

**KE11-A**  
**extended arithmetic**  
**element**

1st Edition, May 1971  
2nd Printing, February 1972  
3rd Printing, May 1972  
4th Printing, November 1972  
5th Printing, April 1973  
6th Printing, July 1973

Copyright © 1971, 1972, 1973 by Digital Equipment Corporation

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DEC	PDP
FLIP CHIP	FOCAL
DIGITAL	COMPUTER LAB
UNIBUS	

# CONTENTS

	Page		Page
<b>CHAPTER 1 GENERAL DESCRIPTION AND INSTALLATION</b>			
1.1	1-1		
1.2	1-1		
1.3	1-1		
1.4	1-1		
<b>CHAPTER 2 PROGRAMMING THE KE11-A</b>			
2.1	2-1		
2.1.1	2-1		
2.1.2	2-1		
2.1.3	2-1		
2.1.4	2-2		
2.1.5	2-3		
2.2	2-3		
2.2.1	2-3		
2.2.2	2-3		
2.2.3	2-3		
2.2.4	2-4		
2.2.5	2-4		
2.3	2-4		
2.3.1	2-4		
2.4	2-5		
<b>CHAPTER 3 THEORY OF OPERATION</b>			
3.1	3-1		
3.2	3-1		
3.3	3-2		
3.4	3-3		
3.4.1	3-3		
3.4.2	3-3		
3.4.2.1	3-4		
3.4.2.2	3-4		
3.4.3	3-4		
3.4.3.1	3-4		
3.4.3.2	3-4		
3.4.3.3	3-5		
3.4.3.4	3-5		
		3.4.3.5	3-5
		3.4.3.6	3-5
		3.4.4	3-5
<b>CHAPTER 4 GENERAL HARDWARE DESCRIPTION</b>			
4.1	4-1		
4.2	4-1		
4.2.1	4-1		
4.2.1.1	4-2		
4.2.1.2	4-3		
4.2.2	4-3		
4.2.2.1	4-4		
4.2.2.2	4-5		
4.2.3	4-5		
<b>CHAPTER 5 DETAILED HARDWARE DESCRIPTION</b>			
<b>CHAPTER 6 MAINTENANCE</b>			
<b>APPENDIX A COMPLEX IC DESCRIPTIONS</b>			
A.1	A-1		
A.2	A-2		
A.3	A-2		
A.4	A-3		
<b>APPENDIX B KE11-A INSTRUCTION SUMMARY</b>			
<b>APPENDIX C NAMES OF MATHEMATICAL TERMS</b>			
<b>ILLUSTRATIONS</b>			
	Figure No.	Title	Art No. Page
	3-1	Multiply Algorithm Flowchart	11-0356 3-4
	3-2	Divide Algorithm Flow Chart	11-0357 3-6
	3-3	Normalize Algorithm Flow Chart	11-0358 3-7
	3-4	Arithmetic Shift Algorithm Flow Chart	11-0359 3-7
	3-5	Logical Shift Algorithm Flow Chart	11-0360 3-8

## ILLUSTRATIONS (cont)

Figure No.	Title	Art No.	Page
4-1	KE11-A Bit-Slice Diagram	11-0365	4-2
4-2	Multiply Implementation Flow Chart	11-0366	4-3
4-3	Divide Implementation Flow Chart	11-0363	4-4
5-1	Clock Rate Adjustment	11-0362	5-4
A-1	7482 2-Bit Binary Full Adders	11-0361	A-1
A-2	74H87 4-Bit True/Complement, Zero/One Element		A-2
A-3	74153 Dual 4-Line-To-1-Line Data Selector/Multiplexer		A-2
A-4	8271 4-Bit Shift Register	11-0364	A-3

## TABLES

Table No.	Title	Page
1-1	Related Documents	1-1
1-2	List of Diagnostic Programs	1-2
2-1	KE11-A Device Registers	2-1
2-2	KE11-A Status Register	2-2
2-3	KE11-A Reloadable Condition Codes	2-3
2-4	KE11-A Address Assignments	2-4
2-5	KE11-A Operation Timing	2-5
5-1	Sign Extension Logic	5-16
6-1	Timing Signal Functions	6-1
A-1	Adder Truth Table	A-1
A-2	Truth Table	A-2
A-3	Truth Table	A-3
A-4	Control State Truth Table	A-3
C-1	Operand Names	C-1

# CHAPTER 1

## GENERAL DESCRIPTION AND INSTALLATION

### 1.1 GENERAL DESCRIPTION

The KE11-A Extended Arithmetic Element is a device that performs arithmetic operations on numerical data. The KE11-A can be included in a PDP-11 System configuration as a device attached to the Unibus <sup>TM</sup> to expedite numerical calculations.

The KE11-A performs five arithmetic operations.

- a. Multiplication
- b. Division
- c. Three different shift operations on data operands of up to 32 bits.

The KE11-A is *not* a processor option; it communicates with the processor entirely through Unibus data transfers and can operate simultaneously with processor operations.

### 1.2 SPECIFICATIONS

The physical and environmental requirements of the KE11-A Extended Arithmetic Element are the same as the corresponding requirements of the processor in the PDP-11 System. All power requirements of the KE11-A are supplied by the system power supply; the KE11-A requires 4A at +5V.

### 1.3 DOCUMENTATION

This manual is divided into six chapters, as follows:

- a. General Description and Installation
- b. Programming the KE11-A
- c. Theory of Operation
- d. General Hardware Description
- e. Detailed Hardware Description – including circuit schematics and logic equations
- f. Maintenance

The KE11-A communicates with other devices in a PDP-11 System via the Unibus; however, the structure and operation of the Unibus is beyond the scope of this manual. A discussion of the Unibus is included in the *Unibus Interface Manual*, DEC-11-HIAB-D.

Other documents that are related to the KE11-A and the PDP-11 System are listed in Table 1-1.

<sup>TM</sup> Unibus is a trademark of Digital Equipment Corporation.

Table 1-1  
Related Documents

<i>PDP-11 Peripherals and Interfacing Handbook</i>	112.01071.1854
<i>PDP-11/20/15/R20 Processor Handbook</i>	112.01071.1855
<i>PDP-11 Conventions Manual</i>	DEC-11-HR6A-D
<i>KE11-A User's Guide</i>	
<i>PDP-11 Paper Tape Software Programming Handbook</i>	DEC-11-GGPA-D

### 1.4 INSTALLATION

The KE11-A Extended Arithmetic Element consists of five modules mounted in a wired PDP-11 System Unit. This system unit connects to the Unibus and to the PDP-11 power supply through standard connectors that occupy standard module slots in the system unit.

The five modules used in the KE11-A are as follows:

Quantity	Module
1	M827 Clock and Status
1	M7211 Register Control
2	M234 Register
1	M7210 Data Control

The KE11-A is installed by putting the system unit in a PDP-11 mounting box, inserting the KE11 module set, the Unibus connectors, and the power connector in the module slots, and verifying correct installation by executing the diagnostic and normal operating programs (refer to Table 1-2).

#### NOTE

The procedure for installing the system unit is described in the *PDP-11 Conventions Manual*, DEC-11-HR6A-D. The KE11-A modules are inserted in the slots as shown on Drawing D-MU-KE11-A-MU.

Table 1-2  
List of Diagnostic Programs

MainDEC No.	Title
MainDEC-11-DOTA	KE11-A Random Exerciser
MainDEC-11-DOSA	KE11-A Basic Logic Test
MainDEC-11-DOQC	PDP-11 System Test 17 (version C or later)

The KE11-A Unibus addresses are hardwired to one of two sets of addresses: 777300 through 777316 or 777320 through 777336. The KE11-A is normally supplied with the lower set of addresses; if a second KE11-A is added to the system, the higher set of addresses is used. The KE11-A does not use the interrupt system and has no interrupt vector address or priorities.

## CHAPTER 2 PROGRAMMING THE KE11-A

The *KE11-A User's Guide* provides detailed programming information for the KE11-A. This chapter provides a summary presentation and describes KE11-A operations. (Appendix B is a reference chart of this information.)

The KE11-A communicates with the Unibus through a set of device registers (refer to Table 2-1). These registers can be read or loaded from the bus. The registers contain both the data used in calculations and information concerning the operation to be performed on the data or an operation that has been performed on the data. The PDP-11 processor controls the KE11-A through the information that it puts in these registers.

The operating speed of the KE11-A is equivalent to one instruction cycle of the processor. Both the data and the operation to be performed on that data must be specified in one data transfer. Therefore, the KE11-A does not require explicit, separate instructions to specify an operation; instead, the operation is determined implicitly from the address to which data is supplied.

An example of this implicit specification is the addressing of the second operand register for multiplication and division. For multiplication, this register contains the multiplicand; for division, it contains the divisor. The register cannot be explicitly addressed and returns all 0s when read. If the register is loaded through address 777306, the KE11-A executes a multiply operation; if the register is loaded through address 777300, the KE11-A executes a divide operation.

Several of the registers in the KE11-A are accessed through more than one bus address. In each case, the different addresses provide different implicit functions that improve the speed and efficiency of communication between the KE11-A and the PDP-11 processor. These functions are discussed for each individual register in the following sections.

### 2.1 KE11-A REGISTERS

The KE11-A contains four explicitly addressable registers and one implicitly addressable register. One of the explicitly addressable registers, the Step Counter (SC), can also be implicitly addressed for several types of operations. All of the registers interact in various ways both during loading and during operation. Table 2-1 lists the registers and their addresses. The individual registers and details of their operation are discussed in the following paragraphs.

Table 2-1  
KE11-A Device Registers

Mnemonic	Register Name	Addresses*
MQ	Multiplier-Quotient	777304 (777324)
AC	Accumulator	777302 (777322)

\*Addresses are shown for the normal assignments; the higher addresses in parentheses are used for a second KE11-A.

Table 2-1 (Cont)  
KE11-A Device Registers

Mnemonic	Register Name	Addresses*
SC	Step Counter	777310 (777330)
SR**	Status Register	777311 (777331)
X	X	none

\*Addresses are shown for the normal assignments; the higher addresses in parentheses are used for a second KE11-A.

\*\*The SR is the high byte of the word located at the SC address.

#### 2.1.1 Multiplier Quotient Register

The Multiplier Quotient (MQ) register contains the multiplier at the beginning of a multiplication and the quotient at the end of a division. The MQ holds the less significant half of the two-word dividend before a division. Two-word numbers contained in the MQ and the accumulator (AC) can be shifted or normalized.

The MQ is explicitly addressed at location 777304 and can be read or loaded. When the MQ is loaded, the value of the most significant bit is reproduced in the AC, thereby extending the sign of one-word operands for division and shift operations. If only the low byte of the MQ is loaded, the sign is extended into both the more significant byte of the MQ and the AC. Loading the high byte extends the sign into the AC but does not disturb the low byte.

#### 2.1.2 Accumulator Register

The AC is used as a storage register for the results of additions and subtractions during multiplication and division. It contains the more significant half of a two-word result following a multiplication and the remainder following a division. Also, the AC contains the more significant half of the dividend before a division. The AC participates in shifts and normalize functions with the MQ.

The AC is addressed explicitly at location 777302 and can be read or loaded. Sign extension from the MQ affects the AC; thus, the AC should be loaded after the MQ and read before the MQ, if the instructions that read the data also load the registers (e.g., adding a number to the contents of the AC and MQ for further processing).

#### 2.1.3 The Step Counter

The step counter (SC) register participates in all KE11-A operations. The SC is directly addressable at location 777310. In addition, the SC can be loaded at locations 777314 and 777316 and read at location 777312 to

implicitly specify an operation that the KE11-A does whenever the particular address is loaded. The locations and their associated operations are as follows:

- a. 777312 – The KE11-A performs a normalize (NOR) operation. (The value loaded into 777312 is ignored, because the SC is set to 0 before the NOR begins.)
- b. 777314 – The KE11-A performs a logical shift (LSH) operation.
- c. 777316 – The KE11-A performs an arithmetic shift (ASH) operation.

The SC is addressed as the low byte of a word. At the explicit SC location (777310), the high byte of the word is the status register. At location 777312 (NOR), the SC can be read as a word quantity, because the SC is always positive after a normalize operation and all more significant bits of the word are 0s. Attempting to read the SC at location 777314 (LSH) or 777316 (ASH) returns all 0s.

The SC can be loaded at one of three locations. Two of these locations, 777314 (LSH) and 777316 (ASH), implicitly specify shift operations. The third location is the explicit address of the SC. The SC cannot be loaded at the explicit address by a byte transfer; it must be loaded as a word in conjunction with the status register. Data loaded into address 777312 (NOR) is lost and does not affect the SC directly.

#### 2.1.4 The Status Register (SR)

Location 777311 provides explicit access to eight bits (indicators) that display various conditions within the KE11-A. These condition codes are different from the condition codes in the PDP-11 processor and indicate only conditions within the KE11-A registers. Table 2-2 lists the eight indicators and describes the meaning of the conditions represented by each indicator. These indicators are especially helpful in determining if the numbers that result from KE11-A operations are within the expected range.

Five of the eight bits in the SR are read-only; they cannot be loaded from the bus, and they always indicate conditions of the AC and MQ registers. The remaining three bits can be read from the bus and loaded, but the data transfer into the SR must be a full word transfer that also loads the SC.

Table 2-2  
KE11-A Status Register

Bit	Indicates	Meaning
0	Carry	Last bit shifted out of combined AC-MQ during ASH or LSH; cleared by multiply, divide, or normalize.
1	AC = MQ15	Single Precision – All bits in the AC are the same as MQ bit 15; thus, the number is contained entirely in the MQ, and the AC is the extended sign.
2	AC = MQ = 0	The contents of the AC and the MQ are entirely 0.
3	MQ = 0	The contents of the MQ register are 0.
4	AC = 0	The contents of the AC register are 0.
5	AC = 177777	The contents of the AC register are entirely 1s.
6	Negative	The results of the last operation were negative. For a multiply or a shift, AC15 is a 1; for a successful divide, MQ15 is a 1; for an unsuccessful

(Continued on next page)

Table 2-2 (Cont)  
KE11-A Status Register

Bit	Indicates	Meaning
6 (Cont)		divide (overflow occurred) SR6 contains the sign of the original dividend (contents of combined AC-MQ).
7	Overflow Exclusive OR Negative	This bit indicates that the result of a division is too large for a 16-bit word, or a significant bit has been lost in a left shift. Note that the overflow indication is affected by SR6; see the discussion below.

The overflow indication provided by the KE11-A is not a simple flag or condition code. The KE11-A overflow indication (bit 7) is combined with the KE11-A sign indication (bit 6) in a manner that permits setting of the PDP-11 overflow (V) and negative (N) condition codes from the KE11-A condition codes. The PDP-11 codes can then be used in branch instructions that refer only to the V and N codes.

The PDP-11 V bit is set during shifting operations in the same manner as the KE11-A recognizes overflow. When the last bit shifted out and the sign bit of the word shifted differ, the V bit is set. If the KE11-A SR is shifted left, the PDP-11 compares bits 6 and 7 of the SR and sets the PDP-11 V bit if they are different. Bit 7 of the KE11-A SR is the exclusive OR of the overflow condition and bit 6; thus, bit 7 has the same value as bit 6 if there is no overflow and the opposite value if there is an overflow.

The PDP-11 negative condition code is set from the contents of SR bit 6, but the PDP-11 carry is set from SR bit 7, not from the KE11-A carry, which is KE11-A SR bit 0.

The KE11-A SR cannot be loaded by a byte transfer. As a result, if a Rotate Left, Byte (ROLB SR) instruction is used to perform the left shift, the PDP-11 V and N condition codes are set from the KE11-A overflow and negative condition codes, respectively, without changing the contents of the KE11-A SR.

Five of the KE11-A condition codes are direct indications of the state of the AC and MQ registers. These codes (SR bits 1 through 5) always indicate the current state of the registers and cannot be loaded from the Unibus. The remaining three condition codes (bits 0, 6, and 7) are supplied from flip-flops that can be read or loaded. These three codes represent conditions that result from preceding KE11-A operations; the reloading capability is provided to permit reentrant KE11-A programming.

#### NOTE

**The SR can only be loaded by a word transfer that also loads the Step Counter. This requirement is a result of blocking byte transfers to allow the ROLB SR instruction to test the SR condition codes without modifying the SR.**

Table 2-3 illustrates the possible effects on the three modifiable condition codes as the result of each KE11-A operation. Because SR07 is the exclusive OR of an overflow condition and SR06, the overflow condition is also shown. The following notation is used in Table 2-3:

Symbol	Meaning
0	The condition is cleared.
C	The indicator is set conditionally.
=	SR07 has the same value as SR06.



Symbol	Meaning
X	SR07 is the exclusive OR of SR06 and the overflow condition (OVFL).
*	If overflow occurs (unsuccessful divide), SR06 is set to the original sign of the dividend. Otherwise, SR06 is set conditionally.

Table 2-3  
KE11-A Reloadable Condition Codes

Operation	C (SR00)	N (SR06)	OVFL	SR07
Multiply	0	C	0	=
Divide	0	*	C	X
Normalize	0	C	0	=
Logical Shift	C	C	C	X
Arith. Shift	C	C	C	X

### 2.1.5 The X Register

The X register in the KE11-A cannot be addressed explicitly. Data loaded into the multiply or divide address is transferred to an internal register, designated the X register, for use in operations on the contents of the AC and MQ registers. The X register cannot be read from the bus; reading location 777300 (divide) or 777306 (multiply) returns all 0s.

## 2.2 KE11-A OPERATIONS

The KE11-A can perform five operations that are implicitly specified by loading data into one of five bus locations (refer to Table 2-4). These operations are:

- a. Multiplication
- b. Division
- c. Normalization
- d. Logical (0 input) shifting
- e. Arithmetic (sign extended) shifting

These operations are explained in the following sections. The explanation specifies the techniques required to initiate the operation, the results of the operation, and the way to recover the results.

### 2.2.1 Multiplication

The KE11-A can multiply two 16-bit numbers in 2's complement binary representation to form a 32-bit (two word), 2's complement product. The 2's complement representation can express numbers in the range  $+2^{15} - 1$  to  $-2^{15}$  using 16 bits; the most significant bit is the sign of the number. The range of 32-bit numbers is  $+2^{31} - 1$  to  $-2^{31}$ .

The multiply operation computes the product of a number in the MQ and a number in the X register. The product remains in the AC and the MQ; the more significant 16 bits are in the AC and the less significant bits are in the MQ. The multiplier (originally in the MQ) is lost.

The multiply operation begins when the multiplicand is loaded into the X register by transferring the number to bus location 777306. The multiplier must be in the MQ at that time. The result can be retrieved from the AC and the MQ. See Section 2.3 for PDP-11 instructions to execute multiplication and Section 2.4 for KE11-A timing information.

### 2.2.2 Division

The KE11-A can divide a 32-bit binary number (in 2's complement representation) by a 16-bit, binary number (in 2's complement representation). If the quotient can be expressed correctly as a 16-bit 2's complement number, a quotient and a remainder are returned; if the quotient requires more significant bits, an overflow indicator is set, and no meaningful results are available. The quotient is adjusted to a value that corresponds to a remainder with the same sign as the original dividend; that is, the division does not terminate until the remainder has the proper sign, at which time the quotient has the correct value.

A division operation begins when a divisor is loaded into location 777300. A dividend must already be in the AC and MQ registers. The more significant half of the dividend is in the AC, the less significant half in the MQ; the sign bit of the dividend is AC15. If the quotient does not overflow, the KE11-A assembles the quotient in the MQ and the remainder in the AC. If overflow occurs, it is shown by the values of Status Register bits 6 and 7. See Section 2.3 for information on testing these bits and on the use of PDP-11 instructions to generate a division operation. Section 2.4 contains timing information.

### 2.2.3 Normalization

A 2's complement binary number is expressed by a sign bit and a number of magnitude bits. If the sign bit, which is the most significant bit, is followed by any bits of the same value, those bits do not contribute to the significant magnitude of the number. For example: in the number 0 000 000 011 101 111, the sign bit is 0, signifying a positive number; however, the next 7 bits are also 0. Thus, the number of significant bits in the magnitude of the number is reduced to 8. Similarly, for the number 1 111 101 001 101 101, there are 4 non-significant bits following the sign bit, and the magnitude is reduced to 11 significant bits.

When numbers are represented in a floating point format that uses an exponent and a fraction, it is important to preserve as many significant bits as possible in the fraction. This is done by shifting the fraction bits to the left until the most significant bit of the fraction is different from the sign bit and then modifying the exponent value to reflect the change in the fraction.

#### NOTE

As the fraction is shifted to the left, it takes on successively greater values. The exponent must be reduced by an equivalent amount to restore the proper value to the number that is represented.

The normalization operation performed by the KE11-A shifts a 32-bit number left until one of the following conditions is true:

- a. The two most significant bits are different.
- b. The two most significant bits are 1s, and all other bits are 0s.
- c. The shift count reaches a maximum of 31 (occurs if the starting number is all 0s).

The number of shifts performed is counted and returned in the Step Counter (SC). Reading this number at the normalize address returns a 16-bit, positive number that can be subtracted from an exponent to correct the value of the number.

**NOTE**

All 32-bit numbers are shown as two 16-bit numbers to illustrate the contents of the individual registers. Each 16-bit number is represented by 6 octal digits.

The normalization operation begins when a data transfer is made to location 777312. The data transmitted is ignored, the 32-bit number in the AC and MQ registers is normalized, and the resulting shift count is available in the SC or through the same location (777312). The following special termination conditions are available:

- a. If the original number was all 1s, the result is 140 000-000 000<sub>8</sub> with a value of 30 in the SC.
- b. If the original number was 111 . . 1100 . . 000<sub>2</sub>, the result is 14 000-000 000<sub>8</sub> and the step counter displays one less than the number of 1 bits in the magnitude (not counting the sign bit). These special terminations prevent the result from being 100 000-000 000, which cannot be negated (the negation produces the same number).
- c. If the original number was 0, the result is 0 with 31 in the SC.

**2.2.4 Logical Shifts**

The KE11-A can shift a 32-bit number left up to 31 positions or right up to 32 positions. As the number is shifted, bits that are vacated are filled with 0s, and the bit that is shifted out of the number is saved in bit 0 of the status register. During a left shift, the status register reflects overflow if the values in bit 0 of the SR (the C bit) and AC15 differ; this indicates that the shift has caused a number to change sign. Bit 6 of the SR shows the sign of the number; it has the same value as AC15 after the shift. (For a right shift, AC15 is loaded with 0s, thus, the result cannot be negative.) Status register bits also display the contents of the AC and MQ after a shift of 0 places.

The direction of shifting is specified implicitly by the sign of the number loaded into the SC. The sign of the SC is bit 05; all more significant bits (bits 06-15) are ignored. If the SC is positive, the number is shifted left, and the SC is decremented once for each shift until it reaches 0. If the SC is negative, the number is shifted right, and the SC is incremented (which decrements the 2's complement negative representation) until the SC reaches 0 by overflowing.

Logical shifting begins when the SC is loaded at location 777314. The 32-bit number must be in the AC and MQ registers before the shift begins. Bit 0 of the AC shifts into bit 15 of the MQ on right shifts; MQ15 shifts into AC0 for left shifts.

The KE11-A can be used to shift shorter numbers (e.g., single words), but the operational properties must be preserved by loading the AC and MQ appropriately. For a left shift, the MQ is cleared (which clears the AC by sign extension), and the AC is loaded with the single word to be shifted. The result of shifting up to 16 places is a word in the AC, and bits 0, 6, and 7 of the SR are properly set. For a right shift, the word is loaded into the MQ, the AC is cleared, and the shift is initiated. The result is a word in the MQ, the C bit (bit 0 of the SR) is set properly, and bits 6 and 7 of the SR (used to indicate negative and overflow conditions) are cleared, as for any logical right shift.

**2.2.5 Arithmetic Shift**

Logical shifts replace the bit that is vacated with a 0; the sign bit is not preserved. The KE11-A also provides a shift operation that preserves the sign bit, extending it during right shifts and signalling if any significant bits are lost during left shifts.

When location 777316 is loaded with the shift count, a 32-bit number in the AC and MQ registers is shifted as follows:

- a. If the SC is negative (SC05 is a 1), the number is shifted right, AC14 is set equal to the sign bit (AC15), and SR00 is set to the previous contents of MQ00. The previous contents of SR00 are lost. The SC is decremented and the operation is repeated if the SC is not 0. SR06 (the N bit) displays the contents of AC15, and SR07 is cleared. To summarize: the sign is extended, the last bit shifted out is in the C bit (SR00), and the number is shifted as many places to the right as designated in the shift count.
- b. If the SC is positive, the number in AC (14:00) and MQ (15:00) is shifted left, MQ00 is cleared, AC15 is not changed, and the C bit takes on the previous value of AC14. The N bit reflects AC15, and the overflow condition is indicated if AC14 and AC15 differ. To summarize: the number is shifted left, the sign is preserved, 0s enter the low bits of the number, and the overflow indication is set if any significant bits are shifted out of the register.

**2.3 PDP-11 INSTRUCTIONS AND THE KE11-A**

The KE11-A is operated in PDP-11 Systems by loading and reading KE11-A registers. Transfers are conducted by addressing the KE11-A using any of the instructions that access bus locations. The most common instruction used for this purpose is the move (MOV) instruction.

Data transfers involving the KE11-A occur whenever the processor addresses certain bus locations. Many of these locations also provide implicit operations in the KE11-A. The order of the KE11-A addresses, is designed to make use of the PDP-11 addressing modes to operate the KE11-A at maximum speed and efficiency.

The multiplication and division operations, which use the most operands, are the most frequently used operations and the most critical. The address assignments shown in Table 2-4 enable very fast operation.

**Table 2-4  
KE11-A Address Assignments**

Address	Operation	Register
777300	Divide	X*
777302	None	AC
777304	None	MQ
777306	Multiply	X*
777310	None	SC
777311	None	SR**
777312	Normalize	SC
777314	Logical Shift	SC
777315	Arithmetic Shift	SC

\* The SC is forced to a specific value for these operations.  
\*\*The SR is the high byte of the word at address 777310; the SC is the low byte.

**2.3.1 Multiplication and Division**

The following example contains a sequence of instructions to perform a multiplication and a sequence of instructions to perform a division. In the multiplication example, A is multiplied by B to produce a two-word result that is stored in C and D, two arbitrary bus locations. Register 0 is used as an address pointer; the contents of register 0 after the multiplication are the same as the contents before the operation.

The division forms the quotient and remainder resulting from dividing the two-word value of  $V$  and  $W$  (where  $V$  is the less significant half and  $W$  is the more significant half) by  $X$ . The result is placed in two arbitrary bus locations at  $Y$  and  $Z$ , register 0 is left with its original value (pointing to the MQ register), and the contents of the SR are available for checking. See Section 2.1.4 for an explanation of operations using the KE11-A condition codes.

```

MOV      #MQ,      R0      ;PUT ADDRESS OF MQ
                          ;IN REGISTER 0
.
.
MULT:    MOV      A,      (0)+ ;PUT FIRST NUMBER IN MQ
        MOV      B,      (0)  ;MULTIPLY BY SECOND NUMBER
        MOV      -(0),   C     ;TRANSFER LOW ORDER PRODUCT
        MOV      -(0),   D     ;TRANSFER HIGH ORDER PRODUCT
        TST      (0)+
.
.
DIVD:    MOV      V,      (0)  ;LOAD LOW ORDER DIVIDEND
        MOV      W,      -(0) ;LOAD HIGH ORDER DIVIDEND
        MOV      X,      -(0) ;DIVIDE
        TST      (0)+,
        MOV      (0)+,   Y     ;TRANSFER REMAINDER
        MOV      (0),    Z     ;TRANSFER QUOTIENT

```

Note that for multiplication, if the product is known to be a single word, the last two instructions are not needed.

## 2.4 KE11-A TIMING

KE11-A operations are initiated by a data transfer from the PDP-11 Unibus. The operations continue concurrent with subsequent PDP-11 operations. The maximum time required for a KE11-A operation is approximately 4.5  $\mu$ s. Table 2-5 provides specific timing information for the five KE11-A operations.

Table 2-5  
KE11-A Operation Timing

Operation	Time	Remarks
Multiply	4 $\mu$ s	None
Divide	4.25 $\mu$ s	No time if immediate overflow
Normalize	0-4 $\mu$ s	Exact time depends on number of shifts required.
Logical Shift	0-4 $\mu$ s	Exact time depends on number of shifts required.
Arithmetic Shift	0-4 $\mu$ s	Exact time depends on number of shifts required.

After the KE11-A begins an operation, it does not respond to Unibus transfers until the operation is complete. If the KE11-A is addressed while it is operating, it delays all response until the end of the operating cycle; this stops all Unibus operation. However, the maximum delay is approximately 2  $\mu$ s, because the PDP-11 processor must first fetch the next instruction before re-addressing the KE11-A. Therefore, programs that operate the KE11-A do not have to provide any explicit delays to allow for completion of the operation; the KE11-A can be re-addressed immediately. This 2  $\mu$ s overlap is the only time contribution that the KE11-A makes in addition to the time implicitly required by PDP-11 instructions.



## CHAPTER 3

### THEORY OF OPERATION

This chapter describes KE11-A theory of operation. A review of the requirements for multiplication and division is presented, as well as the algorithms for the five KE11-A operations (multiplication, division, normalization, and arithmetic and logical shifting). See Appendix C for a review of the names of terms in mathematical operations.

#### NOTE

This chapter is provided for readers who want to fully understand the KE11-A multiplication and division algorithms. Readers who are only interested in the operation of the KE11-A are referred to Chapters 4 and 5.

#### 3.1 BINARY 2's COMPLEMENT NOTATION

The KE11-A requires a numerical notation that expresses both the sign and the magnitude of each number in binary digits. The simplest class of notation that meets this requirement is based on the following property: *a number added to its own negative equals zero*. Thus, adding the negative of a number to another number is the same as subtracting the number. The 2's complement of a number is created by complementing and incrementing the number. Adding a number and its negative in 2's complement notation always produces all 0s (the only representation of the quantity 0 in 2's complement).

#### NOTE

It is important to remember that the representation of a number differs greatly from quantity represented. For example: The quantity -1 is represented in 2's complement notation by 11 111 111 (in eight bits). The quantity +1 has a 2's complement representation of 00 000 001.

Example 1 Adding +1 and -1 yields the following:

```

00 000 001 = +1
+11 111 111 = -1
-----
100 000 000 = 0   (The left most (carry) bit is not a significant
                    bit and is ignored.)

```

Example 2 Adding +5 and -3 yields the following:

```

00 000 101 = +5
+11 111 101 = -3
-----
100 000 010 = +2   (Carry bit is not significant.)

```

A disadvantage of 2's complement notation is that the representation of numbers is not symmetrical. That is, one more negative number than positive number can be expressed. In  $n$  bits, the maximum positive number that can be expressed is  $2^{n-1} - 1$ , but the maximum negative number is  $-2^{n-1}$  (because there is no negative zero).

#### 3.2 MULTIPLICATION

Multiplication is repeated addition. Multiplying 3 times 7 is simply adding 7 three times. However,  $2^{15}$  times a number requires  $2^{15}$  additions; the KE11-A uses a shortcut method that requires only 16 operations.

In practice, the KE11-A adds multiples of the multiplicand. The multiples are formed by shifting. Each time a binary number is shifted one bit to the left, it is multiplied by two; thus, if it is shifted 5 places to the left, it is multiplied by  $2^5$  (32). The multiplier is broken down into individual bits that determine which multiples of the multiplicand are added to form the product.

The multiplication process is complicated by the representation of negative numbers used in PDP-11 Systems. Negative 2's complement numbers cannot be multiplied by the addition of multiples unless a correction step is added at the end. To avoid this step, the KE11-A uses a method that provides for negative numbers and produces the same results as the addition of parts method for positive numbers. This method is based on a different breakdown of a binary number into positive and negative parts.

In binary numbers,  $10 - 1 = 1$ . Representing each 1-bit of a binary number as the difference between that bit and the next most significant bit produces a string of alternating positive and negative powers of two.

For example:

$$\begin{aligned}
 11010111 &= 10000000 - 10000000 + 10000000 - 1000000 \\
 &\quad + 100000 - 10000 + 1000 - 100 + 100 - 10 + 10 - 1 \\
 &= 100000000 - 1000000 + 100000 - 10000 + 1000 - 1
 \end{aligned}$$

Multiplying a multiplicand by each of the numbers in the last string (preserving the signs) and then adding the products of the multiplications is equivalent to multiplying the chosen multiplicand by the original number (11010111). This can be done by shifting the multiplicand left and adding or subtracting at each position that corresponds to one of the numbers in the series.

The series of alternating positive and negative powers of two is easily generated because:

- a. Each pair of powers of 2, one positive and a smaller negative, represents a string of 1s. The positive number is one digit higher than the most significant 1 in the string, and the negative number is in the same position as the least significant 1 in the string.

For example, in the number 11010111:

$$\begin{array}{r} 10000000 - 1000000 = 11000000 \\ 100000 - 10000 = 00010000 \\ 10000 - 1 = \underline{00000111} \\ 11010111 \end{array}$$

- b. Strings of 1s are separated by strings of 0s. Each string is one or more digits long.

For example, in the number 11010111:

$$\begin{array}{r} -1000000 + 100000 = 0 \times 2^5 \\ -10000 + 1000 = \underline{0 \times 2^3} \\ 11010111 \end{array}$$

- c. Thus, each string of 1s can be replaced by a string of 0s with a -1 in the least significant place, and each string of 0s can be replaced by 0s with a +1 in the least significant place.

For example, the digits in the number 11010111 can be replaced as follows:

$$\begin{array}{r} \text{Original digit: } 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \\ \text{Replacement: } 0 \quad -1 \quad +1 \quad -1 \quad +1 \quad 0 \quad 0 \quad -1 \end{array}$$

- d. Therefore if, for any bit of the multiplier, the previous (less significant) bit is the same, the multiplicand is not added to the partial product (or it is multiplied by 0 and 0 is added to the product). If the previous bit is a 1 and the current bit is a 0, the multiplicand is added; if the previous bit is 0 and the current bit is 1, the multiplicand is subtracted (negated and added). In each case the multiplicand is shifted (with respect to the product) before the addition, because the number added to the product is actually a power of two times the multiplicand.

For example, to multiply  $N$  by 11010111, the sum of the products of  $N$  times each replacement digit, times the appropriate power of 2, is the product as follows:

$$\begin{aligned} N \times 11010111 = N \times (0 \times 2^7 - 1 \times 2^6 + 1 \times 2^5 - 1 \times 2^4 \\ + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 - 1 \times 2^0) \end{aligned}$$

### 3.3 DIVISION

Division is repeated subtraction. Division is more complicated than multiplication for two reasons:

- a. The product of two integers is always an integer; the quotient of two integers is rarely an integer. Division produces two results, a quotient and a remainder, that interact. The correct quotient is dependent on a correct remainder.
- b. The maximum value that results from the multiplication of two numbers can be no larger than the square of the maximum number. However, the maximum value that can result from a division is infinite, because the divisor can be much smaller than the dividend. Some quotients cannot be expressed in the number of bits available in the physical representation and are considered to have overflowed.

The quotient in a division is the number of times that the divisor can be subtracted from the dividend without going beyond 0 (changing sign). The result can be determined by counting subtractions until the remainder does go beyond 0 (which produces a condition called *underflow*), then reducing the count by one. The remainder must also be corrected by restoring the value of one subtraction.

Rather than do as many as  $2^{15}$  subtractions, the KE11-A uses a short-cut method similar to that used in multiplication. The results of dividing by multiples of the divisor, where each multiple is a power of two times the divisor, can be combined to form the quotient.

This division procedure operates by subtracting a large multiple ( $2^{n-1}$ ) of the divisor from the dividend. If the remainder does not go beyond 0 (there is no underflow), the next smaller power-of-two multiple of the divisor is subtracted. For each successful subtraction, the quotient is increased by the same multiple (the same power of two) as the multiple of the divisor used in the subtraction.

If a subtraction causes underflow, however, the corresponding quotient bit is cleared (the corresponding power of two is not added to the quotient). However, rather than restoring the previous value of the dividend, the KE11-A now approaches the correct remainder from the opposite direction. Successively smaller multiples of the divisor are added to the remainder (instead of subtracting) until the remainder again underflows, thus restoring the original sign. When the KE11-A is adding, instead of subtracting, the corresponding quotient bits are set only if the sign of the remainder returns to its original value; if the remainder does not change sign, the quotient bit is set to 0.

For example, dividing 17 ( $21_8$ ) by 5 yields a quotient of 3 and a remainder of 2 as follows:

1. Subtract Divisor  $\times 2^3$  from the Dividend.

$$\begin{array}{r} 00\ 010\ 001 = 21_8 \\ \underline{11\ 011\ 000} = -5 \times 2^3 = -50_8 \\ 11\ 101\ 001 \end{array}$$

The partial remainder has the wrong sign (underflow occurred).

2. Add  $0 \times 2^3$  to the quotient = 0.
3. Add Divisor  $\times 2^2$  to the partial remainder.

$$\begin{array}{r} 11\ 101\ 001 \\ \underline{00\ 010\ 100} = 5 \times 2^2 = 24_8 \\ 11\ 111\ 101 \end{array}$$

The partial remainder has the wrong sign (no underflow).

4. Add  $0 \times 2^2$  to the quotient = 0.
5. Add Divisor  $\times 2^1$  to the partial remainder.

$$\begin{array}{r} 11\ 111\ 101 \\ \underline{00\ 001\ 010} = 5 \times 2^1 = 12_8 \\ 00\ 000\ 111 \end{array}$$

The partial remainder has the right sign (underflow occurred).

6. Add  $1 \times 2^1$  to the quotient = 2.
7. Subtract Divisor  $\times 2^0$  from the partial remainder.

$$\begin{array}{r} 00\ 000\ 111 \\ \underline{11\ 111\ 011} = -5 \times 2^0 = -5 \\ 00\ 000\ 010 \end{array}$$

The remainder has the right sign and is 2 (no underflow).

8. Add  $1 \times 2^0$  to the quotient = 3.

This procedure works for positive or negative numbers, provided that the dividend and divisor have the same sign. However, if the signs are originally different, subtracting multiples of the divisor drives the remainder away from 0. Therefore, the KE11-A adds multiples of the divisor until the remainder underflows (at which point, a quotient bit is set) and then subtracts until the remainder regains its original sign.

This procedure can handle any combination of binary numbers, regardless of sign. Implementation of the procedure is simplified by the following considerations:

- a. If the signs of the divisor and the dividend are originally the same, the KE11-A subtracts until they differ (because the sign of the remainder changes), then adds until they are the same. If the signs are originally different, the KE11-A adds until they are the same, then subtracts until they differ.
- b. Therefore, for each operation, the KE11-A compares the signs of the remainder and divisor. If they differ, the KE11-A adds the divisor, shifted to form the proper multiple (power of two times the divisor); if the signs are the same, the KE11-A subtracts.
- c. A quotient bit is set if the sign of the remainder remains the same after a subtraction or changes after an addition; the quotient bit is cleared if the sign changes after a subtraction or remains the same after an addition.
- d. A subtraction is done if the signs of the remainder and divisor are the same, and an addition is done if the signs are different. A changed sign after an addition means that the signs are now the same, while no change after a subtraction also means the signs are the same.
- e. Therefore, after each operation, the corresponding bit of the quotient is set if the signs are the same or cleared if the signs differ.

### 3.4 ALGORITHMS FOR KE11-A OPERATIONS

Figures 3-1 through 3-5 illustrate the sequence of operations for multiplication, division, and shifting. These flow charts emphasize the conceptual organization of the device that does each calculation; Chapter 4 relates the KE11-A logic to these algorithms and explains how the logic structure reduces the hardware and timing requirements.

Beside each flow chart is a map of the logical storage facilities necessary for the operation illustrated. For each register that is operated on as data (added, subtracted, or shifted), a subscript notation is used to describe individual bits or bit strings (e.g.,  $MR_0$  is the 0 bit of the multiplier). Registers that are used as controls or indicators are denoted by a suffix (e.g., SR6 is bit 06 of the Status Register). Comments and explanatory information are provided in parentheses.

Two basic operations are shown in each algorithm: replacement (symbolized by a left arrow  $\leftarrow$ ) and comparison (symbolized by the equals sign  $=$ ). The replacement operation substitutes the value of the expression to the right of the left arrow for the variable to the left of the arrow. Comparison does not modify the value of either operand, but returns a value of true if the operands are equal or false if they are not equal. The arithmetic operators (+, -, \*, and /) are interpreted normally. The exact interpretation is often described in a comment. A circled plus sign symbolizes the exclusive OR logical operation. Each flow chart begins with a set of initialization operations; a LOAD operation is a replacement operation where the new value of a variable is an input to the algorithm and is not necessarily specified explicitly.

#### 3.4.1 Basic Shift Operation

A basic shift operation is used as a primary operation in the sequences for all other operations. The register that is being shifted is treated as a sequence of bits, each shifted separately. The following description illustrates the features of a basic shift:

- a. In general, the bit at a particular location is replaced by another bit that is shifted into that location. No bit of information is moved more than one location.
- b. One bit is shifted out of the register and is lost.
- c. One bit is vacated. The original contents of that bit are shifted to the next bit, and a 0 replaces the previous bit.
- d. The bit lost is at the end toward which the bits are shifted, and the bit vacated is at the end away from which the bits are shifted (bit positions are numbered in ascending order from right to left).

#### 3.4.2 Multiplication

The multiplication algorithm discussed in Section 3.2 is illustrated in Figure 3-1. The two operands, a step counter (SC), and an adder/subtractor are used to produce a sum that is the desired product.

The multiplication consists of  $N$  cycles through a loop, where  $N$  is the number of bits in the multiplier. During each cycle, a different bit of the multiplier is compared with the previous (next less significant) bit, and one of three operations is done. This comparison is done by saving the previous bit in a status bit,  $C$ , and shifting the multiplier right after each cycle to make bit 0 the new current bit and to make the old current bit the new contents of  $C$ .

After an operation, which is either an addition, a subtraction or no change, the multiplicand is shifted left (multiplying it by two), and the cycle is repeated. The adder/subtractor that is used is only  $N$  bits wide, because the maximum number of significant bits in the multiplicand is  $N$ . (The remaining  $N$  bits are either extended sign or low order 0s, and do not require addition to affect the sum.) The  $N$  bits of the sum that are combined with  $N$  bits of the multiplicand differ for each cycle; in effect, the adder is shifted left along the two registers.

The algorithm also illustrates the following points:

- a. On the first cycle,  $C$  is 0 and only a subtraction or no change operation is possible.
- b. The number of significant bits in the sum increases by one (maximum) for each cycle.
- c. The number of bits in the multiplier that have not been examined decreases by one for each cycle.

The flow chart in Figure 3-1 is accompanied by an algorithm that describes the sequence of operations and by an example that illustrates the contents of the registers at the completion of certain steps. The number in the *Step* column of the example corresponds to a number in the *Step* column of the algorithm.

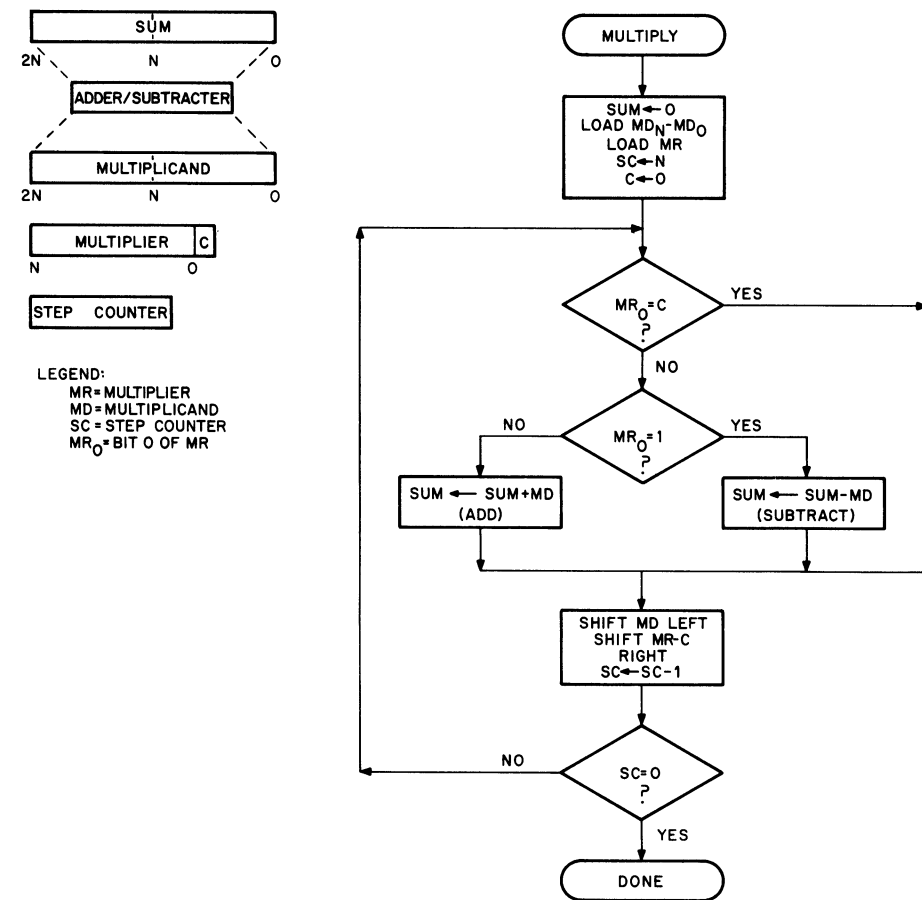


Figure 3-1 Multiply Algorithm Flowchart

3.4.2.1 **Multiply Algorithm** – The sequence of operations is as follows:

Step	Action
1	Clear the sum and the previous bit indicator (C). Load the multiplier and the multiplicand, and load N into the SC.
2	If the current bit 0 of the multiplier is equal to C, go to Step 6.
3	If bit 0 of the multiplier is 1, go to Step 5.
4	Add the multiplicand to the sum. Go to Step 6.
5	Subtract the multiplicand from the sum.
6	Shift the multiplicand left. Shift the multiplier right; the old bit 0 shifts to C. Decrement the SC.
7	If the SC is not 0, go to Step 2.

3.4.2.2 **Multiplication Example** – The following lists the contents of the indicated registers for the example:

5 times 3 equals 15 ( $17_8$ )

MR	C	MD	Sum	SC	Step
0011	0	0000101	00000000	4	1
0011	0	0000101	11111011	4	5
0001	1	00001010	11111011	3	6
0001	1	00001010	11111011	3	2
0000	1	00010100	11111011	2	6
0000	1	00010100	00001111	2	4
0000	0	00101000	00001111	1	6
0000	0	00101000	00001111	1	2
0000	0	01010000	00001111	0	6
0000	0	01010000	00001111	0	7

### 3.4.3 Division

The division algorithm discussed in Section 3.3 is illustrated by Figure 3-2. The algorithm can be divided into four parts:

- The division cycles
- The first overflow check
- The remainder correction
- The second overflow check.

3.4.3.1 **Division Cycle** – Division proceeds through  $N$  cycles, dividing a  $2N$ -bit number by an  $N$ -bit number. In each cycle, the signs of the two operands are compared, and a bit in the quotient is set if the signs are equal. This bit represents the result of the previous cycle of operation and determines whether an addition or subtraction is done in the present cycle. Before the addition or subtraction, the divisor is divided by 2 by shifting right and extending the sign, and the quotient is shifted right to incorporate the result of the last operation.

The divisor is shifted right before the first addition or subtraction. This action corrects for the fact that the dividend can be a number in the range from  $2^{31} - 1$  to  $2^{31}$ , but the divisor can only be in the range from  $2^{15} - 1$  to  $2^{15}$ ; each cycle of the division adds or subtracts a multiple of the divisor; for the first cycle, the multiple used is one half the maximum number, or  $2^{15}$ . This fact requires the divisor to be one place to the right compared to the dividend, because if the divisor is in the left half of the register it is followed by 16 zeros, effectively multiplying it by  $2^{16}$ .

The division algorithm requires only a 16-bit adder, because only 16 bits of the divisor are significant. The remaining bits are extended sign or low order 0s, which do not need to be added or subtracted to combine them with the dividend. In each cycle, the adder/subtractor is used on a different set of bits, one place to the right from the previous operation; in effect, the adder is shifted one place to the right each cycle.

3.4.3.2 **Overflow Detection** – If the dividend is greater than the divisor multiplied by the maximum expressible number, the result of the division is also greater than the maximum expressible number. This condition is called overflow. The KE11-A makes two checks for overflow during a division; one check during the first cycle and one after the remainder correction.



Overflow can take one of two forms. If the correct result is between  $2^n$  and  $2^{n-1}$  ( $2$ 's complement numbers can only express numbers up to  $2^{n-1}$ ), the overflow creates a number with the incorrect sign, but expressible in  $2^{n+1}$  bits. This is readily detectable after the division. If the correct result is greater than  $2^n$ , however, the  $N$  least significant bits may have any values and the number may appear to have the correct sign when the division is completed.

To detect such results, the KE11-A uses a test after the first cycle of the division. If the dividend is larger than  $2^n$  times the divisor, the sign of the remainder after the first cycle is incorrect. Therefore, if the sign of the dividend does not change after the first cycle, an overflow condition is present.

The first overflow check compares the dividend to the divisor times  $2^{n-1}$ , but the maximum expressible number is  $2^{n-1}-1$ . The second overflow check is provided for the remaining possible cases of overflow.

**3.4.3.3 Remainder Correction** – The KE11-A requires that the remainder have the same sign as the original dividend. A remainder of 0 is acceptable for either sign. After  $N$  cycles of operation, the divisor has been shifted  $N$  times, but the quotient has only been shifted  $N-1$  times (plus one shift before the generation of any quotient bits). If the final cycle of the division resulted in a remainder of the correct sign, the last quotient bit is a 1, and a single shift cycle is required to complete the quotient.

If the final division cycle produced a remainder with the wrong sign, or one that can be reduced to 0 by one more operation, the final quotient bit is a 0, according to the criteria in Section 3.3. However, the rest of the quotient may be incorrect and must be corrected by adding the bit resulting from a final addition or subtraction operation to the quotient before the final 0 is shifted.

**3.4.3.4 Division Register Structure** – The structure of the registers used in the division algorithm reveals the following facts:

- a. The divisor register never contains more than  $N$  significant bits; all others are either extended sign or low order 0s.
- b. The number of significant bits in the dividend decreases by at least one each cycle.
- c. The number of bits in the quotient increases by one each cycle.

The flow chart in Figure 3-2 is accompanied by an algorithm that describes the operations done during a division. An example is presented that illustrates the contents of the registers after selected steps. The number in the *Step* column of the example refers to the number of a step in the accompanying sequence.

**3.4.3.5 Division Algorithm** – The sequence of operations is as follows:

Step	Action
1	Load the dividend, and load the divisor into the upper half of the divisor register. Set the SC to $N$ , and set SIGN to the value of the sign of the dividend.
2	If the sign of the dividend and the sign of the divisor are equal, go to Step 4.
3	Set $C$ to a 0. Go to Step 5.
4	Set $C$ to a 1.
5	Divide the divisor by two by shifting right and extending the sign. Shift the quotient left, shifting $C$ into bit 0.
6	If bit 0 of the quotient is a 1, go to Step 8.

(continued on next page)

Step	Action
7	Add the divisor to the dividend. Go to Step 9.
8	Subtract the divisor from the dividend.
9	If this is the first cycle and the dividend is <i>not</i> 0 and the dividend has changed sign, overflow has occurred and the division stops.
10	Decrement the SC. If the SC is not 0, go to Step 2.
11	If the dividend is 0, go to Step 14.
12	If the divisor can be combined with the dividend to result in 0, go to Step 15.
13	If the sign of the remainder (dividend) is not the same as the sign of the original dividend, go to Step 15.
14	Set $C$ to 1, and shift $Q$ left, shifting $C$ into bit 0. Go to Step 19.
15	If the signs of the dividend and the divisor are the same, go to Step 17.
16	Add the divisor to the dividend and set $C$ to 1. Go to Step 18.
17	Subtract the divisor from the dividend and set $C$ to 0.
18	Add $C$ to $Q$ , then clear $C$ and shift $Q$ to the left.
19	If the signs of the original dividend and the divisor are the same and the quotient is positive, or the signs are different and the quotient is negative, the result is correct. Otherwise overflow has occurred.

**3.4.3.6 Division Example** – The following lists the contents of the indicated register for the example:

17 ( $21_8$ ) divided by 5 yields a quotient of 3 and a remainder of 2.

DD	DR	Q	C	SC	Step
00010001	01010000	0000	0	4	1
00010001	00101000	0001	1	4	5
11101001	00101000	0001	1	3	10
11101001	00010100	0010	0	3	5
11111101	00010100	0010	0	2	10
11111101	00001010	0100	0	2	5
00000111	00001010	0100	0	1	10
00000111	00000101	1001	1	1	5
00000010	00000101	1001	1	0	10
00000010	00000101	0011	1	0	14
00000010	00000101	0011	1	0	19

#### 3.4.4 Shift Operations

Figures 3-3 through 3-5 illustrate the three types of shift operations done by the KE11-A. Each figure is self-explanatory. Refer to the appropriate sections in Chapter 2 for a description of the properties of each shift.

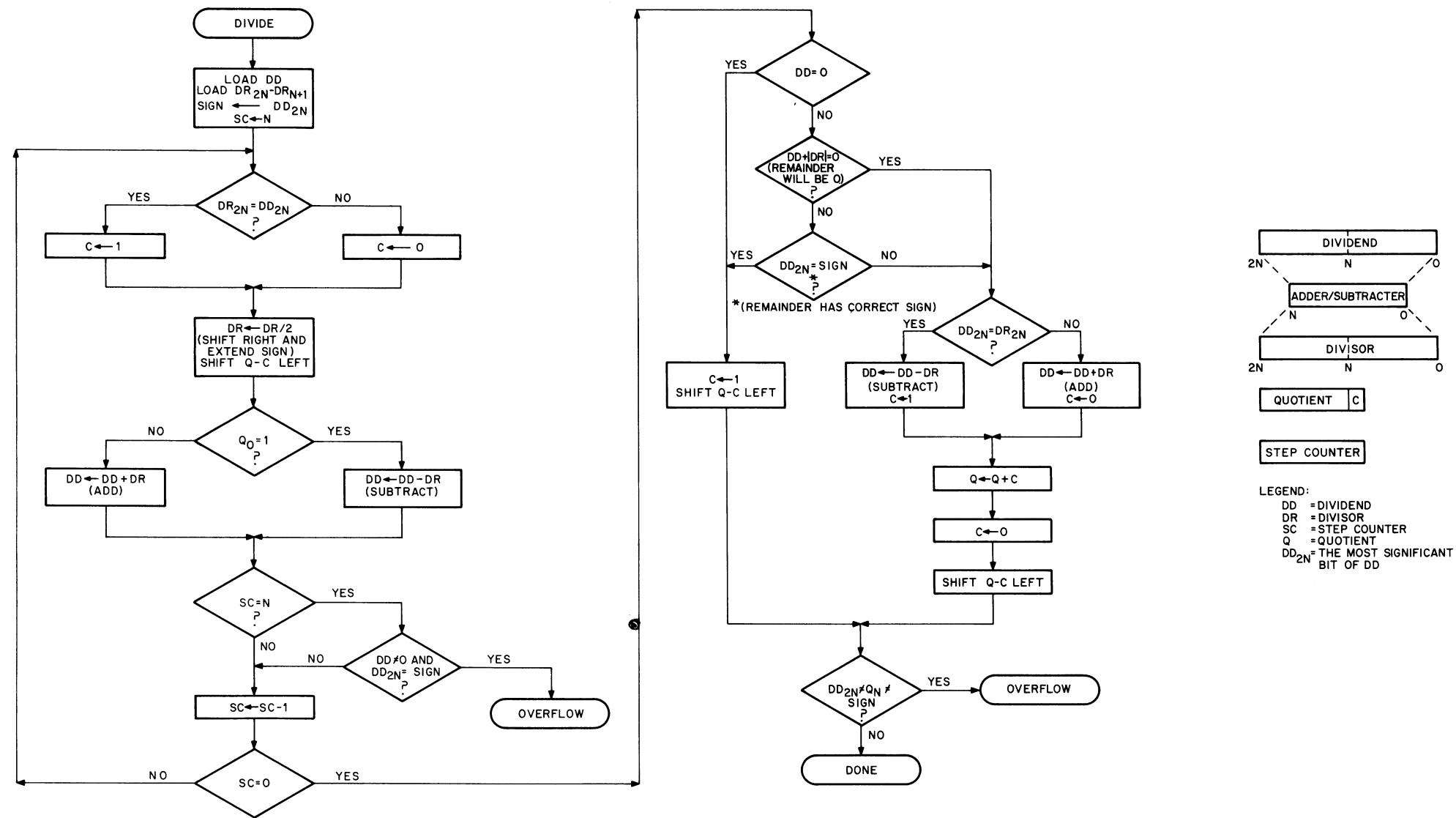


Figure 3-2 Divide Algorithm Flow Chart

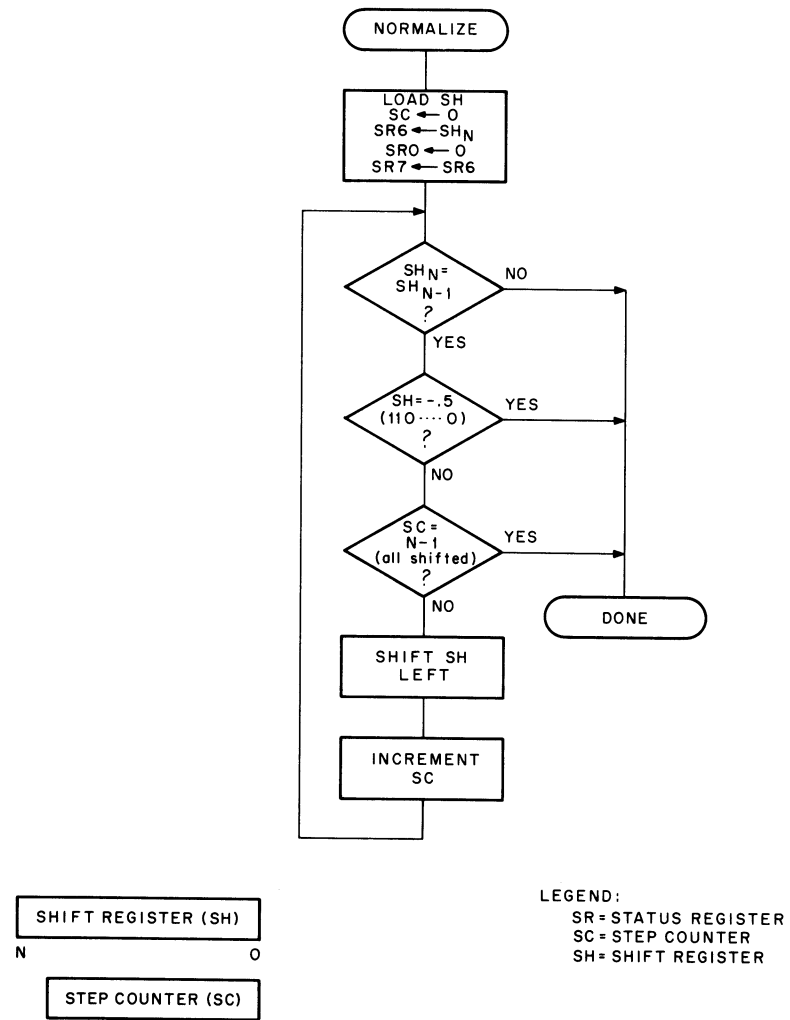


Figure 3-3 Normalize Algorithm Flow Chart

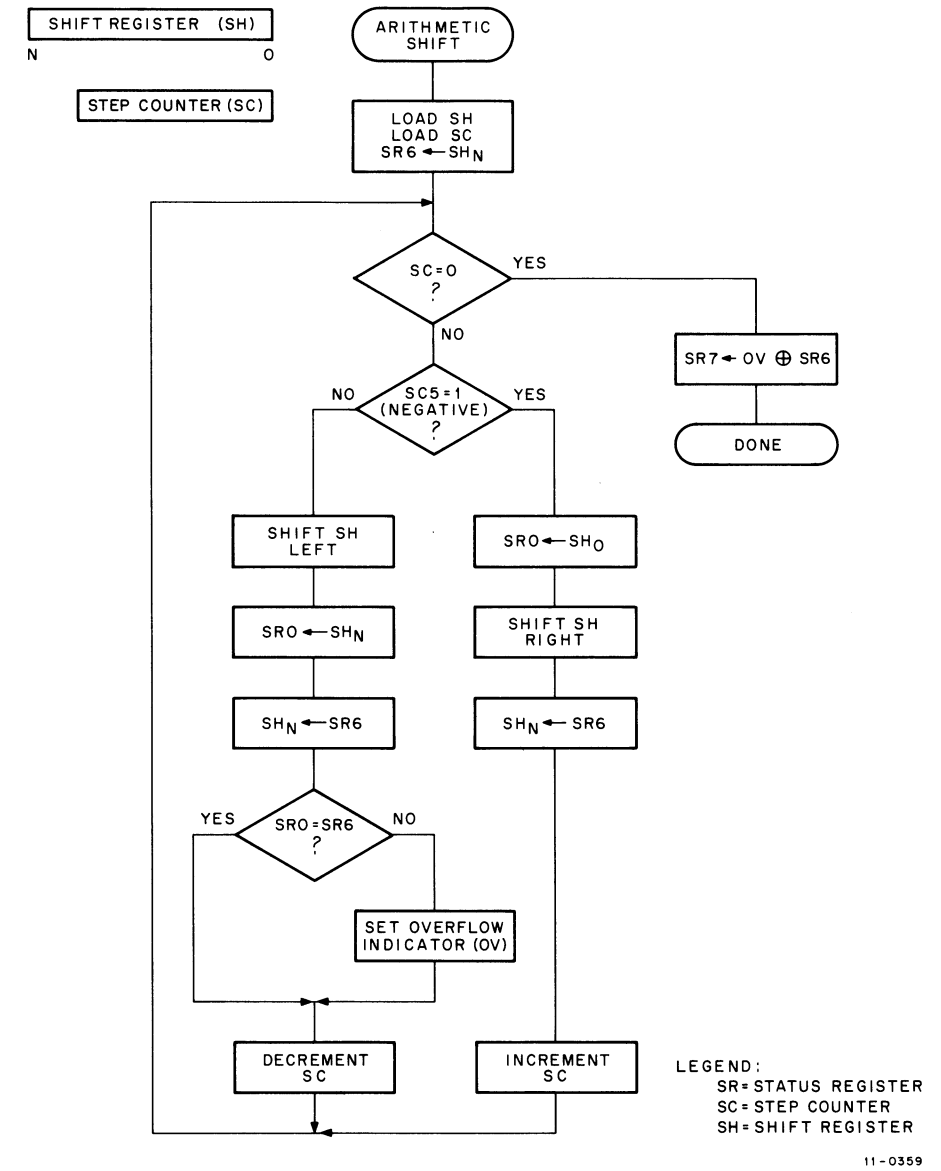
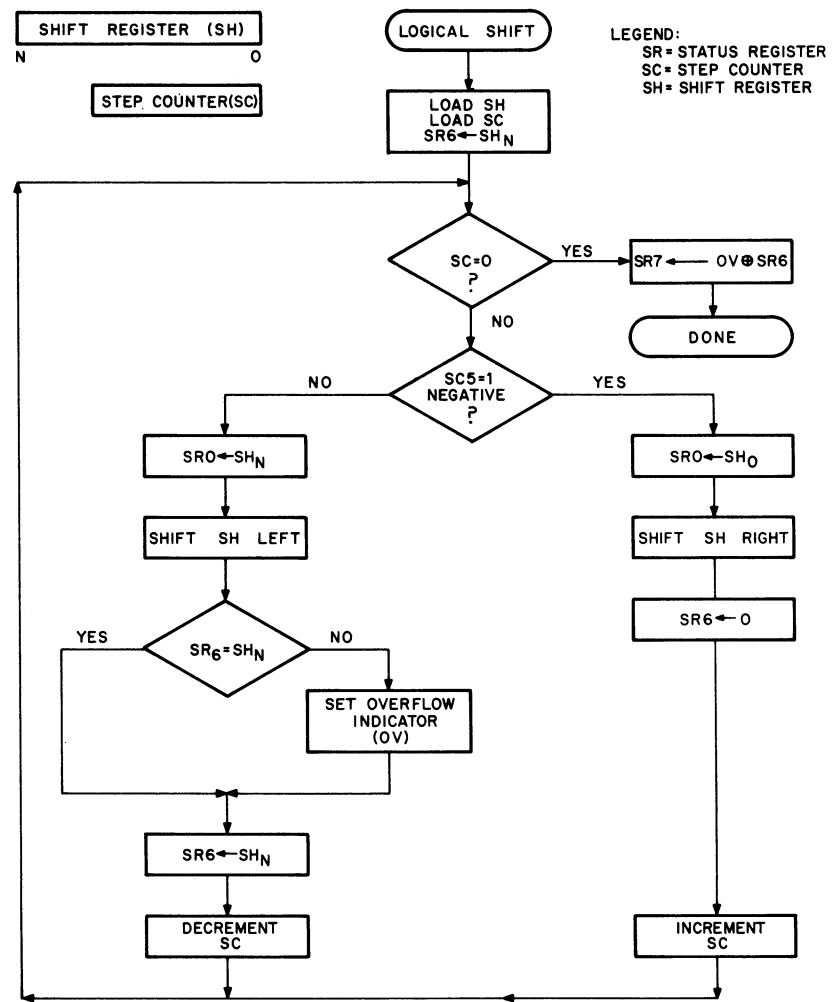


Figure 3-4 Arithmetic Shift Algorithm Flow Chart



11-0360

Figure 3-5 Logical Shift Algorithm Flow Chart

## CHAPTER 4

### GENERAL HARDWARE DESCRIPTION

This chapter explains the implementation of KE11-A operations on a functional level. The algorithms used in the implementation are presented in Chapter 3, and the detailed logic description is included with the KE11-A print set in Chapter 5. This chapter describes the KE11-A logic on a block diagram level (see Dwg. D-BD-KE11-A-BD). The operations that use this logic structure are presented in two forms:

- a. Flow charts of the data operations are given for the multiply and divide operations.
- b. Waveforms of the control signals within the KE11-A are shown for all five KE11-A operations.

See Appendix C for the names of terms in mathematical operations.

#### 4.1 BLOCK DIAGRAM DISCUSSION

The block diagram on Drawing D-BD-KE11-A-BD shows the KE11-A to be divided primarily into five registers (three for data and two for control); the data manipulation and gating associated with the three data registers; and two major control blocks. The Clock and States Control decodes Unibus addresses to determine the required operation and selects the necessary internal operations. The Register Control provides control signals to manipulate the data.

The KE11-A functionally comprises five modules:

- a. Register Control and Clock and States are each a separate module.
- b. The registers, data manipulation logic, and data paths are on two identical modules, which each provide all the logic for 8 bits of each 16-bit register. Communication between the modules is provided for carries in the adder, shifts in the registers, and common control signals.
- c. A third control module is used for gating of data between the KE11-A and the Unibus. This module also contains the Step Counter and Status Registers.

In addition to the AC, MQ, and X registers, the data paths modules include an adder and load and shift gating. These data paths are shown on Figure 4-1, a *bit-slice* block diagram that shows all logic and control signals for one bit of each register and the corresponding data paths. All inputs to the AC register are through the adder; for loading or shifting, the second adder input is disabled and data passes through the adder unmodified. The MQ has separate shift and load gates but can be loaded from the adder for the remainder correction step in a division operation (this step asserts the EXTRA signal). The X register is static; it is loaded directly from the bus receivers. The contents of the X register can be gated to the adder in either a true or complemented form; the adder has a carry input to the least significant bit, which can be used with the complemented input to form the 2's complement negation of the X data.

All connections for carries and shifting are supplied by the input gating to each register. For example, during a left shift, each AC bit is loaded from the next less significant bit. AC bit 0 is loaded from MQ bit 15. During a right shift, MQ bit 15 is loaded from AC bit 0.

#### 4.2 OPERATION CYCLES

This section details the five operations that the KE11-A performs as a sequence of data transformations. The coverage is on a register transfer level; data is moved from a register, through the data paths, to another (or the same register). Data can be modified by addition or shifting during the move. Data transferred from the X register can be negated before the addition. Each discussion refers to the KE11-A waveforms on Dwg. D-TD-KE11-A-WF. In addition, the multiply and divide operations are illustrated by flow charts in Figures 4-2 and 4-3 respectively.

##### 4.2.1 Multiplication

Figure 4-2 is a flow chart of the multiplication procedure used in the KE11-A. Compare this flow chart to Figure 3-1, which illustrates the algorithm that is implemented in the KE11-A.

The implementation is simplified by taking advantage of several features of the algorithm (refer to Section 3.4.2) to reduce the size of the registers needed. In the KE11-A, each register is 16 bits in length, but the multiplication produces a 32-bit product. The multiplicand, which never requires more than 16 significant bits, is held in a static register while the multiplier and product are shifted. This feature permits the use of a static adder/subtractor. The multiplier and the product trade bits, as the product increases by one significant bit and the multiplier decreases by one bit for each cycle of operation.

The sign of the product is extended each time it is shifted right. This sign extension must be modified if the multiplicand is the maximum negative number ( $-2^{15}$ ). When this number is subtracted from a zero partial product, the wrong result is produced, because the negation of the maximum negative number does not produce a positive number (the negative of the maximum negative number is that number itself). That is, the result of subtracting  $-2^{15}$  and then shifting should be  $2^{14}$ , not  $-(2^{15}+2^{14})$ . In this instance, the sign is not extended, and a 0 is shifted into AC15.

Drawing D-TD-KE11-A-WF (Sheet 1) illustrates the waveforms of the KE11-A control signals that effect programmable KE11-A operations in terms of the basic data operations (addition, subtraction, and shifting). The basic timing of the multiply operation is provided by the CLK AC and CLK MQ signals. These signals are driven by a basic timing signal, REG CLK 1, that provides pulses at a 250 ns period (1/4 the basic clock rate).



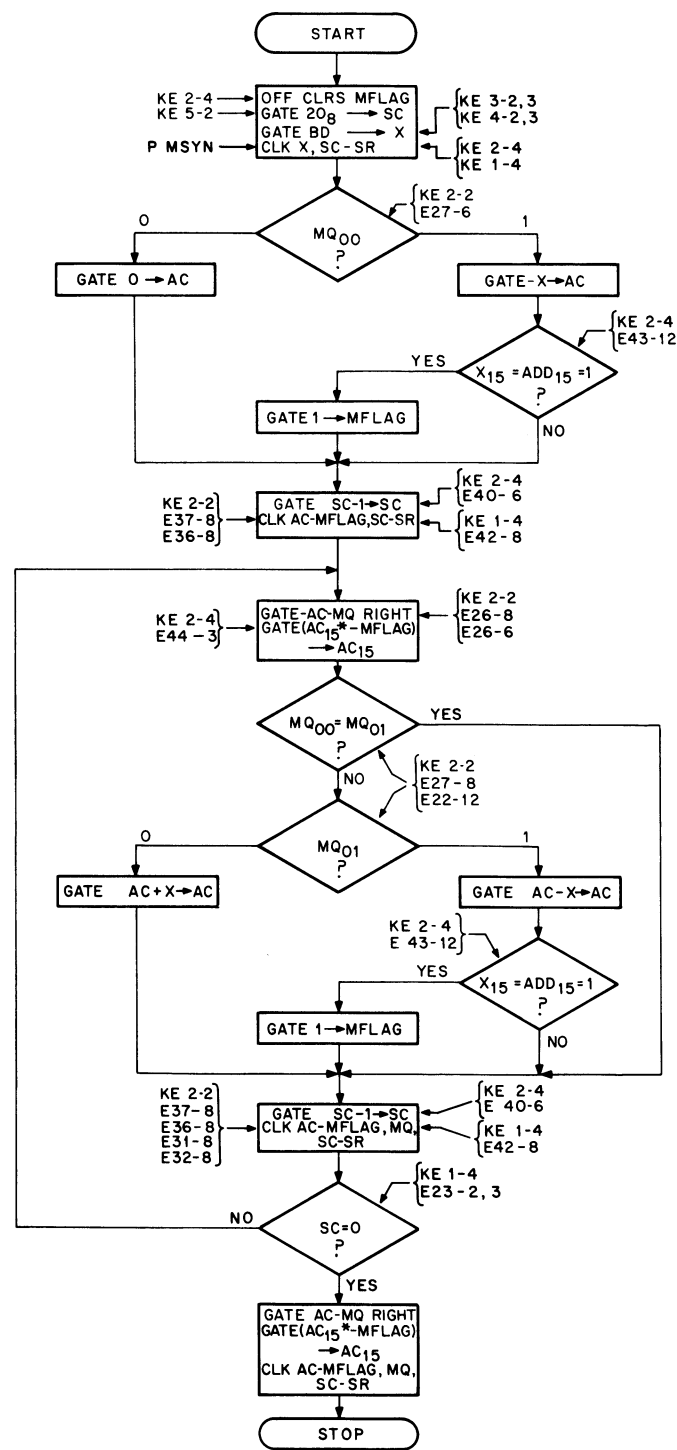


Figure 4-2 Multiply Implementation Flow Chart

- | Step | Action   |
|------|--|
| 2    | If $MQ_{00} = 1$ , gate the 2's complement of $X$ to the AC. Also, gate $a 1$ to MFLAG if $X_{15}$ and $ADDER_{15}$ are both a one. If $MQ_{00} = 0$ , gate zeros to the AC. Gate SC minus 1 to the SC. Clock the AC-MFLAG and SC-SR.              |
| 3    | Gate the AC and the MQ right once. Gate $AC_{15}$ to itself (replication) if MFLAG is clear, or gate $a 0$ to $AC_{15}$ if MFLAG is set.   |
| 4    | If $MQ_{00} = MQ_{01}$ , go to Step 6. (Note: <i>only</i> clocking, <i>not</i> gating, changes the contents of a register. Therefore, the MQ has not changed since entering Step 3).   |
| 5    | If $MQ_{01} = 1$ , subtract $X$ from the AC "gated right" and gate the result to the AC. Also, gate $a 1$ to M FLAG if $X_{15}$ and $ADDER_{15}$ are both a one. If $MQ_{01} = 0$ , add $X$ to the AC "gated right" and gate the result to the AC. |
| 6    | Gate SC minus 1 to the SC. Clock the AC-MFLAG, MQ and SC-SR.   |
| 7    | If $SC \neq 0$ , go to Step 3. Pass $n-1$ times through Step 7 for $X = n$ bits.   |
| 8    | Gate the AC and the MQ right once. Gate $AC_{15}$ to itself (replication) if MFLAG is clear, or gate $a 0$ to $AC_{15}$ if MFLAG is set.   |
| 9    | Clock the AC-MFLAG, MQ and SC-SR. The AC-MQ = $2n$ bit product.  |

4.2.1.2 Multiplication Example – The following lists the contents of the indicated register for the example:

5 times 3 equals  $15_{10}$  ( $17_8$ )

X	AC	MQ	AC-R	MQ-R	MFLAG	SC	Step
?	?	0101	–	–	?	?	MQ loaded
0011	?	0101	–	–	0	4	1
	1101	0101	1110	1010	0	3	2
	0001	1010	0000	1101	0	2	6
	1101	1101	1110	1110	0	1	6
	0001	1110	0000	1111	0	0	6
	0000	1111	–	–	0	-1	9

#### 4.2.2 Division

Figure 4-3 is a flow chart of the division implementation in the KE11-A. Compare this figure with Figure 3-2, which illustrates the corresponding algorithm. The division operation implements most of the steps of the algorithm directly, but the registers are generally single length (16 bits), the adder is static (always operates on the same 16 bits), and 32-bit operands are accommodated by using two 16-bit registers.

The data paths require that shifting occur before addition. This shift is used in the first cycle of the division to shift the divisor so that it is multiplied by  $2^{15}$ , rather than  $2^{16}$ . An extra quotient bit is inserted in the first cycle before the first addition or subtraction; this bit has no significance, because it is shifted out of the MQ by the last shift performed. (The last shift cycle does not shift the AC; thus, the bit does not affect the remainder.)

The KE11-A proceeds through the cycles of the division 16 times, checking for overflow after the first cycle. The logic then examines the remainder to determine which of two alternative sequences should be done:

- a. If no remainder correction is required, only one more cycle is required; thus, the ODD termination sequence is executed.
- b. If remainder correction is required, the EVEN termination sequence provides the final cycles. The requirements of an extra cycle for even quotients, and the reason that a remainder correction is not required for odd quotients, is discussed in Section 3.3.

The waveforms on Drawing D-TD-KE11-A-WF (Sheet 2) illustrate the basic division cycle, the ODD and EVEN terminations, and the case in which overflow is detected during the first cycle. The CLK AC and CLK MQ signals are based on the signal REG CLK1, which operates at a 250 ns period (1/4 the basic clock frequency).

The flow chart in Figure 4-3 is accompanied by a numbered list of steps that describe operations done during a division. Several examples are also presented to clarify the division process. In the examples, one row illustrates the contents of the registers at the end of a selected step of the division. The numbers in the *Step* column refer to the numbered steps in the list accompanying the flow chart. Conventions are described in Section 4.2.1.

#### 4.2.2.1 Divide Implementation – The divide implementation is as follows:

Step	Action
1	a) The AC, MQ and sign of the dividend (SDIVD) were loaded by previous instructions. b) Gate 20 <sub>8</sub> to the SC and gate the Bus Data (divisor) lines to the X register.
2	Gate 1 to SR <sub>00</sub> if Bus Data bit 15 is the same as AC <sub>15</sub> . Gate 0 to SR <sub>00</sub> if Bus Data bit 15 is different from AC <sub>15</sub> . Clock X and SC - SR.
3	Gate the AC and the MQ left once, gating SR <sub>00</sub> to MQ <sub>00</sub> .
4	If SR <sub>00</sub> = 1, subtract X from the AC and gate the result to the AC. If SR <sub>00</sub> = 0, add X to the AC and gate the result to the AC.
5	Gate 1 to SR <sub>00</sub> if the sign of X is the same as ADDER <sub>15</sub> . Gate 0 to SR <sub>00</sub> if the sign of X is different from ADDER <sub>15</sub> .
6	If this is not the first time through this step, go to Step 8.
7	If the adders are not 0 and the carry out of the adder is different from SDIVD, generate DIVD DONE and enable the SR to indicate OVERFLOW.
8	Gate SC minus 1 to the SC. Clock the AC, MQ, and SC - SR. If overflow occurred in Step 7, this is the last step of the invalid divide, and the contents of the MQ and AC are meaningless.
9	If SC ≠ 0, go to Step 3. Pass <i>n</i> times through Step 9 for X = <i>n</i> bits.
10	If SR <sub>00</sub> = 1, subtract X from the AC and gate the result to the AC; if SR <sub>00</sub> = 0, add X to the AC and gate the result to the AC.
11	If the AC is zero, go to Step 15.
12	If the adder is zero, go to Step 14.
13	If the sign of the AC is the same as SDIVD, go to Step 15.
14	a) Gate SC minus 1 to the SC, gate SR <sub>00</sub> to SR <sub>00</sub> . Gate the MQ left once, gating 0 to MQ <sub>00</sub> . Clock the AC, MQ, and SC - SR.

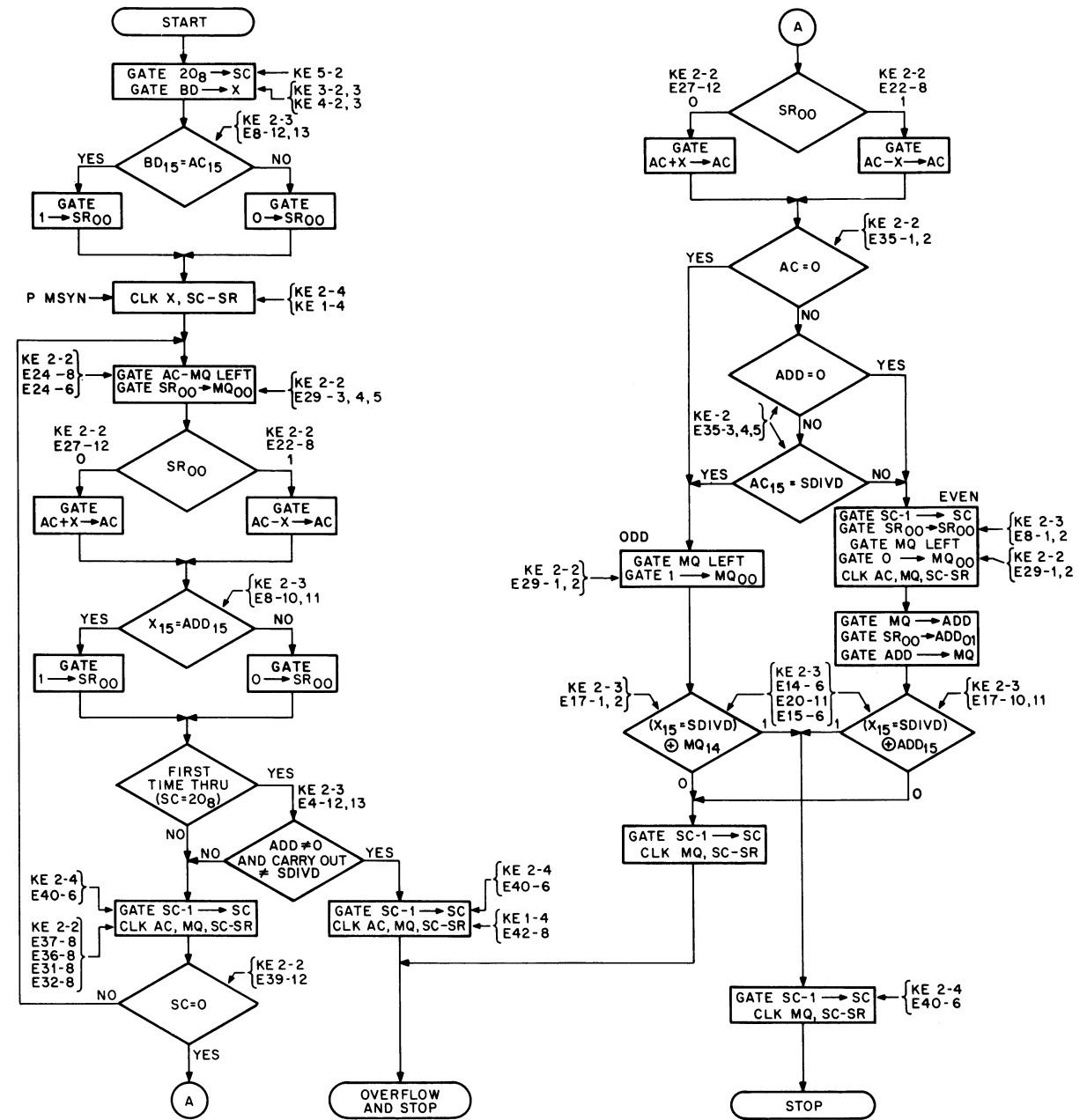


Figure 4-3 Divide Implementation Flow Chart



Step	Action
14 (cont)	b) Gate the MQ to the ADDER, and also gate SR <sub>00</sub> to ADDER <sub>01</sub> . Gate the ADDER to the MQ. Go to Step 16.
15	Gate the MQ left once, gating a 1 to MQ <sub>00</sub> .
16	Generate DIVD DONE and enable the SR to indicate OVERFLOW if: a) the sign of X and SDIVD are the same and the anticipated sign of the final MQ is 1, OR b) the sign of the X and SDIVD are different and the anticipated sign of the final MQ is 0.  Otherwise, the divide was valid.
17	Gate SC minus 1 to the SC. Clock the MQ and SC - SR. If the divide was valid, the MQ = quotient and AC = remainder. Otherwise, the contents of the MQ and AC are meaningless.

4.2.2.2 Division Examples – The following lists the contents of the indicated register for each example:

Example 1

17 (21<sub>8</sub>) divided by 5 yields a quotient of 3 and a remainder of 2.

X	AC	MQ	Carry Out	AC-L	MQ-L	SDIVD	SR <sub>00</sub>	SC	Step
?	0001	0001	–	–	–	0	?	?	AC-MQ loaded
0101	0001	0001	0	0010	0011	↓	1	4	2
	1101	0011	–	1010	0110		0	3	8
	1111	0110	–	1110	1100		0	2	8
	0011	1100	–	0111	1001		1	1	8
	0010	1001	–	–	0011		1	0	8
	0010	0011	–	–	–	↓	0	-1	17

Example 2

35 (43<sub>8</sub>) divided by 4 yields a quotient of 8 (10<sub>8</sub>) and a remainder of 3. The quotient is too large; overflow occurs.

X	AC	MQ	Carry Out	AC-L	MQ-L	SDIVD	SR <sub>00</sub>	SC	Step
?	0010	0011	–	–	–	0	?	?	AC-MQ loaded
0100	0010	0011	–	0100	0111	↓	1	4	2
	0000	0111	–	0000	1111		1	3	8
	1100	1111	–	1001	1110		0	2	8
	1101	1110	–	1011	1100		0	1	8
	1111	1100	–	–	1000		0	0	8
	0011	1000	–	–	–		0	-1	14
	0011	1000	–	–	–	↓	0	-2	17, overflow

Example 3

63 (77<sub>8</sub>) dividend by 3 yields a quotient of 21 or overflow.

X	AC	MQ	Carry Out	AC-L	MQ-L	SDIVD	SR <sub>00</sub>	SC	Step
?	0011	1111	–	–	–	0	?	?	AC-MQ loaded
0011	0011	1111	1	0111	1111	↓	1	4	2
	0100	1111	–	–	–		0	3	8, overflow

4.2.3 Shifts

Shift operations do not require carry propagation in the adders; shift cycles are run from the signal CLK 2-1, which operates at a 125 ns period (1/2 basic clock frequency of the KE11-A). The three types of shift operations are illustrated by one set of waveforms on Dwg. D-TD-KE11-A-WF. The basic shift cycle requires the selection of a gating signal to determine a right or left shift, control signals to determine what bits are shifted into vacated positions, and control signals to determine the setting of KE11-A condition codes. Note that for a 0 shift count (or conditions met in a Normalize operation) at the beginning of a shift, the KE11-A does one cycle of operations that does not affect the contents of the AC or MQ registers, because no CLK AC or CLK MQ signals are generated. This cycle is done to set the KE11-A condition codes.



## CHAPTER 5

### DETAILED HARDWARE DESCRIPTION

This chapter presents a description of KE11-A logic keyed to the circuit schematics contained in a companion document, *KE11-A Extended Arithmetic Element Engineering Drawings*. The major circuits on each drawing are described, and the description is related to the function of each of the major signals that originates on the drawing. Also, logic equations are included for most of the complex combination logic that generates control signals in the KE11-A.

Signal names in the KE11-A include an origin designator and a level designator. The origin of a signal is the engineering drawing on which the signal is generated; each print has been assigned a number of the form of KEx-y, where  $x$  is the number of a module drawing set and  $y$  is the number of the drawing within the drawing set. These numbers are arbitrarily assigned and have no meaning except within the print set. The origin designator precedes the signal name. For example, KE1-4 STOP H is a signal originating on the fourth sheet of the circuit schematic for the first KE11-A module.

The level designator indicates the assertion level of a signal is either high or low, represented by an  $H$  or an  $L$  following the signal name. Some signals refer to high or low bytes; in such cases, the signal name has a two-or three-letter word indicating the byte and a single letter indicating the level. (e.g., CLK AC LO H is a signal that affects the low byte of the AC, but is asserted when high.)

In the logic equations, signal names are reduced to the simplest terms by eliminating the origin designator and the assertion level. Some signals are used as inhibiting levels; the assertion level disqualifies a gate, rather than qualifying it. This function is shown in the logic equations by a NOT symbol (a minus sign is used).

General information concerning circuits that use discrete components for timing (pulsers, delays, and clocks) and complex integrated circuits is included in the *PDP-11 Conventions Manual*, DEC-11-HR6B-D.

## **KE1-1**

### **Clock and States, M827**

The M827 Clock and States Module contains the timing circuits for the KE11-A and the address recognition logic and control signal logic for communication with the Unibus. Address decoding determines when the KE11-A is addressed and the particular operation (if any) implicitly selected. The state of the KE11-A (the operation in progress) is determined by flip-flops on this module.

The transistor clock, which provides the basic timing signals for the KE11-A and the circuits that control the starting and the stopping of the clock are on this module.

This drawing illustrates the address decoder and the operation flip-flops.

**KE1-2 EAE ADRS H** means that the Bus A lines are set to one of the 16 Unibus addresses (8 word-locations) that specify KE11-A addresses. The jumper at print location C7 allows the EAE addresses to be either 777300 through 777317 or 777320 through 777337 (normally the former set of address is used).

**KE1-2 OP H** means that the address implies one of the five EAE operations. Note that it is not gated with EAE ADRS but is only a function of Bus A(3:1).

The following convention applies to the signal names on this print:

- a. Register addresses are *two-character* designations
- b. Addresses that specify operations are *three-character* designations
- c. Outputs from the operation flip-flops are *four-character* designations.

The two-letter and three-letter signal names are address decoder outputs, which are present only when the EAE is addressed by the Bus A lines. The four-letter signal names are outputs from the operation flip-flops, which are present for an entire operation.

The Step Counter (SC) address is not separately decoded because the SC can only be explicitly addressed as the low byte of a word that includes the Status Register (SR) as the high byte. The SC is accessed by the signal SR.

**OP =** -A03\* -A01\* -A02  
 +A02\* -A01\* -A03  
 +A01\* -A02\* -A03

**SHIFTS =** ASHF + LSHF + NORM

### KE1-3

This drawing illustrates the SSYN generator and the clock with the associated turn-on, sync, and frequency divide circuits.

**KE1-3 SELECT H** is set when MSYN is asserted and the KE11-A is addressed, provided that the KE11-A is in maintenance mode or no operation is in process. If a Unibus operation addresses the KE11-A while an operation is in process, the SELECT flip-flop does not set until the operation is done; the Unibus transfer is delayed until that time. Bus SSYN is generated 100 ns after SELECT is set to allow time for the data to be gated to the Unibus if the cycle is a DATI.

**KE1-3 CLK GATE** blocks the first clock cycle, because the circuit that generates the 62.5 ns square wave produces one short cycle before stabilizing. (Clock and timing waveforms are illustrated on Drawing D-TD-KE11-A-WF.)

**KE1-3 CLK2-0** and **KE1-3 CLK2-1** are two square waves with 125 ns periods; the signals differ in phase. CLK2-1 changes state on the high-to-low transition of the clock (at the output of gate E44 at print location B/C5); CLK2-1 changes on the low-to-high transition. The clock rate must be adjusted so that the output at pin D4H02 is a train of pulses of 125 ns period, as shown in Figure 5-1.

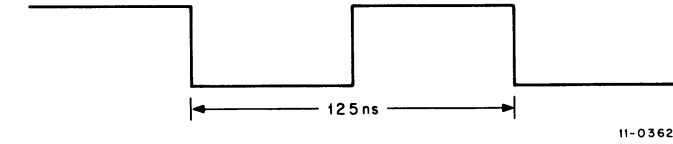


Figure 5-1 Clock Rate Adjustment

**KE1-3 CLK1 H** is a 250 ns period clock that changes state on every low-to-high transition of CLK2-0 H. CLK1 is used for the multiplication and division sequences, which require additional time for carry propagation in the adders.

**KE1-3 REG CLK2 H** provides a pulse at the end of each CLK2-1 cycle that is used to strobe data into the registers of the KE11-A during shift operation sequences.

**KE1-3 REG CLK1 H** provides a pulse at the end of each CLK1 cycle that is used to strobe data into the registers during multiply and divide operation sequences.

**SHIFT ENBL = RUN \* SHIFTS \* – DONE**

See description of waveforms in Chapter 4 for rest of sheet KE1-3.

The circuits shown on this drawing turn off the clock, decode the bus control lines, and detect a zero output from the adder.

**KE1-4 DONE L** indicates the end of the current operation.

**KE1-4 TP1** is a test point for module checking; this signal is not used by the KE11-A logic.

**KE1-4 END H** stops further shifting by blocking transmission of the clock signals to the shift control circuit.

**KE1-4 STOP H** controls the clock turnoff and the operation flip-flops.

**KE1-4 OFF L** turns off the clock at the end of the last phase of the current cycle and resets all clock flip-flops.

**KE1-4 OFF1 L** and **KE1-4 OFF2 L** are buffered signals that reset the operation flip-flops on KE1-2.

**KE1-4 CLK SC H** strobes data into the Step Counter (SC) and into three Status Register (SR) bits. This signal is used to load the SC both at the beginning of a sequence of operations and during each cycle.

**KE1-4 ADD=0 L** provides a signal used to detect remainders of 0 during division. This circuit is placed on this module because of space limitations, not functional affinity.

**KE1-4 OUT H**, **KE1-4 IN H**, **KE1-4 OUT LO H**, **KE1-4 OUT HI H**, and **KE1-4 OUT HI L** are decoded from the Bus C and Bus A00 lines to determine the character of a data transfer that addresses the KE11-A.

<p> <b>DONE</b> = NORM * (AC15 * - AC14 + AC14 * - AC15 + (SC=37) + (AC= - .5))            + (ASHF + LSHF) * (SC = 0)            + MULT * (SC = 0)            + DIVD * (ODD + DIVD DONE)  <b>CLK SC</b> = P MSYN↑ * (SR * OUT LO * OUT HI + OP * OUT LO)            + CLK1↑ * (MULT + DIVD)            + CLK2↑ * (SHIFTS * - END)         </p>
--

**KE2-1**

**Register Control, M7211**

This module comprises the circuits that control the data paths and the registers that hold the operands and results of operations.



The circuits shown on this drawing generate some of the data path control signals.

**KE2-2 MQ00 IN H** controls the value that is shifted into MQ bit 00.

**KE2-2 GATE X H** enables the direct transfer of data from the X register to the adder input.

**KE2-2 GATE -X H** enables the transfer of the complement of the data in the X register to the adder input. When extending the sign of the MQ into the AC, if the sign bit is a 1, both GATE X and GATE -X are produced to generate an AC of all 1s (each bit is ORed with its complement).

**KE2-2 CARRY IN H** is the carry input to the least significant bit of the adder; this signal is generated in conjunction with GATE -X to generate the 2's complement of the number in the X register.

**KE2-2 MQ SHF RT H** enables the data in the MQ register to be shifted one bit to the right.

**KE2-2 AC SHF RT H** enables the data in the AC register to be shifted one place to the right before entering the adders.

**KE2-2 CLK MQ LO H** strobes data into the low byte (8 bits) of the MQ register. This signal, used during loading of the MQ and during operation sequences, is under clock control.

**KE2-2 EXTRA H** and **KE2-2 EXTRA2 H** are buffered signals generated by EXTRA L to control the data paths during the last cycle of a divide operation to increment the contents of the MQ register.

**KE2-2 CLK MQ HI H** clocks the high byte of the MQ (see CLK MQ LO).

**KE2-2 CLK AC LO H** clocks the low byte of the AC (see CLK MQ LO).

**KE2-2 CLK AC HI** clocks the high byte of the AC (see CLK MQ LO).

**KE2-2 ODD H** is generated to represent the last cycle of a division operation that does not require remainder correction. The ODD signal is used to generate the signals that control the data paths during this cycle.

**KE2-2 AC15 SHF LEFT H** enables the left shift input (through the adders) to AC15 during a left shift. This signal is disabled during arithmetic left shifts.

**KE2-2 AC SHF LEFT H** enables the left shift inputs (through the adders) for AC (14:00)

**KE2-2 MQ SHF LFT H** enables the left shift inputs to the MQ.

$$\begin{aligned} \text{ODD} &= (\text{SC} = 0) * ((\text{AC} = 0) + -(\text{ADD} = 0) * -\text{AC15} * -\text{SDIVD} + \text{AC15} * \text{SDIVD}) \\ \text{MQ00 IN} &= \text{DIVD} * (\text{ODD} + -(\text{SC}=0) * \text{SR00}) \\ \text{MULT STEP} &= \text{MULT} * -\text{SC04} * -(\text{SC}=0) \\ \text{ADD X} &= \text{MULT STEP} * -\text{MQ01} * \text{MQ00} \\ &\quad + \text{DIVD} * -\text{EXTRA} * -\text{SR00} \\ \text{SUB X} &= \text{MULT STEP} * \text{MQ01} * -\text{MQ00} \\ &\quad + \text{MULT} * \text{SC04} * \text{MQ00} \\ &\quad + \text{DIVD} * -\text{EXTRA} * \text{SR00} \\ \text{SIGN EXT} &= \text{OUT} * \text{MQ} * \text{D15} \\ \text{CARRY IN} &= \text{SUB X} \\ \text{GATE X} &= \text{ADD X} + \text{SIGN EXT} \\ \text{GATE -X} &= \text{SUB X} + \text{SIGN EXT} \\ \text{MQ SHF RT} &= \text{MULT} + (\text{ASHF} + \text{LSHF}) * \text{SC05} \\ \text{AC SHF RT} &= \text{MULT} * -\text{SC04} + (\text{ASHF} + \text{LSHF}) * \text{SC05} \end{aligned}$$

$$\begin{aligned} \text{CLK MQ LO} &= \text{REG CLK1}\downarrow * (\text{DIVD} + \text{MULT} * -\text{SC04}) \\ &\quad + \text{REG CLK2}\downarrow * \text{SHIFT ENBL} \\ &\quad + \text{P MSYN}\downarrow * \text{MQ} * \text{OUT LO} \\ \text{CLK MQ HI} &= \text{REG CLK1}\downarrow * (\text{DIVD} + \text{MULT} * -\text{SC04}) \\ &\quad + \text{REG CLK2}\downarrow * \text{SHIFT ENBL} \\ &\quad + \text{P MSYN}\downarrow * \text{MQ} * \text{OUT} \\ \text{CLK AC LO} &= \text{REG CLK1}\downarrow * (\text{MULT} + \text{DIVD} * \text{RUN} * -\text{ODD} * -\text{EXTRA}) \\ &\quad + \text{REG CLK2}\downarrow * \text{SHIFT ENBL} \\ &\quad + \text{P MSYN}\downarrow * \text{OUT} * (\text{AC} + \text{MQ}) \\ \text{CLK AC HI} &= \text{REG CLK1}\downarrow * (\text{MULT} + \text{DIVD} * \text{RUN} * -\text{ODD} * -\text{EXTRA}) \\ &\quad + \text{REG CLK2}\downarrow * \text{SHIFT ENBL} \\ &\quad + \text{P MSYN}\downarrow * \text{OUT} * (\text{AC} + \text{MQ}) \\ \text{AC SHF LFT} &= \text{SHIFTS} * -\text{SC05} + \text{DIVD STEP} \\ \text{DIVD STEP} &= \text{DIVD} * -(\text{SC}=0) * -\text{EXTRA} \\ \text{AC15 SHF LFT} &= \text{AC SHF LFT} * -\text{ASHF} \\ \text{MQ SHF LFT} &= \text{SHIFTS} * -\text{SC05} + \text{DIVD} * -\text{EXTRA} \end{aligned}$$

### KE2-3

The circuits shown on this drawing generate some of the status register bits. These flip-flops can be loaded from the Unibus or from the KE11-A logic described in the following paragraphs. The flip-flops are clocked by the CLK SC signal.

**KE2-3 SR07 H** is a status register bit that, in conjunction with SR06, indicates an overflow condition. The output of gate E4 at print location D5 is high if an overflow condition has occurred during the current operation; SR07 is the exclusive OR of this signal with SR06. The overflow detection logic includes a *latch* signal that is tied to pin 10 of gate E4. If an overflow condition occurs during a shift operation, the latch ensures that the signal is present at the end of the operation.

**KE2-3 SR06 H** indicates that the results of an operation are negative. SR06 is loaded with the value of the signal output from gate E18 at drawing location C5.

**KE2-3 SR00** indicates the value of the last bit shifted out during a shift operation; the flip-flop is used during division operations to store a bit that determines whether the next clock cycle is to execute a subtraction or an addition operation.

**KE2-3 DIVD DONE H** is asserted during an *EXTRA* clock cycle or if overflow occurs during a divide.

$\begin{aligned} \text{OVFL} &= \text{DIVD} * \text{-(ADD=0)} * \text{SC04} * \text{-(SC05)} * (\text{CARRY OUT} \neq \text{SDIVD}) \\ &+ (\text{ASHF} + \text{LSHF}) * \text{RUN} * \text{-(SC05)} * \text{CLK2-0} * (\text{AC14} \neq \text{AC15}) \\ &+ \text{DIVD} * \text{ODD} * (\text{SQOUT} \neq (\text{S DIVD} \neq \text{X 15})) \\ &+ \text{EXTRA} * \text{-(ADD=0)} * (\text{SQUOT} \neq (\text{S DIVD} \neq \text{X15})) \\ &+ \text{OVFL} * \text{- STOP} * \text{- DIVD} \\ \text{SQUOT} &= \text{EXTRA} * \text{AC15} + \text{ODD} * \text{MQ15} \\ \text{NEG} &= \text{DIVD} * \text{-OVFL} * \text{SQUOT} \\ &+ \text{DIVD} * \text{OVFL} * \text{SDIVD} \\ &+ (\text{ASHF} + \text{NORM}) * \text{AC15} \\ &+ \text{MULT} * \text{ADD 15} \\ &+ \text{LSHF} * \text{-(SC05)} * \text{AC14} \end{aligned}$	$\begin{aligned} \text{CARRY} &= \text{ASHF} * \text{AC14} * \text{-(SC05)} \\ &+ \text{LSHF} * \text{AC15} * \text{-(SC05)} \\ &+ \text{DIVD} * \text{SR00} * (\text{SC=0}) \\ &+ (\text{ASHF} + \text{LSHF}) * \text{SC05} * \text{MQ00} \\ &+ (\text{MUL} + \text{DIV START}) * (\text{D15=AC15}) \\ &+ \text{DIVD STEP} * (\text{ADD15} = \text{X 15}) \\ \text{SR07} &= \text{RELOAD SR} * \text{D15} \\ &+ \text{-RELOAD SR} * (\text{OVFL} \neq \text{NEG}) \\ \text{SR06} &= \text{RELOAD SR} * \text{D14} \\ &+ \text{-RELOAD SR} * \text{NEG} \\ \text{SR00} &= \text{RELOAD SR} * \text{D08} \\ &+ \text{-RELOAD SR} * \text{CARRY} \\ \text{DIVD DONE} &= \text{EXTRA} + \text{DIVD} * \text{-OVFL} \end{aligned}$
--	--

This drawing illustrates some of the data path control signals.

**KE2-4 AC15 IN H** provides the input to be shifted into AC bit 15 during multiplication and shifts. The **MFLAG** flip-flop provides the correction necessary when the multiplicand is  $2^{-15}$ .

**KE2-4 SDIVD H** is a flip-flop that stores the original sign of the dividend for comparison with the signs of the results to determine the presence of overflow or the need for remainder correction.

**KE2-4 MUL + DIV START L** indicates that a multiply or divide operation is required. The signal is present only until the operation flip-flop is set.

**KE2-4 RELOAD SR L** enables the inputs to SR bits 0, 6, and 7 from the KE11-A logic.

The remaining signals generated on this print are used to gate data into the various registers through various parts of the data paths. The SC can be loaded from the Unibus or with a constant value of  $20_8$  (for multiply or divide operations); during an operation the SC can be incremented or decremented through a separate set of adders.

```

SET MFLAG = CLK AC HI * ADD15 * X15 * GATE - X
AC15 IN = AC15 * - (ASHF + MULT * -MFLAG)
MQ FM D = MQ * OUT * -OPER
AC FM D = AC * OUT * -OPER
BUS TO SC = RELOAD SR + (ASH + LSH) * OUT LO * -OPER
RELOAD SR = SR * OUT LO * OUT HI * -OPER
NUM TO SC = -BUS TO SC * (-NOR + RUN)
SC-1 TO SC = (ASHF + LSFH) * -SC05 + MULT + DIVD
SC+1 TO SC = (ASHF + LSFH) * SC05 + NORM
MUL + DIV START = (MUL + DIV) * -OPER
OPER = SHIFTS + MULT + DIVD
CLK X = (MUL + DIV) * OUT LO * P MSYN↑
AC15 GATE = AC FM ADD * -ASHF + ASHF * -SC05
AC FM ADD = DIVD * (SC=0)

```

## KE3-1

### Register Low Byte, M234

This module and the identical M234 Registers High Byte Module (KE4) make up the three data registers, the main adder, and the data paths that connect these elements. Each module contains one byte of all elements. A *bit slice* diagram of the registers and data paths (see Figure 4-1), which shows all control signals on the register board, is provided in Chapter 4.

Because the Registers Low Byte and Registers High Byte modules are identical, the signal paths for several control signals that are used to provide individual control of the most significant bit in operations on the AC must be present on both modules. For the low byte, these control inputs are connected to the control inputs for the remaining seven bits.

**Register High Byte, M234**

This module is identical to the Register Low Byte Module, KE3. Refer to the description of that module.

**KE5-1**

**Data Control, M7210**

This module contains the bus data receivers and drivers for the KE11-A, as well as the step counter, five bits of the status register, and the associated logic.

This drawing illustrates the circuits that control the loading and modification of the step counter (SC). The SC is a 6-bit register that can be loaded from the Unibus or from an adder network that can add +1 or - 1 to the contents of the SC or load 20 into the SC. Also, several combinational circuits detect various SC states, such as SC all 0s or SC all 1s with a divide operation in progress (EXTRA L). The data gated into the SC is selected by combinational circuits on KE2-4.

**KE5-3**

The combinational logic shown on this drawing detects various states of the contents of the AC and MQ registers.

**KE5-3 AC=-.5 H** detects a condition that terminates a Normalize operation.

**KE5-3 AC=177777 H** is transmitted as SR05.

**KE5-3 AC=0 H** is transmitted as SR04.

**KE5-3 MQ=0 H** is transmitted as SR03.

**KE5-3 SR02** indicates that both the AC and the MQ are all 0s.

**KE5-3 SR01** indicates that the AC is all 1s and MQ15 is a 1 or that the AC is all 0s and MQ15 is a 0; in other words, the result is single precision.



This drawing illustrates the interface between the KE11-A and the Unibus for the low data byte (D (07:00)). Each data line passes through one ungated receiver and can be driven by the output of a multiplexer that selects one of three inputs. The multiplexer allows reading of the AC, the MQ, or the SC; the fourth input is undriven, which permits connection of the drivers to the wired OR bus without disturbance during nontransmitting periods.

**KE5-5**

This drawing illustrates the data interface between the KE11-A and the Unibus for the high byte (D(15:08)). The circuit is similar to drawing KE5-4 with the following changes: the contents of the SR are transmitted by the same control signals that transmit the SC; and the receivers pass through sign extension logic that allows the high byte to be replaced by all 1s or all 0s, depending on the value of D07. Table 5-1 illustrates the output selection for the 74H87 sign extension circuits.

**Table 5-1**  
**Sign Extension Logic**

Inputs		Out	Remarks
B	C		
0	0	-In	never used
0	1	In	OUT HI
1	0	1	-OUT HI * D07
1	1	0	-OUT HI * -D07

## CHAPTER 6 MAINTENANCE

If the KE11-A is not performing in the prescribed manner, run the diagnostic programs listed in Table 1-2 to isolate the malfunction to the following areas:

- a. The KE11-A hardware
- b. Other system hardware
- c. The software.

If a KE11-A hardware fault is indicated, the diagnostic programs often pinpoint the nature and probable causes of the problem.

If the problem cannot be resolved in this manner, or to verify a tentative analysis of a malfunction, several additional procedures are available. The *PDP-11 Conventions Manual* DEC-11-HR6A-D lists a set of maintenance tools and equipments that may be helpful. The list in Appendix A of that manual includes the maintenance module set that contains a W130 Module and a W131 Module. An overlay is available for the KE11-A to indicate the particular signals that are connected to the maintenance modules when they are inserted in slot B02 of the KE11-A system unit.

The maintenance module set slows operation of the KE11-A to operator-discernible speeds. The KE11-A continues to respond correctly to individual Unibus transfers initiated by the processor but does not produce correct results if the processor is running at normal speed. The operator can control the minor clock cycles of the KE11-A which, in turn, control the individual operations of testing, setting, and clearing of flip-flops; addition and subtraction; and shifting. The maintenance module set also provides indicators that display the states of three sets of signals: timing signals; operation indicators; and register contents for the SR and SC registers. Table 6-1 lists the functions of the timing signals that are displayed by the maintenance module set.

**Table 6-1**  
**Timing Signal Functions**

Name	Function
SELECT	Flip-flop set when KE11-A is explicitly addressed
RUN	Flip-flop set when KE11-A operations are implicitly addressed

(continued on next page)

**Table 6-1 (Cont)**  
**Timing Signal Functions**

Name	Function
CLK ON	Controls square wave generator (basic clock)
CLK GATE	Gates square wave to clock flip-flops, ensures clock shutoff in proper state
CLK 2-0	Basic clock divided by two
CLK 2-1	Similar to CLK 2-0 with phase difference
CLK1	Basic clock frequency divided by 4 (1/2 CLK2-0)
END	Controls final step of operations
STOP	Resets all timing and operation flip-flops

The KM11 is a two-module (W130, W131) option to the KE11-A to aid in maintenance. This prewired option is installed by inserting the W130 Module into location B02 and inserting the W131 Module into the W130. Note that the switches and indicators face toward and extend below the console. The bottom cover must be removed with the chassis external to the cabinet.

Labels for the internal machine states lamps are noted on the W131 Overlay. Switches, which provide a manual clock and alter the response to bus transfers, are active when the toggle is up. Normal machine operation requires that all switches be in the off position.

MNT ENBL and MNT CLK provide a manual clock for the KE11-A. MNT ENBL is activated while the EAE is halted. Each toggle of MNT CLK steps the EAE through the smallest EAE clock intervals.

The next highest clock interval, CLK2, is provided by four toggles (2 complete switch cycles) and is indicated by the CLK2-0 and CLK2-1 indicators. Eight toggles are necessary for each CLK1 interval. Normal operation is resumed when MNT CLK and then MNT ENBL are returned to off.



# APPENDIX A

## COMPLEX IC DESCRIPTIONS

This appendix contains logic schematics and truth tables for four types of complex integrated circuits (ICs) used in the KE11-A. The illustrations are supplemented by notes on significant design features.

### A.1 7482 ADDERS

This adder (see Figure A-1) is a two-bit, full-binary adder. Normally the inputs and outputs are high when asserted. The adder has the same truth table (refer to Table A-1) and the same resulting signals, if both the inputs and the outputs are inverted. For example, if A1 is 1, B1 is 0, A2 is 0, B2 is 1, and C0 is 0, then Sum1 is 1, Sum2 is 1, and C2 is 0. Complementing the inputs (A1=0, B1=1, A2=1, B2=0, and C0=1) complements the outputs (Sum1=0, Sum2=0, C2=1).

Table A-1  
Adder Truth Table

INPUT				OUTPUT					
A <sub>1</sub>	B <sub>1</sub>	A <sub>2</sub>	B <sub>2</sub>	WHEN C <sub>0</sub> = 0			WHEN C <sub>0</sub> = 1		
				Σ <sub>1</sub>	Σ <sub>2</sub>	C <sub>2</sub>	Σ <sub>1</sub>	Σ <sub>2</sub>	C <sub>2</sub>
0	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1	0
1	1	0	0	0	1	0	1	1	0
0	0	1	0	0	1	0	1	1	0
1	0	1	0	1	1	0	0	0	1
0	1	1	0	1	1	0	0	0	1
1	1	1	0	0	0	1	1	0	1
0	0	0	1	0	1	0	1	1	0
1	0	0	1	1	1	0	0	0	1
0	1	0	1	1	1	0	0	0	1
1	1	0	1	0	0	1	1	0	1
0	0	1	1	0	0	1	1	0	1
1	0	1	1	1	0	1	0	1	1
0	1	1	1	1	0	1	0	1	1
1	1	1	1	0	1	1	1	1	1

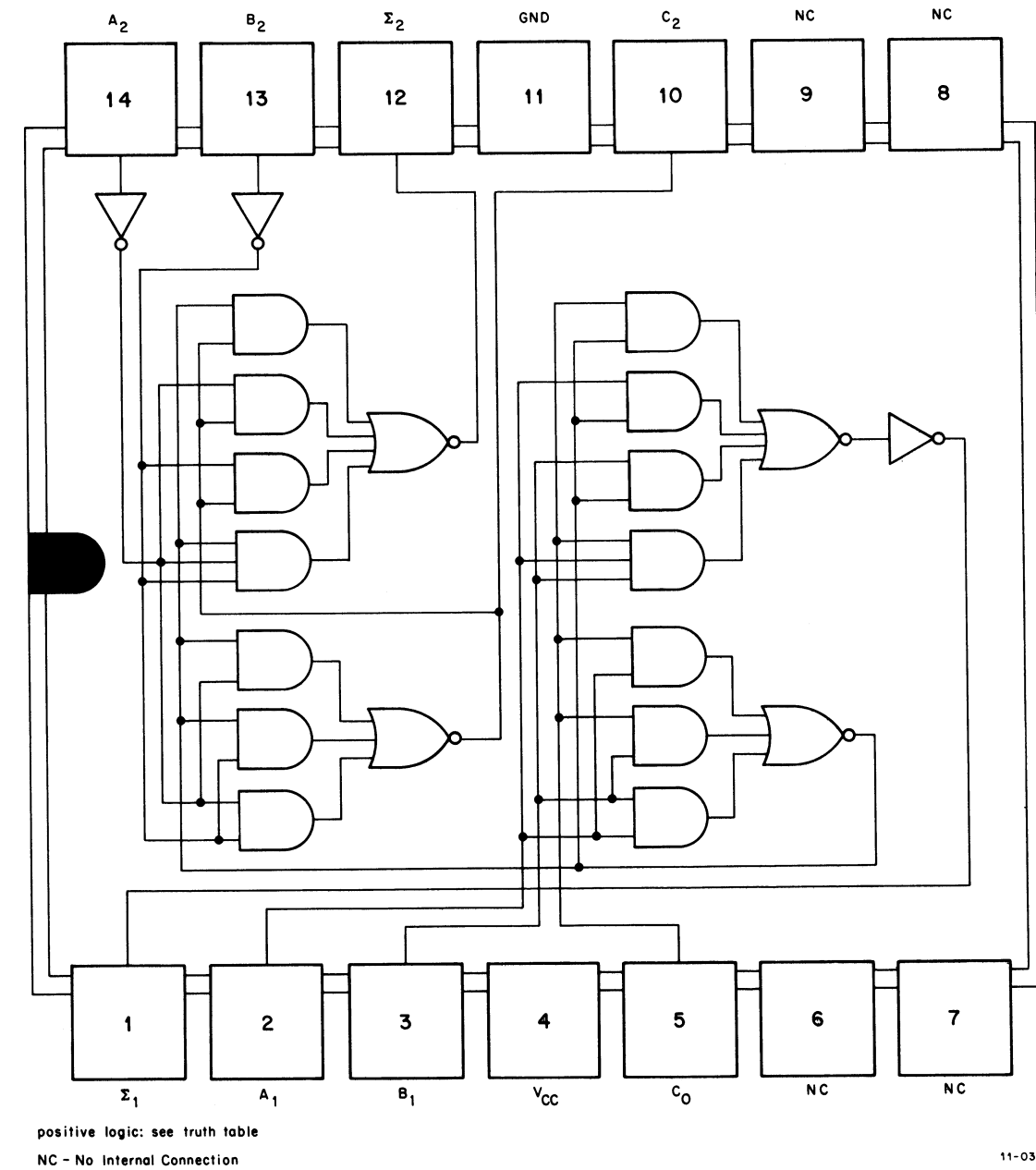


Figure A-1 7482 2-Bit Binary Full Adders

### A.2 74H87 4-BIT SIGN EXTENDER

For an explanation of this element refer to Table 5-1. Figure A-2 is a logic diagram of the sign extender. Table A-2 is the corresponding truth table.

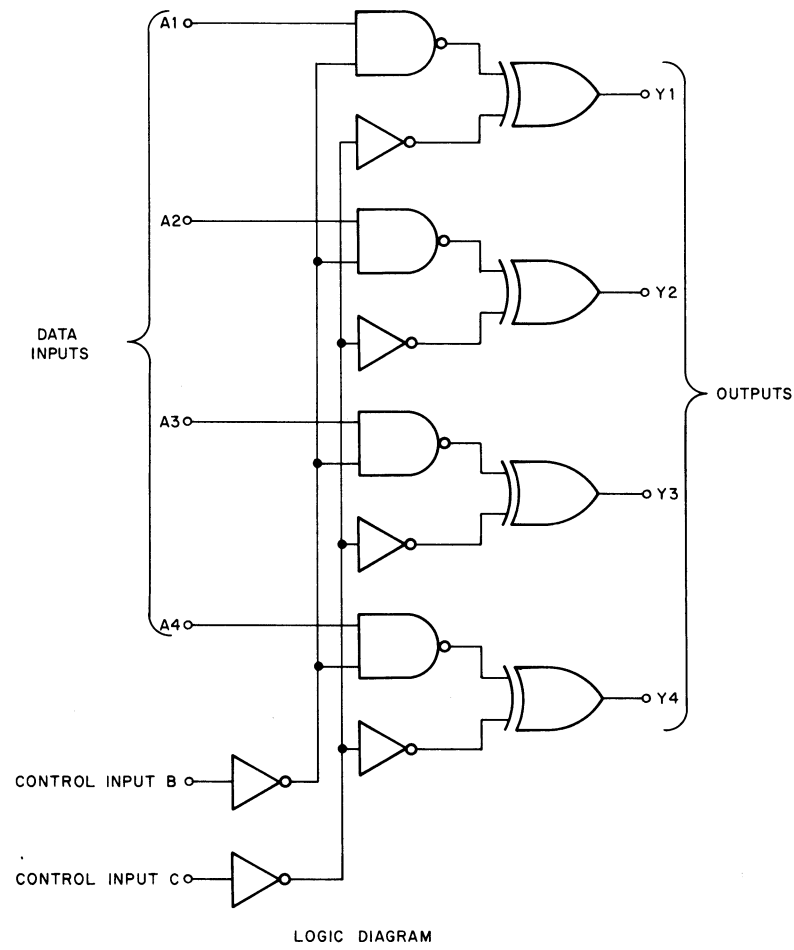


Figure A-2 74H87 4-Bit True/Complement, Zero/One Element

Table A-2  
Truth Table

Control Inputs		Outputs			
B	C	Y1	Y2	Y3	Y4
0	0	$\overline{A1}$	$\overline{A2}$	$\overline{A3}$	$\overline{A4}$
0	1	A1	A2	A3	A4
1	0	1	1	1	1
1	1	0	0	0	0

### A.3 74153 4-WAY MULTIPLEXER

Figure A-3 is a circuit diagram of a Dual 4-Line-To-1-Line Data Selector/Multiplexer. Table A-3 is the corresponding truth table.

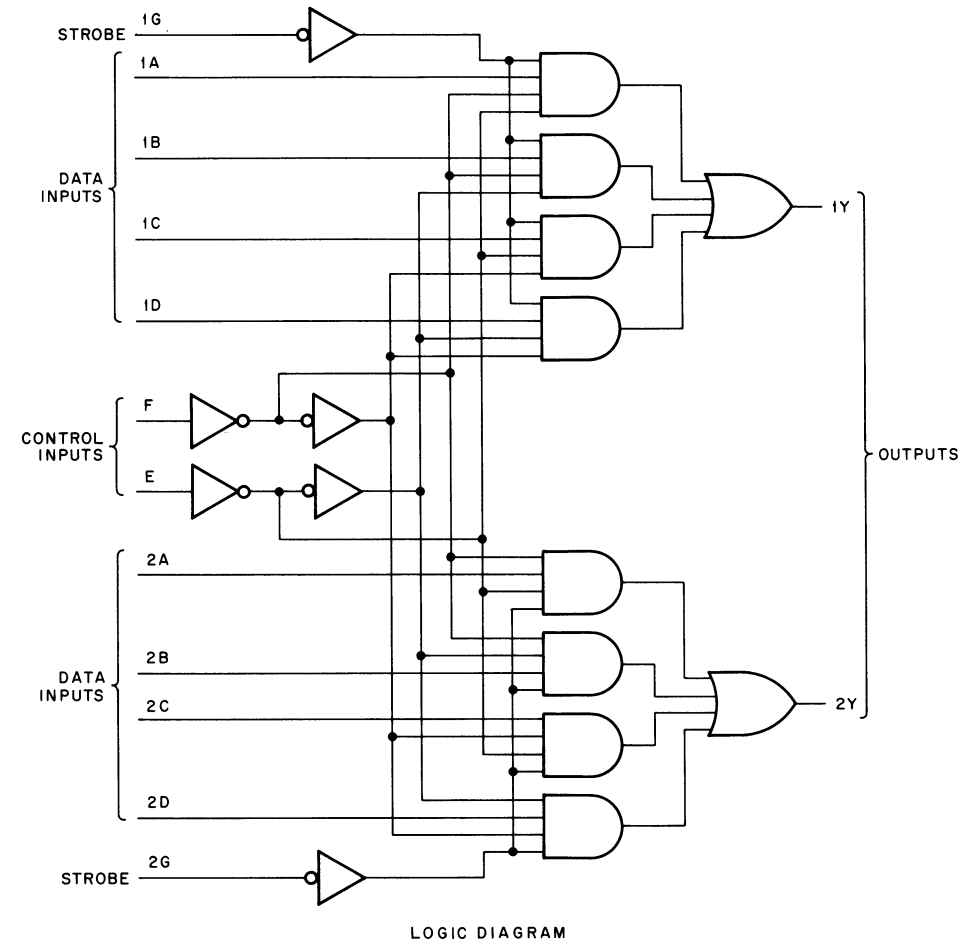


Figure A-3 74153 Dual 4-Line-To-1-Line Data Selector/Multiplexer

Table A-3  
Truth Table

Control Inputs		Data Inputs				Strobe	Output
F	E	A	B	C	D	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

Address inputs A and B are common to both sections.  
H = high level, L = low level, X = irrelevant.

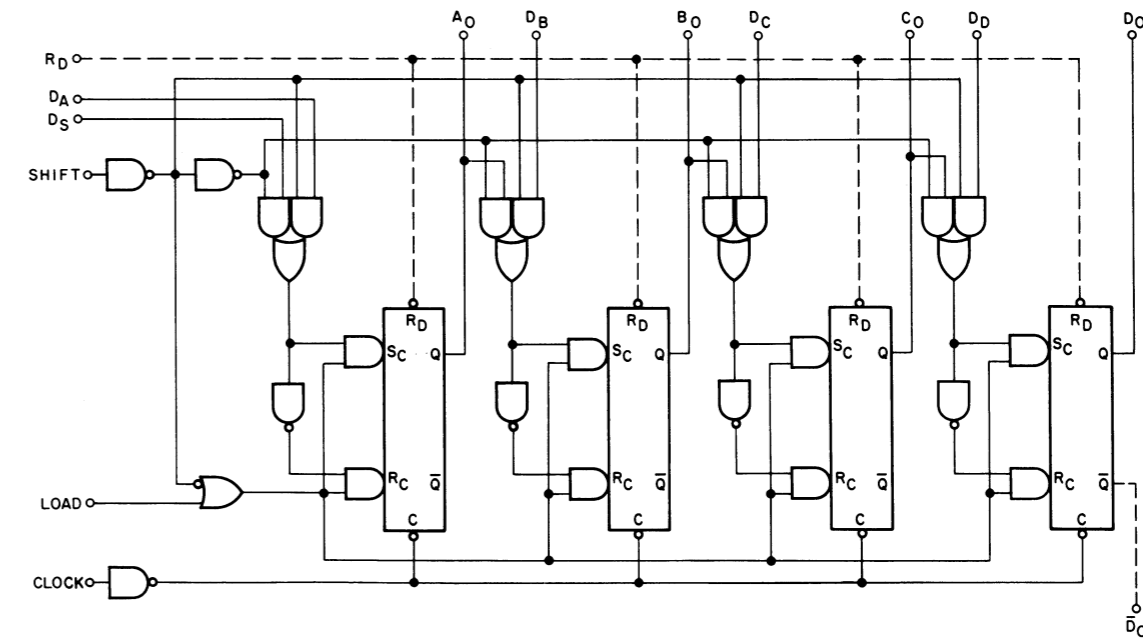


Figure A-4 8271 4-Bit Shift Register

11-0364

#### A.4 8271 4-BIT SHIFT REGISTER

Figure A-4 is a circuit diagram for a 8271 4-bit Shift Register. Mode control logic determines three possible control states. These register states are serial shift right mode, parallel enter mode, and no change (or hold) mode. These states accomplish logical decoding for system control. Table A-4 is a truth table for the control modes. For applications not requiring the hold mode, the load input can be tied high and the shift input used as the mode control.

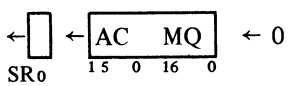
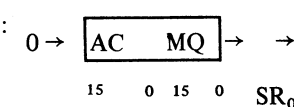
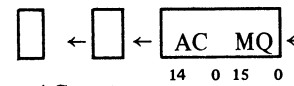
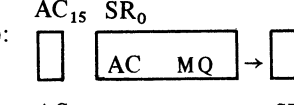
Table A-4  
Control State Truth Table

Control State	Load	Shift
Hold	0	0
Parallel Entry	1	0
Shift Right	0	1
Shift Right	1	1





**APPENDIX B**  
**KE11-A INSTRUCTION SUMMARY**

OP/REG	ADDRESS	DATI or DATIP	DATO	DATOB-Low Byte	DATOB-High Byte	Operation	Status Register (after DATO)							
							N	V	Ovfl	Z	C	AC=1777778	AC=000000	MQ=000000
DIV Divide	777300	Read Zero's	Load X, Start Divide	Load X, Sign Extend, Start Divide	Do nothing	$\frac{AC-MQ}{X} = MQ \text{ rem } AC$	*V	*	*	*	*	*	*	0
AC	777302	Read AC	Load AC	Load AC, Sign Extend	Load AC		-	-	*	*	*	-	*	-
MQ	777304	Read MQ	Load MQ Sign Extend into AC	Load MQ Sign Extend into MQ High Byte and AC	Load MQ, Sign Extend into AC		-	-	*	*	*	*	*	1
MUL	777306	Read Zero's	Load X, Start Multiply	Load X, Sign Extend Start Multiply	Do nothing	$(MQ)(X)=AC-MQ$	0V	*	*	*	*	*	*	0
SC SR	777310 777311	Read SC and SR	Load SC and Load SR Bits 0,6,7	Do Nothing	Do Nothing		*	*	-	-	-	-	-	*
NOR Normalize	777312	Read SC	Start Normalize	Start Normalize	Do Nothing	Shift Left Until: $AC_{15} \neq AC_{14}$ or $AC-MQ = 1100...00$ or $SC=31$	0V	*	*	*	*	*	*	0
LSH Logical Shift	777314	Read Zero's	Load SC, Start Logical Shift	Load SC, Start Logical Shift	Do Nothing	LEFT ( $SC > 0$ ):  RIGHT ( $SC < 0$ ): 	*V	*	*	*	*	*	*	*
ASH Arithmetic Shift	777326	Read Zero's	Load SC, Start Arithmetic Shift	Load SC, Start Arithmetic Shift	Do Nothing	LEFT ( $SC > 0$ ):  RIGHT ( $SC < 0$ ): 	*V	*	*	*	*	*	*	*

- unchanged  
0 cleared  
1 set  
\* set conditionally  
0V no overflow possible  
\*V overflow possible



## APPENDIX C NAMES OF MATHEMATICAL TERMS

The mathematical operations of addition, subtraction, multiplication, and division each produce one result from two operands. The names used to discuss the operands are given in Table C-1. This appendix also includes rules for combining signed numbers in each operation.

Table C-1  
Operand Names

Operation	1st Operand	2nd Operand	Result
Addition	Addend	Addend	Sum
Subtraction	Minuend	Subtrahend	Difference
Multiplication	Multiplicand	Multiplier	Product
Division	Dividend	Divisor	Quotient

The operands can be freely exchanged in addition and multiplication. The order of the operands in the operations is as shown in Figures C-1 through C-4.

$$\begin{array}{r} \text{Addend} \\ +\text{Addend} \\ \hline \text{Sum} \end{array} \quad \text{or} \quad \text{Addend} + \text{Addend} = \text{Sum}$$

Figure C-1 Order of Operands in Addition

$$\begin{array}{r} \text{Minuend} \\ -\text{Subtrahend} \\ \hline \text{Difference} \end{array} \quad \text{or} \quad \text{Minuend} - \text{Subtrahend} = \text{Difference}$$

Figure C-2 Order of Operands in Subtraction

$$\begin{array}{r} \text{Multiplicand} \\ \times \text{Multiplier} \\ \hline \text{Product} \end{array} \quad \text{or} \quad \text{Multiplicand} \times \text{Multiplier} = \text{Product}$$

Figure C-3 Order of Operands in Multiplication

$$\begin{array}{r} \text{Dividend} \\ \hline \text{Divisor} \end{array} = \text{Quotient} \quad \text{or} \quad \begin{array}{r} \text{Quotient} \\ \text{Divisor} \overline{) \text{Dividend}} \end{array}$$

or  $\text{Dividend} \div \text{Divisor} = \text{Quotient}$

Figure C-4 Order of Operands in Division

When operating on signed numbers, the following rules determine the sign of the result:

- For addition: with like signs, add the absolute values and prefix the common sign; with unlike signs, subtract the absolute value of the smaller number, and prefix the sign of the larger.
- For subtraction: change the sign of the subtrahend, and proceed as in addition.
- For multiplication: with like signs, the result is positive; with unlike signs, the result is negative.
- For division: divide the absolute values, and prefix a positive sign for like signs or a negative sign for unlike signs.



**READER'S COMMENTS**

**EXTENDED ARITHMETIC ELEMENT  
DEC-11-HKEA-D**

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

What features are most useful? \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

What faults do you find with the manual? \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

Does this manual satisfy the need you think it was intended to satisfy? \_\_\_\_\_

Does it satisfy *your* needs? \_\_\_\_\_ Why? \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

Would you please indicate any factual errors you have found. \_\_\_\_\_

\_\_\_\_\_  
\_\_\_\_\_

Please describe your position. \_\_\_\_\_

Name \_\_\_\_\_ Organization \_\_\_\_\_

Street \_\_\_\_\_ Department \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip or Country \_\_\_\_\_

-----  
Fold Here  
-----

-----  
Do Not Tear - Fold Here and Staple  
-----

FIRST CLASS  
PERMIT NO. 33  
MAYNARD, MASS.

BUSINESS REPLY MAIL  
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

**digital**

Digital Equipment Corporation  
Technical Documentation Department  
146 Main Street  
Maynard, Massachusetts 01754

