CONSTANT (4/PI OR 2/LN(2)) AND
1059                    *       SUBTRACTS THE NEAREST EVEN INTEGER FROM THE
1060                    *       RESULT.  IF CANCELLATION OCCURS WHICH WOULD
1061                    *       AFFECT THE CALLER'S RESULT, ADDITIONAL
1062                    *       PRECISION IS USED.
1063                    *
1064                    *       CALLING SEQUENCE:
1065                    *
1066                    *            <A = FLAGS>
1067                    *            OPCODE
1068                    *            DEF <RESULT>
1069                    *            DEF <CONST>
1070                    *            DEF <ARG>
1071                    *            <ERROR RETURN>
1072                    *            <NORMAL RETURN>  B=N LOWER.
1073                    *
1074                    *       THE CONSTANTS ARE THE 4-WORD VERSION AND THE
1075                    *       THE 4-WORD VALUE WHICH IS THE CONSTANT MINUS
1076                    *       THE FIRST 28 BITS OF THE CONSTANT.
1077                    *
1078                    *       THE FLAGS ARE:   EXP: -2   TANH: -1   TAN:  0
1079                    *                        SIN: 4,8  COS: 2,6
1080                    *
1081                    *       THE ERROR RETURN IS TAKEN IF N DOES NOT FIT
1082                    *       IN TWO WORDS.
1083                    *
1084                    *       METHOD:  TWO METHODS ARE USED, DEPENDING ON
1085                    *       THE AMOUNT OF CANCELLATION.  IF THE ARGUMENT
1086                    *       IS IN THE RANGE [-16,16) OR TANH IS INDICATED,
1087                    *       IT IS MULTIPLIED BY C AND THE NEAREST EVEN
1088                    *       INTEGER (N) IS FOUND AND SUBTRACTED.  IF THE
1089                    *       CANCELLATION DOES NOT EXCEED ABOUT 4 BITS OR
1090                    *       THE RESULT WILL BE FOR COS, TANH OR EXP, THE
1091                    *       RESULT IS RETURNED.
1092                    *       IF TOO MUCH CANCELLATION OCCURS OR THE
1093                    *       ARGUMENT IS OUTSIDE [-8,8), THE ARGUMENT
1094                    *       AND THE CONSTANT ARE BROKEN INTO TWO PARTS
1095                    *       AND THE PARTIAL PRODUCTS ARE TAKEN.  SINCE
1096                    *       THE FIRST PARTIAL PRODUCT IS EXACT, N MAY BE
1097                    *       SUBTRACTED SAFELY AND THE SIGNIFICANT BITS IN
1098                    *       THE OTHER PRODUCTS ARE NOT LOST IN THE SUM.
```

```
1100                    *       READ X AND C, DETERMINE IF EASY OR HARD.
1101                    *
1102 25353  000 033616  /CMRT         CLFL DEC  X     M          X=ADDR INST
1103 25354  010 075547                   PASS SP    P          SP=INST+2
1104 25355  305 077007            JSB                RDPARAM    START READ C
1105 25356  305 074007            JSB                RD57       FINISH READ C
1106 25357  007 167647                 INC  Y     SP         Y=ERROR RETURN
1107 25360  010 033747                 PASS S     M          S=ADDR C[4]
1108 25361  305 077007            JSB                RDPARAM    READ X
1109 25362  305 073447            JSB                RD14
1110 25363  010 001153                 COV  PASS S4    TAB
1111 25364  321 117602            JMP  CNDX AL0      CMRT3      IN [-.5,+.5)
1112 25365  010 006147                 PASS A     A          TANH ?
1113 25366  320 117602            JMP  CNDX ONES     CMRT3      YES, EASY.
1114 25367  340 000507            IMM       LOW  L     0          L=8-BIT MASK.
1115 25370  014 047547                 SANL SP    S4         EXPONENT*2
1116 25371  343 160507            IMM       LOW  L     370B       -8
1117 25372  003 067547                 ADD  SP    SP         EXPONENT*2-8
1118 25373  327 161542            JMP  CNDX AL15 RJS CMRT6      OUTSIDE [-8,+8)
1119                    *
1120                    *       EASY WAY. DO X*C-N.
1121                    *
1122 25374  305 074347  CMRT3     JSB                BOXIN14C   X IN X*C
1123 25375  010 077707                 PASS P     S          P=ADDR C[4]
1124 25376  305 075107            JSB                BOXIN57B   C
1125 25377  305 076147            JSB                BXOUT811   SAVE X*C
1126 25400  341 024607            IMM       LOW  IRCM 112B       FIX
1127 25401  343 174511            IMM  MPP2 LOW  L     376B       L=1-BIT MASK
1128 25402  305 076447            JSB                WAIT+1
1129 25403  010 020232                 MPP1 PASS B     MPPB       B=N LOWER
1130 25404  327 120302            JMP  CNDX AL15     CMRT4      NEAREST EVEN
1131 25405  267 110207            ENVE      INC  B     B
1132 25406  341 044607  CMRT4     IMM       LOW  IRCM 122B       FLOAT
1133 25407  012 010211                 MPP2 AND  B     B          N
1134 25410  010 010432                 MPP1 PASS MPPB  B
1135 25411  335 127542            JMP  CNDX OVFL     EXITD      IF OFL (TANH ONLY)
1136 25412  340 054647            IMM       LOW  M     026B       FOR X*C-N
1137 25413  305 075347            JSB                BOXIN811   DO IT.
```

```
1139                        *         DETERMINE IF CANCELLATION MATTERS.
1140                        *
1141 25414  010 006147                PASS A    A          CHECK FLAGS
1142 25415  327 125602          JMP   CNDX AL15      CMRT8      EXP, TANH: NO.
1143 25416  320 021142          JMP   CNDX ALZ       CMRT5      TAN: YES.
1144 25417  010 010507                PASS L    B          N
1145 25420  003 006164           R1   ADD  A    A          (N+(S:4,C:2))/2
1146 25421  010 006147                PASS A    A          SIN ?
1147 25422  321 125602          JMP   CNDX AL0       CMRT8      NO.
1148                        *
1149                        *         SEE WHETHER EXCESSIVE CANCELLATION.
1150                        *
1151 25423  010 047147  CMRT5          PASS S4   S4         X IN [-.5,+.5) ?
1152 25424  321 125602          JMP   CNDX AL0       CMRT8      YES, DONE.
1153 25425  340 000517          IMM   STFL LOW  L    0          L=8-BIT MASK
1154 25426  305 076147          JSB                BXOUT811
1155 25427  321 165602          JMP   CNDX AL0  RJS  CMRT8      OUT [-.5,+.5)
1156 25430  007 064507                OP1  L    S11        EXPONENT*2+2
1157 25431  004 166747                SUB       SP         2*(EXP DIF) - 10
1158 25432  327 125602          JMP   CNDX AL15      CMRT8      LOST < 5 BITS
1159                        *
1160                        *         HARD WAY.  DO (XU*CU-N) + (X-XU)*C * XU*CL .
1161                        *
1162 25433  340 104613  CMRT6    IMM   COV  LOW  IRCM 042B       FOR XU*CU
1163 25434  343 160511          IMM   MPP2 LOW  L    370B       MASK FOR XU,CU
1164 25435  010 040432                MPP1 PASS MPPB S1         XU
1165 25436  012 043356                CLFL AND  S8   S2
1166 25437  012 053507                     AND  S11  S6
1167 25440  010 056432                MPP1 PASS MPPB S8
1168 25441  006 037407                     ZERO S9
1169 25442  010 060432                MPP1 PASS MPPB S9
1170 25443  340 000507          IMM        LOW  L    0
1171 25444  014 047347                     SANL S8   S4
1172 25445  010 056432                MPP1 PASS MPPB S8
1173 25446  010 050432                MPP1 PASS MPPB S5         CU
1174 25447  227 176707          READ       INC  PNM  S          READ C[4]
1175 25450  010 064432                MPP1 PASS MPPB S11
1176 25451  010 060432                MPP1 PASS MPPB S9
1177 25452  014 001347                     SANL S8   TAB
1178 25453  010 056432                MPP1 PASS MPPB S8
1179 25454  343 060507          IMM        LOW  L    330B       L=-40
1180 25455  010 006147                PASS A    A
1181 25456  327 163002          JMP   CNDX AL15 RJS  *+2        IF SIN/COS/TAN, -40.
1182 25457  343 160507          IMM        LOW  L    370B       IF EXP/TANH, -8.
1183 25460  003 067547                ADD  SP   SP         CHECK EXPONENT.
1184 25461  327 167542          JMP   CNDX AL15 RJS  EXITD      IF OUT OF RANGE.
1185 25462  305 076147          JSB                BXOUT811  SAVE XU*CU
1186 25463  341 034607          IMM        LOW  IRCM 116B       FIX
1187 25464  343 174511          IMM   MPP2 LOW  L    376B       1-BIT MASK
1188 25465  305 076447          JSB                WAIT+1
1189 25466  010 021572                MPP1 PASS SP   MPPB       N UPPER
1190 25467  010 020232                MPP1 PASS B    MPPB       N LOWER
1191 25470  010 067547                     PASS SP   SP
1192 25471  327 123642          JMP   CNDX AL15      CMRT7      NEAREST EVEN
1193 25472  007 110207                     INC  B    B
1194 25473  321 063642          JMP   CNDX COUT RJS  CMRT7      PROPOGATE CARRY
```

```
1195 25474  267 167547          ENVE       INC  SP   SP
1196 25475  341 054607  CMRT7   IMM        LOW  IRCM 126B       FLOAT
1197 25476  012 010211               MPP2  AND  B    B
1198 25477  010 066432               MPP1  PASS MPPB SP
1199 25500  010 010432               MPP1  PASS MPPB B
1200 25501  335 127542          JMP  CNDX OVFL       EXITD      IF OFL
1201 25502  340 054647          IMM        LOW  M    026B       FOR XU*CU-N
1202 25503  305 075347          JSB                    BOXIN811
1203 25504  305 076147          JSB                    BXOUT811   SAVE
1204 25505  340 044607          IMM        LOW  IRCM 022B       FOR X-XU
1205 25506  343 160511          IMM  MPP2  LOW  L    370B       MASK FOR XU
1206 25507  305 074547          JSB                    BOXIN14A   X
1207 25510  010 040432               MPP1  PASS MPPB S1         XU
1208 25511  012 043047                     AND  S2   S2
1209 25512  010 042432               MPP1  PASS MPPB S2
1210 25513  006 037107                     ZERO S3
1211 25514  010 044432               MPP1  PASS MPPB S3
1212 25515  340 000507          IMM        LOW  L    0
1213 25516  014 047147                     SANL S4   S4
1214 25517  010 046432               MPP1  PASS MPPB S4
1215 25520  306 076402          JSB  CNDX MPP  RJS  WAIT
1216 25521  340 124607          IMM        LOW  IRCM 052B       FOR (X-XU)*C
1217 25522  305 075047          JSB                    BOXIN57A   PASS C, START.
1218 25523  230 074647          READ       PASS M    P          READ CL
1219 25524  305 074007          JSB                    RD57
1220 25525  340 014647          IMM        LOW  M    006B       ADD TO SUM
1221 25526  305 075347          JSB                    BOXIN811
1222 25527  305 076147          JSB                    BXOUT811   SAVE SUM
1223 25530  305 074347          JSB                    BOXIN14C   XU IN XU*CL
1224 25531  305 075047          JSB                    BOXIN57A   CL
1225 25532  340 014647          IMM        LOW  M    006B       ADD TO SUM
1226 25533  305 075347          JSB                    BOXIN811
1227 25534  007 173647  CMRT8                INC  Y    Y          SKIP ERROR RETURN.
1228 25535  325 067007          JMP                    EXITA      WRITE & EXIT.
```

```
1230                    *       /ATLG - COMMON CODE FOR .ATAN & .LOG .
1231                    *
1232                    *       /ATLG COMPUTES THE EXPRESSION X _ (1-X)/(1+X)
1233                    *
1234                    *       CALLING SEQUENCE:
1235                    *
1236                    *            OPCODE
1237                    *            DEF <X>        (DIRECT)
1238                    *
1239                    *
1240 25536  000 033607  /ATLG             DEC  X     M          X=ADDR INST
1241 25537  230 000647               READ  PASS M     TAB        READ X
1242 25540  010 075647                     PASS Y     P          Y=ERROR RETURN.
1243 25541  305 074007               JSB                RD57
1244 25542  306 076402               JSB  CNDX MPP  RJS  WAIT       WAIT FOR BOX.
1245 25543  340 044607               IMM       LOW  IRCM 022B       FOR 1-X
1246 25544  356 177011               IMM  MPP2 CMHI S1   277B       S1-S4 = 1.0
1247 25545  006 037047                     ZERO S2
1248 25546  006 037107                     ZERO S3
1249 25547  353 173147               IMM       CMLO S4   375B
1250 25550  305 074447               JSB                BOXIN14B
1251 25551  305 075047               JSB                BOXIN57A
1252 25552  340 004647               IMM       LOW  M    002B       FOR 1+X
1253 25553  305 076147               JSB                BXOUT811   SAVE 1-X
1254 25554  305 074447               JSB                BOXIN14B
1255 25555  305 075047               JSB                BOXIN57A
1256 25556  340 154656               IMM  CLFL LOW  M    066B       FOR (1-X)/(1+X)
1257 25557  305 075347               JSB                BOXIN811
1258                    *
1259                    *
1260                    *       COMMON CODE TO WRITE RESULT AND EXIT.
1261                    *
1262 25560  007 171547  EXITA              INC  SP    X          SP = ADDR ADDR RESULT.
1263 25561  305 077007  EXITB  JSB                RDPARAM    RESOLVE IND
1264 25562  314 076142  EXITC  JSB  CNDX FLAG RJS  BXOUT811   COPY RESULT
1265 25563  210 056036         WRTE MPCK PASS TAB  S8         WRITE RESULT
1266 25564  007 133707                   INC  P     M
1267 25565  007 174707                   INC  PNM   P
1268 25566  210 060036         WRTE MPCK PASS TAB  S9
1269 25567  007 174707                   INC  PNM   P
1270 25570  210 062036         WRTE MPCK PASS TAB  S10
1271 25571  010 074647                   PASS M     P
1272 25572  210 064036         WRTE MPCK PASS TAB  S11
1273 25573  227 172707  EXITD  READ       INC  PNM   Y          PREPARE TO EXIT.
1274 25574  375 141642         RTN  CNDX OVFL RJS            RTN IF NO OVFL
1275 25575  230 036777         READ IOFF                     TURN OF ALL BUT MP.
1276 25576  323 170002         JMP  CNDX HOI  RJS  *+2       RTN IF NO MP INT
1277 25577  230 036753         READ COV                      COV IF MP AND OVFL
1278 25600  366 036143         RTN  ION  ZERO A              RTN, CLR A, INTS ON
```

```
1280                    *         .FPWR & .TPWR - EXPONENTIATION.
1281                    *
1282                    *         .FPWR & .TPWR RAISE A 2 OR 4-WORD NUMBER TO A POSITIVE
1283                    *         POWER IN THE RANGE [2,65535].  FOR COMPATABILITY WITH
1284                    *         THE SOFTWARE VERSIONS, POWERS ABOVE 32768 SHOULD NOT
1285                    *         BE USED.
1286                    *
1287                    *         CALLING SEQUENCE:
1288                    *
1289                    *              LDA POWER
1290                    *              OPCODE
1291                    *              DEF <ARG>
1292                    *
1293                    *         FOR .FPWR, THE RESULT IS RETURNED IN (A,B).  FOR .TPWR,
1294                    *         THE RESULT OVERWRITES THE ARGUMENT.
1295                    *
1296                    *         OVERFLOW: MAX POS NUMBER RETURNED, OFL SET.
1297                    *         UNDERFLOW: ZERO RETURNED, OFL CLEAR.
1298                    *         OTHERWISE: OFL CLEAR.
1299                    *
1300                    ************************************************************
1301                    *
1302                    *         INITIALIZATION.  START FIRST SQUARE.
1303                    *
1304 25601  340 105516  .TPWR     IMM  CLFL LOW  S11  042B      4-WD MPY OPCODE.
1305 25602  010 075547                PASS SP   P    SKIP RESULT ADDR.
1306 25603  325 070307           JMP                *+3
1307 25604  340 101517  .FPWR     IMM  STFL LOW  S11  040B      2-WD MPY OPCODE.
1308 25605  010 033547                PASS SP   M    SP = ADDR ADDR ARG.
1309 25606  000 033613           COV  DEC  X    M    X = RESTART ADDR.
1310 25607  305 077007           JSB                RDPARAM  READ ADDR.
1311 25610  305 073447           JSB                RD14
1312 25611  010 064607                PASS IRCM S11       SEND BOX OPCODE.
1313 25612  010 001151           MPP2 PASS S4   TAB       (START BOX)
1314 25613  334 070702           JMP  CNDX FLAG RJS  FPWR1     IF 4-WORD.
1315 25614  010 041107                PASS S3   S1    SET UP 2ND ARG 2-WD.
1316 25615  010 043147                PASS S4   S2
1317 25616  305 074547  FPWR1    JSB                BOXIN14A  4-WD: 1ST / 2-WD: BOTH.
1318 25617  314 074542           JSB  CNDX FLAG RJS  BOXIN14A  2ND ARG 4-WD.
1319 25620  010 067647                PASS Y    SP    Y = RETURN ADDR.
1320 25621  010 041007                PASS S1   S1    IF BASE=0, RETURN ZERO.
1321 25622  320 033202           JMP  CNDX ALZ       FPWR5
1322 25623  353 146507           IMM       CMLO L    363B      L=14B
1323 25624  010 165507                     IOR  S11  S11       SQUARE OPCODE.
```

```
1325                        *      LEFT-JUSTIFY POWER, COUNT # BITS.
1326                        *
1327 25625  010 007247  FPWRNORM           PASS S6   A          SAVE POWER FOR INT.
1328 25626  320 033202           JMP  CNDX ALZ       FPWR5      IF POWER=0, RETURN (OPND**2)
1329 25627  010 006207                     PASS B    A          SET UP NORMALIZE.
1330 25630  070 010224           LGS  R1   PASS B    B          UNNORMALIZE POWER
1331 25631  070 010224           LGS  R1   PASS B    B
1332 25632  340 000567           IMM  RPT  LOW  CNTR 0          NORMALIZE.
1333 25633  110 036762           NRM  L1   PASS
1334 25634  353 136525           IMM  DCNT CMLO L    357B       L=16., CNTR=#BITS-18
1335 25635  003 027207                     ADD  S5   CNTR       S5 = # BITS - 2.
1336 25636  010 052147                     PASS A    S6         RESTORE A.
1337 25637  340 114647           IMM       LOW  M    046B       4-WD ACC*ARG OP.
1338 25640  001 110207                     DBLS B    B          SKIP SIGN (=0).
1339                        *
1340                        * THE FOLLWING CHECK IS NECESSARY BECAUSE DMA ACTIVITY MAY
1341                        * CAUSE THE NORMALIZE OP TO TERMINATE INCORRECTLY!!!
1342 25641  327 132302           JMP  CNDX AL15      FPWR3      IF MSB SET, NORM WAS OK
1343 25642  325 071247           JMP                 FPWRNORM   NO, GO TRY AGAIN
1344                        *
1345                        *      LOOP.  SQUARE, THEN MPY BY ARG IF EXP BIT SET.
1346                        *
1347 25643  306 076402  FPWR2     JSB  CNDX MPP  RJS  WAIT       SQUARE ACC.
1348 25644  010 064607                     PASS IRCM S11
1349 25645  340 114651           IMM  MPP2 LOW  M    046B       4-WD ACC*ARG OP.
1350 25646  001 110207  FPWR3               DBLS B    B          EXAMINE NEXT EXP BIT.
1351 25647  327 172742           JMP  CNDX AL15 RJS  FPWR4      IF NOT SET.
1352 25650  314 074402           JSB  CNDX FLAG RJS  BOXIN14    USE "BOXIN14" FOR 4-WD.
1353 25651  334 072742           JMP  CNDX FLAG RJS  FPWR4
1354 25652  306 076402           JSB  CNDX MPP  RJS  WAIT       DO 2-WD INLINE.
1355 25653  340 110607           IMM       LOW  IRCM 044B
1356 25654  010 036751                MPP2
1357 25655  010 040432                MPP1 PASS MPPB S1
1358 25656  010 042432                MPP1 PASS MPPB S2
1359 25657  000 051207  FPWR4               DEC  S5   S5         DEC LOOP CNT (#BITS-2)
1360 25660  327 133202           JMP  CNDX AL15      FPWR5      DONE.
1361 25661  323 172142           JMP  CNDX HOI  RJS  FPWR2      LOOP IF NO INTERRUPT.
1362 25662  336 072142           JMP  CNDX NSNG RJS  FPWR2
1363 25663  325 077347           JMP                 INT        PROCESS. (RESTART)
1364                        *
1365                        *      RETURN RESULT.
1366                        *
1367 25664  334 067002  FPWR5     JMP  CNDX FLAG RJS  EXITA      IF 4-WD, WRITE RESULT.
1368 25665  007 172707                     INC  PNM  Y          SET UP RTN ADDR
1369 25666  306 076402           JSB  CNDX MPP  RJS  WAIT       2-WD.  WAIT FOR BOX.
1370 25667  230 020172           READ MPP1 PASS A    MPPB       (A,B) = RESULT.
1371 25670  370 020232           RTN  MPP1 PASS B    MPPB       RETURN
```

```
1373                        *       COMMON UTILITY ROUTINES.
1374                        ********************************************************
1375                        *       RD14 - FINISH READING TO S1-S4.
1376                        *
1377 25671  010 001007  RD14            PASS S1   TAB
1378 25672  007 133707                  INC  P    M
1379 25673  227 174707          READ    INC  PNM  P
1380 25674  007 174707                  INC  PNM  P
1381 25675  230 001047          READ    PASS S2   TAB
1382 25676  007 174707                  INC  PNM  P
1383 25677  230 001100          READ RTN PASS S3   TAB
1384                        ********************************************************
1385                        *       RD57 - FINISH READING TO S5-S7.
1386                        *
1387 25700  010 001207  RD57            PASS S5   TAB
1388 25701  007 133707                  INC  P    M
1389 25702  227 174707          READ    INC  PNM  P
1390 25703  007 174707                  INC  PNM  P
1391 25704  230 001247          READ    PASS S6   TAB
1392 25705  007 174707                  INC  PNM  P
1393 25706  230 001300          READ RTN PASS S7   TAB
1394                        ********************************************************
1395                        *       BOXIN14 - START BOX: OP=M, OPND=S1-14.
1396                        *
1397 25707  340 104647  BOXIN14C IMM       LOW  M    042B
1398 25710  306 076402  BOXIN14  JSB  CNDX MPP  RJS  WAIT
1399 25711  010 032607  BOXIN14B           PASS IRCM M
1400 25712  010 036751               MPP2
1401 25713  010 040432  BOXIN14A     MPP1 PASS MPPB S1
1402 25714  010 042432               MPP1 PASS MPPB S2
1403 25715  010 044432               MPP1 PASS MPPB S3
1404 25716  370 046432          RTN  MPP1 PASS MPPB S4
1405                        ********************************************************
1406                        *       BOXIN57 - ISSUE MPP2, SEND S5-S7 & (M) TO BOX.
1407                        *
1408 25717  306 076402  BOXIN57  JSB  CNDX MPP  RJS  WAIT
1409 25720  340 014607          IMM       LOW  IRCM 006B      SET UP ADD.
1410 25721  000 075711  BOXIN57A     MPP2 DEC  P    P
1411 25722  010 050432  BOXIN57B     MPP1 PASS MPPB S5
1412 25723  227 174707          READ    INC  PNM  P
1413 25724  010 052432               MPP1 PASS MPPB S6
1414 25725  010 054432               MPP1 PASS MPPB S7
1415 25726  370 000432          RTN  MPP1 PASS MPPB TAB
1416                        ********************************************************
1417                        *       BOXIN811 - START BOX: OP=M, OPND=S8-S11.
1418                        *
1419 25727  306 076402  BOXIN811 JSB  CNDX MPP  RJS  WAIT
1420 25730  010 032607                  PASS IRCM M
1421 25731  010 036751               MPP2
1422 25732  010 056432               MPP1 PASS MPPB S8
1423 25733  010 060432               MPP1 PASS MPPB S9
1424 25734  010 062432               MPP1 PASS MPPB S10
1425 25735  370 064432          RTN  MPP1 PASS MPPB S11
```

```
1427                        *         BOXOUT14 - COPY RESULT TO S1-S4.
1428                        *
1429 25736  306 076402  BOXOUT14 JSB  CNDX MPP  RJS  WAIT
1430 25737  010 021032            MPP1 PASS S1   MPPB
1431 25740  010 021072            MPP1 PASS S2   MPPB
1432 25741  010 021132            MPP1 PASS S3   MPPB
1433 25742  370 021172       RTN  MPP1 PASS S4   MPPB
1434                        ***************************************************
1435                        *         BXOUT811 - COPY RESULT TO S8-S11.
1436                        *
1437 25743  306 076402  BXOUT811 JSB  CNDX MPP  RJS  WAIT
1438 25744  010 021372            MPP1 PASS S8   MPPB
1439 25745  010 021432            MPP1 PASS S9   MPPB
1440 25746  010 021472            MPP1 PASS S10  MPPB
1441 25747  370 021532       RTN  MPP1 PASS S11  MPPB
1442                        ***************************************************
1443                        *         BOX WAIT ROUTINE.
1444                        *
1445 25750  366 000402  WAIT      RTN  CNDX MPP                  RETURN IF DONE.
1446 25751  340 100547           IMM       LOW  CNTR 040B       COUNTER=32
1447 25752  366 002002  WAIT2     RTN  CNDX MPP                  RETURN IF DONE
1448 25753  010 036765           DCNT                           DECR COUNTER
1449 25754  366 000742           RTN  CNDX MPP                  RETURN IF DONE
1450 25755  326 176502           JMP  CNDX CNT8 RJS  WAIT2      LOOP 32 TIMES
1451 25756  007 173707                     INC  P    Y          ** BOX HUNG **
1452 25757  325 003547           JMP                 BAILOUT    USE SIS BOX HUNG CODE.
1453                        ***************************************************
1454                        *         INDIRECT RESOLVER.
1455                        *
1456 25760  230 066647  RDPARAM  READ      PASS M    SP
1457 25761  007 167547                     INC  SP   SP
1458 25762  230 000647  INDIRECT READ      PASS M    TAB        NEXT ADDRESS
1459 25763  367 140002           RTN  CNDX AL15 RJS            RETURN IF M POS.
1460 25764  323 177102           JMP  CNDX HOI  RJS  INDIRECT   LOOP IF NO INT
1461 25765  230 036747           READ                           MUST REREAD
1462 25766  336 077102           JMP  CNDX NSNG RJS  INDIRECT   IGNORE SNGL STEP
1463                        ***************************************************
1464                        *         INTERRUPT HANDLING.
1465                        *
1466 25767  227 170732  INT       READ MPP1 INC  PNM  X          REDO INSTRUCTION.
1467 25770  010 036753            COV                           CLEAR OVERFLOW
1468 25771  320 000007           JMP                 FETCH
1469                        *
1470                        *
1471                                 END

END OF PASS 2: NO ERRORS
```

SYMBOLS=0110 REFERENCES=0286 SOURCE LINES=1471

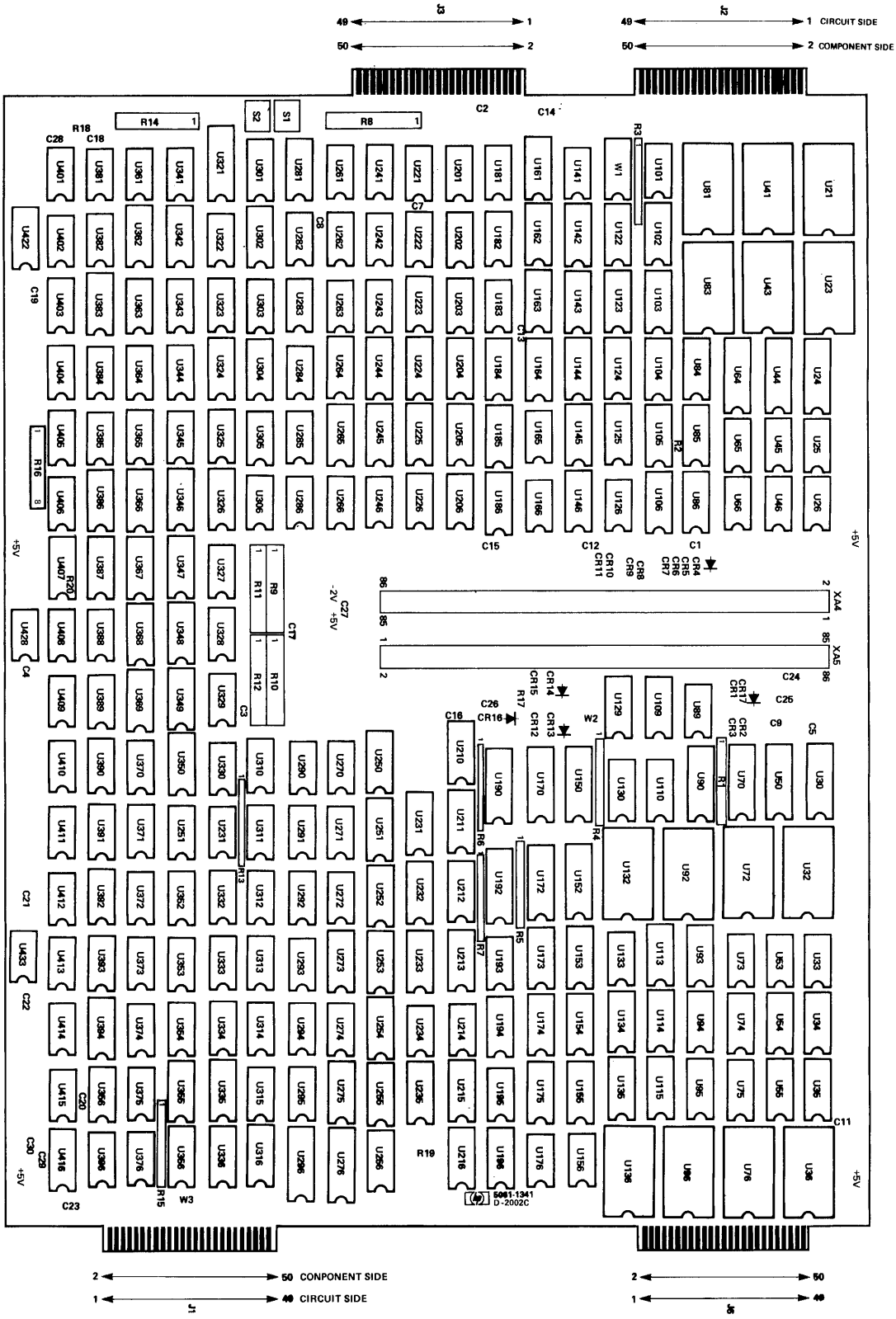
```
CMRT7     1196  1192  1194

CMRT8     1227  1142  1147  1152  1155  1158

COS       0399  0034

COSAG     0414  **NOT REFERENCED**

DISPLAY   0715  0706  0709  0712

DPOLY     0981  0039

ERRET     0386  0489  0567  0789  0866

ESISREVN  0028  0715

EVEN      0736  **NOT REFERENCED**

EXIT1     0375  0370  0479  0481  0559  0584

EXIT2     0678  0670  0675  0898  0929

EXIT3     0937  0729

EXITA     1262  1228  1367

EXITB     1263  1042

EXITC     1264  **NOT REFERENCED**

EXITD     1273  1135  1184  1200

EXP       0798  0036  0887

EXPERR    0861  0824  0827  0853

EXPNXT    0829  0822

FADA2     0219  0522  0628  0896

FADD      0177  0342  0352  0366  0430  0436  0440  0459  0467  0473
          0549  0553  0644  0654  0668  0763  0774  0779  0844  0849
          0910  0925

FDIV7     0189  0523  0629  0669  0769  0778  0846  0897  0917

FDVAC2    0233  0349  0374  0541  0619  0651

FETCH     0026  0142  1468

FLUN      0054  0502  0731

FMPY      0111  0359  0369  0423  0452  0476  0558  0583  0661  0834
          0928

FOPI      0273  0328  0406
```

```
FPWR1       1317  1314
FPWR2       1347  1361  1362
FPWR3       1350  1342
FPWR4       1359  1351  1353
FPWR5       1367  1321  1328  1360
FPWRNORM    1327  1343
FSUB        0261  0624
FSUB2       0165  0534  0673  0837
HORI        0027  0155
INDIRECT    1458  1460  1462
INT         1466  1051  1363
INTRT1      0151  0131
LGNXT       0509  0505
LOGERR      0562  0500  0501
LOOP        1046  1054
LOOP1       0132  0135  0151
LOOP2       1052  1050
LOOP3       1053  1031
LOOPADD     1048  1012
NEG         0941  0934
NEGATE      0257  0480  0676
NEGCHK      0819  **NOT REFERENCED**
NZERO       1040  1023
ODD         0749  0734
PDONE       0097  0071
PNEXT1      0078  0075
PNEXT2      0091  0086  0088
POS         0602  0599  0600
```

```
POSCHK     0824  0817

PWR2       0069  0781  0850

RD14       1377  0989  1109  1311

RD57       1387  0999  1008  1014  1049  1105  1219  1243

RDPARAM    1456  0988  0998  1007  1010  1021  1104  1108  1263  1310

REDUCE     0284  0809

RNXT       0308  0304  0306

SELFTEST   0698  0045

SIN        0403  0035  0400

SINAG      0443  0412

SINERR     0484  0407

SQRERR     0784  0730

SQRT       0727  0031  0704

TAN        0326  0030

TANERR     0381  0329

TANH       0875  0038

TANH1      0900  0860

TANH2      0932  0883

TRNL1      0984  0982

TRNL1A     1006  0991

TRNL1B     1009  1002

VMPY       0245  0431  0437  0460  0468  0550

WAIT       1445  1128  1188  1215  1244  1347  1354  1369  1398  1408
           1419  1429  1437

WAIT1      0130  0069  0111  0165  0177  0189  0201  0219  0233  0245
           0261  0294  0300  0311  0339  0375  0420  0449  0514  0519
           0524  0621  0625  0630  0641  0678  0766  0775  0893

WAIT2      1447  1450

XSQ        0201  0332  0411  0829

ZERO1      0685  0598
```



```
ZERO2      0856  0821
```



E/F Series CPU Assembly 5061-1341/1400

CPU Assembly Parts List 5061-1341/1400 (Sheet 1 of 6)

REFERENCE DESIGNATOR (FIRST SIX)	PART DESCRIPTION	PARENT OPTION	PART NUMBER	COMP. OPTION	LOC	QUANTITY PER
C3-5,9,11-23,26	CAP 1.0UF 20%		0160-0127		U	18
C24,25,27-30	CAP .01UF		0160-2055		U	6
C8	CAP 100PF 5%		0160-2204		U	1
	SPCR #6-32 BR		0380-0593		U	4
R18	RES 1K 1%.125		0757-0280		U	1
R17	RES 100 1%.125		0757-0401		U	1
	SOCKET 16 DIP LO		1200-0482		U	4
	SOCKET 24 PIN		1200-0541		U	6
	SOCKET PC SINGLE		1251-1556		U	9
XA4,XA5	CONN PC2X43.156D		1251-4139		U	2
W3,1	JMPR PLUG .3"C-C		1258-0124		U	2
RC	RES NET 8X10K		1810-0055		U	1
R1-4,8,13-16	RES NET 8X500		1810-0132		U	9
R6,7,9-12	RES NET 8X200		1810-0163		U	6
C1,2	CAP NTWK 220PF		1810-0258		U	2
U408	IC DIGITAL		1813-0081		U	1
U312	IC-HYBRID CKT		1813-0085		U	1
U255	I.C. ROM 4X256		1816-0420		3	1

CPU Assembly Parts List 5061-1341/1400 (Sheet 2 of 6)

REFERENCE DESIGNATOR (FIRST SIX)	PART DESCRIPTION	PARENT OPTION	PART NUMBER	COMP. OPTION	LOC	QUANTITY PER
U132,72,92,132			1820-0999			
U194	IC SN74S158N		1820-1015		U	1
U136,76,96,136	IC SN74198N		1820-1032		U	4
U153	IC SN74S174N		1820-1076		U	1
U210-212,231,232, 250-252,302,346-349 367-369,386,387	IC 8T13B		1820-1080		U	18
U304,416	IC SN74109N		1820-1116		U	2
U93	IC SN74S133N		1820-1130		U	1
U234,314,384,404	IC SN74S51N		1820-1158		U	4
U292	IC SN7428N		1820-1184		U	1
U213,233	IC SN74S175N		1820-1191		U	2
U362	IC SN74LS175N		1820-1195		U	1
U186,266,272,310,344, 371,374,415	IC SN74LS00N		1820-1197		U	8
U183,284,361	IC SN74LS04N		1820-1199		U	3
U322,341,345	IC SN74LS08N		1820-1201		U	3
U293,375	IC SN74LS10N		1820-1202		U	2
U365,372	IC SN74LS11N		1820-1203		U	2
U313	IC SN74LS20N		1820-1204		U	1
	IC SN74LS21N		1820-1205		U	1

REFERENCE DESIGNATOR (FIRST SIX)	PART DESCRIPTION	PARENT OPTION	PART NUMBER	COMP. OPTION	L O C	QUANTITY PER
MADE	FROM 1816-0782		1816-0420			
	IC PROM 4X256		1816-0782		L	1
U114,1	IC RAM 16X4 15,134,135		1816-1005		U	4
U65,126	IC SN74S00N ,186,193,223, 243,262,283,290,295 331,388		1820-0681		U	12
U33,53	IC SN74S04N ,73,165,181, 271,306,334,373, 390,401		1820-0683		U	11
U327-329	IC SN74S05N		1820-0684		U	3
U46,182	IC SN74S10N ,222,291,354, 402,405,409		1820-0685		U	8
U99,241	IC SN74S11N ,305,333		1820-0686		U	4
U176,201	IC SN74S20N ,206,245, 382,406		1820-0688		U	6
U330	IC SN74S22N		1820-0689		U	1
U221	IC SN74S40N		1820-0690		U	1
U274,364	IC SN74S64N ,392,412,414		1820-0691		U	5
U391,411	IC SN74S65N		1820-0692		U	2
U286,332	IC SN74S74N ,350,403,428		1820-0693		U	5
U185	IC 9334DC		1820-0833		U	1
U34,35, 94,95	IC SN74S153N 54,55,74,75,		1820-0998		U	8
	IC SN74S181N		1820-0999		U	4

CPU Assembly Parts List 5061 -1341/1400 (Sheet 4 of 6)

REFERENCE DESIGNATOR (FIRST SIX)	PART DESCRIPTION	PARENT OPTION	PART NUMBER	COMP. OPTION	LOC	QUANTITY PER
U294			1820-1205			
U295	IC SN74LS30N		1820-1207		U	1
U352	IC SN74LS32N		1820-1208		U	1
U410	IC SN74LS51N		1820-1210		U	1
U214	IC SN74LS86N		1820-1211		U	1
U324,326,366	IC SN74LS112N		1820-1212		U	3
U301	IC SN74LS138N		1820-1216		U	1
U184,185,104,105	IC SN74LS151N		1820-1217		U	4
U184,204,205,224,225 244,245,264,265,273	IC SN74S138N		1820-1240		U	10
U124,125,144,145,164 ,342	IC SN74LS153N		1820-1244		U	6
U281	IC DM8096N		1820-1254		U	1
U275	IC DM8098N		1820-1255		U	1
U166,133,156,311,381,	IC SN74S260N		1820-1275		U	5
U161	IC 74LS194		1820-1276		U	1
U215,235	IC 74LS191		1820-1278		U	2
U394	IC SN74LS109N		1820-1282		U	1
U303,323,343,393,413	IC SN74LS54N		1820-1285		U	5
	IC SN74S257N		1820-1301		U	4

CPU Assembly Parts List 5061-1341/1400 (Sheet 5 of 6)

REFERENCE DESIGNATOR (FIRST SIX)	PART DESCRIPTION	PARENT OPTION	PART NUMBER	COMP. OPTION	LOC	QUANTITY PER
U109,110,129,130			1820-1301			
U113	IC SN74S182N		1820-1305		U	1
U385,389	IC SN74S132N		1820-1307		U	2
U154,155,174,175	IC SN74S258N		1820-1309		U	4
U254,316,335,336,355,358	IC SN74S151N		1820-1319		U	6
U45	IC SN74S30N		1820-1323		U	1
U26,165,202,261,315,370	IC SN74S08N		1820-1367		U	6
U325,376	IC SN74LS13N		1820-1415		U	2
U195,270,282	IC SN74LS14N		1820-1416		U	3
U363,396	IC SN74LS132N		1820-1425		U	2
U102,122,142,162	IC SN74LS670N		1820-1447		U	4
U25,203,226,242,263,285,351,353	IC SN74S32N		1820-1449		U	8
U103,123,143,163	IC SN74S163N		1820-1453		U	4
U30,50,296	IC SN74273N		1820-1461		U	3
U253	IC SN74LS157N		1820-1470		U	1
U321	IC SN74S240N		1820-1633		U	1
U172,190,192,276	IC SN74S373N		1820-1676		U	4
	IC SN74S374N		1820-1677		U	3

CPU Assembly Parts List 5061 -1341/1400 (Sheet 6 of 6)

REFERENCE DESIGNATOR (FIRST SIX)	PART DESCRIPTION	PARENT OPTION	PART NUMBER	COMP. OPTION	LOC	QUANTITY PER
U24,44,64			1820-1677			
U383	IC-DIGITAL		1820-2059		U	1
U70,90,150,152,170	IC 74S373N		1820-2184		U	5
CR1-15,17	DIODE SIL		1901-0040		D	16
CR16	DIODE BKDN 2.43V		1902-3005		U	1
	SCR #6-32X.250L		2360-0113		U	4
U173	SW DIP BROCKER		3101-1983		U	1
S1,2	SW-TGL DPDT		3101-2339		U	2
	LABEL-USA		7120-6830		L	1
W2,5	WIRE JUMPERS		8159-0005		D	2
U256	IC 21MXE OSC LOP		5081-2361		3	1
U106	21MXE LUT		5090-0103		3	1
U146	21MXE LUT		5090-0104		3	1

F-5061-1400-51

SYM	REVISIONS	APPROVED	DATE
A	AS ISSUED, PCO-22-1613, DATE CODE IS 1814		

U408 28 - 500 MHz OSC 1813-0081

J-32 CLKXEN

J-30 CLKX

R14-1 +5

R14-2 +5

74S132 U389

74S00 U388

R14-3 +5V

PC 7
PA 7
PC 5
PB 5 7
PB 2 5
PA 2

SHORT CYCLE

LONG CYCLE

74S51 U404

74S51 U384

74S51 U394

74S10 U405

74S00 U388

74S10 U405

1820-2059 (74S175)

PA FF

PB FF

U383 PC FF

PC FF

5 LCMAR
FRZFF

74S132 U389

8 IOGENFF
4 MEEN

SHORT

SYNX J-34

MBUSY XA5-73

S74 U428

S74 MBUSY FF U403

74S10 U405 PAUSE

7 CPUTEN

74S20 U406

74S132 U385

74S10 U409

8 TA
8 TB

REFRESH XA5-78

74S132 U385

8 T8

74S132 U385

DMALO XA5-69

8 ENF

74S132 U385

DMALO 7,8

74S20 U246

4 RDORWT

74S64 U364

4 IAKSP
4 IOGSP

r,11,12,13

R16-6 +5

R16-4 +5

DMACYC XA5-26

74LS14 U282

74S00 U262

100pf C8

74S00 U283

DMACYC-T5 8

DMACYC-T5 3 4 7

8 SIR

74S00 U283

FRZ FF

74S74 U403

FRZ 7

6 BRUNFF

74S00 U349

R9-8

XA4-60 RUN

4 SRUNSF
8 PWUST

74LS00 U415

74S00 U415

6 RUNE

74109 U416 RUNFF

RUNB J1-14

R15-5 C29 .01 +5V

74S132 U389

74S10 4 U354

SHLT P3
B PONB

74LS30 U395

74LS132 U396

P3

HLTIO

74LS132 U396

6 RUNE

HALTB J1-31

R15-3

C30 .01

R14-6 +5V

74LS00 U415

I/I

HLT

R14-7 +5V

RLTPE XA5-8

74LS132 U396

74S20 U382

U401 74S04

P3 3 4 5

P3 4

74S20 U406

P4 4

XA4 31 BIOS

74S04 U165

P4 4

74S10 U402

XA5-47 PANF 8

PE XA5-72

74S20 U382

J1-37 P5 2 4 7

74S04 U401

P5 2 6 7

74S04 U401

74S04 U401

P5B 7

74S10 U402

XA5 - 84 PENF

74S38 U261

P51NF 5

74S11 U305

E1E2 5

74S10 U402

PCW12 6

W34 4

74S04 U401

BPC 8 4 2

74S260 U381

7 ENCP

CP12 5 4

U381 74S260

8,9,10,11

FRZFF 2 4 7

U401 74S04

XA5-76 FRZFF 4

PRESETB J1-16

R15-2 +5

74LS13 U376

PONB

74S32 U351

PCW12
LDREN

74LS00 U415

PRST 7, 8, 9

R15-4 +5

RSPE XA5-30

74LS13 U376

PARFF 2

74LS132 U396

J1-32 PARFF

J3-28

XA5-40 DMAREQ3

W2

J1-4, 10, 41, 47

J6-18, 20, 22, 24

+5

C4 - C5 1.0 µf C9-23

J1-1, 13, 36, 38, 50

J2-1, 2, 36, 49, 50

J3-2, 50, all odd

XA4-1, 2, 5, 6, 51, 52, 69, 70, 85, 86

XA5-1, 2, 85, 86

J6-1, 2 49, 50

C3 .1 µf

-2

MAIN CPU TIMING LOGIC AND DMA, MEMORY CONTROL INTERFACE RUN FLIP FLOP

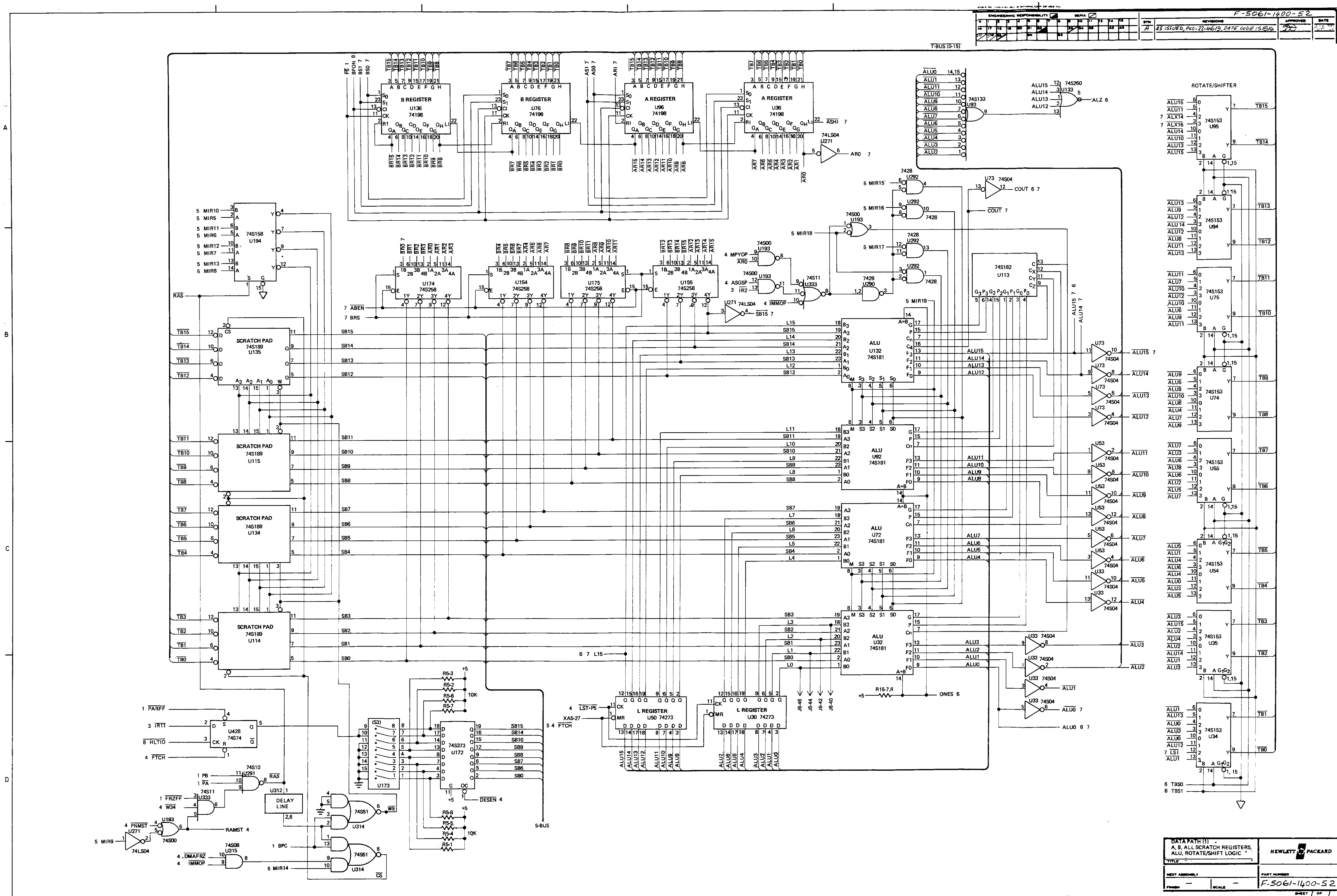
HEWLETT PACKARD

NEXT ASSEMBLY —

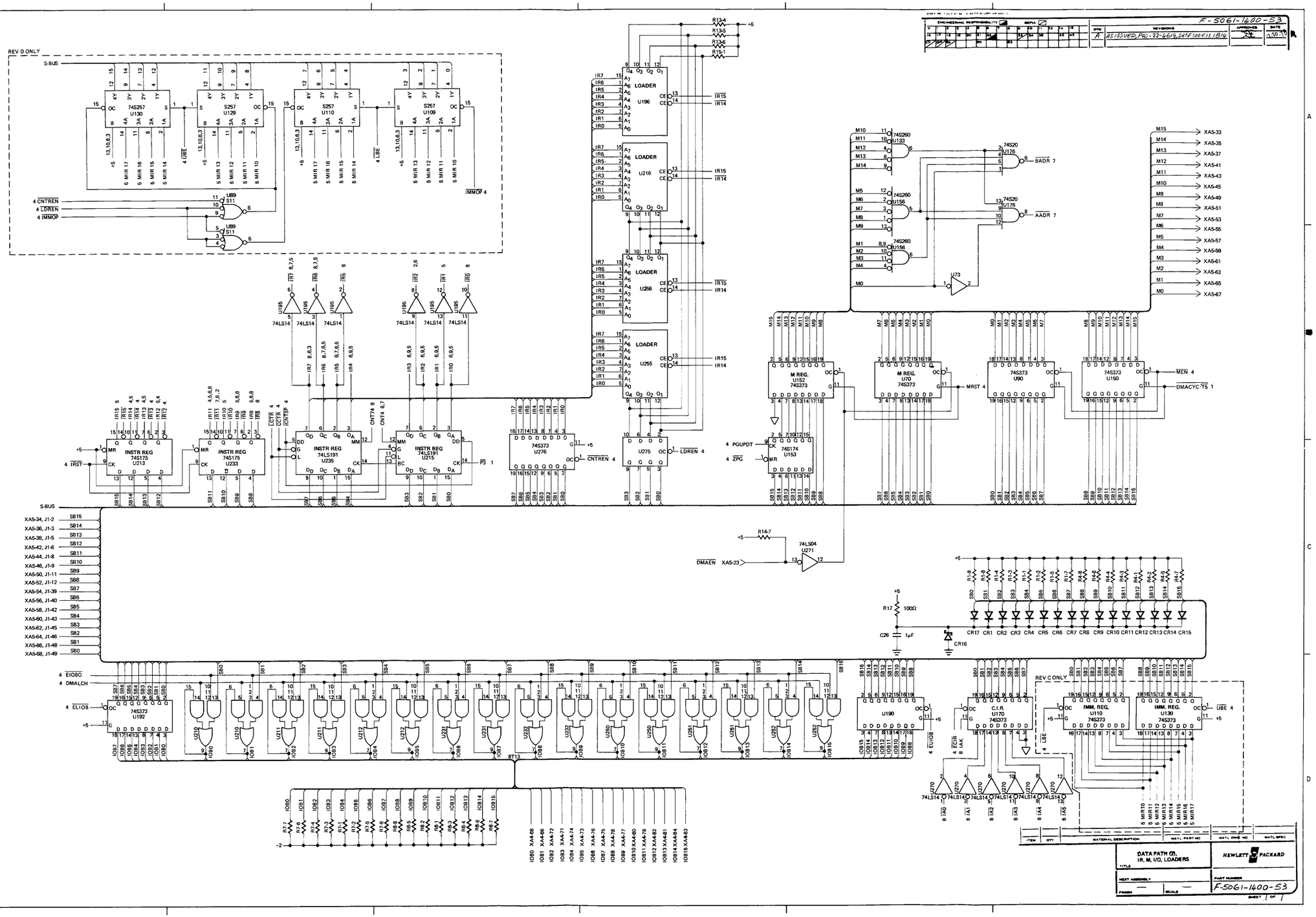
FINISH — SCALE —

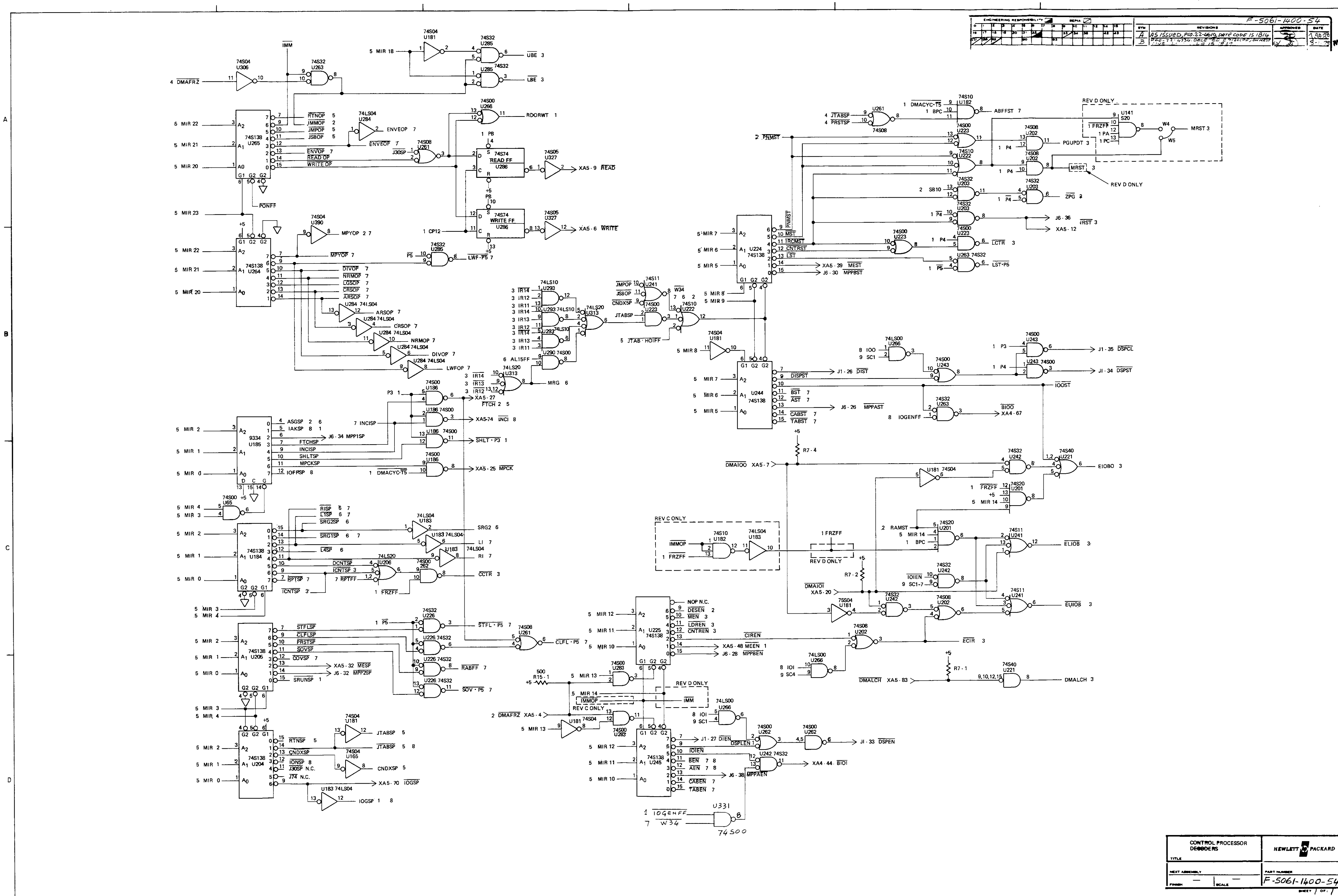
PART NUMBER F-5061-1400-51

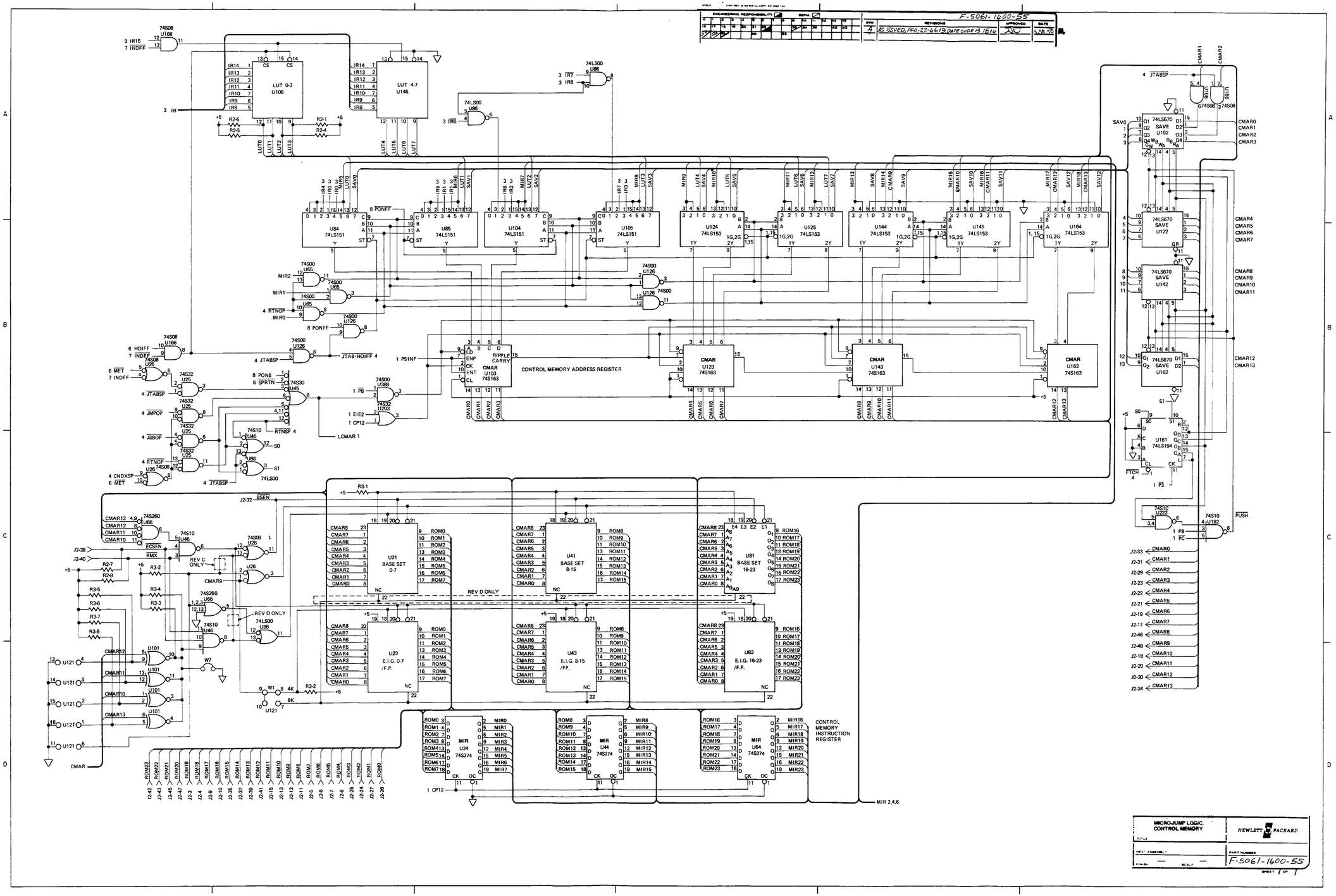
SHEET 1 OF 1

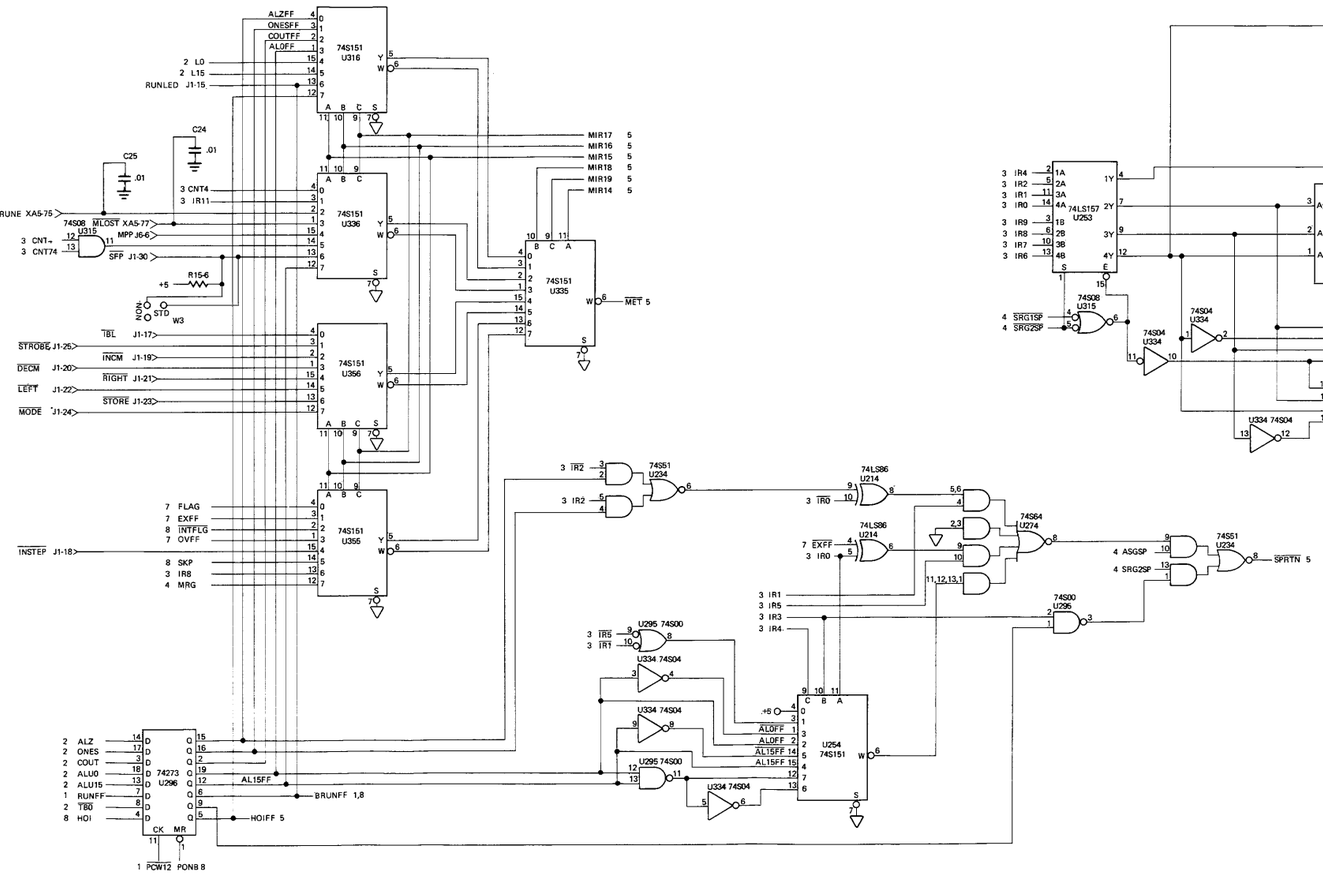


F-5061-1400-52
A ISSUED, PCO-22-6619, DATE CODE 1518U
T-BUS (0-15)
B REGISTER U136 74198
B REGISTER U76 74198
A REGISTER U96 74198
A REGISTER U36 74198
74S158 U194
U174 74S258
U154 74S258
U175 74S258
U155 74S258
SCRATCH PAD 74S189 U135
SCRATCH PAD 74S189 U115
SCRATCH PAD 74S189 U134
SCRATCH PAD 74S189 U114
74S133 U93
74S260 U133
ALZ 6
74S182 U113
ALU U132 74S181
ALU U92 74S181
ALU U72 74S181
ALU U32 74S181
COUT 6 7
ONES 6
L REGISTER U50 74273
L REGISTER U30 74273
74S373 U172
U173
U428 74S74
DELAY LINE
74S51 U314
S-BUS
ROTATE/SHIFTER
74S153 U95
74S153 U94
74S153 U75
74S153 U74
74S153 U55
74S153 U54
74S153 U35
74S153 U34
6 TBS0
6 TBS1
DATA PATH (1)
A, B, ALL SCRATCH REGISTERS,
ALU, ROTATE/SHIFT LOGIC
TITLE
HEWLETT PACKARD
NEXT ASSEMBLY
PART NUMBER
FINISH —
SCALE —
F-5061-1400-52
SHEET 1 OF 1


REV D ONLY
S-BUS
74S257 U130
S257 U129
S257 U110
S257 U109
4 CNTREN
4 LDREN
4 IMMOP
U89 S11
LOADER U196
LOADER U216
LOADER U256
LOADER U255
74LS14
INSTR REG 74S175 U213
INSTR REG 74S175 U233
INSTR REG 74LS191 U235
INSTR REG 74LS191 U215
74S373 U276
U275
4 IRST
CNTREN 4
LDREN 4
P3 1
74S260 U133
74S260 U156
74S260 U156
74S20 U176
74S20 U176
U73
BADR 7
AADR 7
M REG. U152 74S373
M REG. U70 74S373
74S373 U90
74S373 U150
MRST 4
MEN 4
DMACYC 1
4 PGUPDT
4 ZPG
74S174 U153
R13-4
R13-5
R13-6
R15-1
+5
XA5-33
XA5-35
XA5-37
XA5-41
XA5-43
XA5-45
XA5-49
XA5-51
XA5-53
XA5-55
XA5-57
XA5-59
XA5-61
XA5-63
XA5-65
XA5-67
R14-7
74LS04 U271
DMAEN XA5-23
R17 100Ω
C26 1μF
CR16
CR17 CR1 CR2 CR3 CR4 CR5 CR6 CR7 CR8 CR9 CR10 CR11 CR12 CR13 CR14 CR15
XA5-34, J1-2
XA5-36, J1-3
XA5-38, J1-5
XA5-42, J1-6
XA5-44, J1-8
XA5-46, J1-9
XA5-50, J1-11
XA5-52, J1-12
XA5-54, J1-39
XA5-56, J1-40
XA5-58, J1-42
XA5-60, J1-43
XA5-62, J1-45
XA5-64, J1-46
XA5-66, J1-48
XA5-68, J1-49
4 EIOBO
4 DMALCH
4 EIOB
74S373 U192
U190
C.I.R. U170 74S373
IMM. REG. U110 74S373
IMM. REG. U130 74S373
4 EUIOB
4 ECIR
8 IAK
4 LBE
REV C ONLY
DATA PATH (2), I/R, M, I/O, LOADERS
HEWLETT PACKARD
F-5061-1400-53
ENGINEERING RESPONSIBILITY
REVISIONS
APPROVED
DATE


F-5061-1400-54
CONTROL PROCESSOR DECODERS
HEWLETT PACKARD
F-5061-1400-54
REV C ONLY
REV D ONLY


P-5061-1400-55
REVISIONS
A AS ISSUED, ECO-22-4613 DATE CODE IS 1514
CONTROL MEMORY ADDRESS REGISTER
CONTROL MEMORY INSTRUCTION REGISTER
LUT 0-3 U106
LUT 4-7 U146
U84 74LS151
U85 74LS151
U104 74LS151
U105 74LS151
U124 74LS153
U125 74LS153
U144 74LS153
U145 74LS153
U164 74LS153
CMAR U103 74S163
CMAR U123 74S163
CMAR U143 74S163
CMAR U163 74S163
U21 BASE SET 0-7
U41 BASE SET 8-15
U81 BASE SET 16-23
U23 E.I.G. 0-7 /F.P.
U43 E.I.G. 8-15 /F.P.
U83 E.I.G. 16-23 /F.P.
MIR U24 74S374
MIR U44 74S374
MIR U64 74S374
U102 74LS670 SAVE
U122 74LS670 SAVE
U142 74LS670 SAVE
U162 74LS670 SAVE
U161 74LS194
REV C ONLY
REV D ONLY
MICRO-JUMP LOGIC, CONTROL MEMORY
HEWLETT PACKARD
PART NUMBER F-5061-1400-55
SHEET 1 OF 1



ENGINEERING RESPONSIBILITY	SIZE A	REVISIONS	
		SYM A: RELEASED, PCO-82-4619 DATE CODE IS 1814	APPROVED DATE 7/28/82

BRD. CONDITIONAL JUMP LOGIC

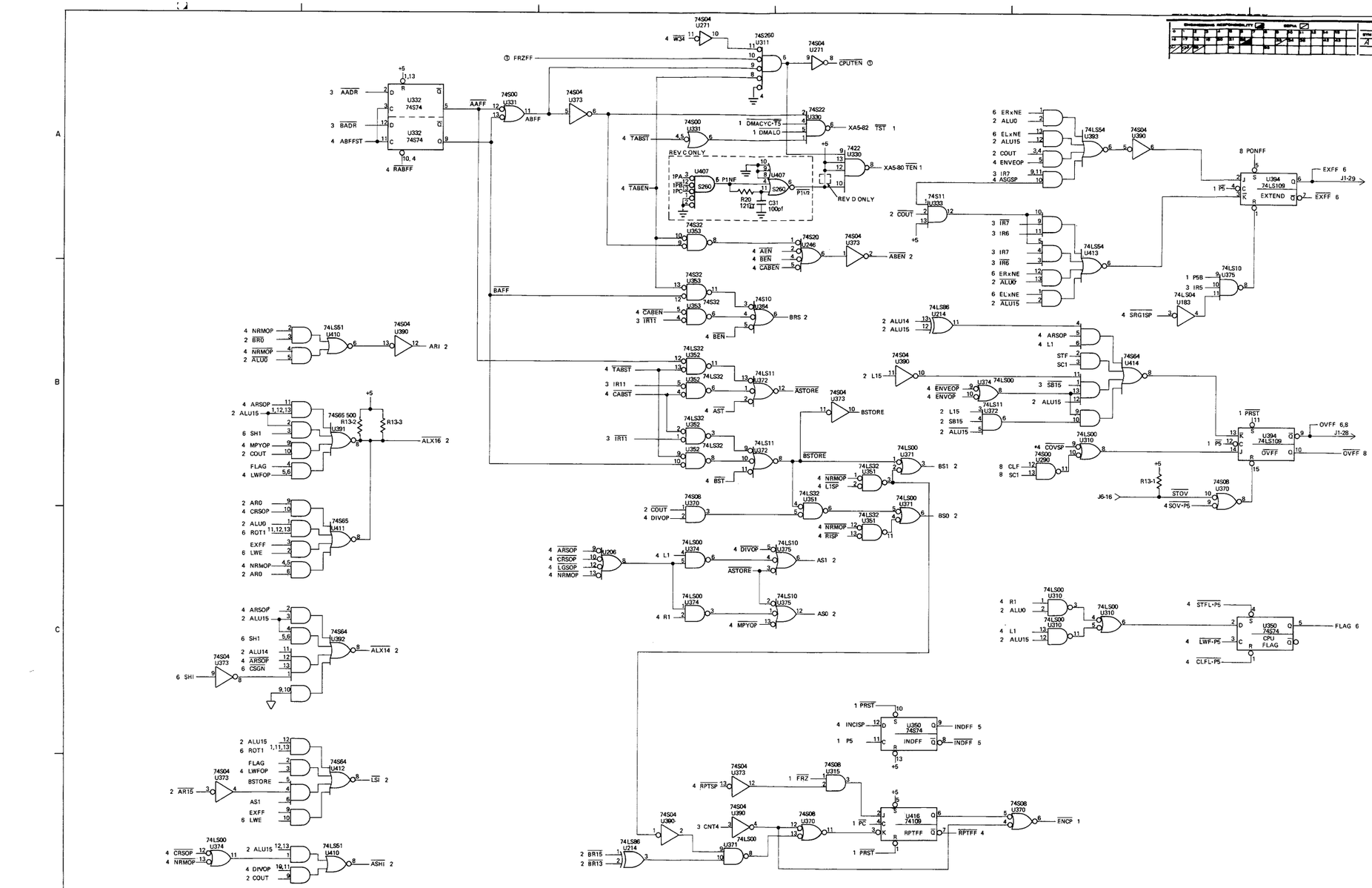
HEWLETT PACKARD

NEXT ASSEMBLY

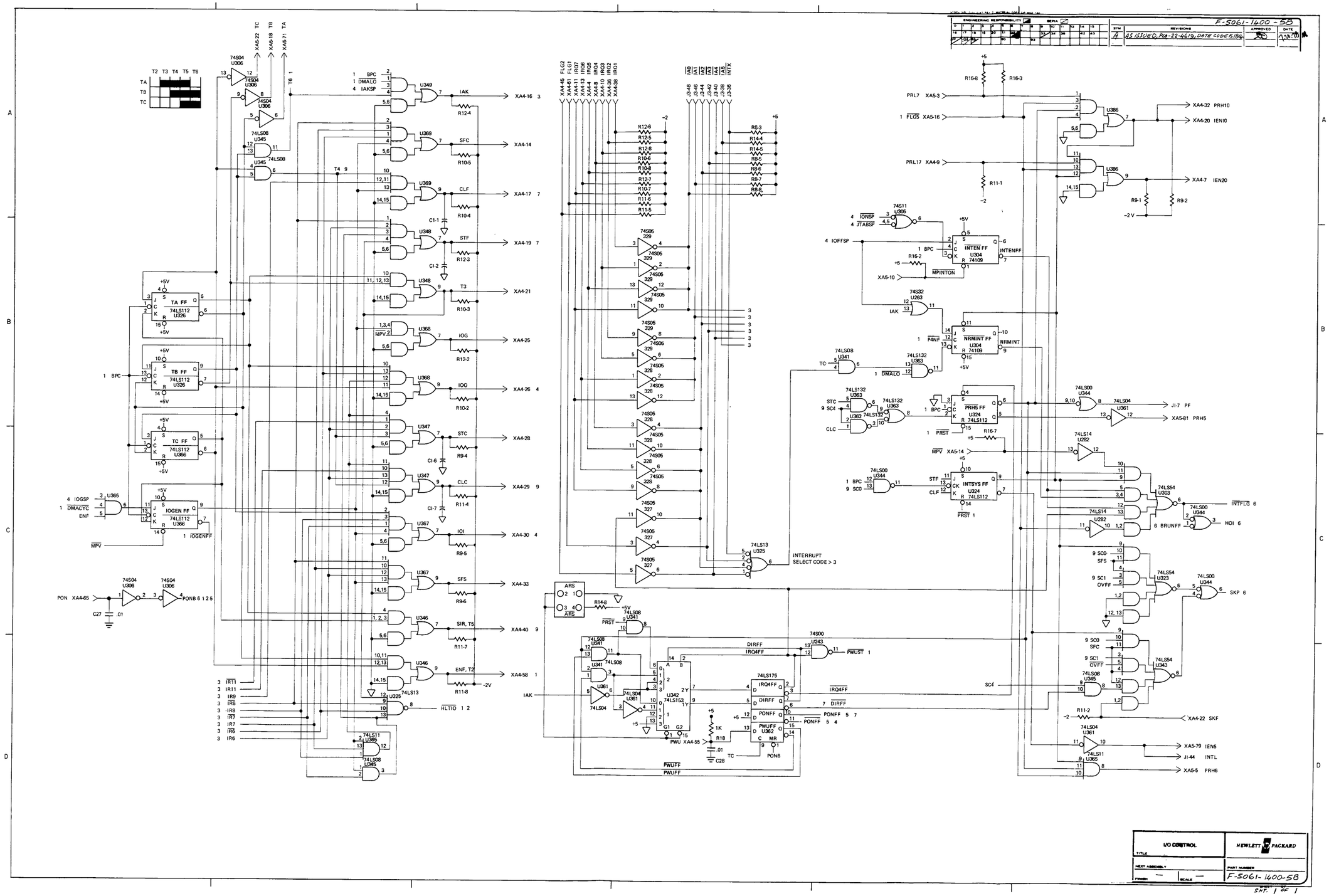
PART NUMBER F-5061-1400-56

FINISH — SCALE —

SHEET 1 OF 1

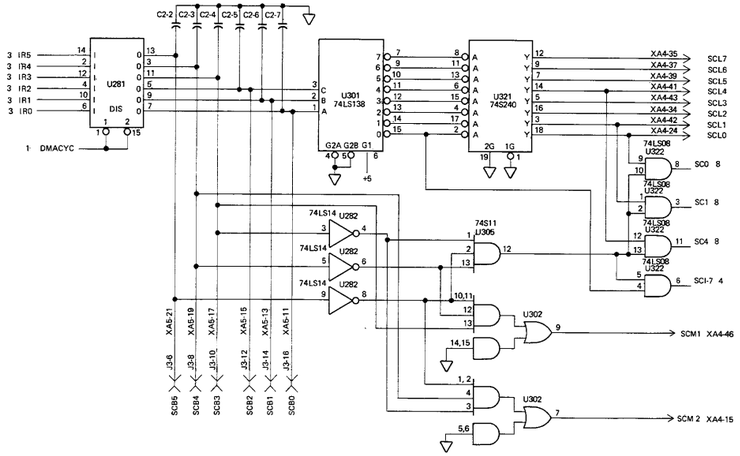


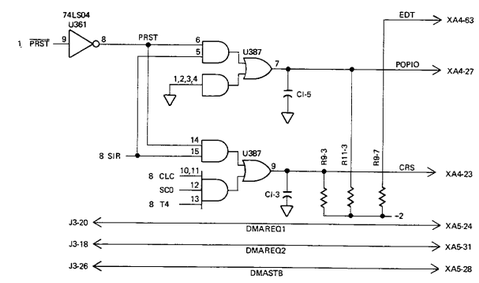
USED: LOGIC A, B ADDRESSABLE; FLAG E, C, FNC, RPF BUFF LOPS		HEWLETT PACKARD	
TITLE			
NEXT ASSEMBLY		PART NUMBER F-5061-1400-57	
FINISH —	SCALE —	SHEET 1 OF 1	


F-5061-1400-58
I/O CONTROL
HEWLETT PACKARD
SHT. 1 OF 1
IA G -23/ - 24

ENGINEERING RESPONSIBILITY	SEPIA

STM	REVISIONS	APPROVED	DATE
A	IS ISSUED, PRO-22463, DATE CODE IS 1814.		7.10.70





TITLE: SELECT - CODE BUS, MISC.	HEWLETT-PACKARD
NEXT ASSEMBLY	PART NUMBER: F-5061-1400-59
FINISH — SCALE —	SHEET 1 / 1