CHAPTER 1
KE8-E EXTENDED ARITHMETIC ELEMENT

SECTION 1 INTRODUCTION

1.1 GENERAL DESCRIPTION

The KE8-E Extended Arithmetic Element option enables the PDP-8/E to perform arithmetic operations at high speeds by incorporating EAE components with the existing central processor logic so that they operate asynchronously. All logic is contained on two quad-size modules, designated M8340 and M8341, which plug directly into the OMNIBUS. The two modules are interconnected by one H851 Connector. A second H851 Connector interconnects the M8341 to the major registers control module (Figure 1-1). This connector carries register gating and controls from the EAE modules to the register controls module. A third H851 Connector interconnects the processor's M8330 Timing Generator Module with the EAE control, supplying clock and IOT functions to the EAE. The basic OMNIBUS signals connect to each module.

1.2 SOFTWARE

The following programs are used in the maintenance of the KE8-E option.

a. KE8-E EAE Test Part 1 (MAINDEC-8E-DOLB) – This program tests all EAE instructions except MUY and DVI.

b. KE8-E EAE Test Part 2 (MAINDEC-8E-DOMB) – This program tests the MUY and DVI instructions.

c. KE8-E EAE Extended Memory Exerciser (MAINDEC-8E-DORA) – The KE8-E Extended Memory Exerciser is a test of the KE8-E "B Mode" instructions which, during the DEFER cycle, use the word following the instruction to obtain the operand. The capability of each instruction to access every memory field from every memory field through nonautoindex and auto-index is tested.

1.3 COMPANION DOCUMENTS

The following documents and publications are necessary in the operation, installation, and maintenance of this option:

a. *PDP-8/E & PDP-8/M Small Computer Handbook* – DEC, 1973
b. *PDP-8/E Maintenance Manual* – Volume 1
c. *Introduction to Programming* – DEC, 1972
d. DEC engineering drawings M8340-0-1 and M8341-0-1
e. KE8-E EAE Test Part 1, MAINDEC-8E-DOLB-D
f. KE8-E EAE Test Part 2, MAINDEC-8E-DOMB-D
g. KE8-E EAE Extended Memory Exerciser, MAINDEC-8E-DORA-D

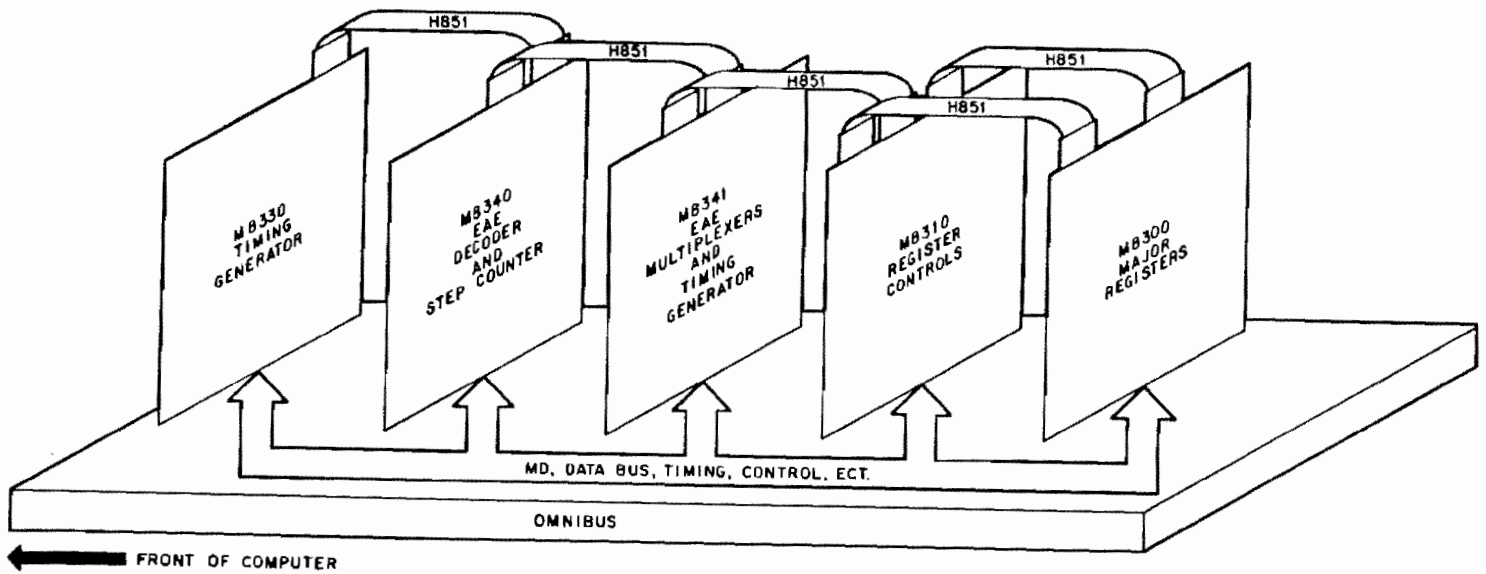


Figure 1-1 EAE Interconnections

SECTION 2 INSTALLATION

The KE8-E EAE option is installed on site by DEC field service personnel. The customer should not attempt to unpack, inspect, install, checkout, or service the equipment.

1.4 INSTALLATION

Perform the following procedures to install the KE8-E options:

Step	Procedure
1	Remove the modules from the shipping containers.
2	Inspect the modules for any apparent damage.
3	Connect the modules as follows:

a. Insert the EAE modules between the Timing Generator and the CP Major Registers and Register Control as follows:

M8330	Timing Generator
M8340	EAE Decoder and Step Counter
M8341	EAE Multiplexers and Timing Generator
M8310	CP Major Registers Control
M8300	CP Major Registers

The five modules *must* be installed in this order with no vacant slots between them for the H851 Connectors to fit properly.

b. Install H851 Connectors (five total) to connect the five modules. All connectors at the top of these modules will be utilized when all H851s are installed.

NOTE

The EAE is a complex instruction decoder that extends the basic PDP-8/E instruction set. It is intimately connected with the basic central processor and relies heavily on an M8300 and M8310 in good condition. Many potential problems can be avoided by running Instruction Test I (MAINDEC-8E-DOAB) and Instruction Test II (MAINDEC-8E-DOBB) before installing the EAE to verify the condition of the CPU. These tests should be run again after EAE installation to verify that the EAE is not malfunctioning and thereby modifying the basic instruction set.

1.5 CHECKOUT

Perform the following procedures to checkout the KE8-E option:

Step	Procedure
1	Verify that both EAE modules have been installed
2	Perform acceptance test procedures provided in Volume 1, Paragraph 2.3.
3	Load MAINDEC-8E-DOLB and perform EAE Test – Part I.
4	Load MAINDEC-8E-DOMB and perform EAE Test – Part II.

(continued on next page)

Step	Procedure
5	Load MAINDEC-8E-DORA and perform EAE extended memory exerciser (even if 4K machine).
6	Make entry on user's log that the acceptance test for the KE8-E was performed satisfactorily.

SECTION 3 SYSTEM DESCRIPTION

The organization of the EAE system block diagram (Figure 1-2) follows the organization of the detailed logic description. The detailed logic is organized by source, route and destination and contains logic diagrams representing each block illustrated in Figure 1-2.

Signals generated within the EAE control the operation of the M8300 Major Registers Module during EAE instructions. The processor timing extension logic causes the processor to halt at TP3 and at the same time starts the EAE Timing Generator. This extends TS3 to enable data to be applied to the adders a number of times. The EAE selects which register is to go into the adders by asserting a combination of signals shown in the EAE source control logic block. What happens to the data when it is on route to its destination is accomplished by asserting a combination of signals in the EAE route control signals block. The destination of the data is either the AC Register or the MQ Register.

The EAE was designed for hardware compatibility with old programs that were written for the PDP-8/I. The MODE flip-flop is cleared, selecting Mode A, when the computer is turned on. Mode A is the PDP-8/I compatible mode. Mode B is selected (via the mode-change instruction) only when using programs developed specifically for this EAE.

To better understand how the EAE functions, 13 of the EAE instructions are described in terms of functional flow to illustrate how the EAE completes each instruction.

NOTE

EAE operation is more integrated with the CPU than most options. Before attempting to study EAE theory of operation, the reader should thoroughly understand CPU theory and review sections of Volume 1 as he is reading this chapter.

1.6 STEP COUNTER LOADING OPERATION (Figure 1-3)

The Step Counter controls the number of shifts performed during the ASR, LSR, and SHL instructions. It also controls the number of steps taken during MUL or DVI, and records the number of shifts required to normalize a number.

The KE8-E provides two methods of loading the Step Counter. The ACS instruction is used by the new, or Mode B, instruction set; the SCL instruction is used by the old, or Mode A, instruction set. The SCL instruction is of interest because this same method of step counter loading is used within the SHL, LSR, and ASR instructions in both modes.

The ACS instruction takes place in a manner similar to an I/O transfer to a peripheral. The contents of the AC are placed on the DATA BUS during TS3. C0 is grounded so that the AC will be cleared. At the leading edge of TP3 the five least-significant bits are loaded into the Step Counter.

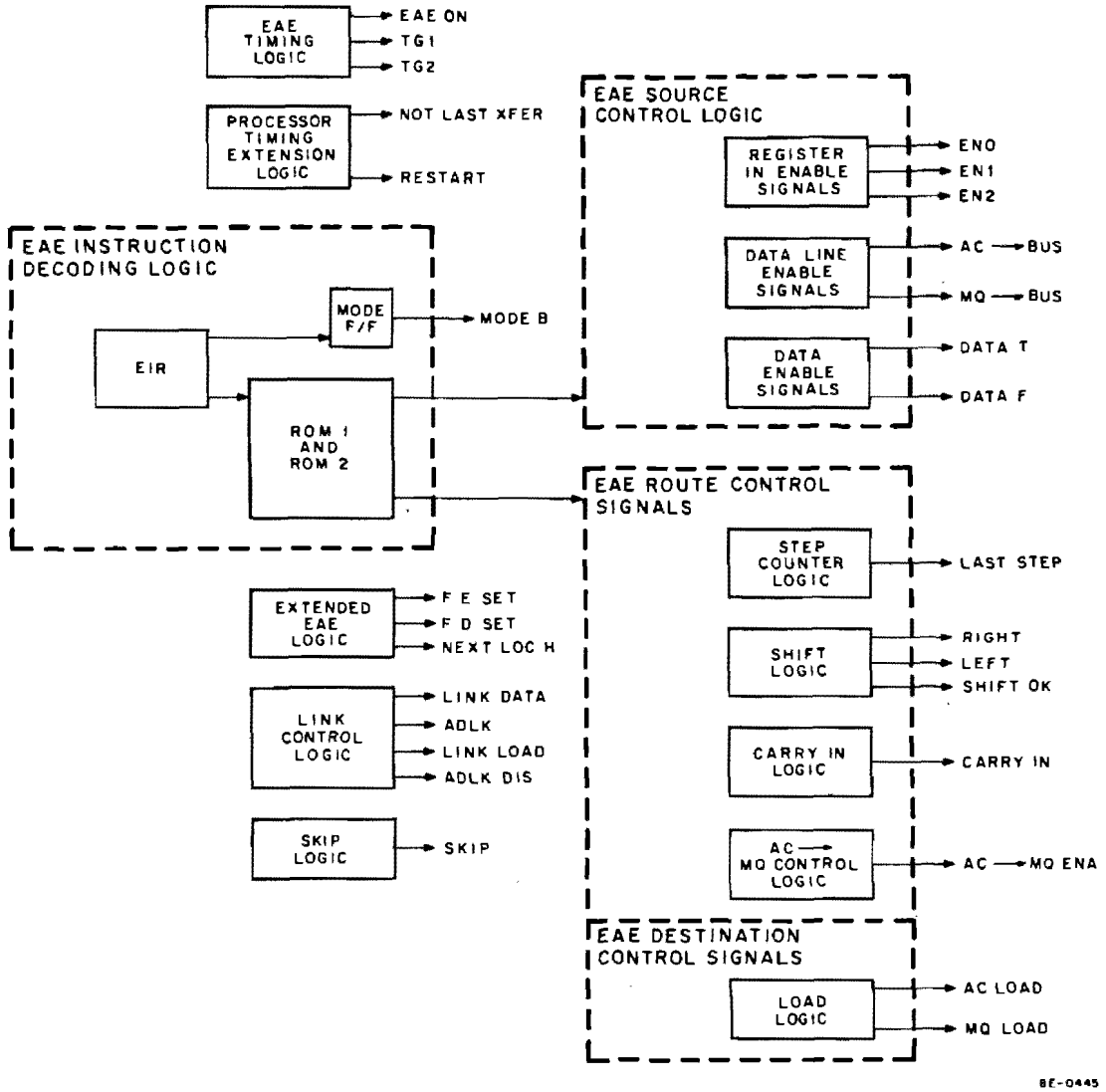


Figure 1-2 EAE System Block Diagram

The SCL instruction is somewhat more complicated. The Step Counter is to be loaded with the 1's complement of the next word in memory. As soon as the instruction is decoded, the SKIP line on the OMNIBUS is grounded. As explained in Paragraph 3.38 of Volume 1, the SKIP line is tested during IOT and OPERATE instructions. Grounding the SKIP line causes the next location (which, in this instance, contains the data for the Step Counter) to be skipped as an instruction.

During TS4 of the FETCH cycle, several control lines into the M8300 Major Registers Module are asserted by the KE8-E via the H851 Connectors. These signals (ENO and CARRY IN) cause the next location in memory to be addressed and treated as an operand (F E SET). During TS2 of the EXECUTE cycle, the contents of the five least-significant MD lines are inverted and applied to the inputs of the Step Counter. At TP2 the Step Counter is loaded.

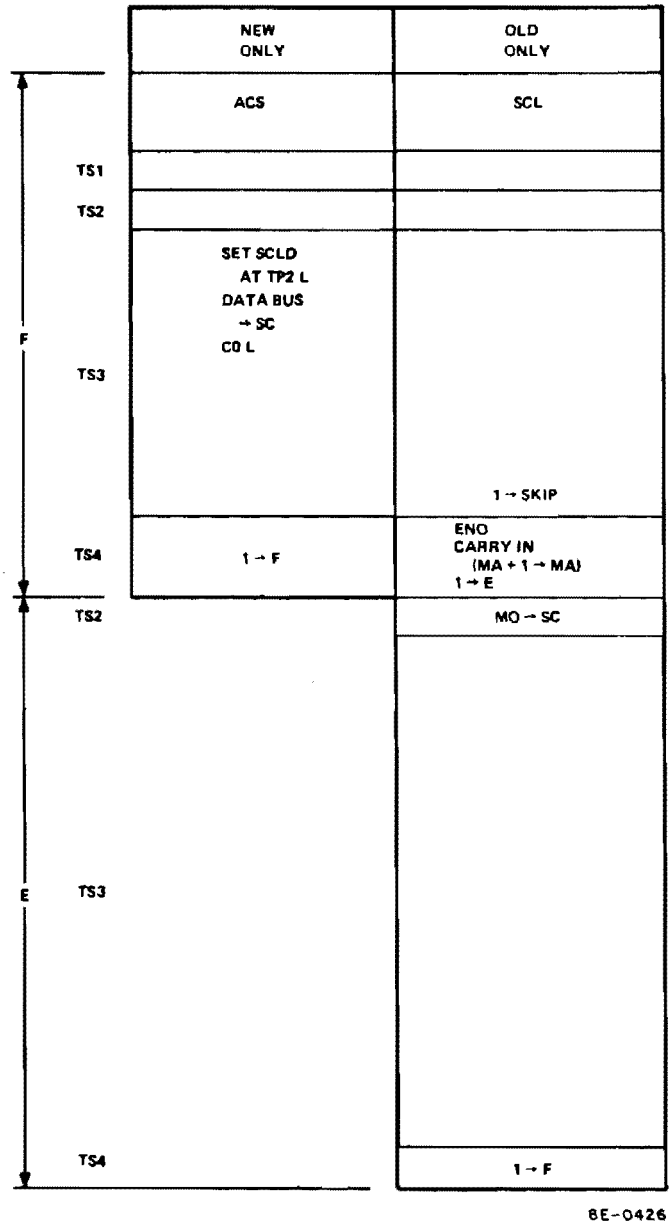


Figure 1-3 Step Counter Loading, Flow Diagram

1.7 STEP COUNTER TO ACCUMULATOR LOADING OPERATION (Figure 1-4)

The contents of the Step Counter are ORed with the contents of the AC and the result transferred to the AC. The entire operation is so similar to an IOT input OR transfer that it will not be discussed further.

1.8 THE SHIFT LEFT OPERATION

The shift left (SHL) instruction (Figure 1-5) is a 2-cycle instruction. The first cycle fetches the instruction word; the second cycle fetches a number that specifies the number of shifts that are to occur. The entire operation is identical to the SCL instruction up to and including TP2 of the EXECUTE cycle.

At the start of TP3 of the EXECUTE cycle, the EAE must shift the contents of the AC, MQ, and Link left by the number of places specified in the Step Counter. Normal machine timing stops at TP3 and EAE timing begins; one shift operation occurs with each clock pulse until the last shift has been performed.

Once the EAE is on, the following signals to the M8300 are asserted:

Signal	Function
LEFT L	Enables left shift gates at output of adders.
SHL + LD EN L	Enables MQ left shift path.
ADLK DIS L	Disconnects the normal Link-AC11 shift path. Also disconnects the AC0-Link shift path.

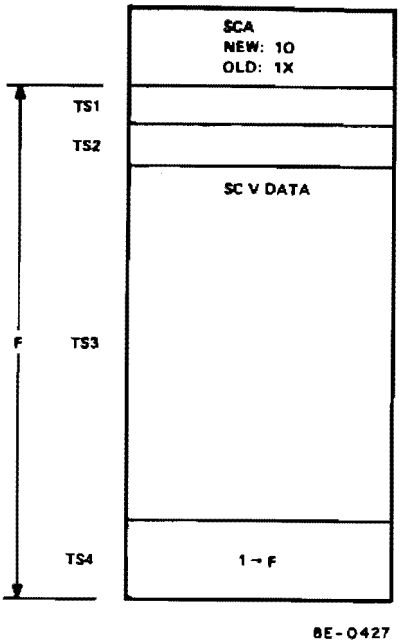


Figure 1-4 Step Counter to AC, Flow Diagram

The following logic functions also occur:

Signal	Function
MQ0 → ADLK L	Establishes shift path from MQ0 to AC11.
AC0 → LINK DATA L	Establishes shift path from AC0 to Link.

MQ DATA is negated (high) so 0 is shifted into the MQ11. For each shift AC LOAD, MQ LOAD and LINK LOAD are developed and 1 is added to the step count. When the step count reaches 37, the EAE starts its shut-down process.

If the instruction mode is A, the EAE merely performs its last shift with NOT LAST XFER high. The processor restarts, and the total number of shifts is one more than the number in the second core location. If the instruction mode is B, a special line within the EAE, SHIFT OK H, is negated. Negating SHIFT OK H negates the signals required to cause AC shifts, and inhibits LINK LOAD and MQ LOAD. Thus, the processor starts without taking the final shift; the number of places shifted is equal to the number in the second core location.

1.9 RIGHT SHIFT OPERATIONS (Figure 1-5)

Two right-shift instructions, ASR and LSR, are available in the EAE option. The only difference between the two instructions is how the Link is handled. The Link is loaded at TP3 of the FETCH cycle via the OMNIBUS. If the LSR instruction (logical right shift) is being processed, no data is placed on the LINK DATA line and thus the Link is cleared. If the ASR instruction (arithmetic right shift) is being processed, AC0 is placed on the LINK DATA line and the Link is thus made equal to AC0.

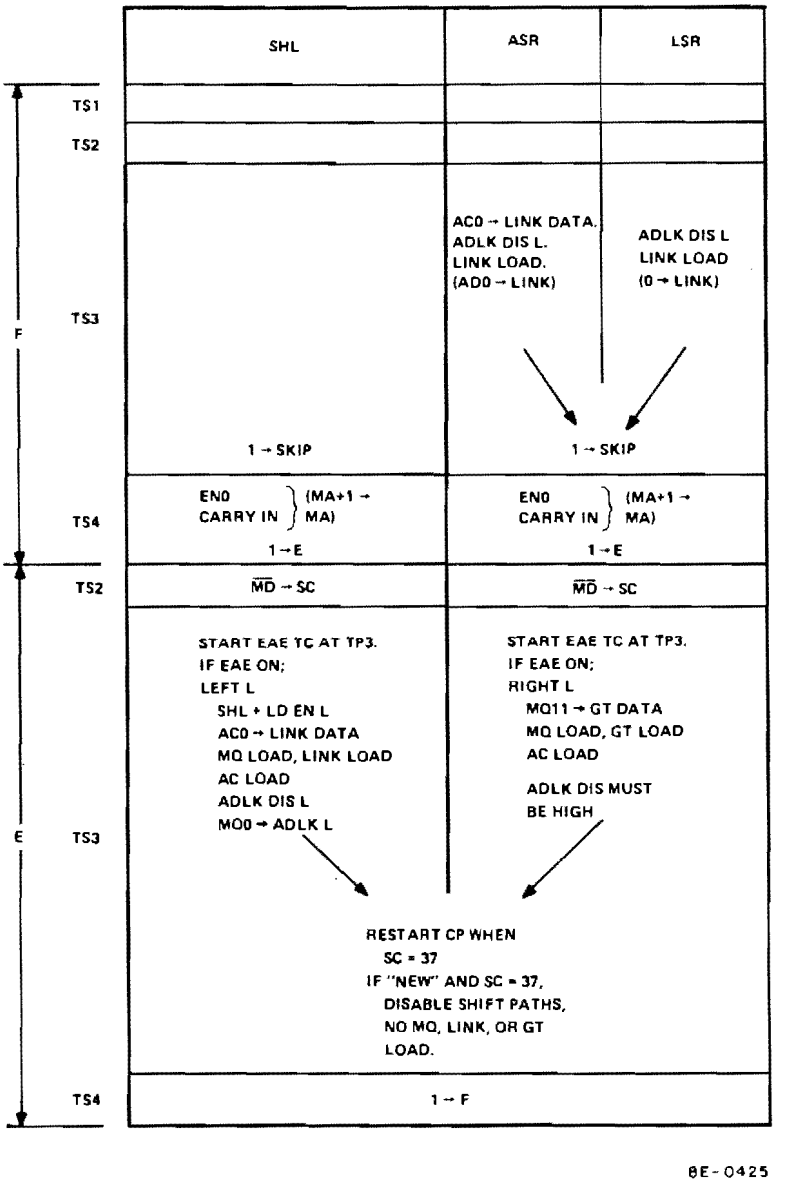


Figure 1-5 SHIFT Operations, Flow Diagrams

As in the case of SHL, the computer enters the EXECUTE cycle to obtain step-count information. When the EAE is turned on at TP3 of the EXECUTE CYCLE, the following signals are asserted:

Signal	Function
RIGHT L	Enables MQ right shift, enables right shift gates at adder outputs.
MQ11 → GT DATA	Enables path from MQ11 to the GT flag.

Like the SHL instruction, AC LOAD and MQ LOAD are generated for each shift and the step count is incremented. GT LOAD is also generated for each shift, although the GT flag is held cleared if it is in Mode A. Notice that the Link is not loaded, and that ADLK DIS L is high. These two conditions mean that the Link is not modified during shifting, but that the output of the Link is coupled to the input of AC0. All other details are similar to those given for the SHL instruction.

1.10 NORMALIZE INSTRUCTION (Figure 1-6)

Normalization is the process by which the 24-bit fixed-point word in the AC and MQ Registers is converted to floating-point format and expressed as a fraction and the corresponding power of two.

The 1-cycle NORMALIZE instruction is completely implemented during TS3 of the FETCH state. Because the final shift count is important to this operation, the Step Counter is initially cleared (zeroed). OMNIBUS signal NOT LAST XFER L is asserted and, at TP3, processor timing comes to a halt and the EAE Timing Generator is started. The Normalize operation only occurs if SHIFT OK H remains H, as determined by comparing AC0 to AC1. If the two are not equal, SHIFT OK H is grounded, thereby causing the EAE timing to halt and restart the processor timing.

As long as AC0 and AC1 are equal, AC, MQ, and Link will shift one place to the left as if they were one long register, as explained for the SHL instruction. Each time a shift occurs, 1 is added to the Step Counter. This continues until the EAE finds AC0 not equal to AC1. Another condition for which the Normalization process is terminated is when AC2-MQ11 are all equal to 0 (the word cannot be normalized). The Normalization process also terminates in Mode B if the 24-bit word in the AC and MQ equals 40000000 (only AC0 is a 1); C0 is grounded during TS3 so that the AC is cleared.

1.11 DOUBLE-PRECISION SKIP IF ZERO (DPSZ) (Figure 1-7)

The 24-bit number in the AC and MQ is tested. If all bits are 0, the next instruction is skipped. If any bit is a 1, the next instruction is executed.

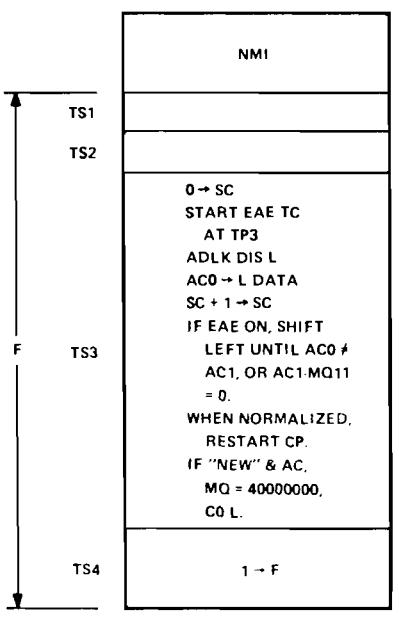


Figure 1-6 NORMALIZE Operation, Flow Diagram

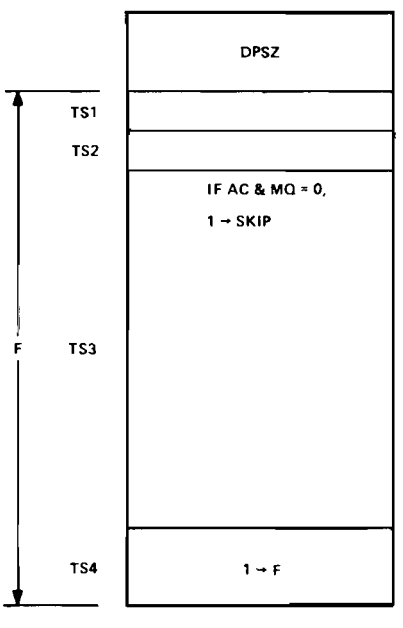


Figure 1-7 DPSZ Instruction, Flow Diagram

1.12 DOUBLE-PRECISION COMPLEMENT (DCM) (Figure 1-8)

The objective of the DCM instruction is to form the 2's complement of the 24-bit word in the AC and MQ. Since the M8300 Major Registers Module is capable of only 12-bit arithmetic, the complete DCM operation requires two passes through the adders. These passes are labeled Step 1 and Step 2 in Figure 1-8 and in the following paragraphs. The entire operation takes place in the FETCH cycle.

The DCM instruction uses the SWP instruction built into the M8310. (One requirement of the DCM instruction is that bit 5, controlling the MQ → AC path, and bit 7, controlling the AC → MQ path, both be 1s.) Thus, as the DCM instruction is decoded, the M8310 causes MQ → BUS L and AC → MQ ENA L (described in Volume 1, Paragraph 3.40). At the KE8-E, two other lines to the M8300 are being controlled. These lines are DATA F L, which is asserted for both operations and CARRY IN L, which is unconditionally asserted for Step 1 and asserted if Link is 1 for Step 2. The KE8-E also disables normal Link gating and places CARRY OUT L from the adders onto the LINK DATA L line of the OMNIBUS. The Link, AC, and MQ are loaded at the conclusion of each step. (AC LOAD and MQ LOAD occurs at the end of Step 1 because of the SWP portion of the instruction.) The processor's timing chain is stopped and the EAE's timing chain is run for one step to provide the extra time and load pulses for Step 2.

One of the more severe tests of the DCM instruction is to perform this operation on a cleared AC and MQ, since such a task requires the carry to propagate through all 24 bits.

1.13 DOUBLE-PRECISION INCREMENT (DPIC) (Figure 1-9)

The DPIC instruction adds 1 to the 24-bit word in the AC and MQ in the same manner as the DCM instruction. The only difference is that DATA F L is not asserted, allowing the contents of the DATA BUS to be applied to the adders without being complemented.

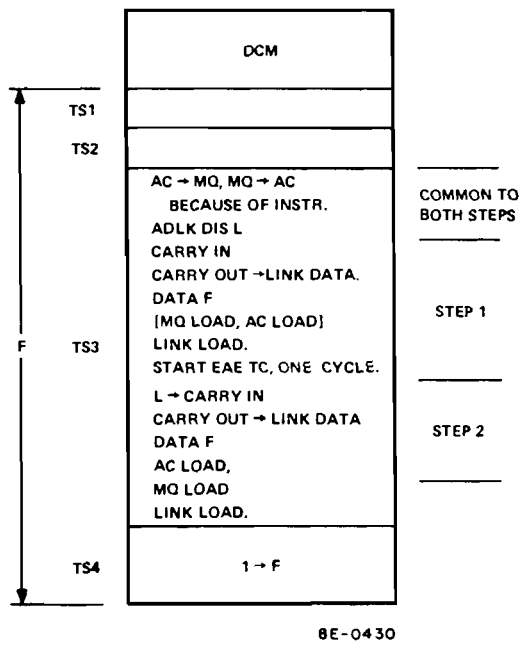


Figure 1-8 DCM Instruction, Flow Diagram

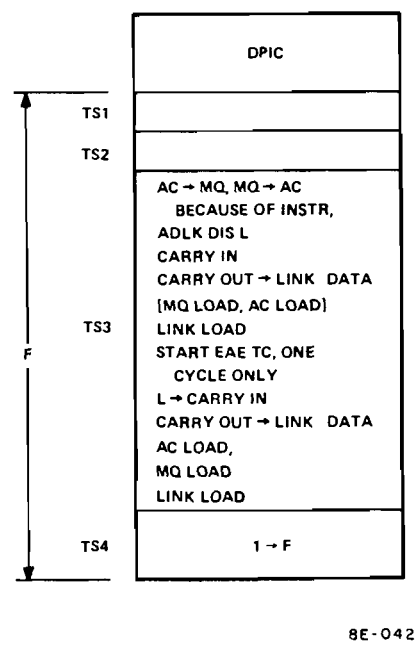


Figure 1-9 DPIC Instruction, Flow Diagram

1.14 DOUBLE-PRECISION STORE (DST) (Figure 1-10)

The contents of MQ and AC are stored at the double-precision location (two consecutive memory locations). The AC, MQ, and Link are not changed by this instruction. When the EAE decodes the DST instruction (Figure 1-10), the next location is accessed in a manner similar to the SCL instruction. Instead of grounding F E SET, however, the DST instruction grounds F D SET, thereby causing the computer to enter the DEFER major state and treat the next location as an address.

At the conclusion of the DEFER cycle, the computer enters the EXECUTE CYCLE. Simultaneously, a flip-flop within the EAE sets. This flip-flop (EX1) grounds F E SET, causing the processor to perform two consecutive EXECUTE cycles and forcing MA + 1 → MA at the end of the first EXECUTE cycle. EX1 is cleared at the end of the first of these EXECUTE cycles, allowing normal processing to resume at the conclusion of the second EXECUTE.

During each of the EXECUTE cycles, the following processor signals are asserted during TS2:

Signal	Function
MQ → BUS DATA T	Gates MQ Register to MB.
MD DIS	Removes the MD, which is normally applied to the MB via the adders.

(AC → MQ ENA L is also grounded, but has no effect since the MQ is not loaded at TP2.)

During TS3, the usual gating is set up to swap the contents of the AC and MQ. Hence, the sequence of events during the EXECUTE cycle is:

a. Store the least-significant twelve bits, presently in the MQ.

b. Swap the AC and MQ.

c. Address the next memory location.

d. Store the most-significant twelve bits presently in the MQ.

e. Swap the AC and MQ to return the bits to their original locations.

1.15 DOUBLE-PRECISION ADD (DAD) (Figure 1-11)

The DAD instruction has many similarities to the DST and DCM instructions. Like the DST instruction, it uses a second memory word as a deferred address. It also requires two EXECUTE cycles to obtain data from two consecutive memory locations. The DAD instruction handles its carry to and from the Link in a manner similar to the DCM instruction.

During TS3 of the FETCH cycle, ADLK DIS is grounded, enabling the OMNIBUS LINK DATA and LINK LOAD inputs. At TP3, LINK LOAD is generated. Since no data was placed on LINK DATA, the Link is cleared. Other than clearing the Link, the DAD instruction process is identical to that of the DST instruction for the first two machine cycles.

During each of the two EXECUTE cycles, a word is obtained from memory and applied to the adders via the MD lines. During TS3, the output of the Link is applied to the carry input of the adders. The contents of the MQ are gated to the other inputs to the adders. The carry output of the adders is applied to the LINK DATA input

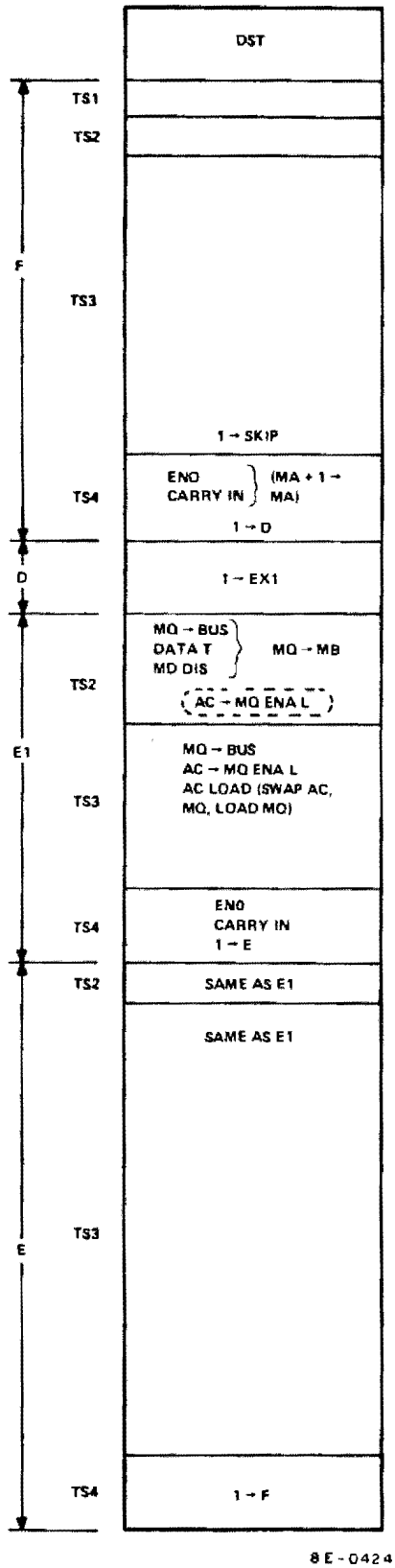


Figure 1-10 DST Instruction, Flow Diagram

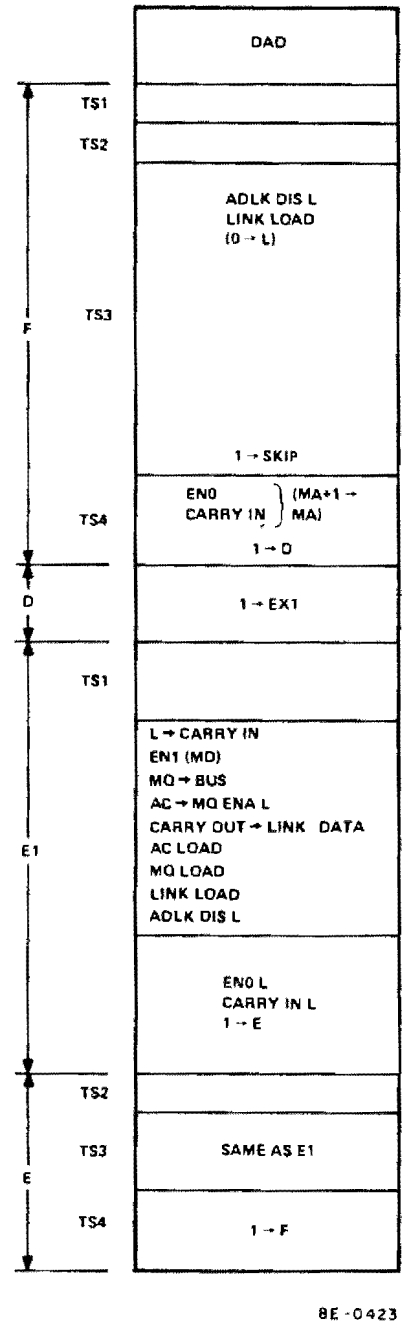


Figure 1-11 DAD Instruction, Flow Diagram

of the Link; the path from the AC to MQ is enabled. At TP3 the AC, Link, and MQ are loaded. Hence, the old AC is moved to the MQ, while the sum of the old MQ and the MD is loaded into the AC. The Link provides and receives carry information.

1.16 SUBTRACT AC FROM MQ (SAM) (Figure 1-12)

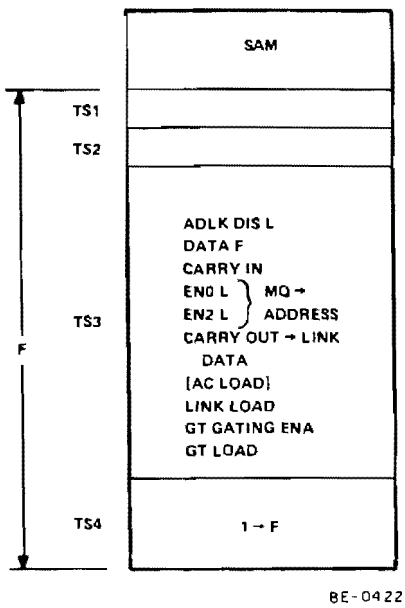


Figure 1-12 SAM Instruction, Flow Diagram

The SAM instruction subtracts AC from MQ and places the result in the AC, Link, and GT flag. The MQ is not modified. The entire operation takes place during the FETCH cycle.

The MQ is gated to the adders by grounding source control lines EN0 and EN2 at the H851 Connectors. As listed in Table 3-4 of Volume 1, the MQ Register is gated to one of the sets of adder inputs. The AC is complemented and introduced to the other set of adder inputs via the DATA BUS by grounding AC → BUS, DATA F, and CARRY IN. A "Greater Than" signal is generated and applied to the GT flag. The carry from the address is applied to the Link inputs. The AC, Link, and GT are loaded, completing the operation.

The "Greater Than" signal is derived as follows:

a. If the MQ and the old AC are of different signs, the MQ is greater than the AC if the MQ is positive. The MQ is less than the AC if the MQ is negative.

b. If the MQ and the old AC are of the same sign, the MQ is greater than (or equal to) the old AC if the output of the most-significant bit of the adder is positive. Otherwise, the MQ is less than the AC.

Logic at the input of the GT flag computes the "Greater Than" signal.

1.17 MULTIPLY INSTRUCTION (MUY) (Figure 1-13)

The MUY instruction combines the multiplicand (which was previously loaded into the MQ Register) with a multiplier (obtained from memory by the MUY instruction), using the rules of binary multiplication. The result is left in the AC and MQ. The multiplication requires twelve (decimal) steps which are counted by the Step Counter. At each step, MQ11 is examined. If it is a 1, the multiplier is added to the AC. Regardless of the state of MQ11, the AC and MQ are shifted right in the same manner as is done for the LSR instruction, except that the GT flag is not loaded. This same process is repeated for the new MQ11 until the twelve steps have been completed. At this point, the AC and MQ contains the 24-bit product.

The MUY instruction requires one FETCH cycle to fetch the instruction, one DEFER cycle (Mode B only) to obtain the multiplier address, and one EXECUTE cycle to obtain the multiplier and accomplish the multiply operation.

The decoded instruction clears the Step Counter and places a 0 in the Link by asserting ADLK DIS L and LINK LOAD L. It then accesses the next location in memory (refer to SCL instruction). If the older, PDP-8/I compatible, Mode A instruction set is in use, the next sequential address contains the multiplier. The EAE, therefore, grounds F E SET L and goes directly to the EXECUTE cycle. If the EAE is in Mode B, F D SET L is grounded and the processor enters the DEFER state for the address of the multiplier. At the conclusion of the DEFER cycle, the processor automatically enters the EXECUTE state.

During the first part of the EXECUTE cycle, the multiplier is read onto the MD lines. At TS3, NOT LAST XFER L is asserted on the OMNIBUS; at TP3, processor timing halts. The EAE timing chain is then started. The right-shift signals are asserted (refer to the LSR instruction for further details). Each time MQ11 is a 1, EN1 is grounded by the EAE. If EN1 is ground, EN0 is grounded by the M8310 Major Registers Control Module. Grounding EN1 and EN0 causes the word on the MD lines to be added to the partial product. This process continues for the twelve steps necessary to complete the multiplication. The last step is made with NOT LAST XFER high, causing the processor to resume its timing.

The data paths for the MUY instruction are illustrated in Figure 1-14. This figure also illustrates the control signals that must be enabled to make this instruction possible.

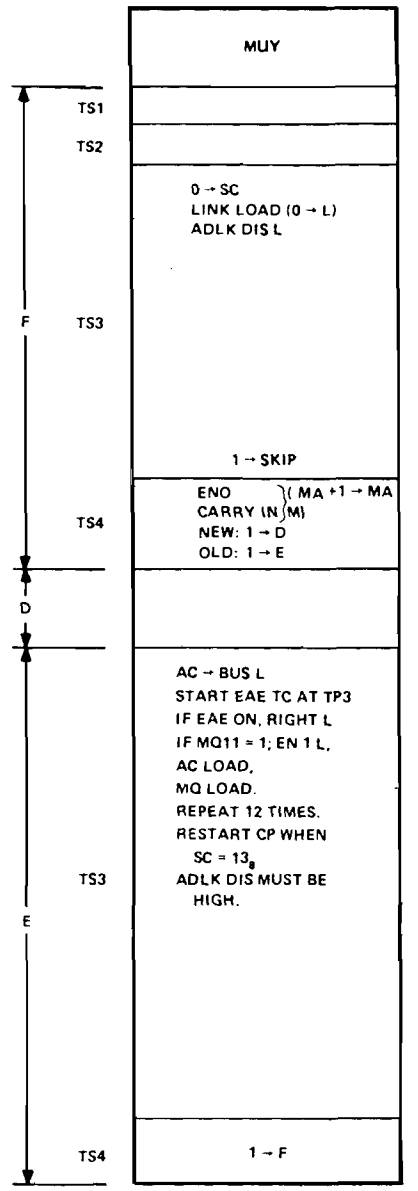


Figure 1-13 MUY Instruction/ Operation, Flow Diagram

1.18 DIVIDE INSTRUCTION (DVI) (Figure 1-15)

There are two common methods of doing binary division:

a. Restoring divide (the standard long-hand method): Try to subtract. If the result is +, place a 1 in the quotient. If the result is −, the subtraction does not take place; place a 0 in the quotient. In either case, shift left.

b. Nonrestoring divide (the method used in PDP-8/E): Always make the subtraction and always shift left. If the result is +, place a 1 in the quotient; the next step will also be a subtract. If the result is −, place a 0 in the quotient; the next step will be an add. This method requires a final correction step if the final remainder is −.

Figure 1-15 illustrates the DVI Instruction. This instruction requires one FETCH, one DEFER (Mode B only), and one EXECUTE cycle. The instruction clears the Step Counter at TP3 of the FETCH cycle. The next memory location is accessed, as explained for the SCL instruction. If the instruction mode is B, the CPU must obtain the operand address by entering the DEFER major state. Otherwise, the CPU goes directly into the EXECUTE state.

The first subtraction takes place at TP3 of the EXECUTE cycle, *before the EAE is turned on.* At the same time, the Link* is set. The EAE is turned on only if there is a carry from the most-significant bit of the adder. Otherwise, a condition known as divide overflow exists, and the quotient cannot be contained in the 12 bits available. If the EAE is turned on, the last divide step clears the Link. Thus, the Link is used as a program flag to indicate whether or not divide overflow occurred.

*Link refers to Processor Link, DIV LNK refers to EAE Link.

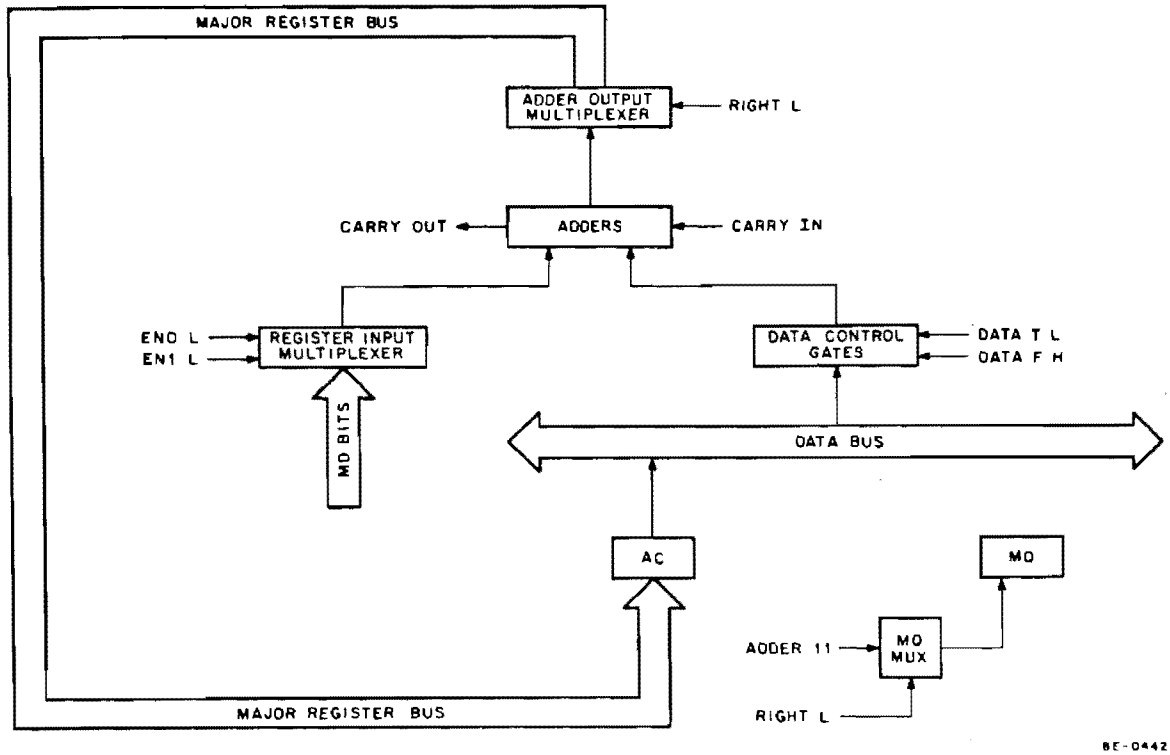


Figure 1-14 MUY Instruction Data Paths

The Major Registers are shown in Figure 1-16. This figure, an expansion of Figure 3-79 of Volume 1, shows the signals that are important to the divide process. Notice that the M8300 has no provision for complementing the MD. The only means of complementing is via the data control gates which are in the AC shift path. In order to subtract, the KE8-E must cause $\overline{AC}$ plus MD → AC. The AC now contains the complement of the result. Successive subtractions merely cause AC plus MD → AC, since the AC is already in complemented form. To change from subtraction to addition, the KE8-E must cause $\overline{AC}$ plus MD → AC. Of course, successive adds are performed by AC plus MD → AC. Complementing is accomplished by grounding DATA F, and must be performed each time the quotient bit changes. The logic merely grounds DATA F if MQ10 ≠ MQ11 after the first two divide steps have established quotient bits in MQ10 and MQ11. DATA F is grounded for the first two steps.

As explained above, AC may be in its true or complemented form as the divide operation progresses. The MQ is always in its true form, and is the source of unprocessed dividend bits that are shifted into AC11. If the word being loaded into the AC is in complemented form, MQ0 must be complemented as it is shifted into AC11. The logic merely examines MQ11. If it is a 1, MQ0 is complemented as it is shifted into AC11.

The fundamental rule governing the quotient bit is as follows:

If the sign of the dividend does not change, MQ11 → MQ11.

If the sign of the dividend changes, $\overline{MQ11}$ → MQ11.

But the sign might have changed because the logic grounded DATA F, so the fundamental must be expanded to:

If DATA F H and no sign change, or if DATA F L and the sign changes,
MQ11 → MQ11.

If DATA F H and the sign changes, or if DATA F L and no sign change,
$\overline{MQ11} \rightarrow MQ11$.

Since DATA F L is caused by MQ10 ≠ MQ11, a little Boolean manipulation yields:

If MQ10 = 0 and no sign change, or if MQ = 1 and the sign changes, 0 → MQ11.

If MQ10 = 0 and the sign changes, or if MQ10 = 1 and no sign change, 1 → MQ11.

The sign change is derived from DIV LNK (the AC sign bit after the shift) XORed with CARRY OUT. All combinations of MQ10, MQ11, DIV LNK and CARRY OUT are shown in Table 1-1, together with the resulting quotient bit.

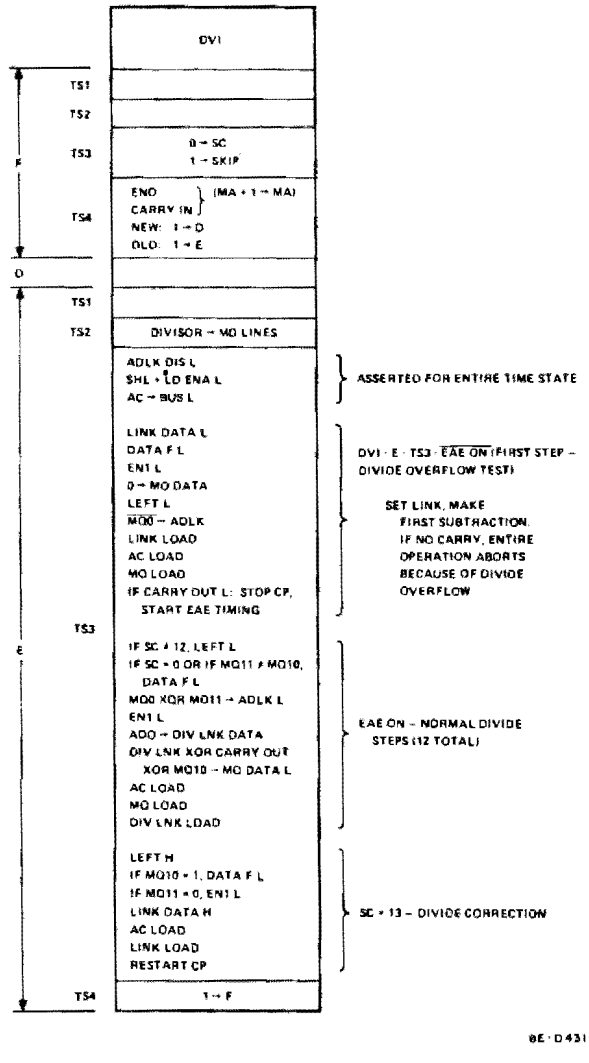


Figure 1-15 DIVIDE Instruction, Flow Diagram

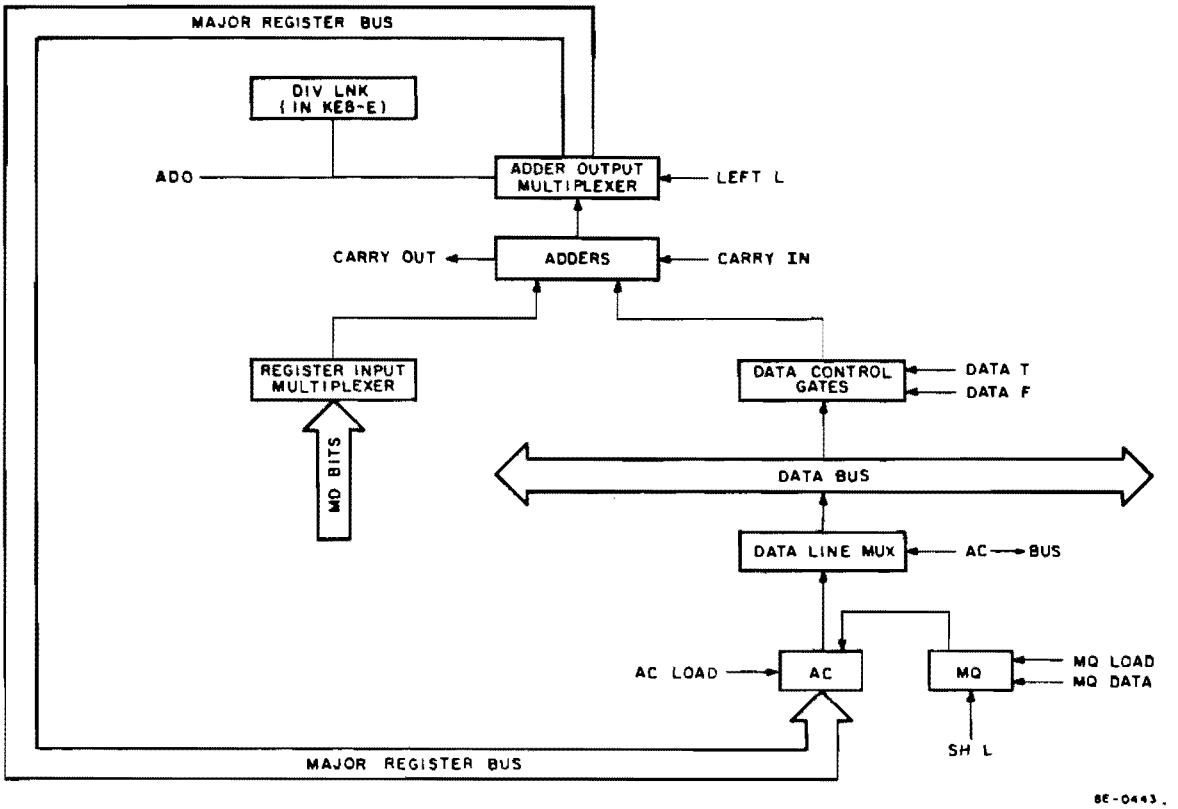


Figure 1-16 Major Registers

Table 1-1
Divide Instruction Table of Combinations

MQ10	MQ11	DIV LNK	CARRY OUT	DATA F	SIGN CHANGE	What goes into MQ11 (MQ DATA)
0	0	0	0	H	no	0
0	0	0	1	H	yes	1
0	0	1	0	H	yes	1
0	0	1	1	H	no	0
0	1	0	0	L	no	0
0	1	0	1	L	yes	1
0	1	1	0	L	yes	1
0	1	1	1	L	no	0
1	0	0	0	L	no	1
1	0	0	1	L	yes	0
1	0	1	0	L	yes	0
1	0	1	1	L	no	1
1	1	0	0	H	no	1
1	1	0	1	H	yes	0
1	1	1	0	H	yes	0
1	1	1	1	H	no	1

XOR XOR XOR

Carry	Link	Accumulator	Multiplier Quotient	Step Counter	Comments
	0	000 000 000 000	000 010 010 001		
		111 111 111 111			$\overline{AC}$ → ADDERS (FIRST STEP)
		000 000 001 100			MD → ADDERS
1		000 000 001 011			
	0	000 000 010 111	000 100 100 010	00 000	MQ0 → AC11
	0	111 111 101 000			
		000 000 001 100			
0		111 111 110 100			
	1	111 111 101 000	001 001 000 100	00 001	
		111 111 101 000			AC → ADDERS (MQ10 = MQ11)
		000 000 001 100			
0		111 111 110 100			
	1	111 111 101 000	010 010 001 000	00 010	
		111 111 101 000			
		000 000 001 100			
0		111 111 110 100			
	1	111 111 101 000	100 100 010 000	00 011	
		111 111 101 000			
		000 000 001 100			
0		111 111 110 100			
	1	111 111 101 001	001 000 100 000	00 100	
		111 111 101 001			
		000 000 001 100			
0		111 111 110 101			
	1	111 111 101 010	010 001 000 000	00 101	
		111 111 101 010			
		000 000 001 100			
0		111 111 110 110			
	1	111 111 101 100	100 010 000 000	00 110	
		111 111 101 100			
		000 000 001 100			
0		111 111 111 000			
	1	111 111 110 001	000 100 000 000	00 111	
		111 111 110 001			
		000 000 001 100			
0		111 111 111 101			
	1	111 111 111 010	001 000 000 000	01 000	
		111 111 111 010			
		000 000 001 100			
1		000 000 000 110			
	0	000 000 001 100	010 000 000 001	01 001	
		111 111 110 011			
		000 000 001 100			$\overline{AC}$ → ADDERS (MQ10 ≠ MQ11)
0		111 111 111 111			
	1	111 111 111 111	100 000 000 011	01 010	
		111 111 111 111			AC → ADDERS (MQ10 = MQ11)
		000 000 001 100			MQ0 → AC11 (MQ11 = 1)
1		000 000 001 011			
	0	000 000 010 110	000 000 000 110	01 011	
		111 111 101 001			$\overline{AC}$ → ADDERS (MQ10 ≠ MQ11)
		000 000 001 100			
0		111 111 110 101			
	1	111 111 110 101	000 000 001 100	01 100	
		111 111 110 101			Since SC = 15_8, MQ10 = 0, and MQ11 = 0;
		000 000 001 100			MQ plus AC → AC
1		000 000 000 001			(Divide Correction).
	0	000 000 000 001	000 000 001 100	01 101	NO SHIFT

Figure 1-17 Divide Example – Divides 221_8 by 14_8

Thirteen divide steps take place (the first step tests for divide overflow; the next twelve steps determine the quotient). A final remainder correction step is made as the CPU is restarted and the Link is cleared. For the correction step, the left-shift signals are all negated. If MQ10 = 1, the last regular divide step was a subtract. The AC is in complemented form before the correction step; hence DATA F must be grounded to re-complement the AC to its true form as a part of the correction process. If MQ11 = 0, the divisor must be added to the remainder (this is the correction step mentioned in the first part of this section).

Figure 1-17 shows an example of the division process.

SECTION 4 DETAILED LOGIC

1.19 EAE INSTRUCTION DECODING LOGIC

The EAE instruction decoding logic consists of the EAE Instruction Register (EIR), the MODE flip-flop, and ROM 1 and ROM 2. The decoding logic recognizes EAE commands from the processor and interprets them in terms of EAE instructions.

1.19.1 EIR Register

The EIR Register (Figure 1-18) comprises 12 D-type flip-flops (IC 74H74). It is loaded at TP2 of the FETCH major state with the 12 Memory Data bits (MD0—11) and provides outputs of EIR N(1) or EIR N(0), where N corresponds to the EIR bit designation. The most active EIR bits correspond to bits MD7—10 and play a dominant role in the EAE coding scheme. If the system is about to answer an interrupt and is not doing a data break, all flip-flops are cleared at TP4. Otherwise, the flip-flops are cleared at TP1 of FETCH.

1.19.2 MODE Flip-Flop Logic

The MODE flip-flop (Figure 1-19) comprises one J-K flip-flop (IC 74H106) which responds to SWAB and SWBA instructions. Two modes of operation were designed into the EAE to accommodate the user having programs that were written for a PDP-8/I or to accommodate the new user. Mode A corresponds to the PDP-8/I type software; Mode B corresponds to the new instructions that are provided. The EAE always starts in Mode A. The MODE flip-flop allows the programmer to switch modes at his convenience. The flip-flop is clocked at the trailing edge of TP2 whenever the basic EAE instruction in a FETCH state is decoded.

1.19.3 ROM Logic

The ROM logic (Figure 1-20) consists of two ICs, each containing a 32 8-bit word capability and selected by the combination of 5 inputs.

Figures 1-21 and 1-22 illustrate ROM operation. ROM 1 is enabled during either a FETCH or EXECUTE cycle, when an EAE instruction has been decoded and the instruction is not a mode-swapping instruction. ROM 2 is enabled during a FETCH cycle, when an EAE instruction has been decoded and the instruction is not a mode-swapping instruction. Each EAE instruction can be easily traced to eight output ROM signals, each representing a specific command to somewhere in the EAE logic. An indication of what each output is doing and where it is going can be seen at the bottom of each matrix. For the purpose of ROM decoding a 0 can be considered active and function as a 1 in normal logic terminology. For example, ROM 26 L causes an MA plus 1 to the MA. The specific EAE instructions causing this ROM instruction can be seen on the matrix.

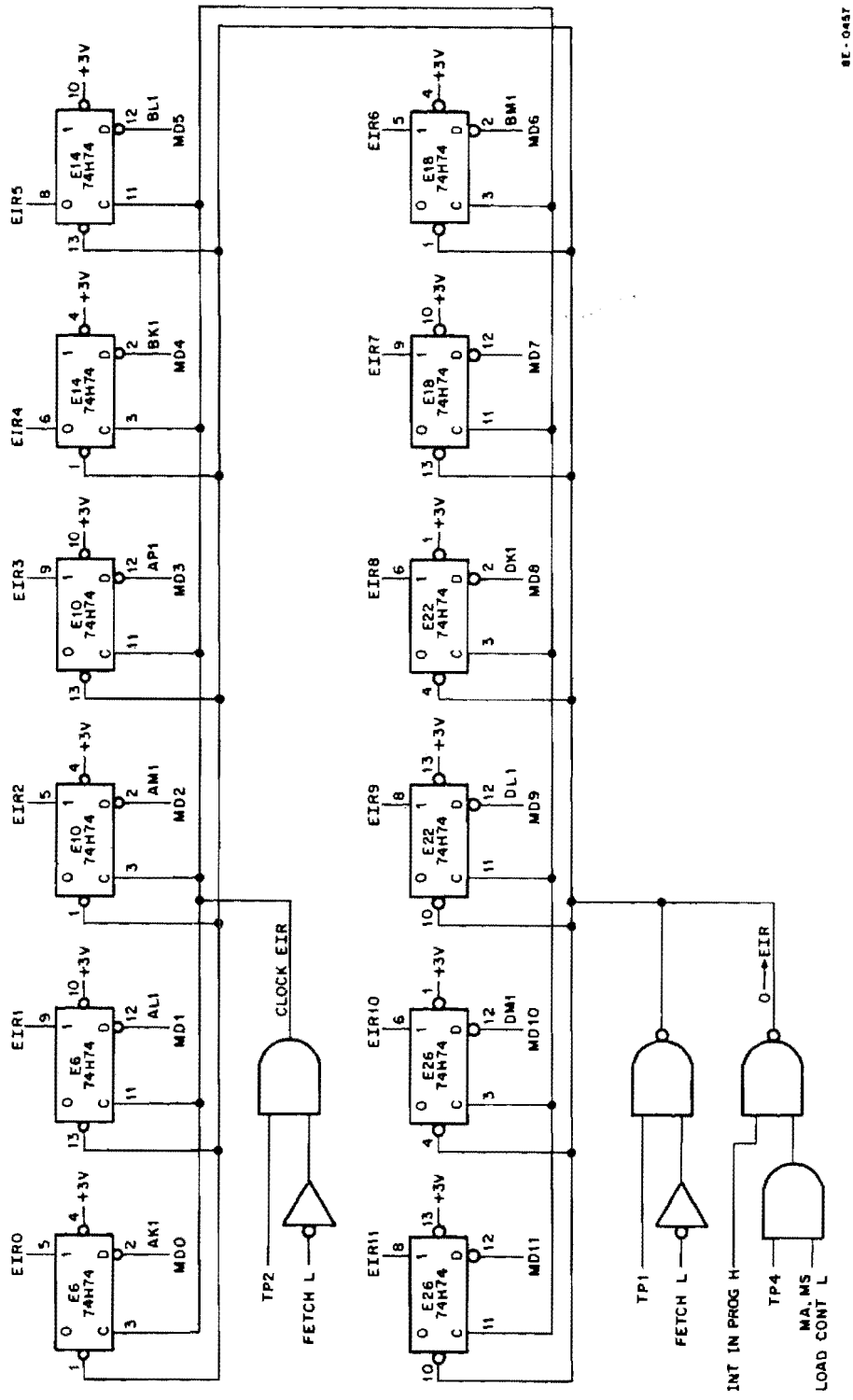


Figure 1-18 EIR Register and Controls

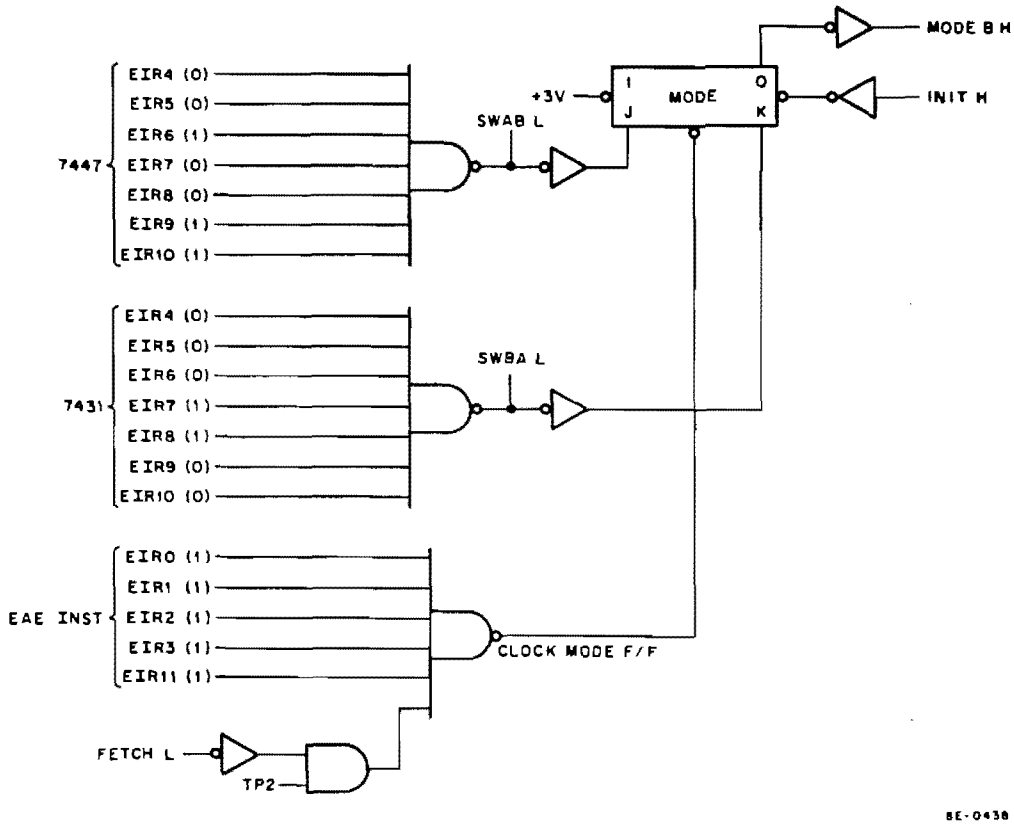


Figure 1-19 MODE Flip-Flop Logic

1.20 EAE TIMING LOGIC

The EAE timing logic is illustrated in Figure 1-23. The components consist of six D-type flip-flops, TG1 through TG4, an E SYNC flip-flop, and an EAE ON flip-flop, plus a variety of control and input gates. Flip-flops TG1 through TG4 are configured as a switch-tail ring counter. TG2(1), TG3(1) and TG4(0) are used to clock major events in the EAE logic. For example, TG2(1) L and TG3(1) L are combined to form ETP (EAE Time Pulse), which is the primary clock pulse to step the Step Counter and load registers.

The length of the switch-tail ring counter is controlled by ROM 12 L, which indicates whether an add (and possibly a shift) or merely a shift operation is taking place. If adds are taking place, the EAE Timing Generator must run at a slower rate to allow time for carries to propagate in the adders of the M8300 Major Registers Module. If ROM 12 L is high, TG1 is disabled and TG4 shifts (complemented) into TG2. Six clock pulses are required to complete the timing generator cycle; hence, ETPs are 300 ns from leading edge to leading edge. If ROM 12 L is low, TG1 is in the Shift Register. The ETPs are then 400 ns from leading edge to leading edge.

1.20.1 EAE Timing Generator Timing Diagram

A timing diagram (Figure 1-24) relates the transition from processor timing to EAE timing. The signal NOT LAST XFER L, which is grounded when the processor is to stop, is not shown on the diagram (refer to Paragraph 1.25 for information on the EAE start/stop logic). NOT LAST XFER L is asserted at TP2D; at the leading edge of

TP3, processor timing halts. The EAE Timing Generator operation begins on the leading edge of TP3, which dc sets the E SYNC flip-flop. EAE timing begins when flip-flop EAE ON is set; the timing chain is started on the next 20-MHz clock input from the processor timing generator. The first ETP occurs when TG4 is set and TG2 is reset.

ETP occurs once every 300 ns or 400 ns (depending upon ROM 12 L) and continues as long as LAST STEP, or SHIFT OK, or DCM + DPIC is not low. Any one of these signals will cause a 0 to be clocked into the E SYNC flip-flop, thus beginning a series of events that ends the EAE timing and restarts the processor timing.

Each time an ETP is generated by timing, the Step Counter is stepped one more time until the total number of shifts have been completed.

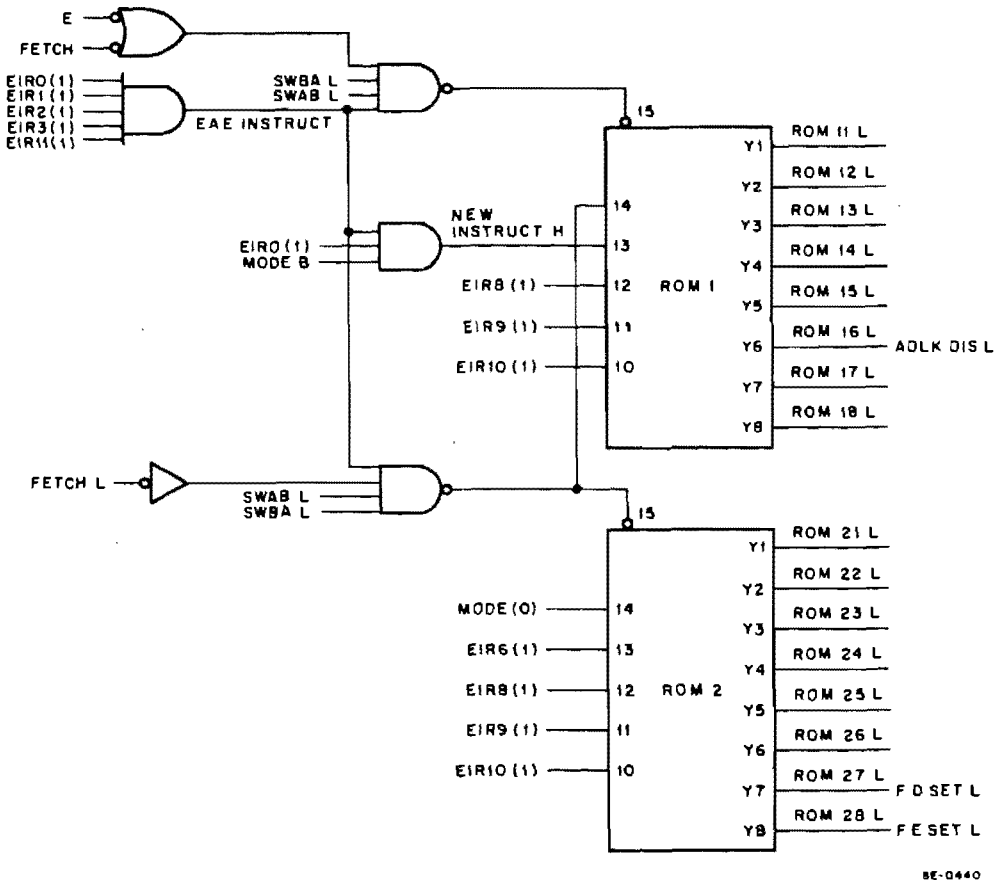
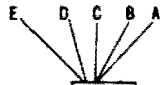


Figure 1-20 ROM Logic

1.21 EAE SOURCE CONTROL SIGNALS

EAE demands on the processor are more extensive than most other options. Data can be selected from the AC, MQ, MD, MB, PC or from the CPMA Register. How the data is selected and its source are illustrated in Figure 1-25. The AC or the MQ can be applied to one set of adder inputs via the DATA BUS. The MD, MQ, PC, or the CPMA Register can be applied to the second set of adder inputs.



	FUNCTION	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	OCTAL
00	F • NOP	1	1	1	1	1	1	1	1	377
01	F • (ACS + SCL)	1	1	1	1	1	1	1	1	377
02	F • MUY	1	1	1	1	1	0	0	1	371
03	F • DVI	1	1	1	1	1	1	1	1	377
04	F • NMI	0	1	1	0	0	0	1	1	143
05	F • SHL	1	1	1	1	1	1	1	1	377
06	F • ASR	0	1	1	1	1	0	0	1	171
07	F • LSR	1	1	1	1	1	0	0	1	371
10	F • SCA	1	1	1	1	1	1	1	1	377
11	F • DAD	1	1	1	1	1	0	0	1	371
12	F • DST	1	1	1	1	1	1	1	1	377
13	NOP	1	1	1	1	1	1	1	1	377
14	F • DPSZ	1	1	1	1	1	1	1	1	377
15	F • DPIC	1	0	0	1	1	0	0	1	231
16	F • DCM	1	0	0	1	1	0	0	1	231
17	F • SAM	1	1	0	1	1	0	0	1	331
20	E • NOP	1	1	1	1	1	1	1	1	377
21	E • SCL	1	1	1	1	1	1	1	0	376
22	E • MUY	1	0	1	1	0	1	1	1	267
23	E • DVI	1	0	1	0	0	0	0	1	241
24	NOT USED									X
25	E • SHL	0	1	1	0	0	0	1	0	142
26	E • ASR	1	1	1	1	0	1	1	0	366
27	E • LSR	1	1	1	1	0	1	1	0	366
30	NOT USED									X
31	E • DAD	1	1	0	1	1	0	0	1	331
32	E • DST	1	1	1	1	1	1	1	1	377
33	NOT USED									X
34	NOT USED									X
35	NOT USED									X
36	NOT USED									X
37	NOT USED									X

0 INDICATES ACB—LINK DATA AT TS3

1 INDICATES TG SLOW—ADD (NOT MERELY SHIFT)

1 INDICATES CARRY COUPLE AT TS3 (L—CARRY IN, CARRY OUT—L DATA)

1 INDICATES LEFT SHIFT

1 INDICATES A SHIFT OPERATION

0 DISABLES CPU ADDER LINK GATING

1 INDICATES LINK LOAD AT TP3

1 INDICATES LOAD SC AT TP2

Figure 1-21 ROM 1 Instructions

FUNCTION	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	OCTAL
NOP	1	1	1	1	1	1	1	1	377
ACS	0	1	1	1	1	1	1	1	177
NEW MUY	1	1	0	1	1	0	0	1	331
NEW DVI	1	1	0	1	1	0	0	1	331
NMI	1	1	0	1	1	1	1	1	337
SHL	1	1	1	1	1	0	1	0	372
ASR	1	1	1	1	1	0	1	0	372
LSR	1	1	1	1	1	0	1	0	372
SCA	1	1	1	1	0	1	1	1	367
DAD	1	1	1	1	1	0	0	1	371
DST	1	1	1	1	1	0	0	1	371
NOP	1	1	1	1	1	1	1	1	377
DPSZ	1	1	1	0	1	1	1	1	357
DPIC	1	0	1	1	1	1	1	1	277
DCM	1	0	1	1	1	1	1	1	277
SAM	1	0	1	1	1	1	1	1	277
NOP	1	1	1	1	1	1	1	1	377
SCL	1	1	1	1	1	0	1	0	372
OLD MUY	1	1	0	1	1	0	1	0	332
OLD DVI	1	1	0	1	1	0	1	0	332
NMI	1	1	0	1	1	1	1	1	337
SHL	1	1	1	1	1	0	1	0	372
ASR	1	1	1	1	1	0	1	0	372
LSR	1	1	1	1	1	0	1	0	372
SCA	1	1	1	1	0	1	1	1	367
SCA+SCL	1	1	1	1	0	0	1	0	362
SCA+OLD MUY	1	1	0	1	0	0	1	0	322
SCA+OLD DVI	1	1	0	1	0	0	1	0	322
SCA + NMI	1	1	0	1	0	1	1	1	327
SCA + SHL	1	1	1	1	0	0	1	0	362
SCA + ASR	1	1	1	1	0	0	1	0	362
SCA + LSR	1	1	1	1	0	0	1	0	362

0 INDICATES ACS

0 INDICATES DCM+SAM+DPIC

0 INDICATES 0→SC AT F • TP3

0 INDICATES DPSZ

0 INDICATES SCA

0 INDICATES MA+1→MA, 1→SKIP

0 INDICATES DSET

0 INDICATES ESET

Figure 1-22 ROM 2 Instructions

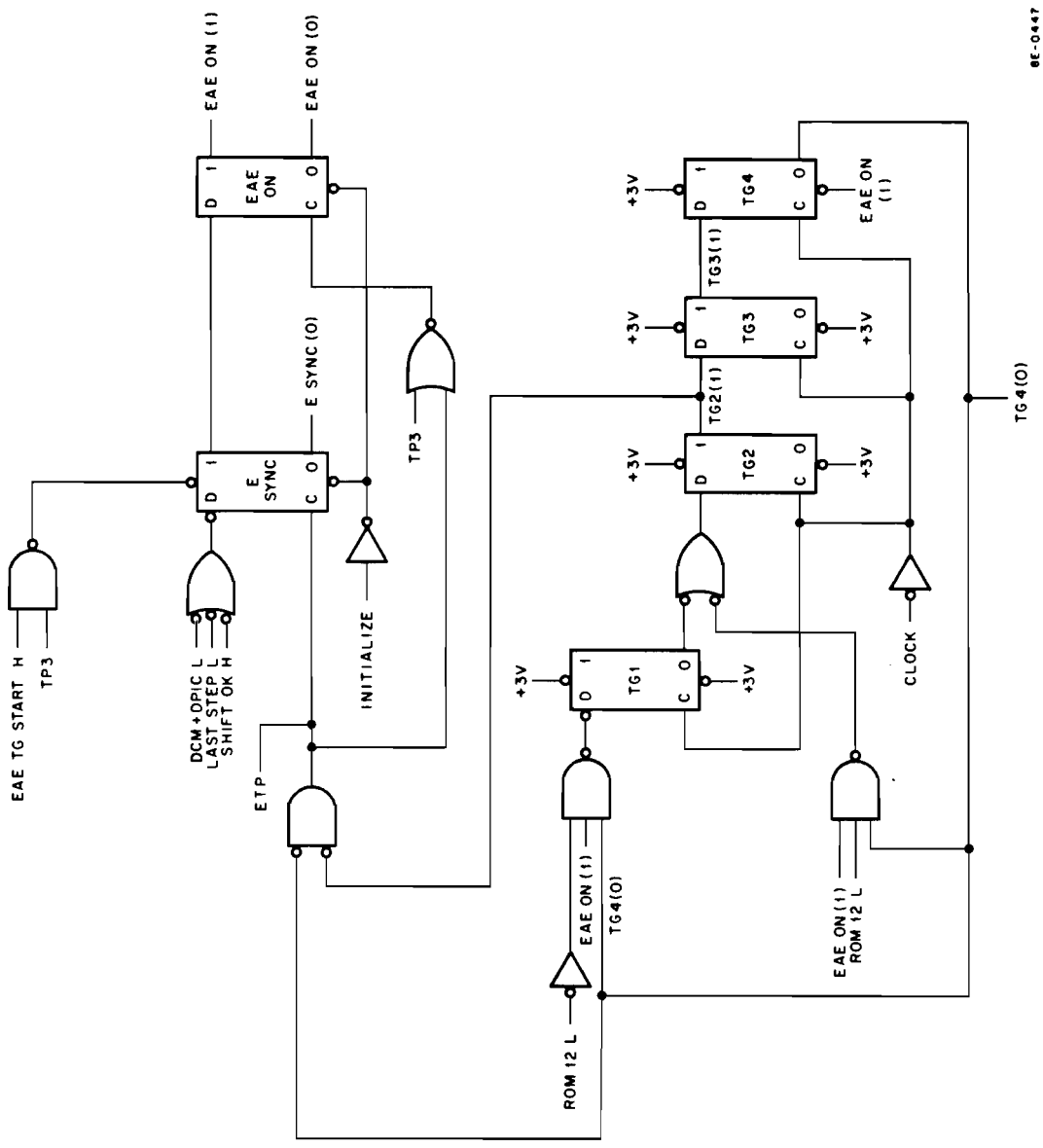


Figure 1-23 EAE Timing Logic

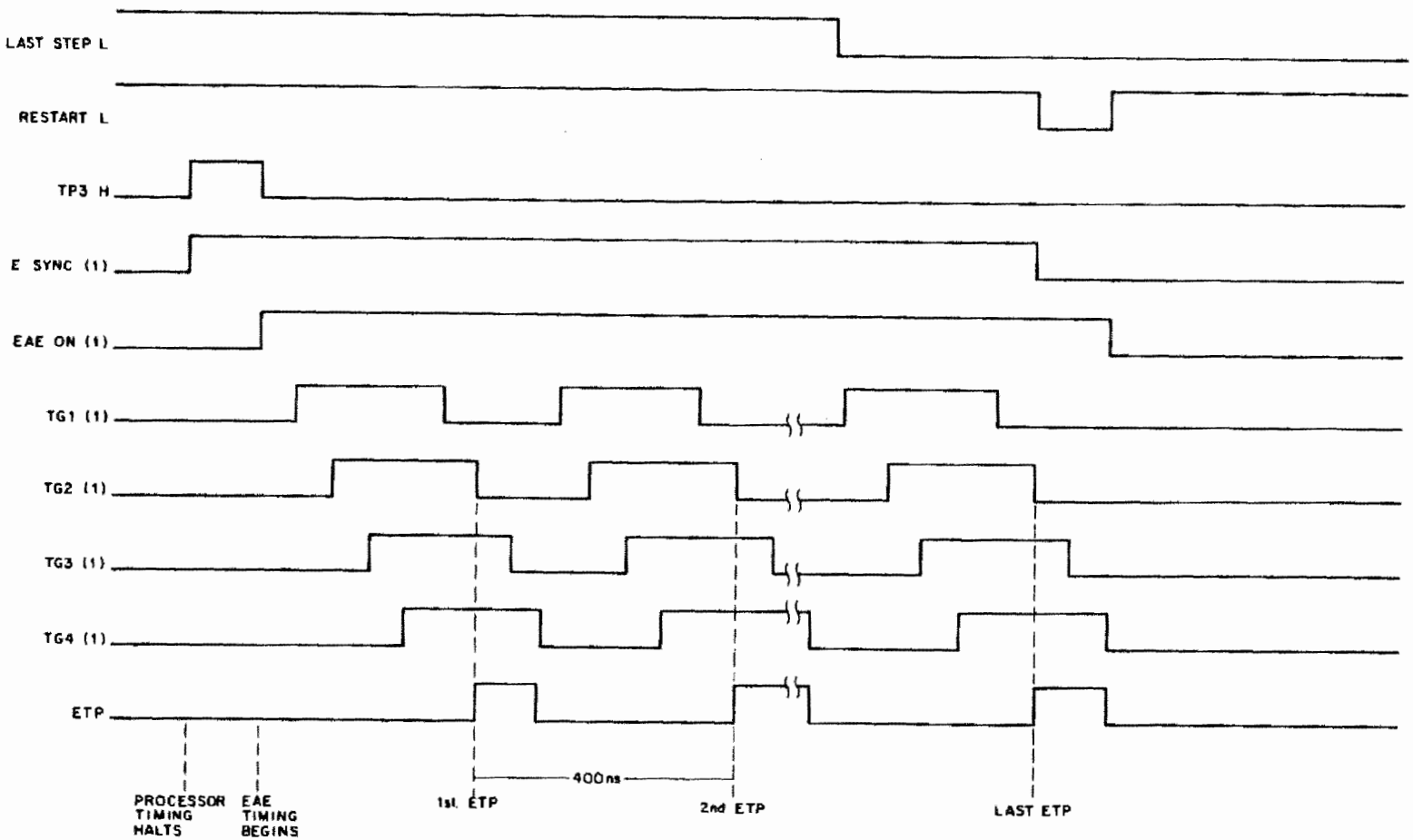


Figure 1-24 EAE Timing Generator Timing Diagram

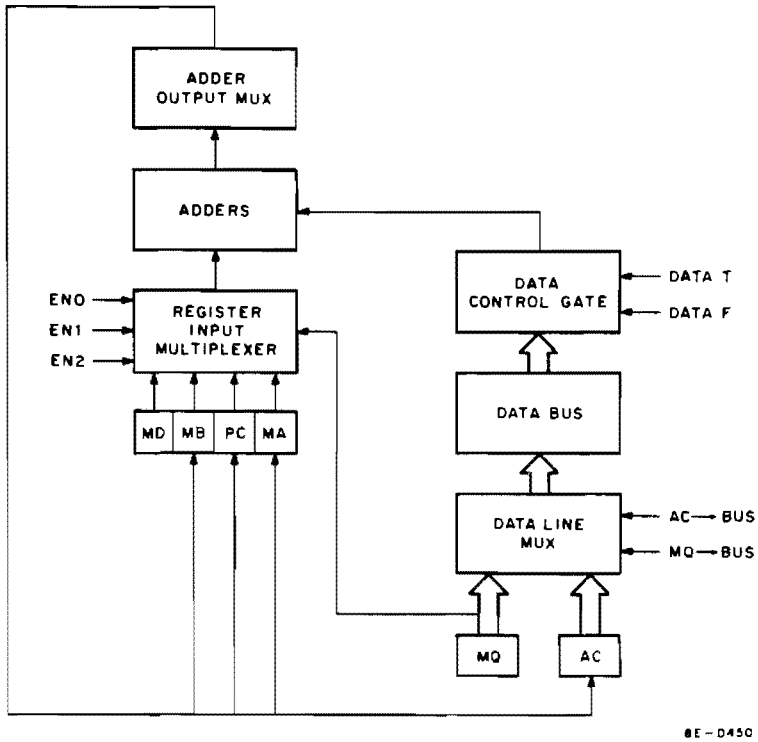


Figure 1-25 Source Control Data Path, Simplified Block Diagram

1.21.1 Register Input Enable Signals

The processor register selection logic is illustrated in Figure 1-26. Signals ENO, EN1, and EN2 determine what data will pass through the Register Input Multiplexer. The data that is selected is illustrated by the decoding scheme shown in Table 1-2.

Table 1-2
Register Select Decoding Scheme

EN0	EN1	EN2	Register or Data Selected
low	low	low	PC Register (not selected by KE8-E)
low	low	high	MD Lines
low	high	low	MQ Register
low	high	high	CPMA Register

NOTE
When EN1 is grounded by EAE, gating within the M8310 Major Registers Control automatically grounds EN0. Thus, the EAE need only ground EN1 to select MD to the adders.

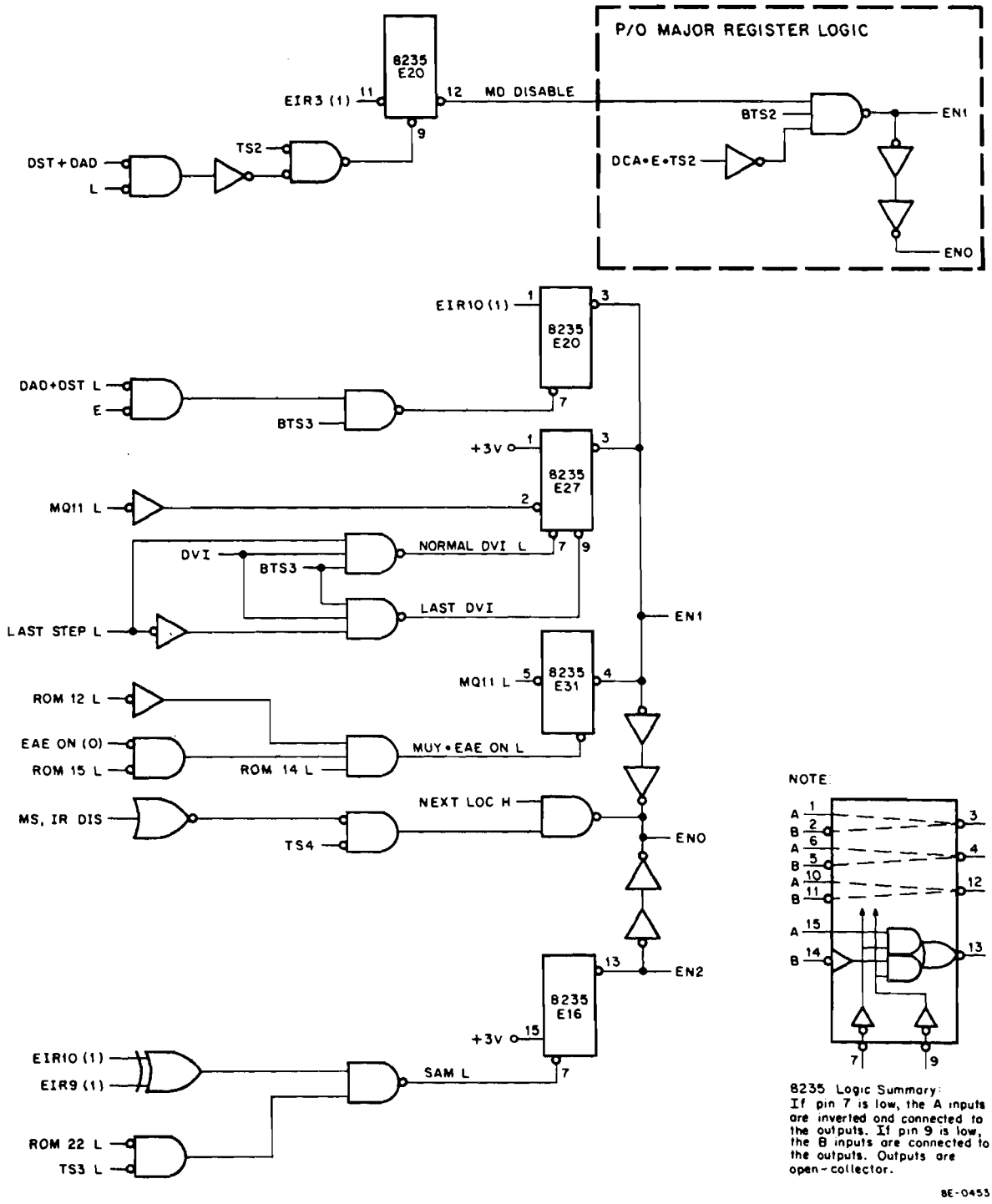


Figure 1-26 Processor Register Selection Logic

1.21.2 Data Line Enable Signals

As illustrated in Figure 1-25, signals AC → BUS and MQ → BUS are used to gate either the contents of the AC Register or the MQ Register to the DATA BUS. The generation of these two signals is shown in Figure 1-27. Signal AC → BUS occurs during all shift instructions, including MUY and DVI, as dictated by signal ROM 15 L during TS3. Signal MQ → BUS is asserted by E and NEW INSTR. E and NEW INSTR also generate AC → MQ ENA L. This arrangement automatically transfers the contents of the MQ Register to the DATA BUS and the contents of the AC Register to the MQ. The DAD and DST instructions follow this procedure. There are other times (DCM and DPIC) when MQ → BUS and AC → MQ L are asserted, but these situations are handled in the M8310 by the MQA and MQL bits (refer to Volume 1).

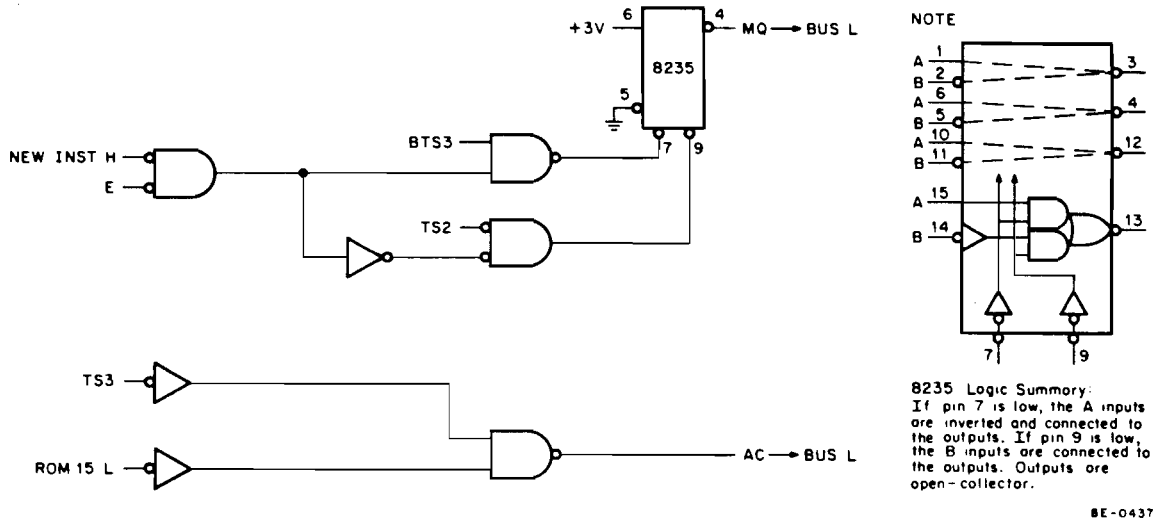


Figure 1-27 Data Line Enable Signals

1.21.3 Data Enable Signals

Signals DATA T and DATA F allow data to move from the DATA BUS to the adders. The type of data being applied to the adders is illustrated in Table 1-3.

Table 1-3
EAE Combinations of DATA T and DATA F

Signal		Type of Data Applied to Adders
DATA T	DATA F	
low	low	Complement of contents of DATA BUS
low	high	Contents of DATA BUS
high	low*	Zero

*DATA F is grounded by a gate in the M8310 Register Control if DATA T is high.

The data usually placed on the DATA BUS by the EAE option will be the contents of the AC Register or the MQ Register. Signals DATA T and DATA F can be asserted by either the Major Registers Control logic or the EAE data control logic (Figure 1-28). Signal DATA T is brought low during TS2 when a DAD + DST instruction is being executed during an EXECUTE cycle. However, DATA T is also brought low during TS3 of all EAE cycles, as described in Paragraph 3.35.3 of Volume 1.

DATA F is pulled low by a SAM, DCM, or DPIC instruction during TS3. During a normal DVI, DATA F is low if MQ10 and MQ11 are alike. During the last divide step (the correction step), LAST STEP L tests MQ10 for a 1.

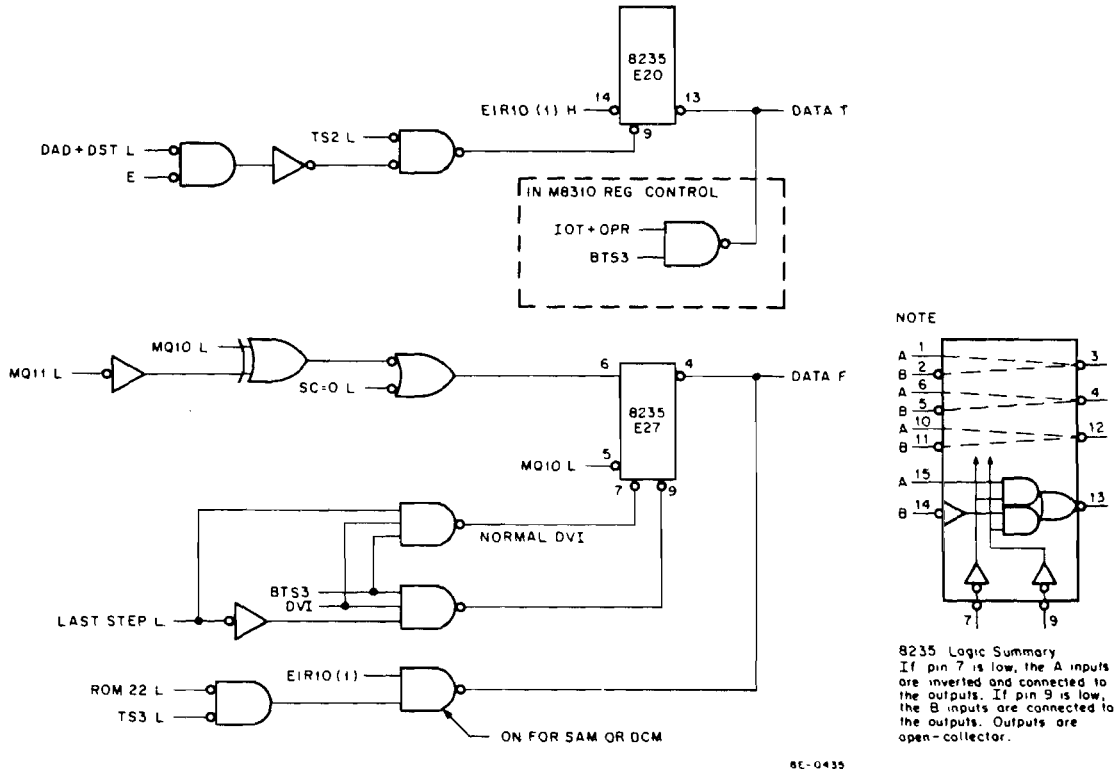


Figure 1-28 EAE Data Control Logic for Processor Data Control Gates

1.22 EAE ROUTE CONTROL SIGNALS

The EAE route control signals control shifting right, shifting left, carry in, and carry out. These elements are represented in the simplified block diagram given in Figure 1-29.

1.22.1 Step Counter Loading and Control Logic

The Step Counter loading and control logic is illustrated in Figure 1-30. The logic controls loading, reading, and incrementing the Step Counter.

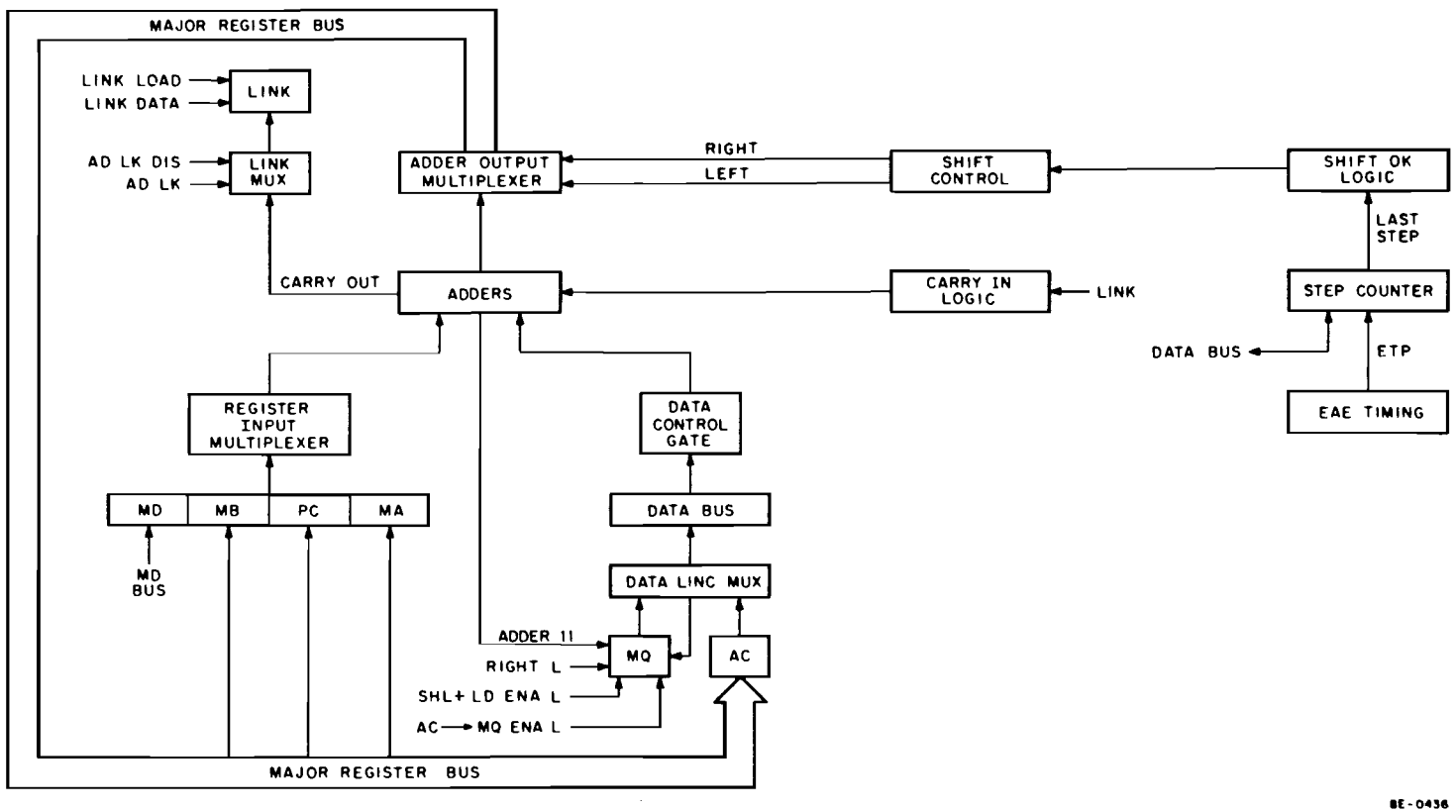


Figure 1-29 EAE Route Control Signal Block Diagram

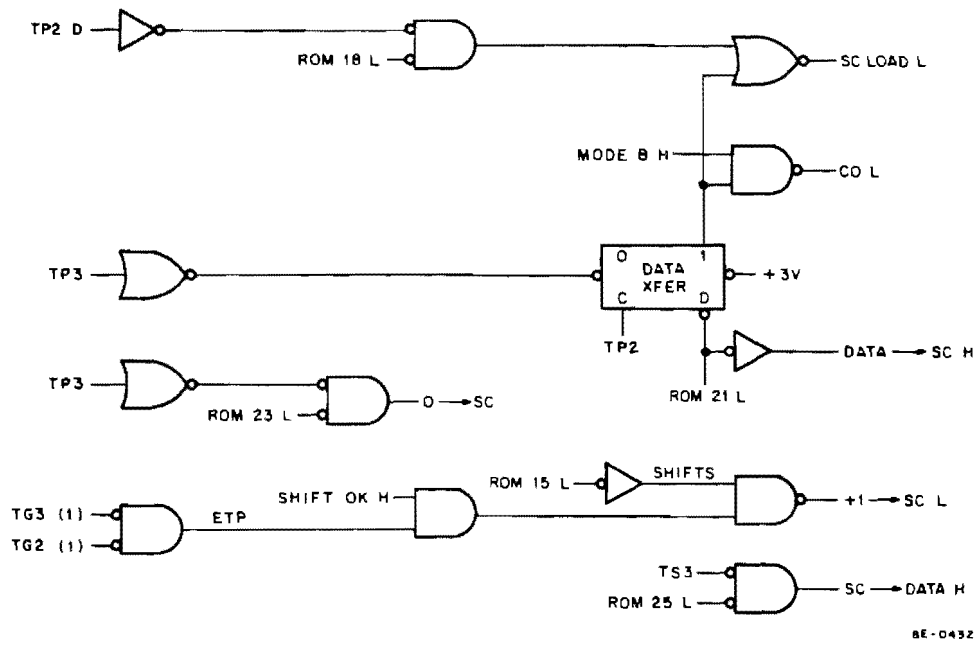


Figure 1-30 Step Counter Loading and Control Logic

Two sources of data (the AC and MD) can be loaded into the SC by two different instructions. If the last five bits of the AC are to be loaded into the Step Counter, the ACS instruction generates ROM 21 L, which sets the DATA XFER flip-flop at TP2. Because ACS is a Mode B instruction, MODE B H and DATA XFER (1) H generate CO L and DATA XFER (1) asserts SC LOAD L. Signal CO L is used to clear the AC Register at the same time the SC is loaded. Signal DATA → SC H is used to gate the contents of the DATA BUS to the Step Counter. At TP3, DATA XFER is cleared and the SC is loaded.

If an SCL, SHL, ASR, or LSR instruction is decoded and the major state is EXECUTE, ROM 18 L is asserted. At TP2D, SC LOAD L is asserted, which causes the complement of the last five MD bits to be loaded into the Step Counter. Signal +1 → SC L is generated at EAE Timing Pulse (ETP) time by SHIFT OK H and ROM 15 L. ROM 15 L is decoded when a shift operation is to take place.

When it is desired to load the AC Register with the contents of the Step Counter, instruction SCA asserts ROM 25 L. During TS3 L, SC → DATA H is asserted, gating the contents of the Step Counter to the DATA BUS.

1.22.2 Step Counter Logic

The Step Counter logic is illustrated in Figure 1-31. IC 8266 transmits the complement of DATA 7 through 11 and the uncomplemented MD bits. When signal DATA → SC goes high, the DATA BUS bits (low for 1) are applied to the Step Counter (high for 1). Otherwise, bits MD7–11 (low for 1) will be complemented and applied to the Step Counter (high for 1). The Step Counter loads the contents of the four input lines when SC LOAD L is received and increments when it receives the signal +1 → SC L. The Step Counter is IC 74193. It is usually used to count up to zero. When the count reaches 0, signal SC = 0 L is asserted.

Signal LAST STEP L is generated when SC = 13_8 during an MUL, when SC = 14_8 during a DVI, and when SC = 37 for all operations.

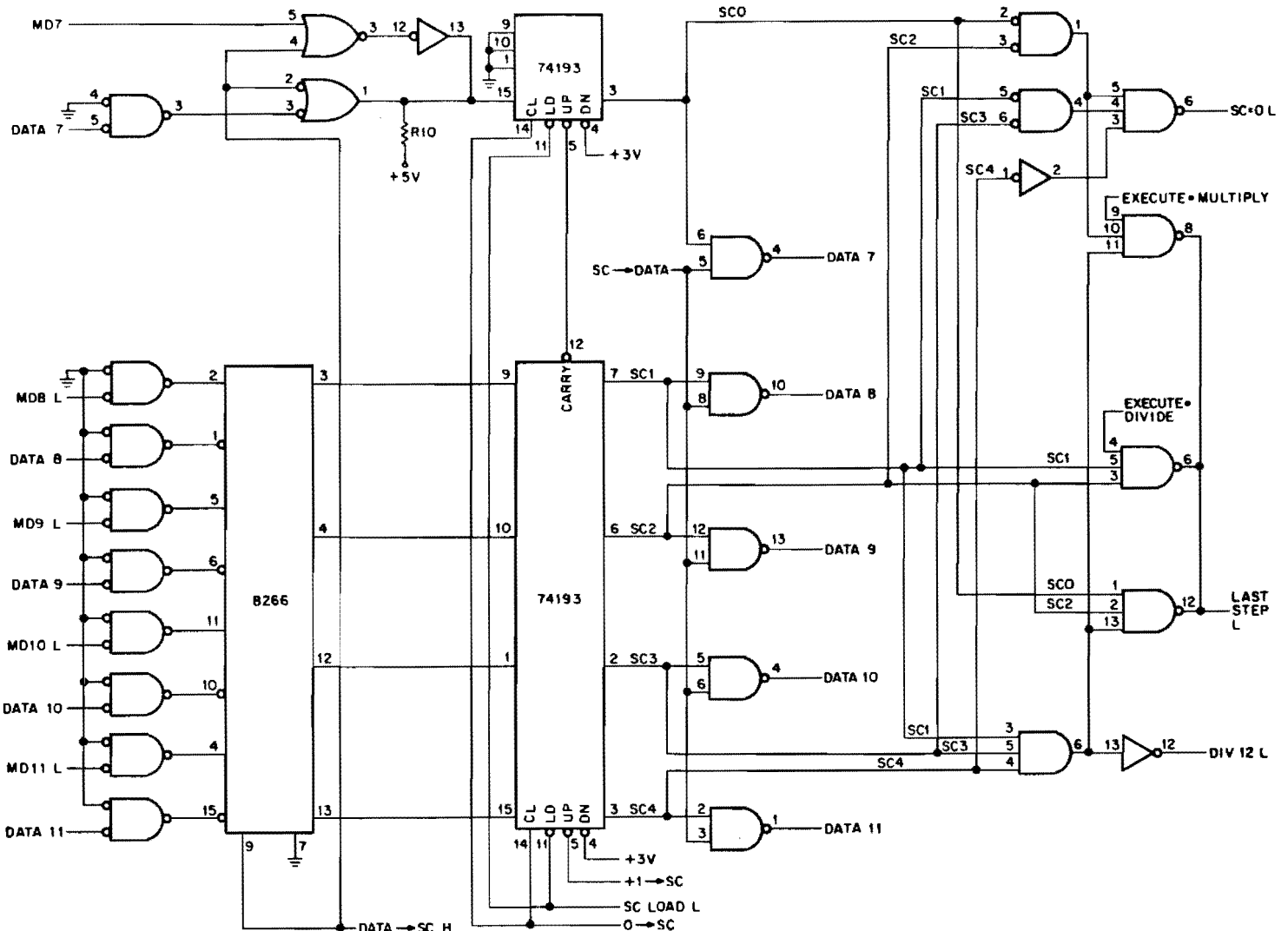


Figure 1-31 EAE Step Counter

1.22.3 Shift Right/Shift Left Control Logic

The shift right/shift left control logic is illustrated in Figure 1-32. The 8235 ICs are multiplexers that receive select signals at pins 7 or 9 to select signals being received at pins 1, 2, 10, or 14. LEFT L is enabled during an SHL, NORMALIZE, or DVI instruction. The shift-right (ASR or LSR) and multiply instructions cause RIGHT L to be asserted.

1.22.4 Shift OK Logic

The shift OK logic (Figure 1-33) monitors the contents of the AC and MQ during the NMI instruction and checks for LAST STEP L. When LAST STEP L becomes low during an SHL, LSR, or ASR instruction, SC = 37. If the MODE flip-flop is set, indicating the new or Mode B instruction set is in use, SHIFT OK H is grounded to prevent the last shift from occurring. During an NMI instruction, SHIFT OK H is grounded when the number becomes normalized, to prevent an extra shift from taking place as the processor is restarted.

1.22.5 EAE Carry In Logic

Signal CARRY IN L is developed by the EAE under the conditions shown in Figure 1-34. ROM output, ROM 22 L is decoded when an SAM, DCM, or DPIC instruction is to be performed. If the EAE is off (SAM or Step 1 of DPIC and DCM) CARRY IN L is generated. ROM 13 L controls the coupling of carries, and introduces a CARRY IN L if the Link is set during Step 2 of DPIC and DCM and the two EXECUTE cycles of the DAD instruction.

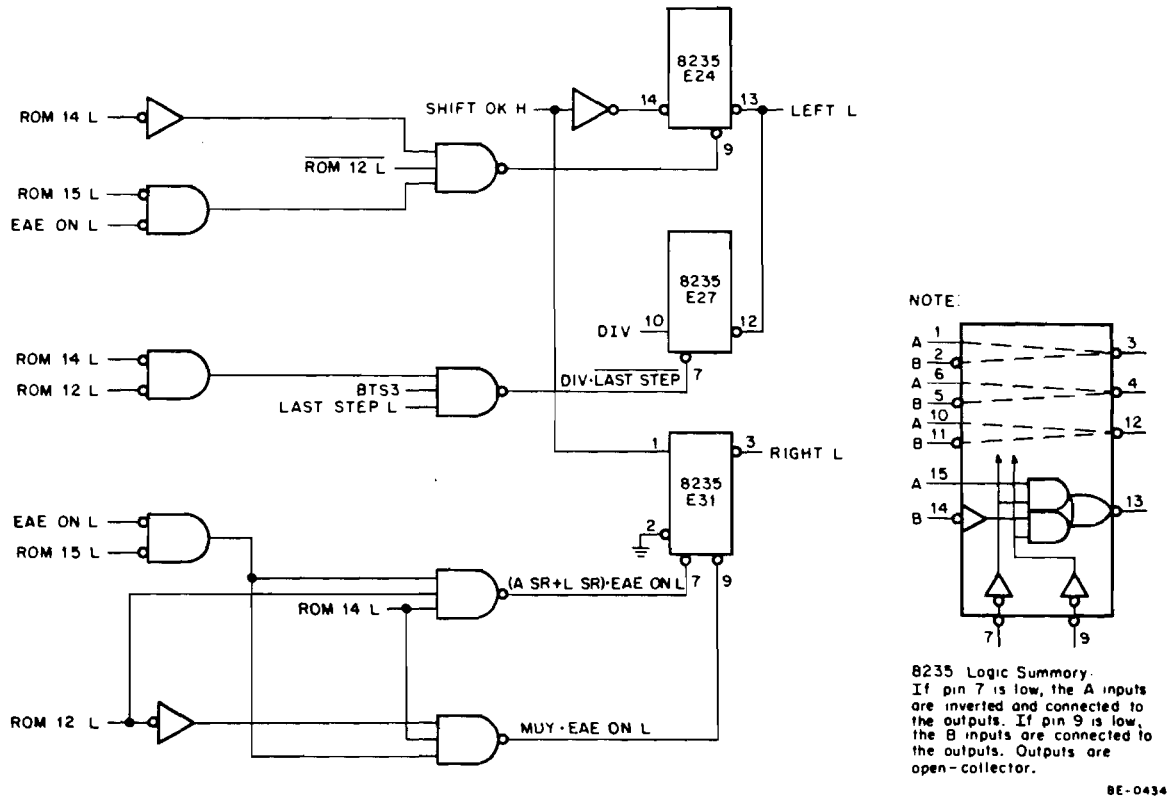


Figure 1-32 EAE Shift Right/Shift Left Control Logic

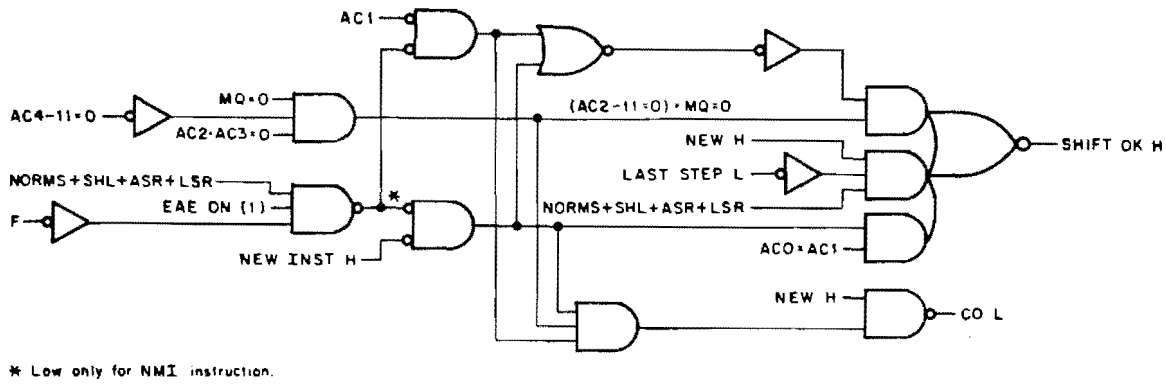


Figure 1-33 Shift OK Logic

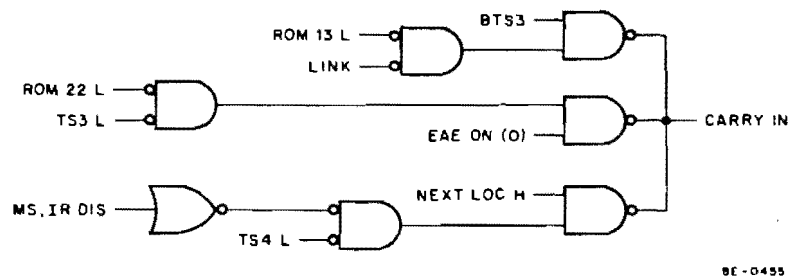


Figure 1-34 EAE Carry In Logic

1.22.6 MQ Register Shift Left Logic

The MQ Register shift left logic is illustrated in Figure 1-35. Decoded outputs ROM 14 L and ROM 15 L, with EAE, select such signals as SHIFT OK H and CARRY OUT L, etc. Signal MQ DATA L provides quotient information to MQ11 during a divide. Otherwise, MQ DATA L remains high, shifting zero into MQ11 for NMI and SHL instructions. Signal SHL + LD EN L forces the MQ Register to shift one place to the left. Also shown with this logic is the DIV LINK and other gating required to generate the quotient bit.

1.22.7 AC to MQ Transfer Signals

Figure 1-36 illustrates the conditions when the signal AC → MQ ENA L can be asserted. This gating is used during the DAD and DST instruction as a part of the AC → MQ swapping process. AC → MQ ENA L is also generated for the DPIC and DCM instructions in the M8310, as described in Volume 1, Paragraph 3.40.

1.23 DESTINATION CONTROL SIGNALS

The signals that actually cause register loading are called destination control signals. In the case of the EAE, only the AC LOAD L and MQ LOAD L signals are developed in the EAE logic. Other loading signals, including MB LOAD L, are asserted by the processor. Figure 1-37 illustrates how the AC, MQ, and MB Register loading signals are generated.

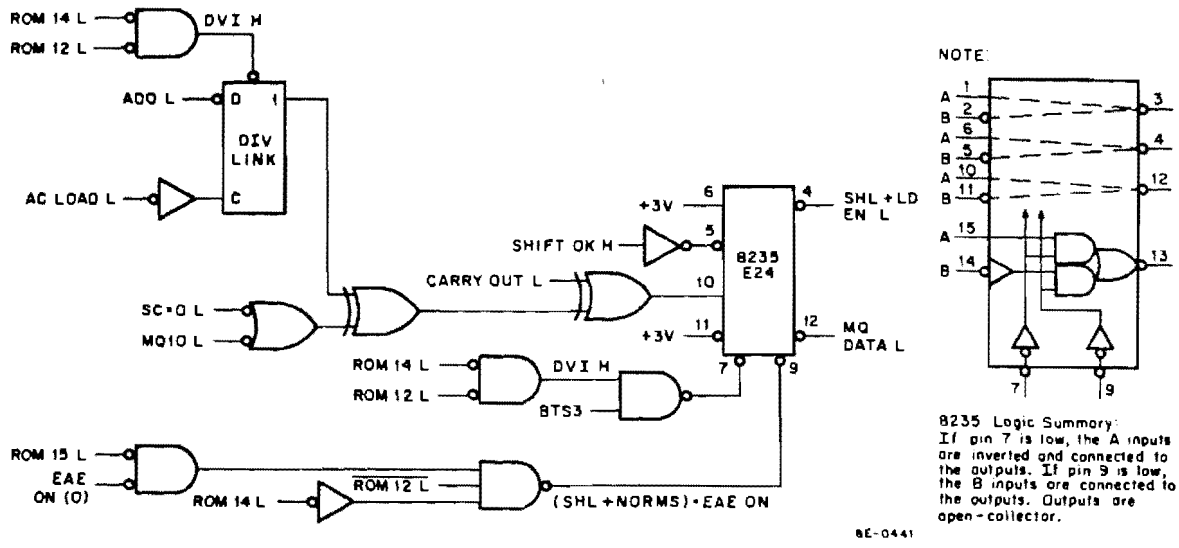


Figure 1-35 MQ Register Shift Left Logic

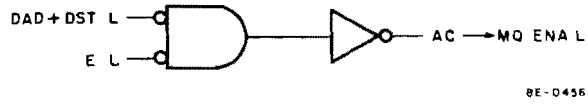


Figure 1-36 AC → MQ TRANSFER Signal

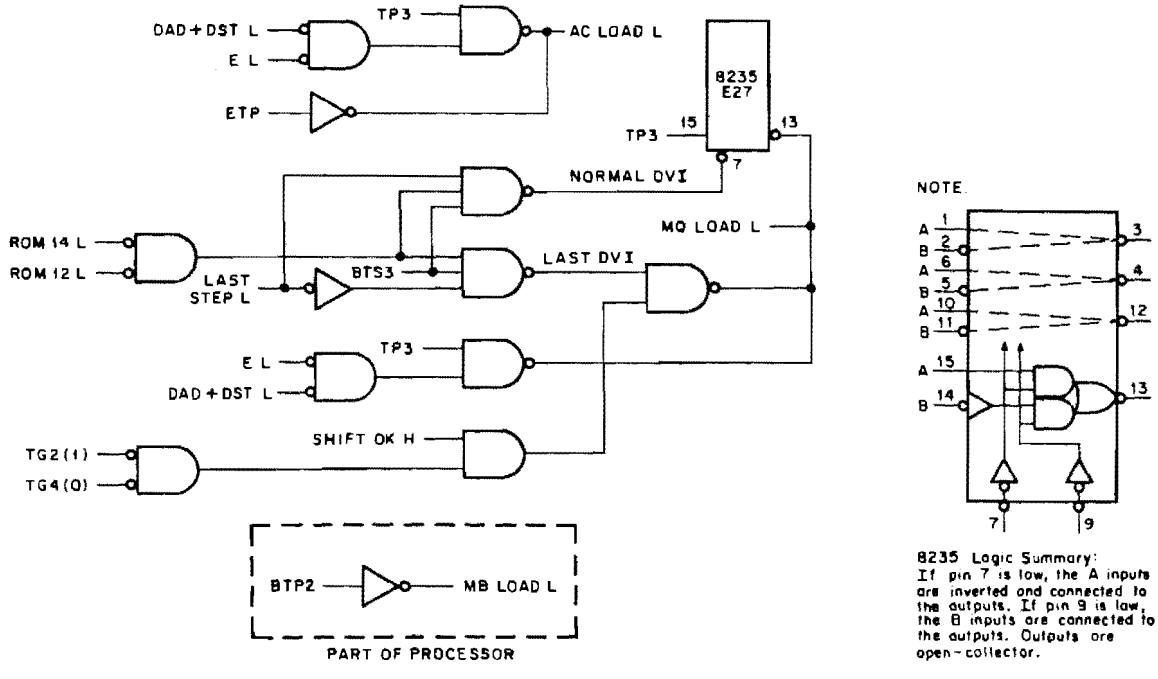


Figure 1-37 Destination Control Logic

1.24 EAE START/STOP LOGIC

The EAE start/stop logic, shown in Figure 1-38, transfers timing generation from the CPU to the EAE. Up to TP3 of certain EAE cycles the generation of all timing signals is under CPU control. At TP3, timing control can be transferred to the EAE to allow high-speed multiple shifts and/or adds. At the conclusion of these special operations, timing is returned to the CPU.

The EAE grounds OMNIBUS signal NOT LAST XFER L before TP3 when the current instruction and major state requires running the EAE timing chain. The NLX flip-flop is clocked 100 ns after the trailing edge of TP2, after the ROMs and associated decoding have had ample time to settle. If the D input to NLX is high, one of the following instructions has been decoded; the major state is the one in which the EAE operation is to take place.

Instructions which start the EAE Timing Chain:

ASR, LSR, SHL, NMI, MUY, DVI, DPIC, DCM

The output of the NLX flip-flop is applied to one input of a two-input NAND gate (labeled A in Figure 1-38) whose output grounds NOT LAST XFER L. At the same time, the other input of gate A is high, unless a Divide Overflow situation is detected by gate B. If NOT LAST XFER L is low at the leading edge of TP3, CPU timing is interrupted as described in Volume 1, Paragraph 3.21. EAE TG START H is ANDed with TP3 and the result used to set the E SYNC flip-flop. At the trailing edge of TP3, the EAE ON flip-flop is clocked and sets. The EAE's timing chain is now running.

The EAE continues to run until some condition within the EAE causes EAE STOP H to go high. At the leading edge of the next ETP, E SYNC clears. At the trailing edge of the same ETP, EAE ON clears and stops the EAE. Signal RESTART L has the same effect on the Timing Generator of the CPU (but not the Major Registers) as does BUS STROBE L – it starts the CPU if NOT LAST XFER L is high. RESTART L is generated twice, once when the EAE starts (it has no effect then), and once when the EAE stops. NOT LAST XFER L is high by the time the second RESTART L signal is generated, because NLX is cleared by one of the EAE's timing generator flip-flops (TG2) a short time after the trailing edge of TP3 and well before the leading edge of the first ETP.

1.25 EXTENDED EAE LOGIC

The extended EAE logic (Figure 1-39) consists of a D-type flip-flop called EX1 and its associated logic. When set, EX1 forces a second EXECUTE cycle and causes the processor to access the next sequential memory location. Signal NEXT LOC H is used to generate CARRY IN L and to ground EX0, which causes MA + 1 to the MA Register.

The logic gating for the EX1 data input is limited to either a DAD or DST instruction. DAD, DST, MUY, and DVI are the only EAE instructions that enter a DEFER cycle. For both MUY and DVI instructions, EIR6(1) is low and, therefore, prevents the EX1 flip-flop from being set.

1.26 EAE LINK CONTROL LOGIC

The EAE link control logic (Figure 1-40) contains all of the Link Control elements required to load the Link and to disable the Link so that it is not affected by certain processor-EAE operations. For a better understanding of Link operation within the processor, refer to Volume 1, Paragraph 3.39.

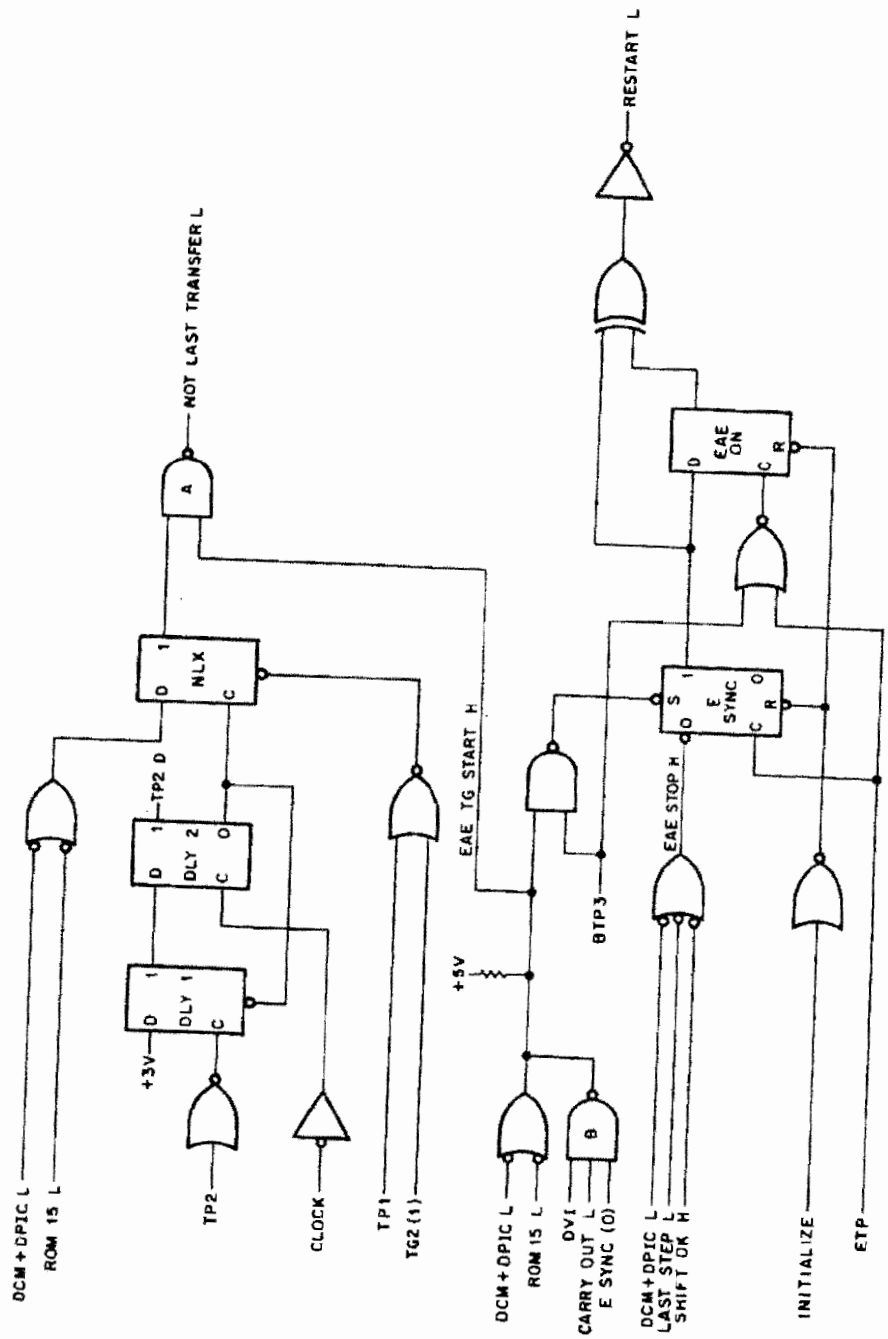


Figure 1-38 EAE Start/Stop Logic

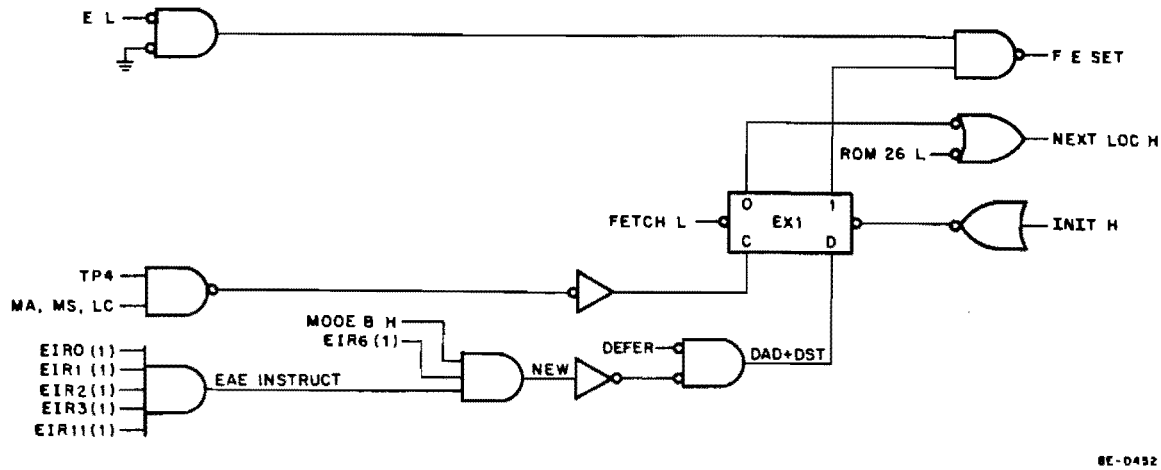


Figure 1-39 Extended EAE Logic

Signal LINK DATA L provides information to be loaded into the Link by the LINK LOAD L pulse. This information may be one of the following:

Enable	Result if Link Loaded	Used By
None	Zero	DVI (to clear overflow indication) MUY, LSR, DAD
ROM 11 L	Carry from adders	DAD, DPIC, DCM
ROM 13 L	AC0	ASR, NMI
DVI EAE ON (0)	One	DVI (to anticipate overflow)

The LINK LOAD L signal is generated at TP3 if ROM 17 is low (usually to preset the Link) and at ETP time during left shifts. During the execution of DVI, the Link is set at TP3 (for possible overflow indication) and cleared if the DVI process reaches LAST STEP L (meaning a legal divide has occurred). During the right shift and MUY instructions, the Link is not modified.

During left shifts, data is introduced from MQ0 to AC11 via a line called ADLK L. During SHL or NMI instructions, MQ0 is gated directly to ADLK L. For DVI, MQ0 is sometimes inverted before being applied to ADLK L. The gating logic, which depends on MQ11 and whether the first divide step has taken place (EAE ON), is also shown in Figure 1-40.

The ADLK DIS L signal disables the normal Link gating described in Volume 1, Paragraph 3.39. ADLK DIS L must be low any time the LINK LOAD L/LINK DATA L inputs are used or whenever left shifts are performed in order to avoid conflict with the normal link gating. This line is grounded by ROM 16 L.

1.27 EAE SKIP LOGIC AND GT FLAG

The EAE skip logic is illustrated in Figure 1-41. There are two methods of generating a SKIP L signal. Signal NEXT LOC H, which is generated by the extended EAE logic, is inverted and applied to the SKIP L line. Signal NEXT LOC H also generates CARRY IN L.

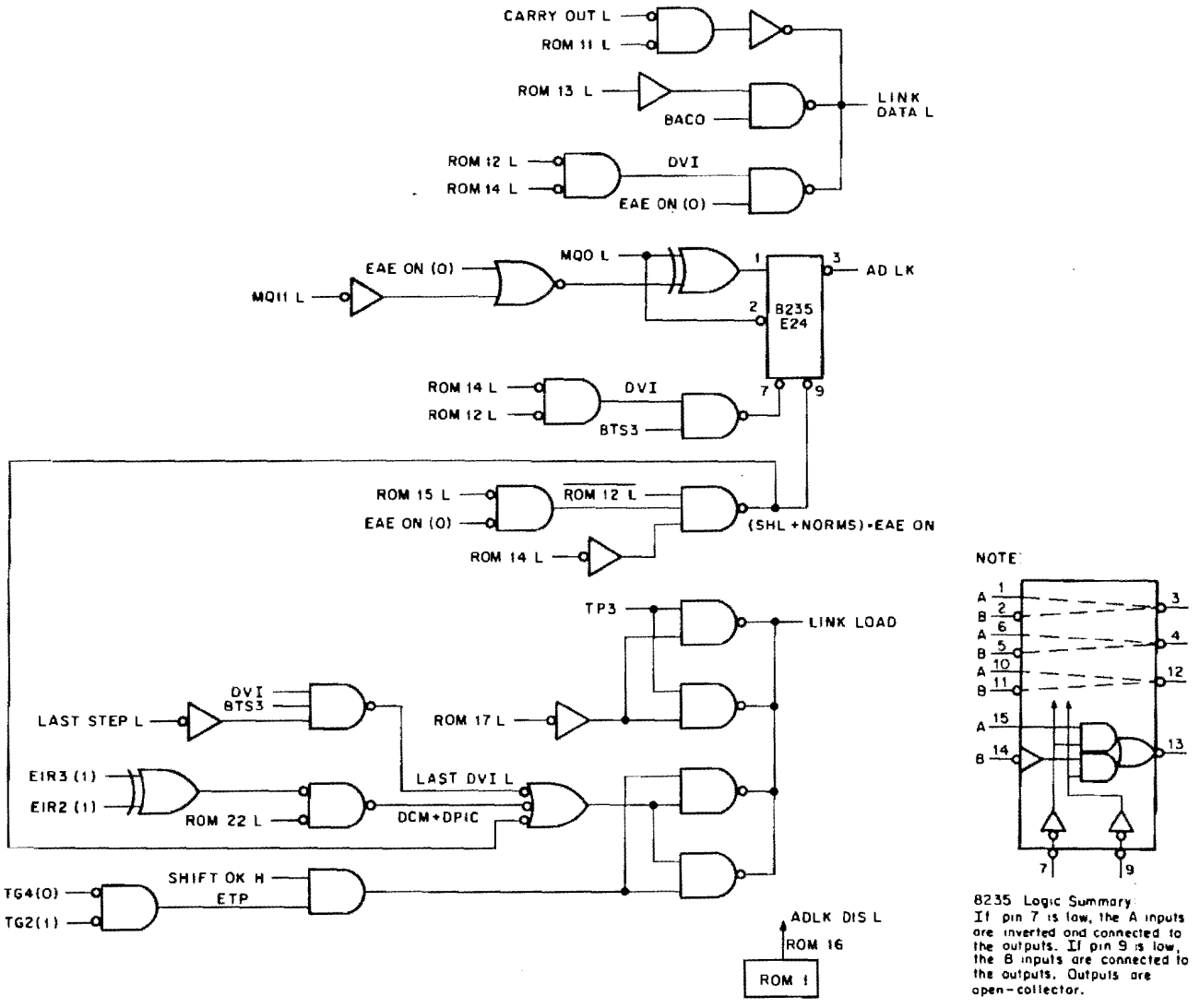


Figure 1-40 EAE Link Control

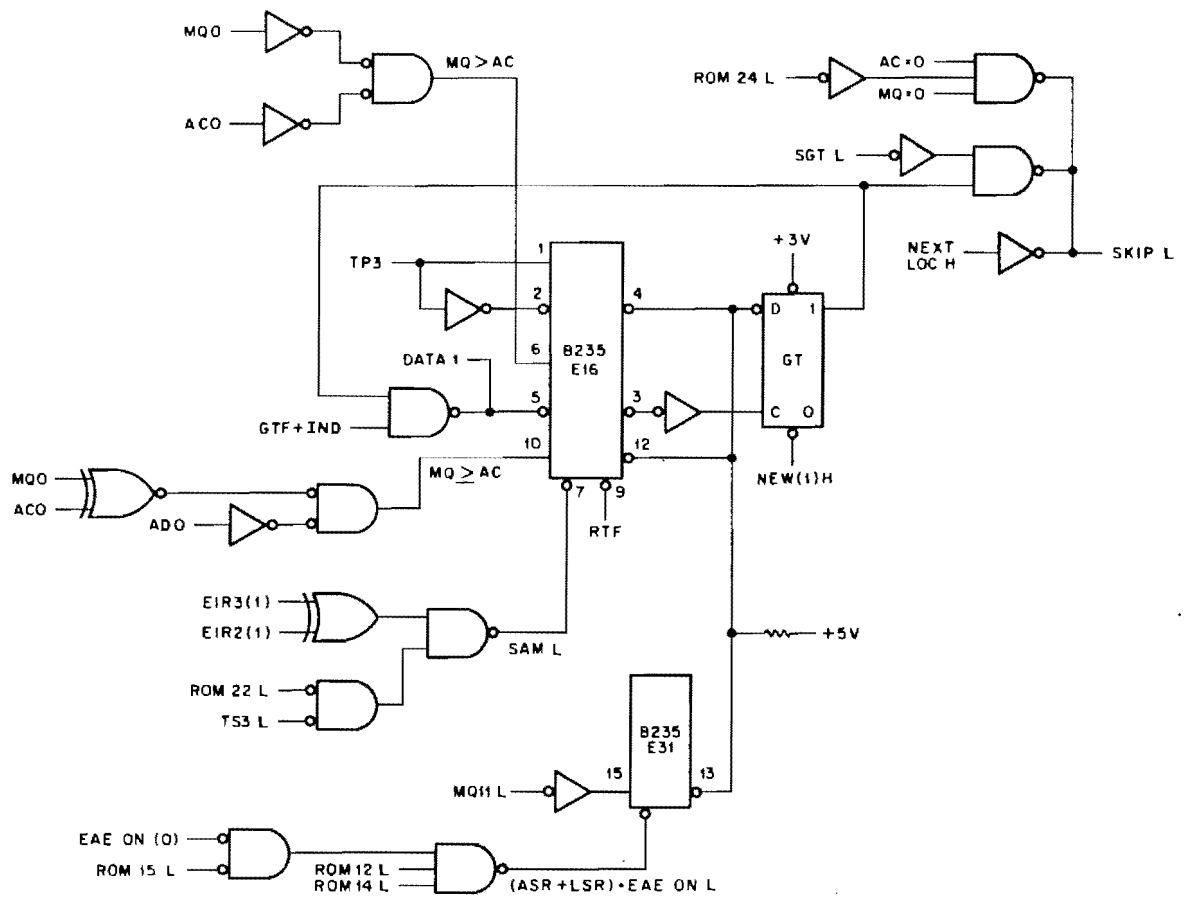


Figure 1-41 EAE Skip Logic

The SKIP line is also grounded when a DPSZ instruction is being performed. This instruction tests the AC and the MQ for 0. If both registers equal zero, the SKIP L signal will be asserted. SKIP L is used to set the SKIP flip-flop in the processor's logic at TP3. For information on the processor's skip logic, refer to Volume 1, Paragraph 3.38.

The more complex part of the skip logic involves the Greater Than (GT) flag. When the GT flip-flop is set, instruction SGT forces the SKIP line to go low. The GT flag can be changed by one of three methods:

Method	Source of Information
RTF instruction	DATA 1
SAM instruction	Set if MQ greater than or equal to AC; cleared otherwise.
ASR or LSR	Previous MQ11

The GT flag is cleared if the EAE is in Mode A; hence, the flag is active only for Mode B instructions.

SECTION 5 MAINTENANCE

Since the EAE is physically connected to the Central Processor Timing Generator and OMNIBUS, a definite possibility exists that some EAE malfunctions will not be caused by the EAE modules. For this reason, the following procedure is suggested.

Step	Procedure
1	Remove M8340 and M8341 from the OMNIBUS.
2	Perform processor-related diagnostics to ensure processor reliability.
3	Insert M8340 in OMNIBUS and connect it to the Timing Generator via the "J" top connector.
4	Perform PDP-8/E Instruction Tests 1 and 2.
5	Insert M8341 in OMNIBUS and connect it to M8340 connector H.
6	Perform PDP-8/E Instruction Tests 1 and 2.
7	Connect M8341 to M8310 via "F" connector.
8	Perform PDP-8/E Instruction Tests 1 and 2.

NOTE

If problems are encountered during this procedure, they can be isolated by troubleshooting the processor, and tracing the malfunction back to the last module or connector that was added to the system.

When this procedure fails to isolate a malfunction, perform EAE Instruction Tests 1 and 2, and the EAE Extended Memory Test. One of these tests should give some idea of the problem.

Once the problem is pinpointed, write and toggle in a simple program using the malfunctioning instruction.

The following basic ideas can be built upon or modified to suit any special purpose:

1. Mode Changing Instructions

```
0/  7431    SWAB
    7447    SWBA
    5000    JMP 0
```

2. SCL or ACS (dependent on mode)

Mode A

```
0/  7604    LAS
    3003    DCA.+2
    7403    SCL
    XXXX    OPERAND
    5000    JMP 0
```

Mode B

```
0/  7431    SWAB
    7604    LAS
    7403    ACS
    5001    JMP .-2
```

3 SCA or SCA, CLA

Mode A

```
0/  7604    LAS
    3003    DCA .+2
    7403    SCL
    XXXX    OPERAND
    7441 or SCA or SCA, CLA
    7641
    7000 ┐
     ↓   } 5 or 6 NO OPS WILL HOLD THE AC
    7000 ┘ FOR OBSERVATION
    5000    JMP 0
```

Mode B

```
0/  7431    SWAB
    7604    LAS
    7403    ACS
    7441 or SCA or SCA, CLA
    7641
    7000 ┐
     ↓   } NO OPS TO HOLD AC FOR OBSERVATION
    7000 ┘
    5001    JMP 1
```

4. SHL Shift Left

Mode A

```
0/  7604    LAS (Shift Count = one more than the last five bits of the location following SHL)
    3006    DCA
    1050    TAD MQ
    7421    MQL
```

(continued on next page)

```
1051    TAD AC
7413    SHL
XXXX    SHIFT COUNT
7000  ┐
 ↓    } NO OPS WILL HOLD AC AND MQ FOR OBSERVATION
7000  ┘
7621    CAM
5000    JMP 0
```

Mode B

```
0/ 7604    LAS (Shift Count = last five bits of location following SHL)
   3007    DCA
   7431    SWAB
   1050    TAD MQ
   7421    MQL
   1051    TAD AC
   7413    SHL
   XXXX    SHIFT COUNT
   7000  ┐
    ↓    } NO OPS WILL HOLD AC AND MQ FOR OBSERVATION
   7000  ┘
   7621    CAM
   5001    JMP 1
```

5. NMI

```
0/ 7431    SWAB (Start here if Mode B)
1/ 1050    TAD MQ (Start here if Mode A)
   7421    MQL
   1051    TAD AC
   7411    NMI
   7000  ┐
    ↓    } HOLDS AC AND MQ FOR OBSERVATION
   7000  ┘
   5001
```

6. ASR or LSR

```
0/ 7604      LAS (Shift Count = One more than this number in location following ASR or LSR
             if Mode A)
   3007      DCA (Shift Count = Number in location following ASR or LSR if Mode B)
   7431      SWAB (Start here if Mode B)
   1050      TAD MQ (Start here if Mode A)
   7421      MQL
   1051      TAD AC
   7415 or   ASR or LSR
   7417
   XXXX      SHIFT COUNT
```

(continued on next page)

```
7000
 ↓      } NO OPS TO HOLD AC AND MQ FOR OBSERVATION
7000
5000
```

7. MUY

Mode A

```
0/  7406    LAS (Multiplier)
    3006    DCA
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7405    MULTIPLY
    XXXX    MULTIPLIER
    7000
     ↓      } HOLD AC AND MQ FOR OBSERVATION
    7000
    5000    JMP
```

Mode B

```
0/  7431    SWAB
    7604    LAS (Multiplier)
    3100    DCA 100
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7405    MUY
    100     ADDRESS OF MULTIPLIER
    7000
     ↓      } HOLD AC AND MQ FOR OBSERVATION
    7000
    5000    JMP
```

8. DVI

Mode A

```
0/  7604    LAS (Divisor)
    3006    DCA
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7407    DVI
    XXXX    DIVISOR
    7000
     ↓      } HOLD AC AND MQ FOR OBSERVATION
    7000
    5000    JMP
```

(continued on next page)

Mode B

```
0/  7431    SWAB
    7604    LAS (Divisor)
    3100    DCA 100
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7407    DIVIDE
    100     ADDRESS OF DIVISOR
    7000  ┐
     ↓    } HOLD AC AND MQ FOR OBSERVATION
    7000  ┘
    5001    JMP
```

9. SAM

```
0/  7431    SWAB
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7457    SAM
    7000  ┐
     ↓    } HOLD AC, MQ AND STATUS FOR OBSERVATION
    7000  ┘
    5001    JMP
```

10. DAD or DLD

```
0/  7431    SWAB
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7443 or DAD or DLD
    7763
    XXXX    ADDRESS OF WORD
    7000  ┐
     ↓    } HOLD AC AND MQ FOR OBSERVATION
    7000  ┘
    5001
```

11. DST — CONFIGURE AC AND MQ

```
0/  7431    SWAB
    7445    DST
    100     LOCATION OF WORD
    7763    DLD
    100
    7000  ┐
     ↓    } HOLD AC AND MQ FOR OBSERVATION
    7000  ┘
    5001
```

(continued on next page)

12. DPIC

```
0/  7431    SWAB
    7573    DPIC
    5001    JMP
```

13. DCM

```
0/  7431    SWAB
    1050    TAD MQ
    7421    MQL
    1051    TAD AC
    7575    DCM
    7000    HOLD AC AND MQ FOR OBSERVATION
     ↓
    7000
    5001    JMP
```

14. DPSZ

```
0/  7431    SWAB
    7621    CAM
    7451    DPSZ
    7402    HLT
    5001    JMP
```

The programs listed above, when used in conjunction with the flowchart, ROM encoding matrix, and print set, provide simple, repetitive troubleshooting instructions.

Use the following check list as a guideline.

a. Instruction was properly loaded into the Op-decoder.

b. ROM address is correct.

c. ROM outputs are correct.

d. Step Counter decodes last step properly.

e. Control and loading signals listed on the flow chart are occurring at the correct time in relation to time states and bit configurations.

SECTION 6 SPARE PARTS

Table 1-4 lists recommended spare parts for the KE8-E. These parts can be obtained from a local DEC office or from DEC, Maynard, Massachusetts.

Table 1-4
Recommended KE8-E Spare Parts

DEC Part No.	Description	Quantity
19-05585	IC DEC 7476	1
19-05576	IC DEC 7410	1
19-09955	IC DEC 7412	1
19-10018	IC DEC 74193	1
19-09934	IC DEC 8266	1
19-09267	IC DEC 74H11	1
19-05635	IC DEC 74H20	1
19-05586	IC DEC 74H40	1
19-09486	IC DEC 384	1
19-09004	IC DEC 7402	1
19-09667	IC DEC 74H74	1
19-09059	IC DEC 74H30	1
19-09973	IC DEC 97401	1
19-09485	IC DEC 380	1
23-001A1	IC Encoded ROM (Drives ROM 11–18)	1
23-002A1	IC Encoded ROM (Drives ROM 21–28)	1
19-09930	IC DEC 7405	1
19-09705	IC DEC 8881	1
19-05515	IC DEC 7400	1
19-07686	IC DEC 7404	1
19-09062	IC DEC 74H53	1
19-10011	IC DEC 7486	1
19-09935	IC DEC 8235	1
13-00295	Resistor 330Ω 1/4W, 5%	1
13-00365	Resistor 1K, 1/4W, 5%	1
13-00317	Resistor 470Ω, 1/4W, 10%	1
10-00067	Capacitor 6.8 μF, 5V, 20% Solid Tantalum	1
10-01610	Capacitor 0.01 μF, 100V, 20% Ceramic Disk	1