DEC-8/I-HOCA-D

KE8/I
EXTENDED ARITHMETIC ELEMENT

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

EXTENDED ARITHMETIC ELEMENT

INTRODUCTION

The KE8/I Extended Arithmetic Element (EAE) option for the PDP-8/I enables the central processor to perform arithmetic operations at higher speeds. These higher speeds are made possible by incorporating the EAE components with the existing central processor logic in such a way that they operate asynchronously. The EAE components consist of: a 12-bit Multiplier Quotient (MQ) register, a 5-bit Step Counter (SC), a 3-bit Instruction Register (EAE IR), and the EAE timing and control logic.

The EAE logic is contained in twenty-five modules located in the central processor mainframe, module locations C13 through C24 and D12 through D24. The MQ and SC registers are displayed on the PDP-8/I front panel.

The instructions for the EAE are a class of Group 3 Microinstructions containing binary ones in MB bits 00 through 03 and 11. These bits, always being set, distinguish the EAE instruction from all other instructions. These instructions can be subdivided further into two basic groups: the register load group (single-cycle instructions not requiring EAE timing) and EAE operation group (two cycle instructions when EAE timing is started and the central processor is set to the Pause state). All register load instructions have a binary one in MB bits 04, 05, 06, or 07 and binary zeros in bits 08, 09, and 10 except SCL which has a binary one in MB_{10}. All other EAE instructions have binary zeros in MB bits 04, 05, 06 and 07 and a binary one in one or more of MB bits 08, 09, and 10. The EAE microinstructions are augmented instructions and they are microprogrammable. Therefore, they can be combined to perform non-conflicting logical operations.

LOGIC DESCRIPTION

The KE8/I EAE logic circuits are used asynchronously to, but in conjunction with, the accumulator (AC), link (L), and memory buffer (MB) performing parallel arithmetic operations on positive binary numbers. Figure 1 is a simplified block diagram of the KE8/I EAE option.

General Logic and Instructions

The transfer of information between most of the KE8/I EAE registers and the PDP-8/I central processor (CP) occurs during the EAE instruction Fetch cycle. All arithmetic operations, with the exception of NMI (normalize), require an Execute cycle for referencing the next memory location, thereby, obtaining one of the operands for a multiply (MUY), divide (DVI), or the number of shifts to be performed during the long-shift feature. The PAUSE flip-flop, in the CP, is set by the EAE microinstructions when microprogrammed for arithmetic operations because the completion of an arithmetic operation requires at least 7.8 μs. Setting the PAUSE flip-flop prevents the CP from advancing in its cycle until the arithmetic operation is completed. At the finish of the arithmetic operation the EAE logic sets the EAE END flip-flop which direct sets the TS4 flip-flop, clearing the PAUSE flip-flop, and allowing the CP to continue the program.

EAE Instruction Register (EAE IR)

The EAE IR is a 3-bit register consisting of flip-flops that are set to the contents of MB bits 08, 09, and 10 during the Fetch cycle of an EAE instruction. MB bits 08, 09, and 10 determine if the EAE instruction is an SCL, MUY, DVI, NMI, SHL, ASR, or LSR instruction.

All EAE instructions cause MB bits 08-10 to be loaded into the EAE IR at TP3 of the Fetch cycle (D-BS-KE8I-0-2 B-1,2,3). The input to the EAE IR0 flip-flop data input is MB_{08}, EAE IR1 flip-flop data input is MB_{09}, and EAE IR2 flip-flop data input is MB_{10}. The clock inputs are pulsed by the output of a negative NOR gate whose input is from a two-input NAND gate. The two inputs of the NAND gate are EAE INST and TP3. EAE INST is high during TP3 of the Fetch cycle of an EAE instruction (D-BS-KE8I-0-2 C-4). A four-input NAND gate has B FETCH (1), OPR (EAE instructions are a class of OPR instruction), MB_{03} (1), and MB_{11} (1) as input and EAE INST as its output. The last two inputs are true only for the EAE instructions.

Multiplier Quotient (MQ) Register

The MQ is a 12-bit bidirectional shift register that acts as an extension of the AC during EAE operations. The MQ contains the multiplier at the beginning of a multiplication and the least significant half of the product at the conclusion. The MQ contains the least significant half of the dividend at the start of a division and the quotient at the end. The MQ contains the least significant part of a number during a shift or a normalize operation.

Step Counter (SC) Register

The SC is a 5-bit register loaded with the complement of the contents of MB bits 07 through 11 for the ASR, LSR, SCL, and SHL instructions and is set to contain all zeros for the DVI, MUY, and NMI instructions. It is used to record the number of shifts performed and stops the shifting process after the correct number of shifts.

Single-Cycle Instructions

The single-cycle instructions are CAM, CLA, MQA, MQL, and SCA and require a Fetch cycle only. These will be discussed first so that the logic used by these and other instructions can be developed slowly and used as steps for the more difficult instructions. Logical sequence one is during or before TS3, logical sequence two is at TP3, and logical sequence three is after TP3.

Clear the Accumulator (CLA)

The CLA instruction (7601_8) clears the AC during logical sequence 1, therefore, this instruction can be microprogrammed with other EAE instructions that load the AC during logical sequence 2 such as SCA or MQA. Since the EAE instructions are a class of group 3 instructions OPR is always present during the Fetch cycle of an EAE instruction and OP2 is present during TS3 of the Fetch cycle. AC ENABLE is inhibited by MB_{04} being set (D-BS-8I-0-4 C-2). AC ENABLE is high and present only when the four-input AND gate has all four inputs high, that is, OP2, $\overline{\text{AC} \rightarrow \text{MQ ENABLE}}$, and MB_{04} (0). AC → MQ ENABLE is inhibited by MB_{04} (0) and MB_{07}(1)

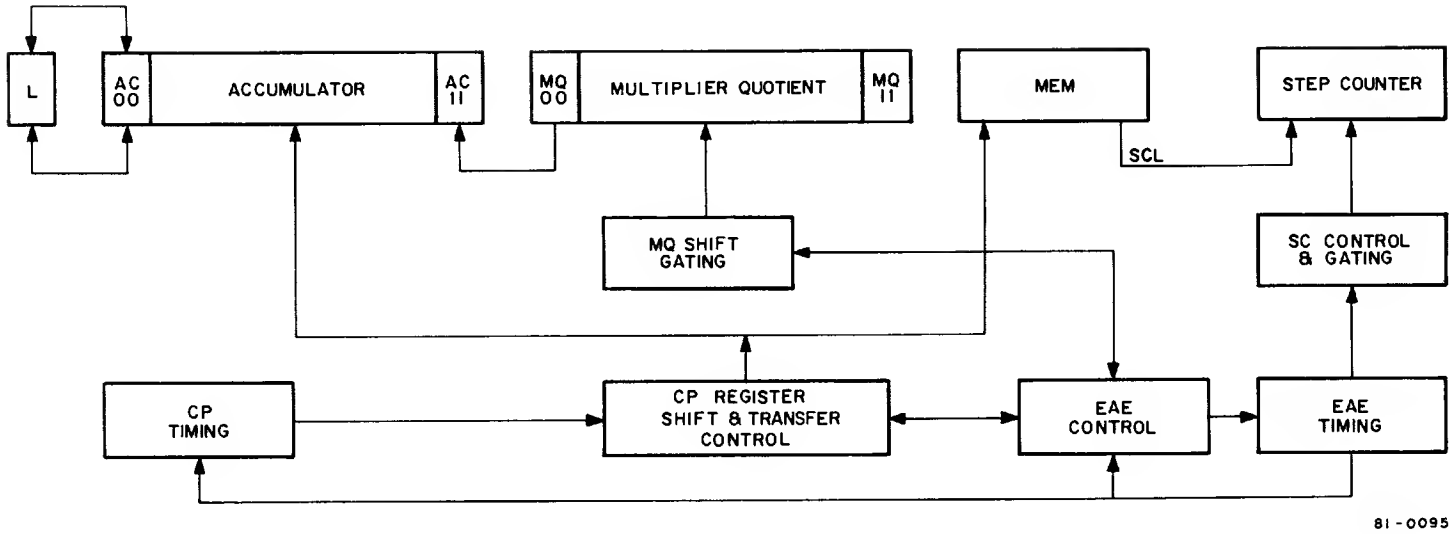


Figure 1 Simplified Block Diagram, KE8/I EAE

(D-BS-KE8I-0-2 C-6). The NAND gate must have all input signals high to produce AC → MQ ENABLE from the negative NOR gate it drives. $MB_{11}(1)$ and OP2 are high but $MB_{04}(0)$ and $MB_{07}(1)$ are low (being 1's) therefore, AC → MQ ENABLE is not present. AC LOAD is generated by four-input AND gate (D-BS-8I-0-6, D-7) with TP3, B FETCH (1), and OPR as inputs. This AND gate output is inverted by a NOR gate and used to load the AC.

Clear Accumulator and Multiplier Quotient (CAM)

The CAM instruction (7621_8) clears the AC during logical sequence 1, as in CLA, and during logical sequence 2 (at TP3) the MQ is cleared by MB bit 07 (1) and EAE INST. AC ENABLE is inhibited by MB_{04} (1) (D-BS-8I-0-4, C-2), AC →MQ ENABLE is also inhibited by MB_{04} (1) (D-BS-KE8I-0-2, C-7). MQ LOAD is present (D-BS-KE8I-0-2, D-2) because MB_{07} (1) and EAE INST are both high during TP3, therefore, MQ LOAD is present during logical sequence 2. AC LOAD is present during TP3 also (D-BS-8I-0-6, D-6 and 7) the output of a four-input AND gate with inputs of TP3, B FETCH(1), and OPR drives a NOR gate. The output of this NOR gate is connected to a negative NOR gate whose output is AC LOAD.

0 = >AC, 0 = > MQ.

Multiplier Quotient Load into Accumulator (MQA)

The MQA instruction (7501_8) loads the contents of the MQ into the AC. This command is given to load the 12 least significant bits of the product into the AC after a multiplication or to load the quotient into the AC after a division. The AC should be cleared prior to issuing this command or the CLA instruction can be combined with the MQA to clear the AC then load the MQ into the AC. If the AC is not cleared prior to this command the contents of the MQ are inclusively ORed with the contents of the AC. During this instruction MB bits 08 through 10 are loaded into the EAE IR.

MQ ENABLE is developed by a three-input NAND gate (D-BS-KE8I-0-2 D-3) with MB_{05} (1), OP2, and $MB_{11}(1)$ as inputs. The output of this NAND gate is connected to a negative NOR gate whose output, MQ ENABLE, is developed by a four-input AND gate (D-BS-8I-04 C-2) with inputs of OP2, $\overline{\text{AC → MQ ENABLE}}$, and $MB_{04}(0)$. $\overline{\text{AC → MQ ENABLE}}$ is high because (D-BS-KE8I-0-2 C-6) MB_{07} is a 0. The output of this AND gate is connected to a NOR gate whose output (low) goes to a negative NOR giving AC ENABLE. AC LOAD is developed by a four-input AND gate (D-BS-8I-0-6 D-7) with inputs of TP3, B FETCH (1), and OPR. The output of this AND gate drives the input of a NOR gate whose output is connected to a negative NOR whose output signal is AC LOAD. AC LOAD is developed for all EAE instructions in this manner at TP3 of the Fetch cycle.

MQ V AC = >AC

Load Multiplier Quotient (MQL)

The MQL instruction (7421_8) clears the MQ, logical sequence 1, loads the contents of the AC into the MQ, logical sequence 2, then the AC is cleared. AC → MQ ENABLE (high) is developed by a four-input NAND gate (D-BS-KE8I-0-2 C-6) with inputs of $MB_{11}(1)$, $MB_{07}(1)$, $MB_{04}(0)$, and OP2. This output is $\overline{\text{AC → MQ ENABLE}}$ and is connected to a negative NOR gate whose output is AC → MQ ENABLE. AC → MQ ENABLE causes the AC bits to go directly to the data inputs of the MQ flip-flops (D-BS-KE8I-0-3 C-8). AC → MQ ENABLE appears on one side of the two-input AND gates with an AC bit on the

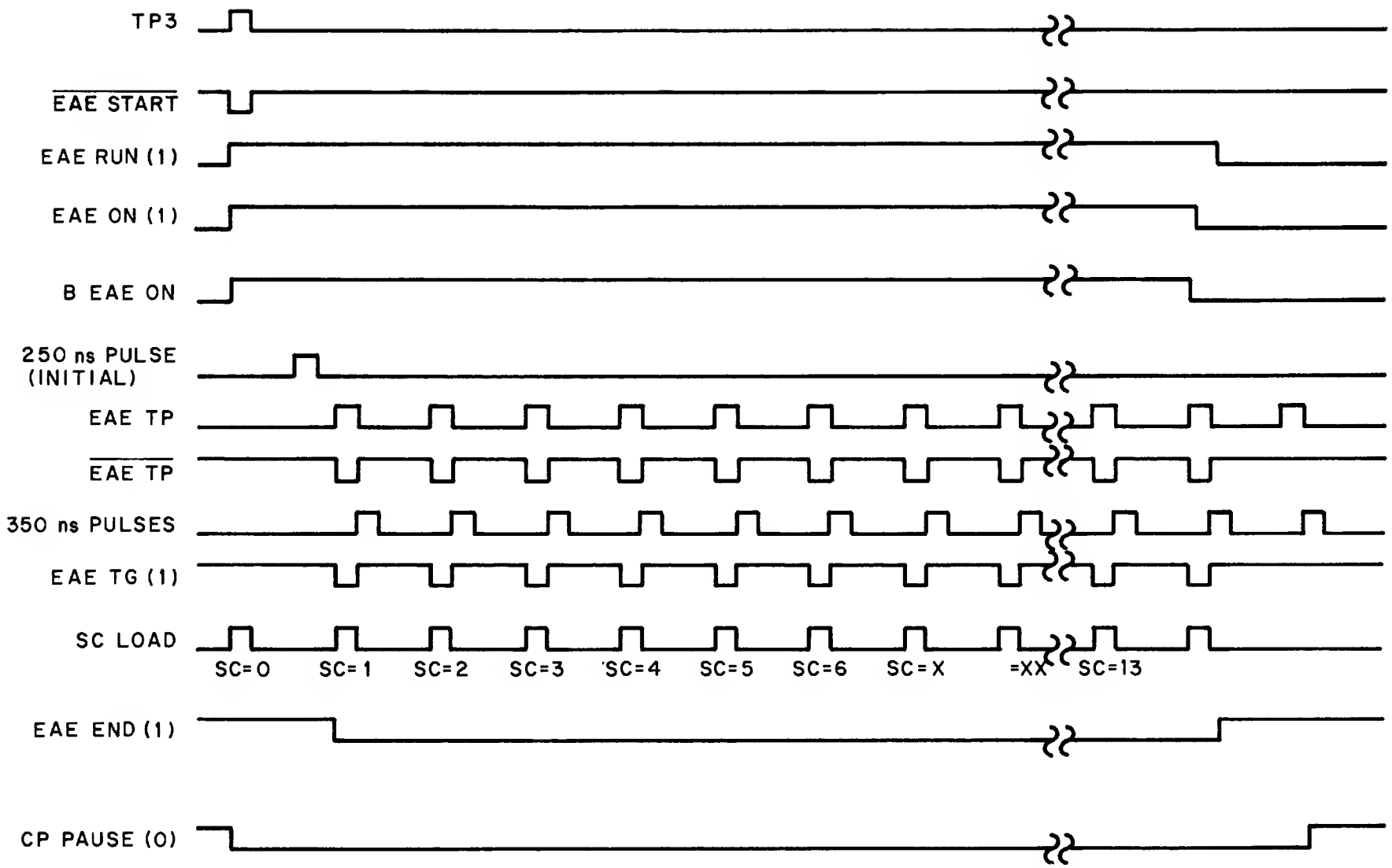


Figure 2 Timing Diagram, KE8/I EAE

other side. MQ LOAD is developed during logical sequence 2 here the same as in the CAM instruction. Zero's are loaded into the AC by inhibiting AC ENABLE (D-BS-8I-0-4 C-2) because AC → MQ ENABLE is low at this time.

0 = >MQ, AC = >MQ, 0 = >AC

Step Counter Load into Accumulator (SCA)

The SCA instruction (7441$_8$) loads the contents of the SC into the AC. The AC should be cleared prior to this instruction or the CLA instruction may be combined with the SCA to clear the AC during logical sequence 1. The transfer of the SC to the AC happens during logical sequence 2 so both instructions may be combined. SC ENABLE is developed (D-BS-KE8I-0-2 D-2) by a three-input NAND gate with inputs of $MB_{11}(1)$, OP2, and $MB_{06}(1)$. The output of this NAND gate goes into a negative NOR gate whose output is SC ENABLE. AC LOAD is developed in this instruction exactly as in the MQA instruction.

SC V AC = >AC.

EAE Timing

All two cycle instructions start EAE timing therefore, EAE timing is covered here with the understanding that all of these instructions accomplish the starting of EAE timing in a similar manner. During this discussion of EAE timing see drawing D-BS-KE8I-0-2, Figure 1 Block Diagram, KE8/I EAE, and Figure 2 Timing Diagram, KE8/I EAE.

During an NMI instruction or any one of the two-cycle instructions one of two NAND gates (D-BS-KE8I-0-2 A-5) is qualified either by NMI (the normalize instruction) and NORM (the normalize condition) or OPR and B EXECUTE (1). When either of these NAND gates qualifies, a negative NOR gate qualifies giving EAE BEGIN. EAE BEGIN combines with $\overline{SCL}$ giving $\overline{EAE\ SET}$. EAE BEGIN also combines with $\overline{SCL}$ and TP3 giving $\overline{EAE\ START}$. $\overline{EAE\ START}$ is inverted and applied to the clock input of the EAE ON flip-flop and since the data input is high (because $\overline{EAE\ COMPLETE}$ is high) the EAE ON flip-flop is set. At the same time $\overline{EAE\ START}$ is applied to the set input of the EAE RUN flip-flop this transition from high to low sets the EAE RUN flip-flop. During this time the pulse applied to the clock input of the EAE ON flip-flop is also applied to a tapped time delay network. Approximately 250 ns later this pulse comes from the time-delay to an amplifier having approximately a 100 ns delay. This pulse (EAE TP) is applied to one side of a NAND gate (D-BS-KE8I-0-2 6-5). The other side of the NAND gate is high at this time because of EAE RUN (1). The EAE TP pulse is inverted through a negative NOR gate and applied to the clock input of the EAE ON flip-flop and to the input of the tapped time-delay line. Therefore, approximately each 350 ns this pulse will circulate through this timing chain and EAE timing will continue. Once EAE timing is started it is a continuous cycle until stopped.

The tapped delay line also has a 350 ns output. This output appears at nearly the same time as the EAE TP and is applied to an amplifier where it is delayed approximately 100 ns and then applied to the clock input of the EAE RUN flip-flop. The data input of the EAE RUN flip-flop is connected to the EAE ON flip-flop (1) output therefore, as long as the EAE ON flip-flop is set the EAE RUN flip-flop will be clocked and set approximately every 450 ns. This ensures that the EAE RUN flip-flop stays set during the circulation of EAE TP throughout the EAE timing chain and that EAE RUN is turned off after EAE TP.

When the EAE TP pulse comes out of the NAND gate we have an EAE TP that is applied, through a negative NOR gate, to the clock input of the EAE TG (time generator) flip-flop. The EAE TG flip-flop data input is low as long as DVI LAST and EAE RUN (1) are present at the NAND gate connected to it. So EAE TP resets this flip-flop, but approximately 100 ns later the 350 ns pulse appears from the tapped time-delay network and is inverted and applied to the set input of the EAE TG flip-flop thereby, setting this flip-flop. EAE TG (1) provides a pulse (pulse width 100 ns) supplying an MQ LOAD and SC LOAD. This happens approximately each 450 ns during EAE timing.

EAE Timing Termination

In order to terminate EAE timing EAE COMPLETE (D-BS-KE8I-0-2 C-6) must go low. When EAE COMPLETE goes low it is inverted and applied to a NAND gate (D-BS-KE8I-0-2 C-5). The other input of this NAND gate is connected to the EAE RUN flip-flop (1) output. This NAND gate's output, now low, is connected to the data input of the EAE ON flip-flop. The next EAE TP pulses the clock input of this flip-flop causing it to reset. This same EAE TP goes on to the time-delay line where it appears again, in 350 ns, as another EAE TP. This EAE TP again pulses the clock input of the EAE ON flip-flop assuring that it is reset as the data input is still low. At this time at the 350 ns time-delay line output, there appears another pulse delayed approximately 100 ns by the amplifier and then it pulses the EAE RUN flip-flop clock input. At this time the EAE RUN flip-flop data input, which is connected to EAE ON flip-flop (1), is low, therefore, the EAE RUN flip-flop is reset. This same pulse is inverted and applied to the set input of the EAE TG flip-flop once more setting this flip-flop. At this time, there is one more EAE TP circulating in the time-delay line. When this EAE TP is applied to the amplifier it generates the last EAE TP pulse. This last pulse is applied to the input of the NAND gate, to clock EAE ON flip-flop, but at this time the other input of the gate is low, since the EAE RUN flip-flop (1) side is 0 having been reset, therefore, the EAE TP pulse dies. One more pulse appears from the delay network 350 ns output pulsing the clock input of the EAE RUN flip-flop and the direct set input of the EAE TG flip-flop. The EAE TG flip-flop is set; the EAE RUN flip-flop data input is low and does not set.

The first EAE TP pulse, after EAE COMPLETE goes low, pulses the clock input of the EAE END flip-flop. At this time the EAE END data input, connected to EAE COMPLETE, is low, therefore, this EAE TP clocks the EAE END flip-flop resetting it. When EAE END (1) goes low its output pulse is applied to a negative NOR gate (D-BS-8I-0-2 D-4) and then to a NAND gate and applied to the TS 4 flip-flop direct set input and CP timing is restarted.

EAE COMPLETE goes low, to end EAE timing, under any of several conditions. EAE COMPLETE is the output of a NOR gate (D-BS-KE8I-0-2 C-6). There are four inputs which can cause it to qualify. On the left hand side the input is tied to an AND gate whose inputs are MUY, SC1(1), SC3(1), and SC4(1). During an MUY instruction MUY is high. As soon as SC1, SC3, and SC4 all become ones, signifying the next to the last operation of this instruction, this AND gate qualifies giving EAE COMPLETE low and thus EAE timing is terminated.

The second input from the left is connected to a two-input AND gate. One of these inputs is high when performing a DVI instruction; the other input is DIV LAST. DIV LAST is given by a negative NOR gate when either input goes low. The left input comes from a

NAND gate whose inputs are SC0 -3 = 0, ADDER L, and SC4(0); these inputs, when high, qualify this NAND gate giving $\overline{\text{EAE COMPLETE}}$ and terminating EAE timing. In other words if the AC is equal to or larger than the MB this gate stops EAE timing. The right hand input is SC1(1), SC4(1), and SC2(1); this condition is met when only one operation remains to be performed by the DVI instruction. EAE timing is halted and the one remaining shift is performed thus completing the EAE DVI instruction.

The third input from the left of the NOR gate remains low at all times. The fourth input from the left is connected to an AND gate with inputs of SC FULL and SC0(1). This condition is met during an ASR, LSR, or SHL instruction when only one shift remains to be performed. When one of these instructions is being performed, the SC is loaded with the complement of the number of shifts to be performed. SC FULL is generated (see D-BS-KE8I-0-3 B-7) by SC1, SC2, SC3, and SC4 all being set to ones. These inputs qualify a NAND gate that, in turn, qualifies a negative NOR gate giving SC FULL. This signal qualifies one input to the AND gate on the fourth input. When SC0(1) goes high both inputs of this AND gate are qualified giving $\overline{\text{EAE COMPLETE}}$ terminating EAE timing.

The fifth input from the left or the extreme right hand input of the NOR gate causes termination of EAE timing during the NMI instruction. This input receives its levels from a three-input AND gate. One input is tied high, the middle input is high at all times during the NMI instruction, and the third input is connected to a negative NOR gate. This NOR gate has three inputs each being driven by a NAND gate, therefore, any time one of the following three conditions is met EAE timing is terminated. First condition: $AC_{01}(0)$ and $AC_{02}(1)$. Second condition: $AC_{03}(0)$, MQ & LOW AC=0, and MID AC = 0. The last two signals are developed (D-BS-KE8I-0-2 B-8) by NAND gates that sample the contents of different MQ bits and (D-BS-8I-0-6 B and C-4) NAND gates that sample AC bits 4 through 11. When AC bits 08 through 11 are all 0's (D-BS-8I-0-6 C-4) a NAND gate qualifies giving a low out which is inverted by a negative NOR gate giving LOW AC = 0. Immediately above these gates is another pair. One of these, a NAND gate, samples AC bits 04 through 07 and when all are zero this gate has a low output qualifying a negative NOR gate whose output signal is MID AC = 0. When MQ_{00} through $_{03}$ (D-BS-KE8I-0-2 B-8) are all 0s a NAND gate qualifies a negative NOR gate whose output is connected to one input of a four-input NAND gate. The second input of this NAND gate receives a signal from a negative NOR gate when MQ_{04} through 07 are all 0s. The third input, of the four-input NAND gate, receives a signal from a negative NOR gate when MQ_{08} through $_{11}$ are all 0s. The fourth input is high when LOW AC = 0. When these four conditions are met simultaneously and the activated NAND gate turns on the negative NOR gate above it, the signal MQ & LOW AC = 0 is present. This signal is connected to the middle input of the three-input NAND (D-BS-KE8I-0-2 A-5 and 6) combining with $AC_{03}(0)$ and MID AC = 0, developed above, thus terminating EAE timing. A number is loaded into the AC and the MQ and is shifted left until $AC_{00} \neq AC_{01}$. Since AC_{00} is the sign bit, AC_{01} is the first data bit. Therefore, shifting occurs until the most significant bit reaches AC_{01}. The control of EAE timing thus depends on AC_{01} and AC_{02} because there is one more shift performed after $\overline{\text{EAE COMPLETE}}$ is generated. The middle NAND gate terminates EAE timing if all bits to the right of AC_{03} are zeros. Third condition: When AC_{01} (1) and AC_{02} (0) EAE timing is terminated by the lost EAE TP.

Special Case Instructions

There are two EAE instructions that do not fit into either the single-cycle or two-cycle instruction group. These are the Normalize (NMI) and Step Counter Load from Memory (SCL). These, with their differences, are discussed below.

Normalize (NMI)

The NMI instruction (7411_8) is used, in part, to convert a binary number to a fraction (and its exponent) for use in floating-point arithmetic. The AC and MQ are acted on as one long register and their contents are shifted left, by this command, until the content of AC_{00} is not equal to AC_{01}. A "zero" is shifted into bit 11 of the MQ each shift. When this instruction is completed, the SC contains a number equal to the number of shifts performed. The contents of the L are lost. During the NMI instruction EAE timing is started but, the CP stays in the Fetch cycle and the PAUSE flip-flop is set. This is the only EAE instruction that starts EAE timing, but does not require an Execute cycle. During logical sequence 1 MB bits 08, 09, and 10 are loaded into the EAE IR as in all EAE instructions. At this time the SC is cleared. A normalize test tree (D-BS-KE8I-0-2 B, C, and D-8) checks the number contained in the AC and MQ to determine if it is already in the normalized condition. There are three NAND gates (D-BS-KE8I-0-2 C-8) with outputs connected to a three-input negative NOR gate. The NAND gate on the left-hand side checks $AC_{00}(0)$ and $AC_{01}(1)$ and the right-hand NAND gate checks $AC_{01}(0)$ and $AC_{00}(1)$. If AC_{00} does not equal AC_{01} neither of these NAND gates will turn on. The center NAND gate has inputs of MQ & LOW AC = 0, MID AC = 0, $AC_{03}(0)$, and $AC_{02}(0)$, if none of these conditions exist this NAND gate does not qualify. The output of the three-input negative NOR gate is low and is inverted by a NAND gate giving the signal $\overline{\text{NORM}}$.

$\overline{\text{NORM}}$ and NMI combine in a two-input NAND gate (D-BS-KE8I-0-2 A-5) whose output is connected to a negative NOR gate. The output of this NOR gate is the signal EAE BEGIN and at TP3, of the Fetch cycle, $\overline{\text{EAE START}}$ is generated and EAE timing is started. This is the only condition where EAE timing is started without an Execute cycle. The normalize test tree (D-BS-KE8I-0-2 B, C, and D-8) keeps constantly testing the AC and MQ for the normalized condition. When this condition is reached EAE timing is terminated as was discussed under EAE Timing Termination.

For each shift during NMI the following signals are generated: AC ENABLE, AC LOAD, AC LEFT SHIFT, MQ LOAD, $\overline{\text{MQ LEFT SHIFT}}$, SC LOAD, and increment SC. AC ENABLE is produced by $\overline{\text{EAE AC ENABLE}}$ (D-BS-KE8I-0-2 D-5). A two-input NAND gate with inputs of B EAE ON, always present with EAE timing, and $\overline{\text{EAE AC ENABLE}}$. $\overline{\text{EAE AC ENABLE}}$ comes from the three-input NAND gate to the left. This NAND gate's output is always high except during an EAE DVI instruction. This input, $\overline{\text{EAE AC ENABLE}}$ and B EAE ON gives $\overline{\text{EAE AC ENABLE}}$ during EAE timing. $\overline{\text{EAE AC ENABLE}}$ is one of the inputs to a four-input negative NOR gate (D-BS-8I-0-4 D-1) giving AC ENABLE. AC LOAD is generated by $\overline{\text{EAE TP}}$ (D-BS-8I-0-6 D-7). During EAE timing each $\overline{\text{EAE TP}}$ pulse into the negative NOR gate gives AC LOAD. $\overline{\text{EAE LEFT SHIFT ENABLE}}$ is generated by a four-input NAND gate (D-BS-KE8I-0-2 D-7). The inputs of this gate are: $\overline{\text{EAE RIGHT SHIFT ENABLE}}$, B EAE ON, $\overline{\text{DIV LAST}}$, and the output of another three-input NAND gate. $\overline{\text{EAE RIGHT SHIFT ENABLE}}$ is high when either EAE IR1 (1) or $\overline{\text{DVI}}$ is low. EAE IR1(1) is low during the NMI instruction, therefore, the output of this NAND gate is high during this EAE instruction. B EAE ON is always present during EAE timing. $\overline{\text{DIV LAST}}$ is high at all times except the last division process of the EAE DVI instruction. The other input is connected to the output of a three-input

NAND gate. The three inputs to this NAND gate are: SC1(1), DVI, and SC2(1). The output of this NAND gate will be high except under these conditions during the EAE DVI instruction, therefore, $\overline{\text{EAE LEFT SHIFT ENABLE}}$ is one of the inputs to a four-input negative NOR gate (D-BS-8I-0-5 D-4) which gives AC LEFT SHIFT. MQ LOAD is developed as the output of a four-input negative NOR gate (D-BS-KE8I-0-2 D-2) one input is EAE TG(1). As was shown in the discussion of EAE timing EAE TG(1) goes low for approximately 100 ns at each $\overline{\text{EAE TP}}$ pulse, therefore, during this time pulse MQ LOAD is generated. $\overline{\text{LEFT SHIFT}}$ is generated by a two-input NAND gate (D-BS-KE8I-0-3 B-8). The two inputs are: $\overline{\text{EAE RIGHT SHIFT ENABLE}}$, developed earlier, and B EAE ON, also developed earlier, giving $\overline{\text{LEFT SHIFT}}$ which is inverted by a negative NOR gate and goes to one side of a bus connecting six two-input NAND gates. The other inputs of these NAND gates are MQ_{01} through MQ_{06}. There are six other two-input NAND gates bussed together and connected to the output of a negative NOR gate (D-BS-KE8I-0-3 C-1, 2, 3, and 4) with $\overline{\text{LEFT SHIFT}}$ as its input. The other inputs of these NAND gates are MQ_{07} through MQ_{11} and the output of a four-input NOR gate. The right-hand input is from a four-input AND gate with three inputs tied high. The fourth input is $\overline{\text{DVI}}$. $\overline{\text{DVI}}$ is high except during an EAE DVI instruction therefore, the output of this AND gate is high except for this instruction, making the output of the NOR gate low so zeros are placed into MQ_{11} during any left shift operation.

SC LOAD is caused by one input of a negative NOR gate being pulsed low by EAE TG(1) each time the EAE TG flip-flop clears (D-BS-KE8I-0-2 D-1). The SC register (D-BS-KE8I-0-3 B and C-2 through 7) has a negative NOR gate connected to the data input of each flip-flop. Each of these NOR gates has three inputs except the one driving the data input of the SC4 flip-flop. The left-hand and center inputs are connected to a three-input NAND gate. One input of each NAND gate is bussed to EAE ON(1) giving a high to each of these inputs during EAE timing. The other input(s) samples either one side or the other of the associated flip-flop and the "one" side of each flip-flop following it, that is, SC0 samples SC1, SC2, SC3, and SC4; SC1 samples SC2, SC3, and SC4; SC2 samples SC3, SC4, and etc. The set (one) and reset outputs of the flip-flops are sampled directly by one NAND gate and the reset and set outputs are sampled by the other NAND gate. Thus, for each SC LOAD generated the SC will count one binary number higher, recording the number of shifts performed.

$$AC_i = AC_i - 1,\ AC_{00} = L,$$
$$MQ_{00} = AC_{11},\ MQ_i = MQ_i - 1,$$
$$0 = MQ_{11} \text{ until } AC_{00} \neq AC_{01}.$$

Step Counter Load From Memory (SCL)

The SCL instruction (7403_8) requires two sequential memory locations, one containing the instruction and the following one containing the number to be loaded into the SC. This instruction loads the complement of the memory word bits 07 through 11, the word located in the next sequential memory address, into the SC. This instruction is a two-cycle instruction since it goes into an Execute cycle but, it does not start EAE timing. During the Fetch cycle, the memory address is incremented by one and loaded into the PC; then the memory word is loaded into the MB and IR. The contents of the PC are loaded into the MA and the Execute state is entered. During the Execute cycle, the contents of the memory word (next memory address) are complemented and loaded into the SC.

During the Fetch cycle, the signal EAE INST is generated (D-BS-KE8I-0-2 C-4) by a four-

input NAND gate with B FETCH(1), OPR, $MB_{03}(1)$, $MB_{11}(1)$; inverting the output gives EAE INST. All EAE instructions during TP3, $MB_{08, 09}$, and 10 are loaded in the EAE IR. $\overline{\text{EAE E SET}}$ (D-BS-KE8I-0-2 C-3) goes low because of the negative NOR gate whose inputs are $MB_{09}(0)$ (now high), and $MB_{10}(0)$ (now low). This output combines with EAE INST in a NAND gate giving $\overline{\text{EAE E SET}}$. During logical sequence 3 PC ENABLE is generated by $\overline{\text{EAE E SET}}$ (D-BS-8I-0-4 A-7) (inverted by a negative NOR gate) and combined with TS4 (1) and a NAND gate. The output of this NAND gate qualifies a negative NOR giving PC ENABLE. E SET is generated by $\overline{\text{EAE E SET}}$ into a negative NOR gate (D-BS-8I-0-3 C-4) whose output is E SET. Therefore, at TP4 of the Fetch cyle the EXECUTE flip-flop is set. During the Execute cycle, B EXECUTE(1) combines with OPR in a NAND gate (D-BS-KE8I-0-2 C-2) whose output is $\overline{\text{EAE EXECUTE}}$. The PC is incremented by one, as in two cycle instructions, by $\overline{\text{EAE EXECUTE}}$ as an input to a negative NOR gate (D-BS-8I-0-5 A-5 and 6) whose output is PC INCREMENT. MB_{07} through $_{11}$ are complemented and loaded into the SC. $\overline{\text{MUY + DVI}}$, EAE ON(0), OPR, and B EXECUTIVE(1) are the four inputs to a NAND gate (D-BS-KE8I-0-2 C-7) whose output is inverted and applied to a negative NOR gate giving MB → SC ENABLE. It should be noted that the SCL instruction is the only instruction which will give B EXECUTE(1) and EAE ON(0) concurrently. SC LOAD is developed by a two-input NAND gate (D-BS-KE8I-0-2 D-1) with inputs of TP3 and EAE BEGIN.

The output of this NAND gate is connected to a negative NOR gate giving SC LOAD. EAE BEGIN is generated by a negative NOR gate (D-BS-KE8I-0-2 B-4). One of the inputs to this NOR gate is the output of a two-input NAND gate with OPR and B EXECUTE(1) as inputs. In order to prevent EAE timing from starting, EAE BEGIN combines with $\overline{\text{SCL}}$ in two NAND gates. Thus, when in the SCL instruction Execute cycle $\overline{\text{SCL}}$ is low and prevents the generation of $\overline{\text{EAE SET}}$ and $\overline{\text{EAE START}}$. This is the circuit that inhibits EAE timing during the SCL instruction.

MB_{07} through $_{11}$ are loaded into the SC by two-input NAND gates (D-BS-KE8I-0-3 B-2 through 7). The right-hand NAND gate for each SC flip-flop has one input bussed to MB → SC ENABLE. The other input is connected to the "zero" output of MB_{07} through $_{11}$ flip-flops. When MB → SC ENABLE is generated, all NAND gates which are high on both inputs (that is, if the MB bit sampling is not set), have outputs inverted by negative NOR gates and these outputs are applied to the data-input of their respective flip-flops. Thus if MB_{07} is actually a zero, the $MB_{07}(0)$ side will be high, qualifying its NAND gate, inverted by the NOR gate, and applied to the data-input of the SC0 flip-flop. When the clock input is pulsed the SC0 flip-flop will set ("one" side high) giving the complement of MB_{07}.

The contents of the PC will be loaded into the MA because F SET (D-BS-8I-0-3 D-5) is now present. Although the CP is in the Execute cycle and B FETCH(1) is now low, EAE INST, and therefore, $\overline{\text{EAE E SET}}$ are no longer available to cause E SET. $\overline{\text{F SET}}$ is an input to a negative NOR gate (D-BS-8I-0-4 A-7), the output combines with TS4(1); in a NAND gate giving PC ENABLE and when inverted by a negative NOR gate it gives PC ENABLE.

$$AC_i = AC_i - 1, \quad AC_{00} = L,$$
$$MQ_{00} = AC_{11}, \quad MQ_i = MQ_i - 1,$$
$$0 = MQ_{11} \text{ until } AC_{00} \neq AC_{01}.$$

Two-Cycle Instructions

There are five two-cycle instructions that start EAE timing. These are: ASR, LSR, SHL, MUY, and DVI. These instructions are coded in bits 08 through 10 of the instruction word. This coding transfers timing from the CP to EAE timing at TP3 of the Fetch cycle with the operation being performed during the Execute cycle. Some EAE microinstructions can be microprogrammed to perform one two-cycle and one or more single-cycle micro-instructions if they are logically compatible.

Arithmetic Shift Right (ASR)

The ASR instruction (7415_8) causes the combined contents of the AC and MQ to shift right one position more than the number contained in the next sequential core memory location. MQ bits shifted past MQ_{11} are lost and the sign bit, contained in AC_{00}, is placed in the L and AC_{00}. During the Fetch cycle, the contents of MB bits 08 through 10 are loaded into the EAE IR, the L is zeroed, the contents of the AC are loaded directly back into the AC, and the EXECUTE flip-flop is set. During the Execute cycle the contents of MB_{08} through $_{11}$ are complemented and loaded into the SC. EAE timing is then started, setting the PAUSE flip-flop, and causing CP timing to be suspended during the rest of this instruction.

During logical sequence 1 of the Fetch cycle, the L is zeroed by AC ENABLE, AC LOAD, NO SHIFT, and L ENABLE (inhibited). AC ENABLE is generated by $\overline{\text{AC} \rightarrow \text{MQ ENABLE}}$, MB04(0), and OP2 as inputs to an AND gate the output of which is inverted (D-BS-8I-0-4 C-2) and applied to a negative NOR gate whose output is AC ENABLE. $\overline{\text{AC} \rightarrow \text{MQ ENABLE}}$ is inhibited by $MB_{04}(0)$ (D-BS-KE8I-0-2 C-6). AC LOAD is developed by a four-input AND gate (D-BS-8I-0-6 D-6) with TP3, B FETCH(1), and OPR as inputs. The output of this AND gate is inverted and applied to a negative NOR gate giving AC LOAD. NO SHIFT is generated by a four-input NAND gate (D-BS-8I-0-5 D-2) with inputs of EAE NO SHIFT ENABLE, $\overline{\text{TT SHIFT ENABLE}}$ and the high output of a NAND gate. EAE NO SHIFT ENABLE is generated by B EAE ON (D-BS-KE8I-0-2 D-6) when applied to a negative NOR gate whose output is EAE NO SHIFT ENABLE. This signal is generated during the Fetch cycle of all EAE instructions and is present until approximately 100 ns after TP3 of the Fetch cycle.

L ENABLE is inhibited by EAE L DISABLE (D-BS-8I-04 A-2). EAE L DISABLE is generated by EAE INST and $MB_{09}(1)$, inputs to a NAND gate (D-BS-KE8I-0-2 D-4) whose output is connected to a NOR gate giving EAE L DISABLE. When L ENABLE goes low the L is not enabled because L ENABLE is one of the inputs to an AND gate; therefore, the output of this AND gate goes low (D-BS-8I-0-8 A-8) and the L is "zeroed".

The PC is loaded into the MA (as discussed under the SCL instruction) during TS4 of the Fetch cycle. The EXECUTE flip-flop is set by B FETCH(1), OPR, $MB_{03}(1)$, and $MB_{11}(1)$ (D-BS-KE8I-0-2 C-4). These inputs to a NAND gate give a low output inverted by a negative NOR giving EAE INST. EAE INST combines with the output of a negative NOR gate (high because of $\overline{MB_{10}(0)}$ in a NAND gate whose output is $\overline{\text{EAE E SET}}$. $\overline{\text{EAE E SET}}$ is applied to a negative NOR gate (D-BS-8I-0-3 C-4) giving E SET. At TP4 the EXECUTE flip-flop is set to a "1".

During TS3 of the Execute cycle MB_{07} through $_{11}$ are complemented and loaded into the SC as was discussed in the SCL instruction. EAE timing is started by OPR and B EXECUTE(1), now high (D-BS-KE8I-0-2 A-4). The PAUSE flip-flop is set to a "1" as was discussed in EAE timing. When EAE timing is started, EAE IR(1), $\overline{\text{DVI}}$, and B EAE ON combine in

a NAND gate (D-BS-KE8I-0-2 D-8) generating EAE RIGHT SHIFT ENABLE. When the output of this NAND gate goes low it disables the NAND gate to the right, thereby inhibiting EAE LEFT SHIFT ENABLE. It also inhibits LEFT SHIFT (D-BS-KE8I-0-3 B-8) by causing the output of this NAND gate to go high. At the same time it is applied to a negative NOR gate (D-BS-8I-0-5 D-5) giving RIGHT SHIFT. EAE AC ENABLE is generated by a two-input NAND gate (D-BS-KE8I-0-2 D-5) with inputs of B EAE ON and the high output of a NAND gate to the left. The output of this NAND gate is EAE AC ENABLE. It is now high because one of the inputs is DVI, which is high only during the EAE DVI instruction. EAE AC ENABLE is applied to a negative NOR gate (D-BS-8I-0-4 D-1) generating AC ENABLE.

ASR ENABLE is generated by a negative NOR gate driven by a three-input NAND gate (D-BS-KE8I-0-2 D-4). This NAND gate has inputs of B EAE ON, EAE IR0(1), and EAE IR1(1). ASR ENABLE is applied to a three-input NAND gate (D-BS-KE8I-0-2 A-8) and combines with EAE IR2(0) and $AC_0(1)$. When $AC_0(1)$ is high ASR L SET is low, but when $AC_0(1)$ is low ASR L SET is high. ASR L SET is applied to a four-input AND gate (D-BS-8I-0-8 C-8) where it combines with RIGHT SHIFT and the output of a NAND gate (high because EAE ON(0) is low). The output of this AND gate is inverted by a NOR gate and applied to the data-input of the LINK flip-flop. With each AC LOAD generated the LINK flip-flop will set if AC_{00} is a "1" and will reset if AC_{00} is a "0". Each time AC_{00} is a "1" the NAND gate turns on (D-BS-KE8I-0-2 A-8). ASR L SET goes low, turning off the AND gate, giving a high (D-BS-8I-0-8 C-8) out of the NOR gate, and putting a high on the data-input of the LINK flip-flop. ASR ENABLE, along with $AC_{00}(1)$, B EAE ON, and EAE IR2(0) qualifies an AND gate (D-BS-8I-0-8 A-7) whose output (high when AC_{00} is a "1", low when AC_{00} is a "0") is inverted by a NOR gate and appears at the inputs of two different gates. The NAND gate has the other input tied high so it inverts the signal and applies it to an AND gate. The other input of this gate is connected to CARRY OUT 0. The second gate (an AND gate) has CARRY OUT 0 for its other input, therefore, when CARRY OUT 0 and $AC_{00}(1)$ are both either high or low one of the two AND gates qualifies turning on the NOR gate above giving ADDER L (low). When CARRY OUT 0 or $AC_{00}(1)$ are opposite (one high and the other low) the AND gates are inhibited and the output of the NOR gate (ADDER L) is high. ADDER L and RIGHT SHIFT are inputs to an AND gate (D-BS-8I-0-9 sheet 1 C-8). When ADDER L is high and this AND gate qualifies, a NOR gate is turned on and puts a 0 on REG BUSS 00. REG BUSS 00 is connected to the data-input of the AC_{00} flip-flop. Therefore, if AC_{00} is a 1, the L is set to a "1" (ASR ENABLE and $AC_0(1)$ giving ASR L SET low thus a high is applied to the data-input of the LINK flip-flop), and a "1" is set back into AC_{00}. (ASR ENABLE and $AC_{00}(1)$ combine and the resultant combines with CARRY OUT 0 giving ADDER L low; ADDER L then combines with RIGHT SHIFT applying a "1", through the REG BUSS 00, to the data-input of the AC_{00} flip-flop.) At the same time each AC bit is shifted right one place. This operation is repeated until the number in the SC register generates SC FULL and SC0 (1) and causes termination of EAE timing.

$$AC_{00} = AC_{00},\ AC_i = AC_i + 1,$$
$$AC_{11} = MQ_{00},\ MQ_i = MQ_i + 1,$$
$$MQ_{11} \text{ lost.}$$

Logical Shift Right (LSR)

The LSR instruction (7417_8) causes the combined contents of the AC and the MQ to shift right one position more than the number in the next sequential core memory location after the instruction. MQ bits shifted past

MQ_{11} are lost and the L is loaded with a "0" during this operation. During the Fetch cycle the contents of MB bits $_{08}$ through $_{10}$ are loaded into the EAR IR register, the L is "zeroed", the contents of the AC are loaded directly back into the AC, and the EXECUTE flip-flop is set. The contents of MB_{07} through $_{11}$ are complemented and loaded into the SC during the Execute cycle. Then EAE timing is started and CP timing is suspended. The contents of the AC and MQ are then shifted the required number of times to the right.

During logical sequence 1 of the Fetch cycle the L is "zeroed" by AC ENABLE, AC LOAD, NO SHIFT, and L ENABLE (inhibited) as discussed under the ASR instruction. The PC is loaded into the MA (as discussed under the SCL instruction) during TS4 of the Fetch cycle. EAE timing is started and CP timing is suspended as discussed under EAE timing. The logic for this instruction is the same as the logic for the ASR instruction with the following exception: ASR ENABLE is generated, ($\overline{\text{ASR L SET}}$ remains high at all times during this instruction because the four-input NAND gate (D-BS-KE8I-0-2 A-8) is disqualified by EAE IR2 (0)). The NAND gate is disqualified as it is a "1" for the LSR instruction. Thus the data-input of the LINK flip-flop remains low and the L stays reset. The proper gates for the shifting of the AC and MQ bits to the right are discussed under the ASR instruction.

$$0 = AC_{00},\ AC_i = AC_i + 1,$$
$$MQ_i = MQ_i + 1.$$

Shift Left (SHL)

The SHL instruction (7413_8) causes the combined contents of the AC and the MQ to shift left one position more than the number in the next sequential core memory location after the instruction. The following occurs for each shift:

(1) the content of the L is lost,
(2) the content of AC_{00} is loaded into the L
(3) all other AC bits move one place left,
(4) MQ_{01} is loaded into AC_{11}; all other MQ bits move one place left, and
(5) MQ_{11} is loaded with a "0".

During the Fetch cycle the contents of MB_{08} through $_{10}$ are loaded into the EAR IR, the contents of the PC are loaded into the MA, and the EXECUTE flip-flop is set. The contents of MB_{07} through $_{11}$ are complemented and loaded into the SC during the Execute cycle. Then the CP PAUSE flip-flop is direct set and EAE timing is started. The contents of the AC and MQ are then shifted left the required number of times.

The logic is the same as for the ASR and LSR instructions during the Fetch cycle. During the Execute cycle the logic is the same as the ASR and LSR, until the EAE timing is started.

AC ENABLE is generated by $\overline{\text{EAE AC ENABLE}}$. $\overline{\text{EAE AC ENABLE}}$ is generated by a NAND gate (D-BS-KE8I-0-2 D-5) with B EAE ON and $\overline{\text{EAE AC ENABLE}}$ (because DVI is low). $\overline{\text{EAE LEFT SHIFT ENABLE}}$ is generated by a four-input NAND gate with inputs of B EAE ON, $\overline{\text{DIV LAST}}$, $\overline{\text{EAE RIGHT SHIFT ENABLE}}$, and the high output of a NAND gate (D-BS-KE8I-0-2 D-7) (high because one input is DVI which is low).

$\overline{\text{EAE LEFT SHIFT ENABLE}}$ is applied to the input of a negative NOR gate (D-BS-8I-0-5 D-4) giving LEFT SHIFT. $\overline{\text{EAE RIGHT SHIFT ENABLE}}$ (high) combines with B EAE ON in a NAND gate (D-BS-KE8I-0-3 B-8) giving $\overline{\text{LEFT SHIFT}}$. This signal is inverted and applied to two negative NOR gates (D-BS-KE8I-0-3 C-1 and 8). This high output is bussed to one input of a series of two-input NAND gates. The other input to each of these NAND gates (except the one for MQ_{11}) is MQ_{i-1}. The NAND gate for MQ_{11} is connected to a NOR gate (D-BS-KE8I-0-3 B-1)

and this NOR gate has inputs from four AND gates. The right-hand AND gate has inputs of -3V and $\overline{DVI}$. Since $\overline{DVI}$ is high except during the DVI instruction, this AND gate has a high output. The output of this AND is connected to the input of a NOR gate whose output is connected to the input of a NAND gate with LEFT SHIFT as the other input. Since this NAND gate remains "turned off" except during the DVI instruction the negative NOR gate it is driving will keep the data input of the MQ_{11} flip-flop low for each MQ LOAD pulse. Therefore, each time LEFT SHIFT is present and we are not doing a DVI instruction the MQ_{11} flip-flop will reset to a "0". This means that for each shift 0s are loaded into MQ_{11} and transferred from MQ_{11} to the vacated positions in the MQ.

$$AC_i = AC_i - 1,\ AC_{00} = L,$$
$$MQ_{00} = AC_{11},\ MQ_i = MQ_i - 1,$$
$$0 = MQ_{11}$$

Multiply (MUY)

The MUY instruction (7405_8) multiplies the number in the MQ by the number held in the next successive core memory location after the MUY instruction. At the end of the instruction, the twelve most significant bits of the product are contained in the AC and the twelve least significant bits of the product are contained in the MQ. During the Fetch cycle the usual operations necessary for an EAE instruction are performed: MA + 1 → PC, MEM → MB, MEM → IR, MB_{08} through $_{10}$ → EAE IR, 0 → L, and the Execute major state is entered. During the Execute cycle the usual operations for the two-cycle instructions are performed: MA + 1 → PC, MEM → MB, 1 → EAE ON, 1 → PAUSE, PC → MA, and 1 → F. The only difference during the Execute cycle for MUY and DVI is that instead of loading $\overline{MB7\text{-}11}$ → SC, the SC is cleared to receive the number of multiply operations. For each operation the SC is incremented. As will be shown the SC will never need to count more than 11_{10}.

Before discussing the actual logic of the MUY instruction some discussion of binary multiplication, as performed by the EAE, is necessary. To multiply two numbers together (12_{10} times 12_{10}) with the EAE, the following operations are necessary. These numbers in binary form are: $001\ 100_2$.

Step 1

```
 001  100
X001  100
------------
 000  000      partial product
```

Step 2

```
 001  100
X001  100
------------
 000  000      (from Step 1)
0000  00
------------
0000  000      second partial product
```

Step 3

```
 001  100
X001  100
------------
 000  000
0000  00       (from Step 2)
00110  0
------------
00110  000      third partial product
```

Step 4

```
   001  100
  X001  100
  ------------
   000  000
  0000  00
 00110  0        (from Step 3)
001100
------------
010010  000      fourth partial product
```

Step 5 and 6

```
   001  100
  X001  100
  ------------
   000  000      (from Step 1)
  0000  00       (from Step 2)
 00110  0        (from Step 3)
001100           (from Step 4)
000000           (from Step 5)
000000           (from Step 6)
------------
00010010  000    final product (144₁₀)
```

In Step 1 the least significant bit in the multiplier is multiplied by the multiplicand and forms a partial product. Since binary numbers (0s and 1s) are being used by the computer some definite rules can be established. When the least significant bit in the multiplier is a 1 the multiplicand is added to the partial product. When the least significant bit in the multiplier is a 0 then 0s are added to the partial product. The number of operations necessary to perform this multiplication is six. When the same multiplication is performed using 12-bit numbers, the number of operations necessary is twelve. Since the SC must increment one time less than the number of operations performed, the SC reaches 12 to terminate EAE timing. Even when multiplying the largest binary numbers that can be expressed in 12-bits the number of operations is the same, and the number of bits in the product will not exceed twenty-four. When an addition of either 0s or the multiplicand is made to the partial product, the least significant bit does not change. In order to determine whether to add 0s or to add the multiplicand to the partial remainder, the least significant bit of the multiplier is examined, and if it contains a 1 the signal EAE MEM ENABLE must be generated. If it contains a 0 then EAE MEM ENABLE must be inhibited, allowing 0s to be added to the contents of the AC. The EAE actually shifts the contents of the AC and MQ to the right to add 0s. The least significant bit is examined, and if it is a 0, then the L, AC, and MQ are shifted to the right. If the LSB is a 1, then the MEM is added to the AC, and the L, AC and MQ are shifted to the right.

When MQ_{11} (the LSB) is a 0 MEM ENABLE is inhibited and AC ENABLE, AC LOAD, MQ LOAD, and SHIFT RIGHT are generated. When MQ_{11} is a 1, MEM ENABLE, AC ENABLE, AC LOAD, MQ LOAD, SHIFT RIGHT are generated. During an MUY instruction, if MQ_{11} is a 1 then EAE MEM ENABLE (D-BS-KE8I-0-2 D-6) is low. MQ_{11} (0) and MUY are two inputs to a NAND gate. When MQ_{11} (0) is low this NAND gate has a high output giving EAE MEM ENABLE (low). EAE MEM ENABLE is one of the inputs to a negative NOR gate (D-BS-8I-0-4 D-7) giving MEM ENABLE 0-4 and MEM ENABLE 5-11. Therefore, if MQ_{11} is a 1 MEM ENABLE 0-11 is generated. AC ENABLE is generated by EAE AC ENABLE (D-BS-8I-0-4 D-1). EAE AC ENABLE is generated by B EAE ON, and EAE AC ENABLE (D-BS-KE8I-0-2 D-5). AC LOAD is generated by EAE TP (D-BS-8I-0-6 D-7). MQ LOAD and SC LOAD are generated by EAE TG(1) (D-BS-KE8I-0-2 D-1). RIGHT SHIFT is generated by EAE RIGHT SHIFT ENABLE (D-BS-8I-0-5 D-5). EAE RIGHT SHIFT ENABLE is generated by EAE IR1(1), DVI, and B EAE ON (D-BS-KE8I-0-2 D-8). As each operation is performed, a four-input AND gate tests the SC register until MUY, SC1(1), SC3(1), and SC4(1) are all high (D-BS-KE8I-0-2 B-7). When this condition exists, it signifies that the multiplication is finished and terminates EAE timing.

Y X MQ = AC, MQ. 0 → L

Divide (DVI)

The DVI instruction (7407_8) divides the 24-bit dividend contained in the AC (12 most significant bits) and the MQ (12 least significant bits) by the number located in the next sequential core location after the DVI instruction.

Divide Algorithm

The simplest divide algorithm, although not the one used in the KE8/I, is that of subtraction test, subtractions and shifts. Assume that a double-precision number has been loaded into the AC and MQ, and that the dividend is present in the MEM register. A trial subtraction is taken (for example, MEM + AC is placed on the register bus). If an overflow results, the subtraction cannot be made without causing a change in sign. The entire AC and MQ are shifted left without actually executing the subtraction, and a 0 is shifted into MQ_{11} to form a part of the quotient. If overflow does not result, the subtraction is performed before, or as a part of, the shift, and a 1 is loaded into MQ_{11}. The process is very similar to long-hand decimal division.

In the KE8/I, the algorithm used is slightly more complicated, although faster. A subtraction and shift is performed. If overflow does not result (indicated at ADDER L), a 1 is loaded in MQ_{11}. Subtractions and shifts continue until overflow occurs. When overflow occurs, a 0 is loaded in MQ_{11}, and the next arithmetic operation is an add and shift. The adding process continues until another overflow is encountered and 0s are loaded into MQ_{11} for each addition. The 0s are loaded into the MQ for each addition. Overflow indicates that the partial remainder is now positive and that the arithmetic operation should become a subtraction. At the conclusion of the divide operation (SC = 13), the remainder (now in the AC) may have to be corrected, and the LINK cleared. The correction depends upon the last two bits in the quotient. If the last two bits are $MQ_{10} = 0$ and $MQ_{11} = 1$, the remainder is correct as is. If the last two bits are both 1s, the remainder must be complemented. Other combinations of MQ_{10} MQ_{11} result in an extra addition or subtraction.

The first subtraction in any divide must produce a negative result; otherwise, a condition known as "divide overflow" exists. If divide overflow occurs (division by zero is a classical example), the resulting quotient cannot be contained in 12 bits, and the EAE signifies this fact by immediately exiting from the division with the LINK set.

Before EAE timing is started at TP3 of the Execute cycle TP3 and EAE BEGIN combine in a NAND gate (D-BS-KE8I-0-2 D-1) giving SC LOAD. Therefore, the SC is cleared before the DVI instruction is started. Figure 3, KE8/I EAE DVI Algorithm, shows the operations performed by the EAE to divide 145_{10} by 12_{10}. The quotient will be 12_{10} (in the MQ) with a remainder of 1_{10} (in the AC). During the first operation $\overline{\text{AC ENABLE}}$ is always generated because $\overline{\text{SC0-3 = 0}}$ is one of the inputs to a three-input negative NOR gate (D-BS-KE8I-0-2 D-5). This output is connected to a NAND gate above giving $\overline{\text{EAE AC ENABLE}}$. $\overline{\text{EAE AC ENABLE}}$ is one input to a negative NOR gate (D-BS-8I-0-4 B-1) giving AC ENABLE. Therefore AC ENABLE gates the AC flip-flop "0" outputs to the ADDER bus giving the complement of the AC which combines with the contents of MEM. The result is a subtraction with the complement of the result left in the AC. MEM ENABLE 0-11 is generated by $\overline{\text{EAE MEM ENABLE}}$ as inputs to two negative NOR gates (D-BS-8I-0-4 D-5 and 6). $\overline{\text{EAE MEM ENABLE}}$ is generated by a NAND gate (D-BS-KE8I-0-2 D-6) with inputs of EAE IR0, B EAE ON, and the high outputs of two NAND gates. The

Link	Carry	Accumulator	Multiplier Quotient	Memory	Step Counter	Comments
0		000 000 000 000	000 010 010 001	000 000 001 100		
		111 111 111 111				$\overline{AC}$ ENABLE
		000 000 001 100				MEM ENABLE
	1	000 000 001 011				Output of ADDERS
0		000 000 010 111	000 100 100 010	000 000 001 100	00 001	AC and MQ LEFT SHIFT
						End of 1st operation
0		111 111 101 000				$\overline{AC}$ ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 100				Output of ADDERS
1		111 111 101 000	001 001 000 100	000 000 001 100	00 010	AC and MQ LEFT SHIFT
						End of 2nd operation
		111 111 101 000				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 100				Output of ADDERS
1		111 111 101 000	010 010 001 000	000 000 001 100	00 011	AC and MQ LEFT SHIFT
						End of 3rd operation
		111 111 101 000				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 100				Output of ADDERS
1		111 111 101 000	100 100 010 000	000 000 001 100	00 100	AC and MQ LEFT SHIFT
						End of 4th operation
		111 111 101 000				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 100				Output of ADDERS
1		111 111 101 001	001 000 100 000	000 000 001 100	00 101	AC and MQ LEFT SHIFT
						End of 5th operation
		111 111 101 001				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 101				Output of ADDERS
1		111 111 101 010	010 001 000 000	000 000 001 100	00 110	AC and MQ LEFT SHIFT
						End of 6th operation
		111 111 101 010				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 110				Output of ADDERS
1		111 111 101 100	100 010 000 000	000 000 001 100	00 111	AC and MQ LEFT SHIFT
						End of 7th operation
		111 111 101 100				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 111 000				Output of ADDERS
1		111 111 110 001	000 100 000 000	000 000 001 100	01 000	AC and MQ LEFT SHIFT
						End of 8th operation
		111 111 110 001				AC ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 111 101				Output of ADDERS
1		111 111 111 010	001 000 000 000	000 000 001 100	01 001	AC and MQ LEFT SHIFT
						End of 9th Operation
		111 111 111 010				AC ENABLE
		000 000 001 100				MEM ENABLE
	1	000 000 000 110				Output of ADDERS
0		000 000 001 100	010 000 000 001	000 000 001 100	01 010	AC and MQ LEFT SHIFT
						End of 10th operation
		111 111 110 011				$\overline{AC}$ ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 111 111				Output of ADDERS
1		111 111 111 111	100 000 000 011	000 000 001 100	01 011	AC and MQ LEFT SHIFT
						End of 11th operation
		111 111 111 111				AC ENABLE
		000 000 001 100				MEM ENALBE
	1	000 000 001 011				Output of ADDERS
0		000 000 010 110	000 000 000 110	000 000 001 100	01 100	AC and MQ LEFT SHIFT
						End of 12th operation
		111 111 101 001				$\overline{AC}$ ENABLE
		000 000 001 100				MEM ENABLE
	0	111 111 110 101				Output of ADDERS
1		111 111 110 101	000 000 001 100	000 000 001 100	01 101	MQ LEFT SHIFT ONLY
						End of 13th operation
		111 111 110 101				Since SC = 13 and MQ10=0
		000 000 001 100				and MQ11=0 MEM and $\overline{AC}$ → AC.
	1	000 000 000 001				
0		000 000 000 001	000 000 001 100	000 000 001 100	01 101	NO SHIFT

Figure 3 KE8/I EAE DVI Algorithm

right-hand NAND gate has a high output because one of its inputs is MUY. The left-hand NAND gate has four inputs: $MQ_{11}(1)$, $SC_1(1)$, (low until at least the 8th operation when the SC = 8), and DIV LAST (low until the SC = 13 or divide overflow occurs). $SC_1(1)$ is an input for this gate because if divide overflow occurs then $SC_{0-3} = 0$, ADDER L, and $SC_4(0)$ (D-BS-KE8I-0-2 A-7) are inputs to a NAND gate giving DIV LAST. Since $SC_1(1)$ is low for divide overflow this gate is inhibited at this time. Also since one or more of these inputs is high until DIV LAST goes high $\overline{\text{EAE MEM ENABLE}}$ is present until the SC = 13. Figure 4, KE8/I EAE DVI Flow Chart, shows the decisions made by the EAE logic for each divide operation. If the SC = 0 or $MQ_{11} = 1$ and $MQ_{00} = 0$ then the necessary gates are enabled to place a "0" in AC_{11}. It follows that, the number is complemented as it is shifted from MQ_{00} to AC_{11} if a subtraction is being performed. This step is necessary because the AC is, in effect, complemented instead of complementing MEM. A NAND gate with inputs of $SC_{0-3} = 0$ and $SC_4(0)$ (D-BS-KE8I-0-2 C-3) has an output of $\overline{SC = 1}$ or more during the DVI instruction operation 2 through 13 this gate has a high output except during operation number 1. A negative NOR gate, to the right, has $MQ_{11}(0)$ as an input so that MQ_{11} is set to a "1" this input is low giving a high output that combines with DVI in a two-input NAND gate. The output of this gate is applied to two (2) three-input NAND gates. The left-hand NAND gate has $MQ_0(0)$ and B EAE ON as its other inputs. $MQ_0(0)$ is high if MQ_{00} is a "0". Therefore when $MQ_{11}(0)$ is high ($MQ_{11} = 0$) and $MQ_{00} = 0$, $\overline{\text{EAE MQ0 ENABLE}}$ is generated. The NAND gate on the right has inputs of B EAE ON, and $MQ_0(1)$, while the NAND gates output is inverted. It follows that when $MQ_{11}(0)$ is low ($MQ_{11} = 1$), a high is applied to this NAND gate. If $MQ_{11} = 1$ and $MQ_0(1)$ is high (MQ00 = 1) then $\overline{\text{EAE MQ0 ENABLE}}$ is generated. Consequently if $MQ_{11} = 0$ and $MQ_{00} = 0$ or $MQ_{11} = 1$ and $MQ_{00} = 1$, one of these two signals is generated. A three-input negative NOR gate (D-BS-8I-0-9 sheet 4 C-1) has inputs of: $\overline{\text{EAE MQ0 ENABLE}}$, and the output of NAND gate (high during DVI because EAE ON (0) is low during EAE timing). When either of these signals is low this NOR gate has a high output which combines with LEFT SHIFT in an AND gate and is inverted by the above NOR gate causing AC_{11} to be cleared. If neither of these signals is low ($MQ_{11} = 0$ and $MQ_{00} = 1$ or $MQ_{11} = 1$ and $MQ_{00} = 0$) then AC_{11} is set.

$\overline{\text{ADDER L}}$ is generated by a series of gates (D-BS-8I-08 A, B, C - 7 and 8). The two AND gates at the bottom of the series have inputs sampling the condition of the LINK flip-flop. When $\overline{\text{AC ENABLE}}$ is present L ENABLE is generated by a negative NOR gate (D-BS-8I-0-4 B-4) with $\overline{\text{EAE AC ENABLE}}$ as an input. L ENABLE is generated by a NOR gate because when $\overline{\text{EAE AC ENABLE}}$ is high EAE L DISABLE is low giving a low out of the AND gate it drives making L ENABLE go high. The AND gates (D-BS-8I-0-8 A-7 and 8) will have low outputs unless either LINK(1) and L ENABLE or LINK (0) and $\overline{\text{L}}$ ENABLE are high. The outputs of these gates go to a NOR gate above whose output is applied to two AND gates (one through an inverter) and combine with $\overline{\text{CARRY OUT 0}}$. When there is a carry from ADDER 00, $\overline{\text{CARRY OUT 0}}$ is low turning off the right-hand AND gate. $\overline{\text{CARRY OUT 0}}$ is inverted and applied to the left-hand AND gate. Therefore, one of these AND gates will always have one input high, or low, depending on $\overline{\text{CARRY OUT 0}}$. The other input is either high or low depending on the status of L. If $\overline{\text{CARRY OUT 0}}$ is high and LINK(0) and $\overline{\text{L}}$ ENABLE or LINK (1) and L ENABLE are high the low output of the NOR is inverted turning on the right-hand AND gate making $\overline{\text{ADDER L}}$ low. $\overline{\text{ADDER L}}$ will be low if there is a carry and $\overline{\text{L}}$ ENABLE and LINK(0) are both high or L ENABLE and LINK(1) are both high. $\overline{\text{ADDER L}}$ will also be low if there is not a carry and if either

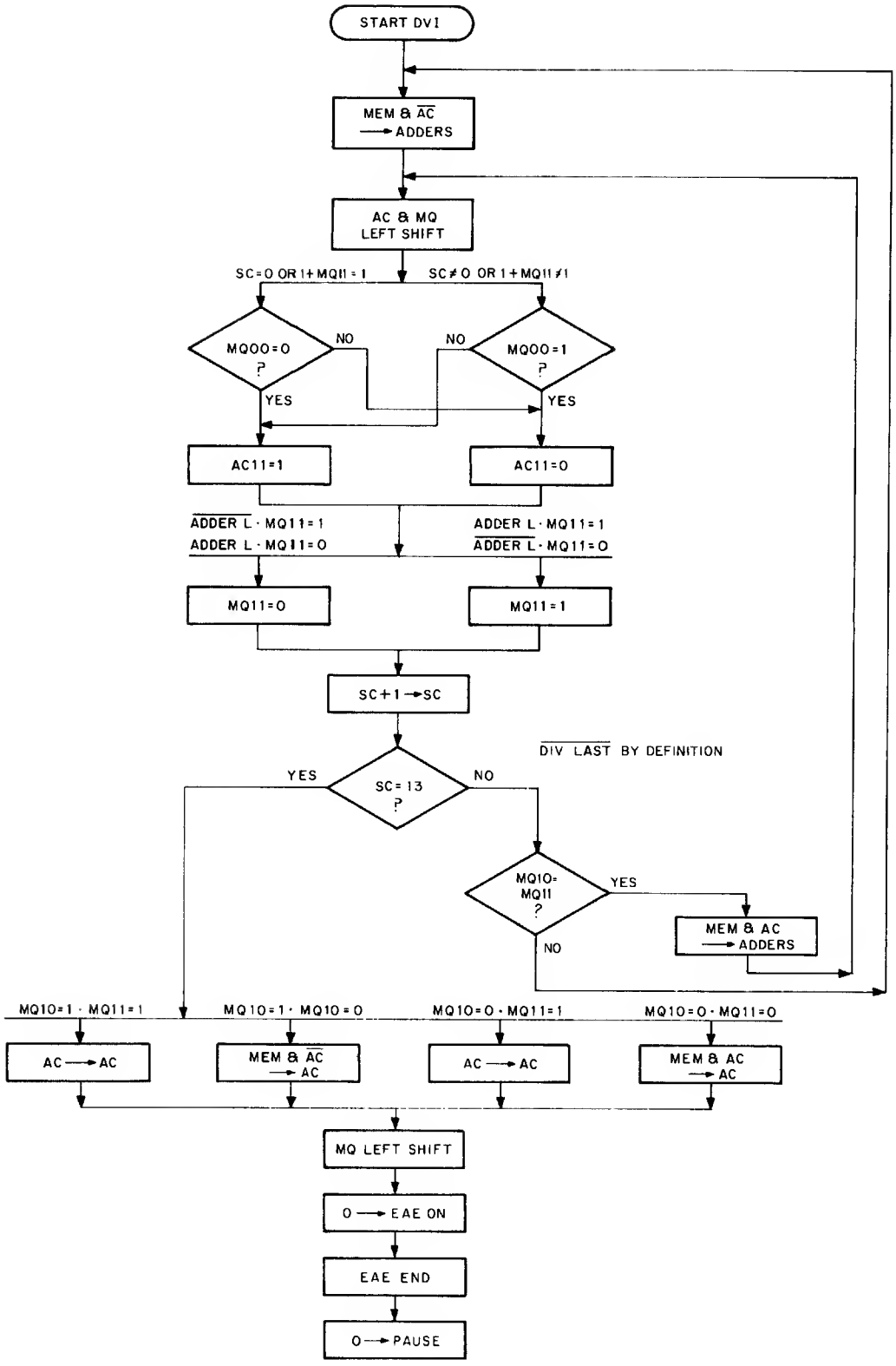


Figure 4 KE8/I EAE DVI Flow Chart

$\overline{L}$ ENABLE or LINK(0) is low or if L ENABLE or LINK(1) is low.

When SC = 0 there are three gates (D-BS-KE8I-0-3 B-1) which check the status of the SC. The bottom gate has inputs of $\overline{\text{EAE TP}}$, SC1(0), SC2(0), and SC3(0) giving an output of SC0-3 = 0 which combines with SC4(0). This output is connected to a NOR gate with its output connected to a NAND gate, inverted and applied to the data-input of the MQ_{11} flip-flop. Thus when SC = 0 MQ_{11} is cleared. During Steps 2 through 12 the other three NAND gates determine the content of MQ_{11}.

$\overline{\text{ADDER L}}$ is an input to an AND gate and a negative NOR gate (D-BS-KE8I-0-3 B-1). The output of the negative NOR is ADDER L. Therefore $\overline{\text{ADDER L}}$ is combined with MQ_{11}, determining whether to set or clear the MQ_{11} flop-flop. As an example, during operation two $\overline{\text{ADDER L}}$ is high because $\overline{\text{CARRY OUT 0}}$ is low, $\overline{L}$ ENABLE is high and LINK(0) is high (since the L is a "0") giving $\overline{\text{ADDER L}}$ high. $\overline{\text{ADDER L}}$ is combined with $MQ_{11}(1)$ in an AND gate. Since $MQ_{11} = 1$ before the shift, the AND gate's output is high when inverted by a NOR and applied to a NAND gate above. This turns the previously mentioned NAND gate off, its high output is inverted by a negative NOR and clears the MQ_{11} flip-flop. Thus MQ_{11} contains a "0" at the end of the second operation.

During operation 10 (see Figure 4) $\overline{\text{CARRY OUT 0}}$ is low, L ENABLE is high, and LINK(1) is high giving $\overline{\text{ADDER L}}$ high (D-BS-8I-0-8). $\overline{\text{ADDER L}}$ puts a high input on the left-hand AND gate and a low input on the right-hand AND gate (D-BS-KE8I-0-3 B-1), but $MQ_{11} = 0$, therefore, $MQ_{11}(1)$ is low and both AND gates have low outputs. The NOR gate above has a high output which turns on the NAND above it setting MQ_{11}.

After each operation the SC is incremented by one and tested to see if it equals thirteen.

When the SC reaches thirteen EAE timing is terminated as discussed under EAR timing. If SC = 13 then a series of gates (D-BS-KE8I-0-2 C and D-5) test to determine if the AC is to be complemented. A NAND gate (with inputs of DIV LAST and DVI) has a high output until the last operation, or until DIV LAST goes high. This output combines with $MQ_{10}(0)$ in another NAND gate. If $MQ_{10}(0)$ is high ($MQ_{10} = 0$) this NAND gate has a low output, which is inverted and combined with $MQ_{11}(1)$. Consequently if $\overline{\text{DIV LAST}}$ is low, and $MQ_{10} = 0$, $MQ_{11} = 1$, then $\overline{\text{EAE AC ENABLE}}$ is generated. An AND gate to the right has inputs of $MQ_{10}(1)$ and $MQ_{11}(0)$. When these inputs are high ($MQ_{10} = 1$ and $MQ_{11} = 0$) this AND gate generates $\overline{\text{EAE AC ENABLE}}$. A NAND gate, further to the right, goes low on the last division if $MQ_{10} = 1$ thus generating EAE $\overline{\text{AC}}$ ENABLE. Therefore if it is not the last operation and $MQ_{10} \neq MQ_{11}$ or if it is the last operation and $MQ_{10} = 1$ the AC is complemented.

During the last operation, or when SC = 13, a four input NAND gate (D-BS-KE8I-0-2 D-6) tests MQ bit 11 to determine whether or not MEM ENABLE is to be generated. DVI is high for all of the DVI instruction, DIV LAST is high only during the last operation or divide overflow. SC1(1) is also high during the last instruction so if $MQ_{11}(1)$ is high MEM ENABLE is inhibited, but if $MQ_{11}(1)$ is low or if $MQ_{11} = 0$ then MEM ENABLE is generated.

During operation 12, a three-input NAND gate (D-BS-KE8I-0-2 D-7) with inputs of SC1(1), DVI, and SC2(1) qualifies and inhibits $\overline{\text{EAE LEFT SHIFT ENABLE}}$ then generates EAE NO SHIFT ENABLE giving NO SHIFT. However, since $\overline{\text{EAE RIGHT SHIFT ENABLE}}$ is still high (MQ) $\overline{\text{LEFT SHIFT}}$ is still present. During operation 13 since $\overline{\text{EAE LEFT SHIFT ENABLE}}$ and EAE NO SHIFT ENABLE are still high, NO SHIFT is still present, therefore, the AC is either complemented and

loaded back into the AC or just loaded back. NO SHIFT combines with $\overline{\text{ADDER L}}$ in an AND gate (D-BS-8I-0-8 C-7) clearing the L to a "0" indicating no overflow.

As soon as the DVI instruction is completed EAE timing is stopped and CP timing is started as explained under EAE Timing Termination.

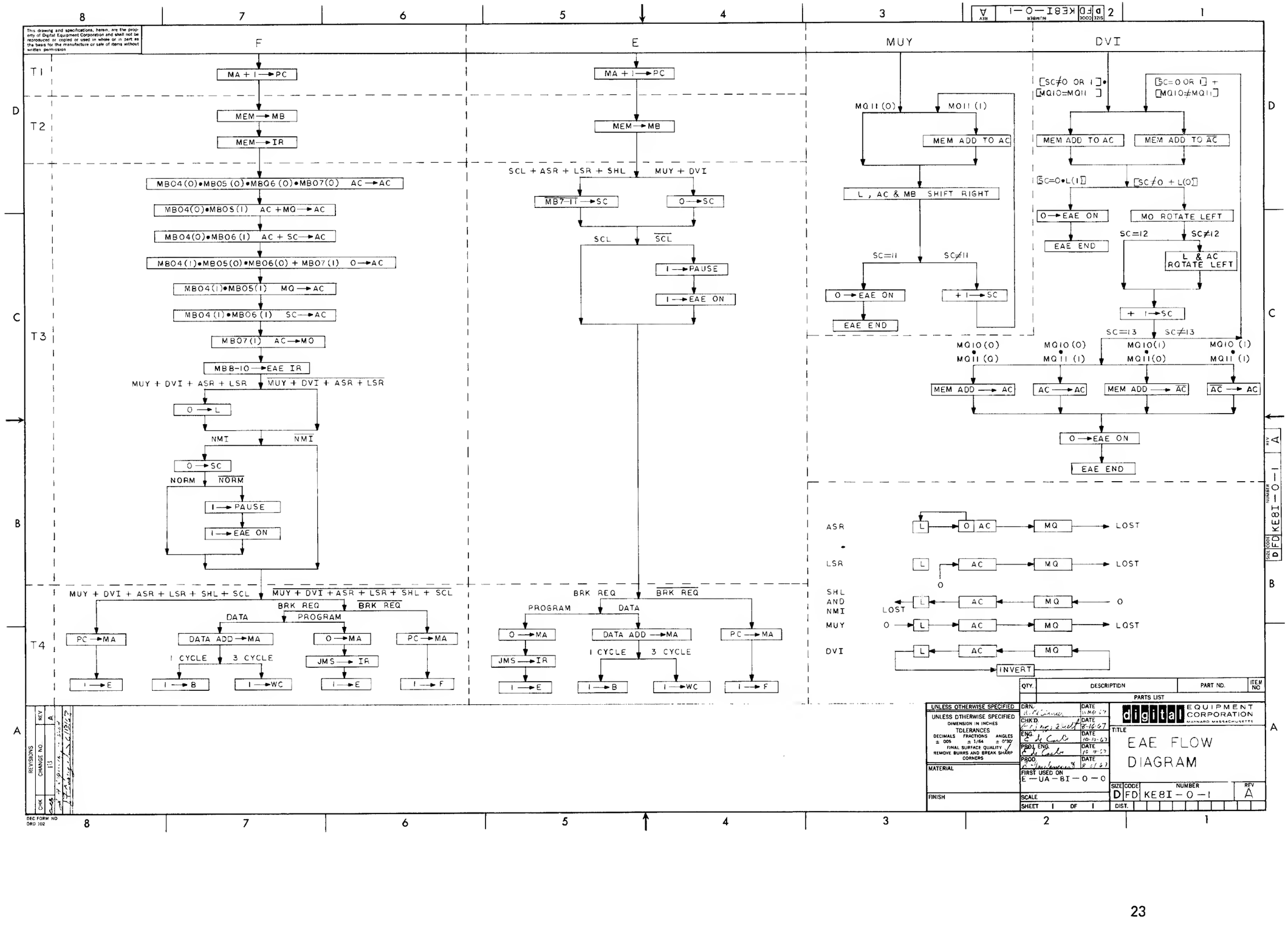
MAINTENANCE

The maintenance procedures for the PDP-8/I should be followed for the KE8/I control logic.

ENGINEERING DRAWINGS

The following drawings pertaining to the KE8/I are included in this section.

Drawing No.	Title	Rev.
D-BS-KE8I-0-1	EAE Flow Diagram	A
D-BS-KE8I-0-2	EAE Control	K
D-BS-KE8I-0-3	Multiplier Quotient and Step Counter	E



QTY.	DESCRIPTION	PART NO.	ITEM NO.

PARTS LIST

REVISIONS	REV	CHANGE NO	CHK
	A	13	

UNLESS OTHERWISE SPECIFIED
DIMENSION IN INCHES
TOLERANCES
DECIMALS ± .005 FRACTIONS ± 1/64 ANGLES ± 0°30'
FINAL SURFACE QUALITY
REMOVE BURRS AND BREAK SHARP CORNERS

MATERIAL

FINISH

DRN.		DATE 11-NOV-67
CHK'D.		DATE 8-16-67
ENG.		DATE 10-10-67
PROJ. ENG.		DATE 10-9-67
PROD.		DATE 8-16-67

FIRST USED ON E—UA—8I—0—0

SCALE

SHEET 1 OF 1

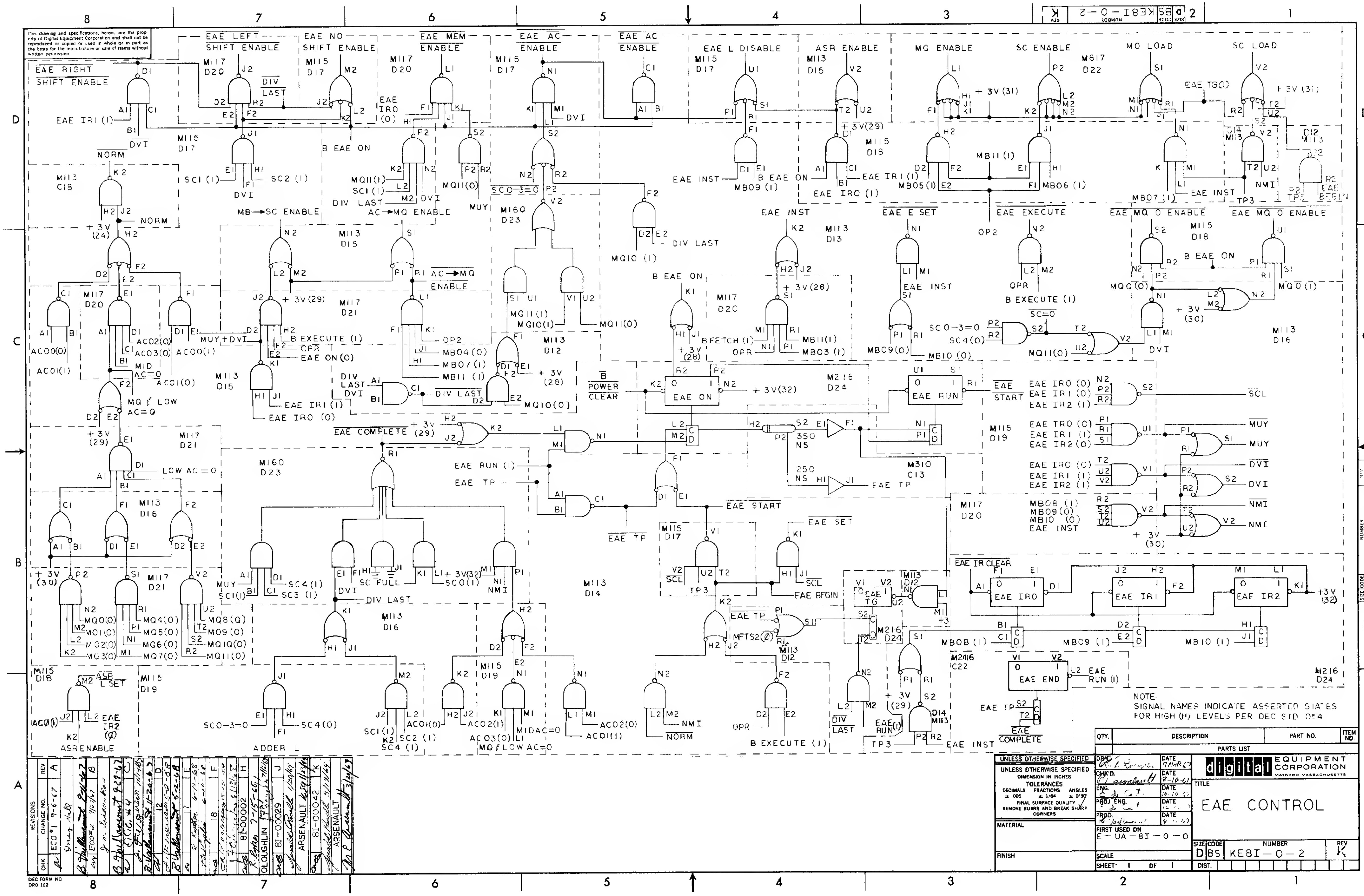
digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE
EAE FLOW DIAGRAM

SIZE D | CODE FD | NUMBER KE8I—0—1 | REV A

DIST.

DEC FORM NO DRD 102


This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission
EAE RIGHT SHIFT ENABLE
EAE LEFT SHIFT ENABLE
EAE NO SHIFT ENABLE
EAE MEM ENABLE
EAE AC ENABLE
EAE AC ENABLE
EAE L DISABLE
ASR ENABLE
MQ ENABLE
SC ENABLE
MQ LOAD
SC LOAD
MB→SC ENABLE
AC→MQ ENABLE
EAE E SET
EAE EXECUTE
EAE MQ 0 ENABLE
EAE MQ 0 ENABLE
EAE ON
EAE RUN
EAE COMPLETE
EAE START
EAE TP
EAE SET
EAE BEGIN
EAE TG
EAE IR CLEAR
EAE IR0
EAE IR1
EAE IR2
EAE END
B POWER CLEAR
ADDER L
ASR ENABLE
SC FULL
MQ & LOW AC=0
LOW AC = 0
NOTE:
SIGNAL NAMES INDICATE ASSERTED STATES FOR HIGH (H) LEVELS PER DEC STD 0F4
QTY. DESCRIPTION PART NO. ITEM NO.
PARTS LIST
UNLESS OTHERWISE SPECIFIED
DIMENSION IN INCHES
TOLERANCES
DECIMALS ±.005 FRACTIONS ±1/64 ANGLES ±0°30'
FINAL SURFACE QUALITY
REMOVE BURRS AND BREAK SHARP CORNERS
MATERIAL
FINISH
FIRST USED ON E-UA-8I-0-0
digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS
TITLE
EAE CONTROL
SIZE D CODE BS NUMBER KE8I-0-2 REV K
SCALE
SHEET 1 OF 1
DIST.
REVISIONS
DEC FORM NO DRD 102

NOTE:
SIGNAL NAMES INDICATE ASSERTED STATES FOR HIGH(H) LEVELS PER DEC STD 054

RIGHT SHIFT
AC → MQ ENABLE
LEFT SHIFT
EAE RIGHT SHIFT ENABLE
SC LOAD
EAE ON (1)
MB → SC ENABLE
MQ LOAD
SC FULL

REV	REVISIONS	CHK	CHANGE NO
A			
B	ECO#1 7-27-67		
C	ECO#2 9-21-67		
D	ECO#3 11-23-67		
E	ECO#4 2-22-68		
F	ECO#5 7-12-68		

QTY.	DESCRIPTION	PART NO.	ITEM NO.

PARTS LIST

UNLESS OTHERWISE SPECIFIED
UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES
TOLERANCES
DECIMALS ± .005 FRACTIONS ± 1/64 ANGLES ± 0°30'
FINAL SURFACE QUALITY / REMOVE BURRS AND BREAK SHARP CORNERS
MATERIAL
FINISH

		DATE
DRN	R. Cormier	7/12/67
CHK'D	C. Hainsworth	8-16-67
ENG.	P. J. Costa	10-10-67
PROJ. ENG	P. J. Costa	10-10-67
PROD.	R. Hollowell	10-11-67

FIRST USED ON
E -UA -8I-0-0

SCALE
SHEET 1 OF 1

digital EQUIPMENT CORPORATION
MAYNARD MASSACHUSETTS

TITLE
MULTIPLIER QUOT & STEP COUNTER

SIZE	CODE	NUMBER	REV.
D	BS	KE8I-0-3	E

DIST.

DEC FORM NO DRD 102