DEC-15-HQFA-D

PDP-15 SYSTEMS
FP15
FLOATING POINT PROCESSOR
MAINTENANCE MANUAL
VOLUME 1

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Edition June 1971

CONTENTS

ILLUSTRATIONS

ENGINEERING DRAWINGS

TABLES

CHAPTER 1
INTRODUCTION

1.1 GENERAL

This chapter provides a physical and functional description of the FP15 Floating-Point Processor. The physical description includes lists of FP15 system parameters and special features.

1.2 FLOATING-POINT PROCESSOR PHYSICAL DESCRIPTION

The FP15 Floating-Point Processor consists of four racks of Medium Scale Integrated logic (MSI) and TTL logic located as shown in Figure 1-1. The interconnecting cabling associated with the FP15 is shown in Figure 1-2. The floating-point processor logic uses an operating voltage of +5 Vdc that is supplied from an H721 Power Supply, with 115V or 220V input and +5 Vdc output fused at 20A. A 716 Power Supply provides the power for the indicator panel. The operating characteristics of the FP15 are listed in Table 1-1; Table 1-2 includes some of the more significant features of the FP15.

Table 1-1
FP15 System Characteristics

Operating Characteristics	
Power Requirements	115V, ±15% 12A 50 ± 1 Hz, 60 ± 1.2 Hz Single Phase 230V, ±15% 6A 50 ± 1 Hz, 60 ± 1.2 Hz Single Phase
Power Consumption	1.4 kW max
Temperature Range	50° - 120°F
Relative Humidity	10 - 95%
Heat Dissipation	4800 btu/hr
Physical Characteristics	
Size	19-in. wide by 21-in. high
Weight	50 lb
No. of Racks	4
Type of Logic	TTL and MSI

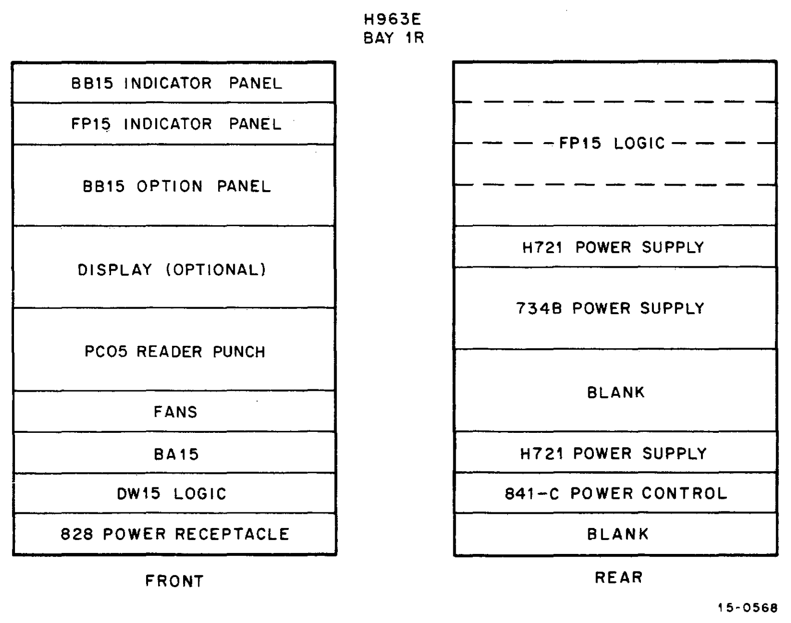
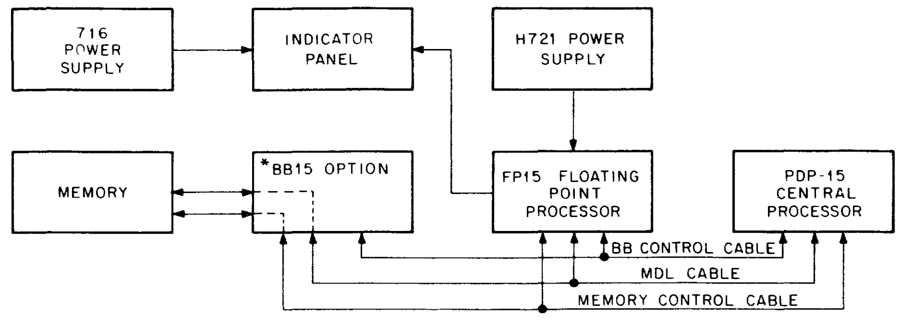


Figure 1-1 Cabinet Housing FP15 Logic



*If BB option is not installed, cables are directly routed to memory.

Figure 1-2 System Interconnecting Cabling

Table 1-2
FP15 System Features

- Directly or indirectly addressable up to 128K of core.
- Performs arithmetic operations on 18- or 36-bit integers and 36- or 54-bit floating-point numbers.
- Allows execution of in-line code--CPU instructions and floating-point instructions may be interspersed as desired.
- I/O Processor can access memory on a shared basis with the floating-point processor; however, the I/O Processor takes priority over the FP15.
- When an undesired condition (Underflow, Overflow, Abnormal Division, or Memory Protect Violation) occurs, the FP15 interrupts the CP stored program and automatically identifies the source of the interrupt.
- Worst-case multiplication and division times on normalized operands do not exceed 24 μs.
- Possesses ability to convert floating-point numbers to integers and integers to floating-point numbers.
- Remainder, product, and align bits in FMQ are accessible by appropriate software.
- Unnormalized and unrounded arithmetic may be specified.
- A class of non-memory reference instructions is available. These instructions use existing contents of FMA and FMB and require no memory reference.
- Built-in maintenance logic (maintenance mode) allows single or multiple substeps of an instruction. All major registers and control can be examined at the end of each step.
- Designed to operate with existing PDP-15 options (Memory Protect, Memory Relocate, etc.) with no increase in cycle time.

1.3 FUNCTIONAL DESCRIPTION

The FP15 Floating-Point Processor functional block diagram is shown in Figure 1-3. Before describing each of the major elements in the diagram, it is necessary to introduce the various operating cycles in the FP15; they are:

a. FETCH
b. OPAND
c. EXP
d. FUN
e. NOR
f. WRITE
g. INTERRUPT

1.3.1 Operating Cycles

During a floating-point instruction, the FP15 is in one of the operating cycles. Each cycle is approximately 900 ns and is divided into three time states (300 ns per time state). The cycles can be extended in time due to shifting and aligning. In turn, each time state is subdivided into four phases (75 ns per phase). The following paragraphs provide a brief description of the major events that occur during each cycle.

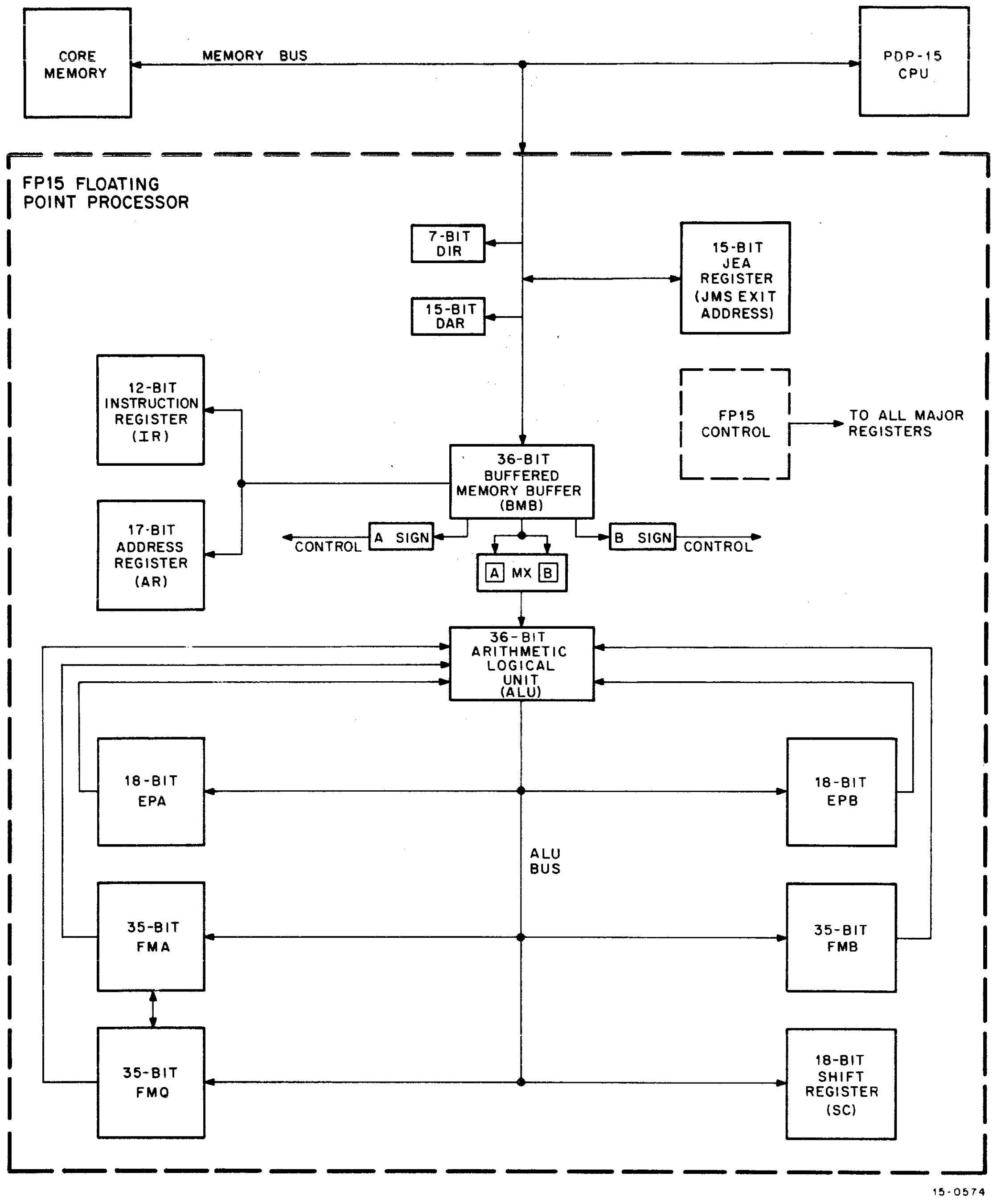
FETCH - In the FETCH cycle the instruction word (first word) is loaded into the FP15 Instruction Register (IR) and the address of the operand is loaded into the FP15 Address Register (AR). If indirection (indirect addressing) is requested, the FP15 remains in the FETCH cycle to obtain the effective address.

OPAND - In the OPAND cycle the operand(s) is transferred from memory to the FP15. The number of operands transferred depends on the format in Table 1-3.

Table 1-3
Operand Transfer and Cycle Time

Format	No. of Operands	Cycle Time
Single-precision integer	One operand	(1) 1.2 μs
Double-precision integer and Single-precision floating-point	Two operands	(2) 2.4 μs
Double-precision floating-point	Three operands	(3) 3.6 μs

If non-memory reference instructions are specified, the OPAND cycle is bypassed and no operands are transferred from memory to the FP15.



CORE MEMORY
MEMORY BUS
PDP-15 CPU
FP15 FLOATING POINT PROCESSOR
7-BIT DIR
15-BIT DAR
15-BIT JEA REGISTER (JMS EXIT ADDRESS)
FP15 CONTROL
TO ALL MAJOR REGISTERS
12-BIT INSTRUCTION REGISTER (IR)
17-BIT ADDRESS REGISTER (AR)
36-BIT BUFFERED MEMORY BUFFER (BMB)
CONTROL
A SIGN
B SIGN
CONTROL
A MX B
36-BIT ARITHMETIC LOGICAL UNIT (ALU)
18-BIT EPA
18-BIT EPB
ALU BUS
35-BIT FMA
35-BIT FMB
35-BIT FMQ
18-BIT SHIFT REGISTER (SC)
15-0574

Figure 1-3 FP15 Functional Block Diagram

EXP - In the EXP cycle, during floating-point addition and subtraction, the mantissa with the smaller exponent is aligned with the mantissa having the larger exponent. Alignment occurs by right-shifting the smaller mantissa.

In the EXP cycle, during floating-point multiplication and division, the exponent is calculated. In integer format, negative integers in 2's complement format are converted to sign and magnitude numbers during the EXP cycle.

FUN - In the FUN cycle, the actual arithmetic or logical operation is performed. The cycle time required is the basic 900 ns, plus the additional time required for shift, multiply, and divide operations.

NOR - In the NOR (normalize) cycle, the FMA is normalized by shifting. Rounding may also be requested. The basic NOR cycle requires 900 ns, plus an additional 150 ns for each shift necessary to normalize.

WRITE - During the WRITE cycle, the operands are transferred to memory. The operands transferred from the FP15 to memory are:

Single-precision integer--one 2's complement operand

Double-precision integer--two 2's complement operands

Single-precision floating-point--2's complement exponent and high-order mantissa

Double-precision floating-point--2's complement exponent and high-order and low-order mantissas.

Each transfer requires about 1.2 μs.

1.3.2 Major Register Functional Descriptions

Buffered Memory Buffer Register (BMB) - The 36-bit Memory Buffer Register is loaded from the memory bus 18 bits at a time. The output of this register is connected to the ALU, the instruction register, and the address register. All inputs from the memory pass through the memory buffer.

Instruction Register (IR) - The 12-bit Instruction Register stores bits 6 through 17 of the instruction word retrieved from memory during the FETCH cycle. Bits 6 through 17 remain in the IR until another instruction is fetched from memory.

Address Register (AR) - The 17-bit Address Register stores the effective address used in fetching or storing operands.

Arithmetic Logic Unit (ALU) - The 36-bit ALU performs both arithmetic and logic operations in the FP15. The output of the ALU is connected to all major registers via the ALU bus. Most major registers are available as inputs to the ALU.

EPA - The 18-bit EPA is a synchronous up-down counter used to store the 2's complement exponent associated with the mantissa loaded in the FMA. The most significant bit of the EPA represents the sign of the exponent. For single-precision floating-point format, the most significant bit of the exponent is bit 9. The value of this bit is extended from bit 9 through bit 0. The EPA is loaded from the ALU bus and keeps track of the exponent associated with the mantissa in the FMA.

FMA - The 35-bit FMA stores an integer operand during integer arithmetic or a mantissa during floating-point arithmetic. The FMA is loaded from the ALU and can be shifted left or right. The FMA can also be loaded and shifted simultaneously from the ALU bus during multiplication and division. The EPA and A SIGN/FMA are the floating-point accumulator.

A SIGN - The 1-bit A SIGN register stores the sign of the operand loaded into the FMA. A 1 in this register indicates a negative number; a 0 indicates a positive number.

FMQ - The FMQ is a 36-bit extension of the FMA or FMB and is used primarily during arithmetic operations. Bits shifted out of the FMA or FMB, during alignment for addition and subtraction, are shifted into the FMQ. The most significant bit in the FMQ is used for rounding, if requested. The FMQ can be loaded from the ALU bus, or directly from the FMA, and has a shift-left and shift-right capability.

EPB - The 18-bit EPB register is loaded from the ALU bus and stores the 2's complement exponent associated with the mantissa loaded in the FMB. The most significant bit of the EPB represents the sign of the exponent. For single-precision floating-point format, the most significant bit of the exponent is bit 9. The value of this bit is extended from bit 9 through bit 0.

FMB - The 35-bit FMB register stores an integer operand during integer arithmetic or a mantissa during floating-point arithmetic. Unlike the FMA, the FMB can only be shifted right for alignment. The FMB is loaded directly from the ALU bus. The EPB and B SIGN/FMB are a second operand register.

B SIGN - The 1-bit B SIGN register stores the sign of the operand loaded into the FMB. A 1 in this register represents a negative mantissa; a 0 represents a positive mantissa.

Shift Counter - The shift counter performs the following functions:

a. Keeps track of the number of words to be fetched from memory during the OPAND cycle.
b. Keeps track of the number of words written into memory during the WRITE cycle.
c. Keeps track of the number of shifts required for multiply and divide operations.
d. Limits the number of shifts during normalizing to a maximum of 35_{10}.
e. Controls the number of shifts required during alignment.
f. Checks for exponents having differences which exceed 35_{10}.

JEA - The 15-bit JEA address register points to the interrupt handling routines in core that service the floating-point interrupts (underflow, overflow, abnormal divide, and FP memory trap). This register is loaded by software control.

Diagnostic Instruction Register (DIR) - The 7-bit DIR determines the number of steps through which an instruction is to be sequenced.

Diagnostic Address Register (DAR) - The 15-bit DAR specifies the address in core where the contents of the registers are to be stored.

CHAPTER 2
MODULE DESCRIPTIONS

2.1 GENERAL

This chapter provides descriptions of the following modules used in the FP15 Floating-Point Processor:

M238	SYNCHRONOUS UP/DOWN COUNTER
M159	ARITHMETIC LOGIC UNIT
M191	CARRY LOOK-AHEAD GENERATOR
M248	RIGHT-SHIFT PARALLEL LOAD REGISTER
M1701	DATA SELECTOR
M1713	16-To-1 DATA SELECTOR

2.1.1 M238 Synchronous Up/Down Counter

The M238 Synchronous Up/Down Counter consists of two DEC 74193 4-bit synchronous up/down counters. Synchronous operation is provided by having all flip-flops in the counter clocked simultaneously so that the outputs change at the same time. The flip-flops are master-slave flip-flops and the outputs are triggered by a positive-going transition of one of two count (clock) inputs. One input is designated U (up count); the second input is designated D (down count). The direction of counting is determined by the count input that is pulsed while the other count input is high.

The outputs of the flip-flops may be preset to any desired state by entering the data at the data inputs while the load input (L) is low. The output will change to reflect the input, regardless of the count pulses.

A clear input (CLR) forces all outputs low on receipt of a high clear input. The clear input is independent of the count and load inputs.

Both borrow and carry outputs are available for cascading the up-counting and down-counting operations. When counter underflow occurs, the borrow output produces the same width pulse as the down-count input. When counter overflow occurs, the carry output produces the same width pulse as the count-up input. Cascading is accomplished by connecting the borrow and carry inputs to the count-down and count-up inputs, respectively, of the next counter.

The M238 Counter is used in the EPA, DIR, and DAR registers in the FP15 Floating-Point Processor. Figure 2-1 is an example of how the M238 Counter is used in the DIR register (see drawing D-BS-FP15-0-14).

NOTE

The up count is inhibited by +3V in the DIR register, indicating that this register can only be decremented.

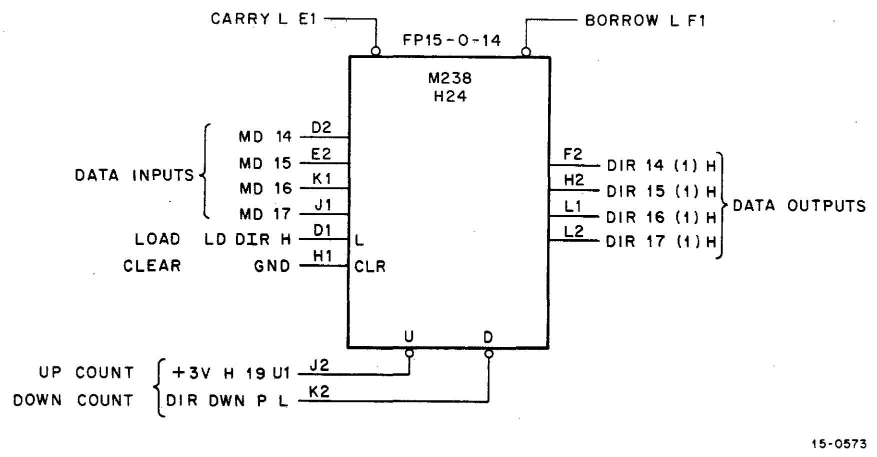


Figure 2-1 M238 Synchronous Up/Down Counter

2.1.2 M159 Arithmetic Logic Unit

The M159 4-bit Arithmetic Logic Unit (ALU) contains a single DEC 74181 integrated circuit. Nine of these ALU modules are used in the FP15 Floating-Point Processor to perform 36-bit arithmetic and logic operations, as shown on drawings D-BS-FP15-0-19 through D-BS-FP15-0-27.

This integrated circuit performs 16, 4-bit arithmetic operations when the MODE control (MC) input is low and 16 logic functions when the MC input is high. The functions are selected by applying combinations of function select inputs S0 through S3. For FP15 applications, the function select and MC inputs are generated by the ALU control logic shown on drawing D-BS-FP15-0-33.

Only two arithmetic operations, A plus B and A minus B minus 1, are selected in the FP15; five logic functions, A, -A, B, -B, and logical 0 are performed in the FP15. The combined ALU truth table for FP15 arithmetic operations and logic functions is listed as follows:

Mode Control	Function Select Inputs				Output Function
	S3	S2	S1	S0	
0	1	0	0	1	A plus B (arithmetic operation)
0	0	1	1	0	A minus B minus 1 (arithmetic operation)
0	0	0	0	0	A (logic function)
1	0	0	0	0	-A (logic function)
1	1	0	1	0	B (logic function)
1	0	1	0	1	-B (logic function)
1	0	0	1	1	Logical 0 (logic function)

In addition, a comparator output, A=B, is provided when the four A inputs are equal to the four B inputs if the function A=B=1 is selected. A full-carry look-ahead provides fast, simultaneous carry generation by the M191 module.

Figure 2-2 shows the ALU configuration for bits 00 through 03 in the FP15 Floating-Point Processor.

2.1.3 M191 Carry Look-Ahead Generator

The M191 Carry Look-Ahead Generator, consisting of two DEC 74182 integrated circuits, is a high-speed generator capable of anticipating a carry through a group of ALUs. A 13-ns delay occurs for each look-ahead level. The M191, when used in conjunction with the M159 ALU, provides carry, generate-carry, and propagate-carry functions for 36-bit words.

Figures 2-3 and 2-4 show how the M191 is used.

Each carry look-ahead circuit is associated with four ALUs (16 bits). Each circuit generates the anticipated carry through its respective group of ALUs, as well as providing a Generate (G) and Propagate (P) input to a third carry look-ahead circuit associated with the last ALU; hence, the term full-carry look-ahead in three levels (36 bits).

Depending on the selected function of the ALUs, the carry look-ahead circuitry determines whether a carry will be propagated through the particular ALU, or whether the selected function will generate a

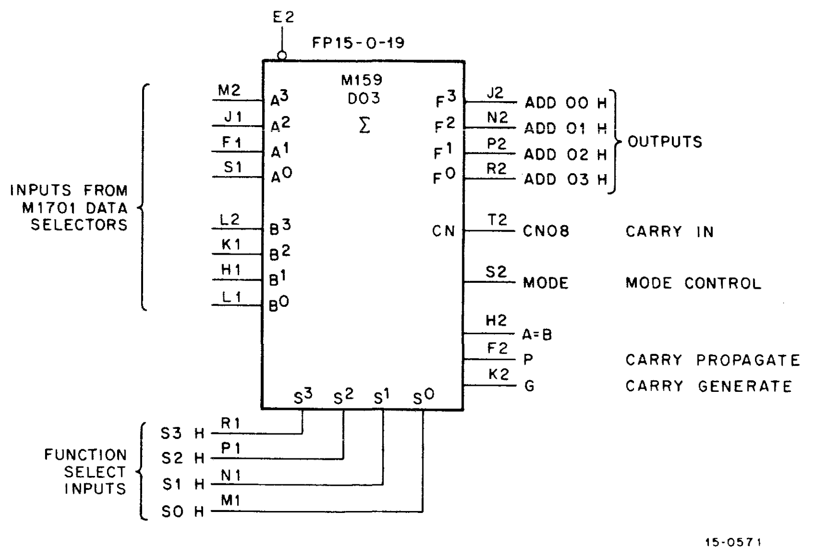


Figure 2-2 M159 Arithmetic Logic Unit

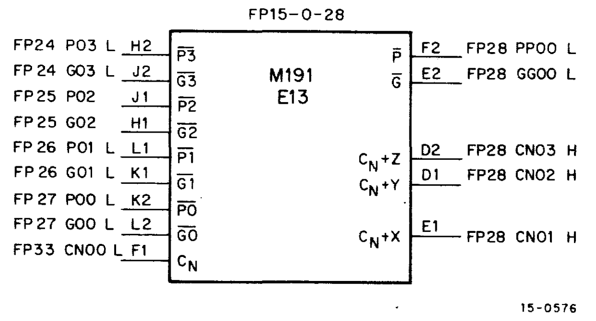


Figure 2-3 M191 Carry Look-Ahead Generator

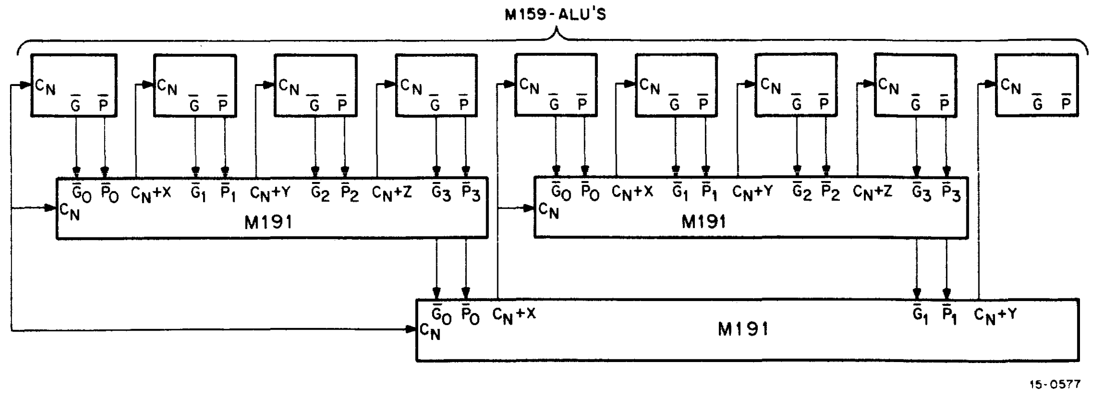


Figure 2-4 36-Bit ALU, Full-Carry Look-Ahead in Three Levels carry. If a carry is produced, it is directed into the next ALU in line. This sequence is continued for each of the four ALUs in the section. The carry look-ahead circuitry then "looks" at the G and P signals of all four ALUs and determines whether a carry should be inserted into the next four ALUs and into the third level of carry look-ahead. This process is continued for the second section of ALUs (next 16 bits). Finally, the third level of carry look-ahead determines whether a carry should be inserted into the final ALU by examining the resulting G and P inputs of the other two look-ahead circuits.

The truth table for the first-stage carry is as follows:

True Carry Insert = L

P00	G00	C_{N00}	C_{N+X}
L	L	L	H
L	L	H	H
H	H	H	L
L	H	L	L

True Carry Insert = Low

P00	G00	C_{N00}	C_{N+X}
L	L	L	H
H	L	L	H
L	H	L	L
H	H	L	L
L	L	H	H
H	L	H	H
L	H	H	H
H	H	H	L

The following are the logic equations for a carry look-ahead stage:

$$\overline{C_{N01}} = \overline{C_{N00}} * \overline{G_0} + \overline{G_0} * \overline{P_0}$$

$$\overline{C_{N02}} = \overline{G_1}*\overline{P_1} + \overline{P_0}*\overline{G_0}*\overline{G_1} + \overline{G_1}*\overline{G_0}*\overline{C_N}$$

$$\overline{C_{N03}} = \overline{P_2}*\overline{G_2} + \overline{G_1}*\overline{G_2}*\overline{P_1} + \overline{G_0} + \overline{G_1}*\overline{G_2}*\overline{P_0} + \overline{G_0}*\overline{G_1}*\overline{G_2}*\overline{C_N}$$

$$\overline{GG00} = \overline{P_3}*\overline{G_3} + \overline{P_2}*\overline{G_3}*\overline{G_2} + \overline{P_1}*\overline{G_3}*\overline{G_2}*\overline{G_1} + \overline{G_3}*\overline{G_2}*\overline{G_1}*\overline{G_0}$$

$$\overline{PP00} = \overline{P_3} + \overline{P_2} + \overline{P_1} + \overline{P_0}$$

where

$\overline{C_{NXX}}$ = True L

$\overline{GXX}$ = True H

$\overline{PXX}$ = True H

$\overline{GGXX}$ = True H

$\overline{PPXX}$ = True H

2.1.4 M248 Right-Shift Parallel Load Register

The M248 Right-Shift Parallel Load Register consists of two 4-bit DEC 7495 Right-Shift Parallel Load Registers connected to allow right-shifting between 4-bit sections. The registers perform load or right-shift operations, depending on the logical input to the MC. When a logical 0 is applied to the MC input, the output of each flip-flop is connected to the succeeding flip-flop and right-shift operation is performed by clocking at the input designated RS. During this time, the input designated LS is inhibited. When a logical 1 is applied to the MC input, the flip-flops are decoupled (to prevent right-shift); the register is loaded with parallel inputs when the input designated LS is clocked. The register can be configured for left-shift operation by connecting the output of each flip-flop to the parallel input of the previous flip-flop.

The M248 Right-Shift Parallel Load Register is used in the EPB, FMA, FMB, and FMQ registers in the FP15 Floating-Point Processor. Each module is capable of handling 8 bits. Figure 2-5 shows a sample of the application of this module in the FP15 Floating-Point Processor.

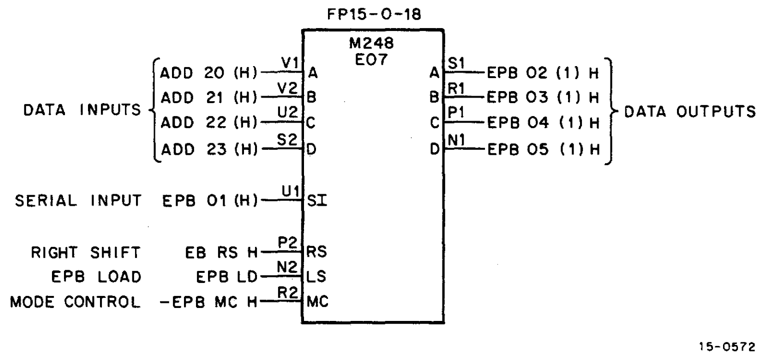


Figure 2-5 M248 Right-Shift Parallel Load Register

2.1.5 M1701 Data Selector

The M1701 Data Selector contains two DEC 74153 Dual 4-Line-to-1-Line Data Selector/Multiplexer integrated circuits. These integrated circuit modules comprise input multiplexers A and B of the

36-bit ALU in the FP15 Floating-Point Processor. They are also used as input multiplexers to the shift counter, MPO, FMA, and FMQ registers. A complete block schematic of the input multiplexers is shown on drawings D-BS-FP15-0-19 through D-BS-FP15-0-27.

For each section of each IC, one of four data inputs is selected by combinations of address input signals A and B. The selected data input is strobed to the output by a low strobe signal. Refer to the following truth table for a typical input multiplexer A section.

Address Inputs		Data Inputs				Strobe	Output
A	B	0	1	2	3		
x	x	x	x	x	x	1	0
0	0	0	x	x	x	0	0
0	0	1	x	x	x	0	1
1	0	x	0	x	x	0	0
1	0	x	1	x	x	0	1
0	1	x	x	0	x	0	0
0	1	x	x	1	x	0	1
1	1	x	x	x	0	0	0
1	1	x	x	x	1	0	1
x indicates irrelevancy.							

Address input signals A and B are common to both sections of each IC. Figure 2-6 is a typical application of the M1701 Data Selector in the FP15 Floating-Point Processor.

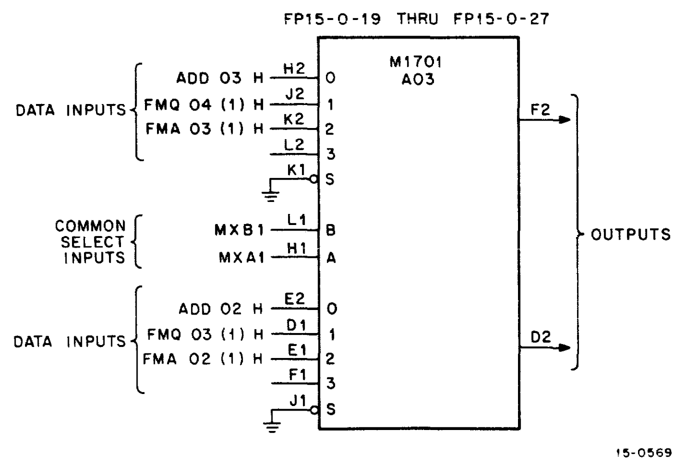


Figure 2-6 M1701 Data Selector

2.1.6 M1713 16-To-1 Data Selector

The M1713 16-To-1 Data Selector contains a single DEC 74150 integrated circuit. It is used in the output multiplexer section of the FP15 Floating-Point Processor, where up to 16 major register outputs are selected for transfer to the common MPO bus. The block schematic of the output multiplexer is shown on drawing D-BS-FP15-0-03.

Data inputs are selected by combinations of data select signals MXA, MXB, MXC, and MXD, which are generated by the multiplexer control logic shown on drawing D-BS-FP15-0-05. The strobe inputs are wired to ground so that each IC is always enabled. A typical truth table for the 16-to-1 Data Selector follows:

Data Select Inputs				Data Input* Selected
MXD	MXC	MXB	MXA	
0	0	0	0	DIR12
0	0	0	1	JEA12
0	0	1	0	ADD30
0	0	1	1	ADD12
0	1	0	0	FMQ30
0	1	0	1	FMQ12
0	1	1	0	FMB30
0	1	1	1	FMB12
1	0	0	0	EPB12
1	0	0	1	FMA30
1	0	1	0	FMA12
1	0	1	1	EPA12
1	1	0	0	IR12
1	1	0	1	BMB30
1	1	1	0	BMB12
1	1	1	1	MPI12
* Signal mnemonics vary as shown on drawing D-BS-FP15-0-03.				

Figure 2-7 is a typical example of the manner in which the M1713 Data Selector is utilized.

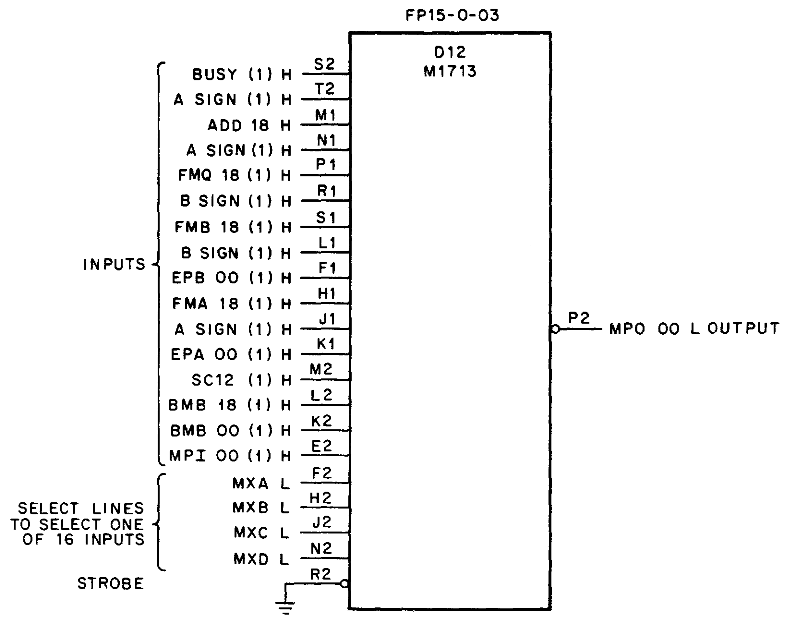


Figure 2-7 M1713 16-To-1 Data Selector

CHAPTER 3
FP15/PDP-15 INTERFACE

3.1 INTRODUCTION

This chapter describes the interface between the CPU, FP15, and memory. This interface is described by discussing the major events that occur during the FETCH, OPAND, WRITE, and Interrupt (INT) cycles, followed by a flow diagram description of each cycle. The EXP, FUN, and NOR cycles, internal to the FP15, are described in Chapter 4. Figure 3-1 shows the various control signals associated with the interface.

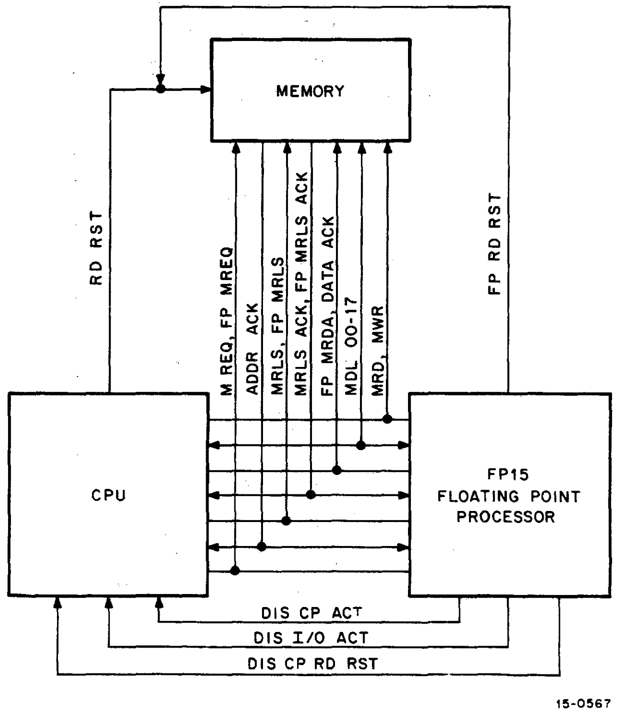


Figure 3-1 Major Signal Interface Diagram

3.2 FETCH CYCLE INTERFACE

Prior to the FETCH cycle, the floating-point instruction from memory is strobed into the FP15 BMB. During the FETCH cycle, the operand address is strobed into the FP15 Address Register (AR) (see Figure 3-2). If indirection is specified, a second FETCH cycle is performed to obtain the effective address.

Every instruction is monitored by both the CPU and the FP15, which are in parallel on the memory bus. Bits 00 through 05 of the instruction are examined for an octal code of 71. The 71 is recognized by the CPU as a NOP and by the FP15 as a floating-point instruction. The CPU strobes the instruction into the memory input (MI) register and then into the instruction register (IR), while the FP15 strobes the instruction into the BMB register.

The CPU executes the $71XXXX_8$ (NOP) and makes a second memory reference to the next location, as if it were fetching the next instruction. This memory request (M REQ) actually fetches the operand address that is the second half of the two-word FP15 instruction. The normal interface signals between the CPU and memory take place; i.e., the CPU specifies an address, READ cycle, and issues M REQ. After M REQ is placed on the memory bus, the contents of the BMB in the FP15 are strobed into the IR; the DIS CP ACT and DIS CP RD RST signals are generated to inhibit the CPU from making further memory requests. Address Acknowledge (ADDR ACK) is returned from memory to clear M REQ in the CPU. The memory then places the operand address on the memory data line (MDL) and issues RD RST. The operand address is strobed into the BMB in the FP15.

The CPU does not see the operand address because DIS CP RD RST prevents RD RST from loading the MI and halts the CPU in Time State 3, Phase 3 (TS03*PH03).

SUBCYCLE	CENTRAL PROCESSOR	FLOATING POINT UNIT	MEMORY	REMARKS
	M REQ, ADDR, RD			CPU REQUESTS MEM. CYCLE
			ADDR ACK	MEM. ACKNOWLEDGES ADDRESS
	0 → M REQ			
	REMOVE ADDR FROM MDL		0 → ADDR ACK	CPU REMOVES ADDR FROM MDL
	SET FETCH		INSTR ON MDL AND RD RST ISSUED	MEM PLACES INSTR. ON MDL
FLOATING POINT INSTRUCTION	$71XXXX_8$ → MI	$71XXXX_8$ → BMB		$71XXXX_8$ RECOGNIZED AS FLOATING POINT INSTR.
	MRLS		MRLS ACK	AND IS STROBED IN BMB OF FPU AND MI OF CPU. CPU
	0 → MRLS		0 → MRLS ACK	COMPLETES REST OF MEM CYCLE.
	$71XXXX_8$ → IR (EXECUTED AS NOP)			$71XXXX_8$ STROBED INTO IR IN CPU
OPERAND ADDRESS	M REQ	DIS CP ACT, DIS RD RST	ADDR ACK	M REQ IS MADE FOR LOCATION CONTAINING OPERAND ADDR.
	0 → M REQ	BMB → IR	0 → ADDR ACK	CPU RD RST IS DISABLED, WHICH INHIBITS RD RST FROM RESTARTING CPU CLOCK.
		MDL → BMB	OPERAND ADDR ON MDL & RD RST ISSUED	OPERAND ADDR STROBED INTO BMB. $71XXXX_8$ STROBED INTO IR IN FPU.
		FP MRDA	MRLS ACK	REST OF MEM CYCLE COMPLETED.
		0 → FP MRDA	0 → MRLS ACK	
		BMB → AR		OPERAND ADDR STROBED INTO AR
DUMMY SETUP		REMOVE DIS CP RD RST		CPU BECOMES ACTIVE
		FPU ISSUES FP RD RST WITH 710000_8 ON MDL		FP RD RST LOOKS LIKE RD RST TO CPU. CPU
	710000_8 → MI			STROBES 710000_8 INTO MI.
	MRLS	FP MRLS ACK		THE EXECUTION OF THE 710000 INSTRUCTION WAITING FOR CPU
	0 → MRLS	0 → FP MRLS ACK		ACTIVE TO SET (WAITING TO ISSUE M REQ)

Figure 3-2 Memory Interface--FETCH Cycle

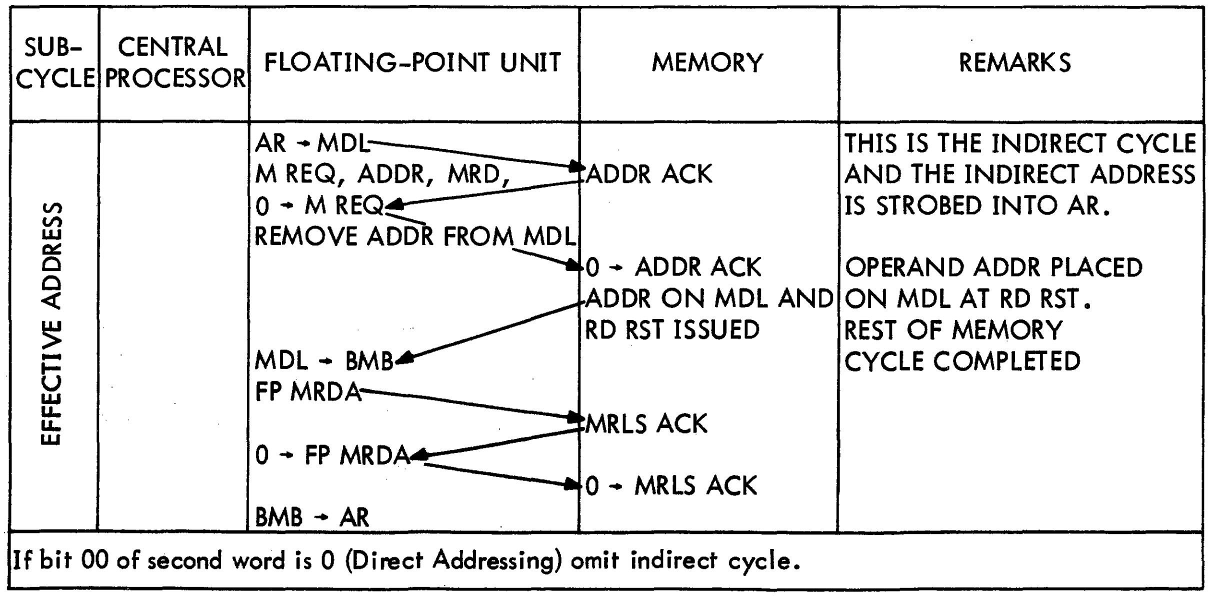
The FP15 now issues MRDA (Memory Release and Data Acknowledge) which releases the memory for additional requests and acknowledges receipt of the data (operanda ddress). The memory cycle is completed when the memory issues MRLS ACK, clears MRDA in the FP15 which, in turn, clears MRLS ACK (Memory Release Acknowledge). The operand address, which was loaded into the BMB, is now strobed into the FP15 AR. Since the CPU did not receive the operand address, it is still waiting for data from memory. The FP15 places a 710000_8 on the MDL, clears DIS RD RST, and sends RD RST to the CPU. The CPU loads the 710000_8 into the MI and generates MRLS. The FP15 generates MRLS ACK to allow the CPU to complete its cycle.

The CPU executes the 710000_8 but is prevented from making a M REQ because of DIS CP ACT. The CPU waits in TS03*PH02 until completion of the FP15 instruction. The memory interface is now free for I/O memory requests.

3.3 FETCH (INDIRECT) CYCLE INTERFACE

If bit 00 of the second FP15 word (address) is a 1, specifying indirection, a second FETCH (indirect) cycle is performed. This word, which is in the FP15 AR, and is the address of the effective address, is placed on the MDL (see Figure 3-3). The FP15 requests a memory cycle and the contents of the operand address (effective address) are accessed from memory, placed on the MDL, and RD RST issued. The address is then strobed into the BMB register. The FP15 issues FP MRDA to memory, which releases memory for further requests. The effective address, which was transferred into the BMB, is now strobed into the AR and represents the address of the first operand.

If bit 00 of the second FP15 word (address) is a 0, no indirection is specified and this cycle is omitted.



SUB-CYCLE	CENTRAL PROCESSOR	FLOATING-POINT UNIT	MEMORY	REMARKS
EFFECTIVE ADDRESS		AR → MDL M REQ, ADDR, MRD, 0 → M REQ REMOVE ADDR FROM MDL MDL → BMB FP MRDA 0 → FP MRDA BMB → AR	ADDR ACK 0 → ADDR ACK ADDR ON MDL AND RD RST ISSUED MRLS ACK 0 → MRLS ACK	THIS IS THE INDIRECT CYCLE AND THE INDIRECT ADDRESS IS STROBED INTO AR. OPERAND ADDR PLACED ON MDL AT RD RST. REST OF MEMORY CYCLE COMPLETED
If bit 00 of second word is 0 (Direct Addressing) omit indirect cycle.				

Figure 3-3 Memory Interface--FETCH Cycle (Indirect)

3.4 FETCH CYCLE DESCRIPTION

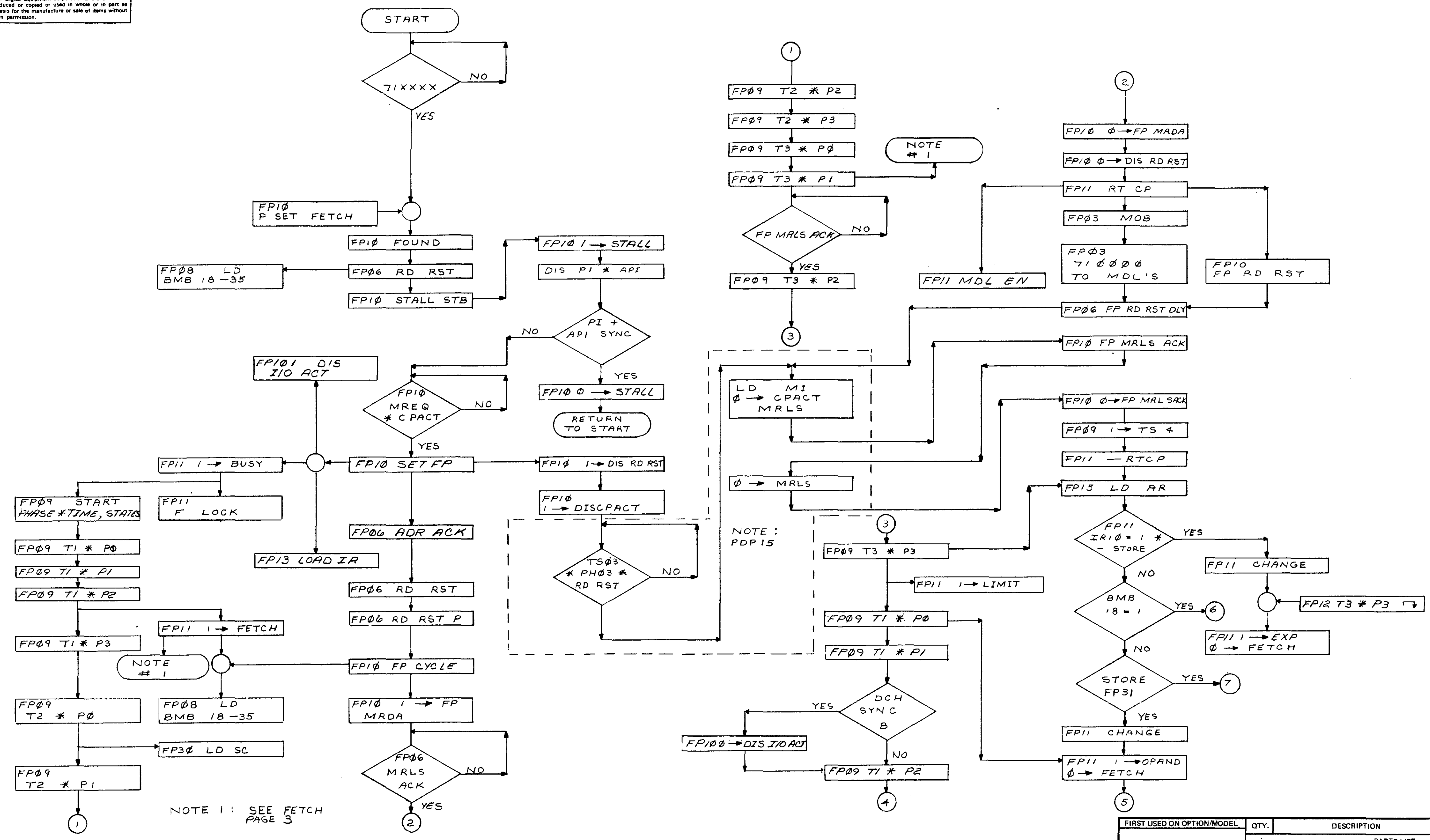
The FP15 detects a floating-point instruction by monitoring MDL bits 00 through 05 for a $71XXXX_8$ while FP SET FETCH is true (see drawings D-FD-FP15-0-45 through D-FD-FP15-0-47). FP SET FETCH indicates that the CPU is fetching an instruction. When the $71XXXX_8$ is detected, the contents of the MDLs are strobed into the FP15 BMB bits 18 through 35 and the floating-point operation is started. The PI and API facilities are disabled at this point, to prevent an interrupt during the floating-point instruction.

The CPU executes the $71XXXX_8$ as a NOP and makes a memory request for the next instruction that is actually the operand address associated with the floating-point instruction. The FP15 sets BUSY, DIS CP ACT, DIS RD RST, and loads the contents of BMB bits 18 through 35, which contain the floating-point instruction, into its IR. BUSY starts the floating-point phases and time states and DIS RD RST prevents the CPU from seeing the RD RST of the memory request for the operand address. The CPU waits in TS03*PH03 for RD RST. When RD RST is returned by memory, the FP15 strobes the contents of the MDL into BMB bits 18 through 35 and issues MRDA to memory. The memory responds by issuing MRLS ACK which clears MRDA. The FP15 now completes the CPU memory request by clearing DIS RD RST, enabling 710000_8 (NOP) onto the MDL, and issuing FP RD RST, which strobes the NOP into the CPU's MI. The CPU responds with MRLS and the FP15 returns MRLS ACK. The CPU begins to execute the NOP but cannot issue a M REQ because DIS CP ACT holds the CPU in TS03*PH02, thus allowing I/O memory requests to be made.

The FP15 loads BMB bits 18 through 35 (operand address) into the AR and determines if an I/O memory request is pending. If one is pending, the FP15 waits; if not, the FP15 determines if indirection has been requested. When indirection has not been requested, the FP15 enters the OPAND cycle, if an operand FETCH is requested, or the EXP cycle, if no operand FETCH is requested.

If indirection has been requested, the FP15 places the contents of the AR onto the MDL and issues M REQ. When RD RST is received from the memory, the contents of the MDL are strobed into BMB bits 18 through 35 and then loaded into the AR. The FP15 then enters OPAND or EXP, as described above.

Two other operations are also performed in the FETCH cycle. If the instruction is ADD, SUBTRACT, or FIX, the FMQ is cleared during TS03 of the FETCH cycle. If the instruction is a Reverse Divide, the FMA is loaded into the FMQ; if the instruction is a Reverse Subtract, the FMA is loaded into the FMB during TS02.



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL

FINISH

		DATE
DRN	E Wilson	4/20/71
CHK'D	L.P Hollos	5/14/71
ENG	[illegible]	5/14/71
PROJ. ENG.	[illegible]	5/14/71
PROD.	[illegible]	5/14/71

NEXT HIGHER ASSY.

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE FETCH CYCLE 1

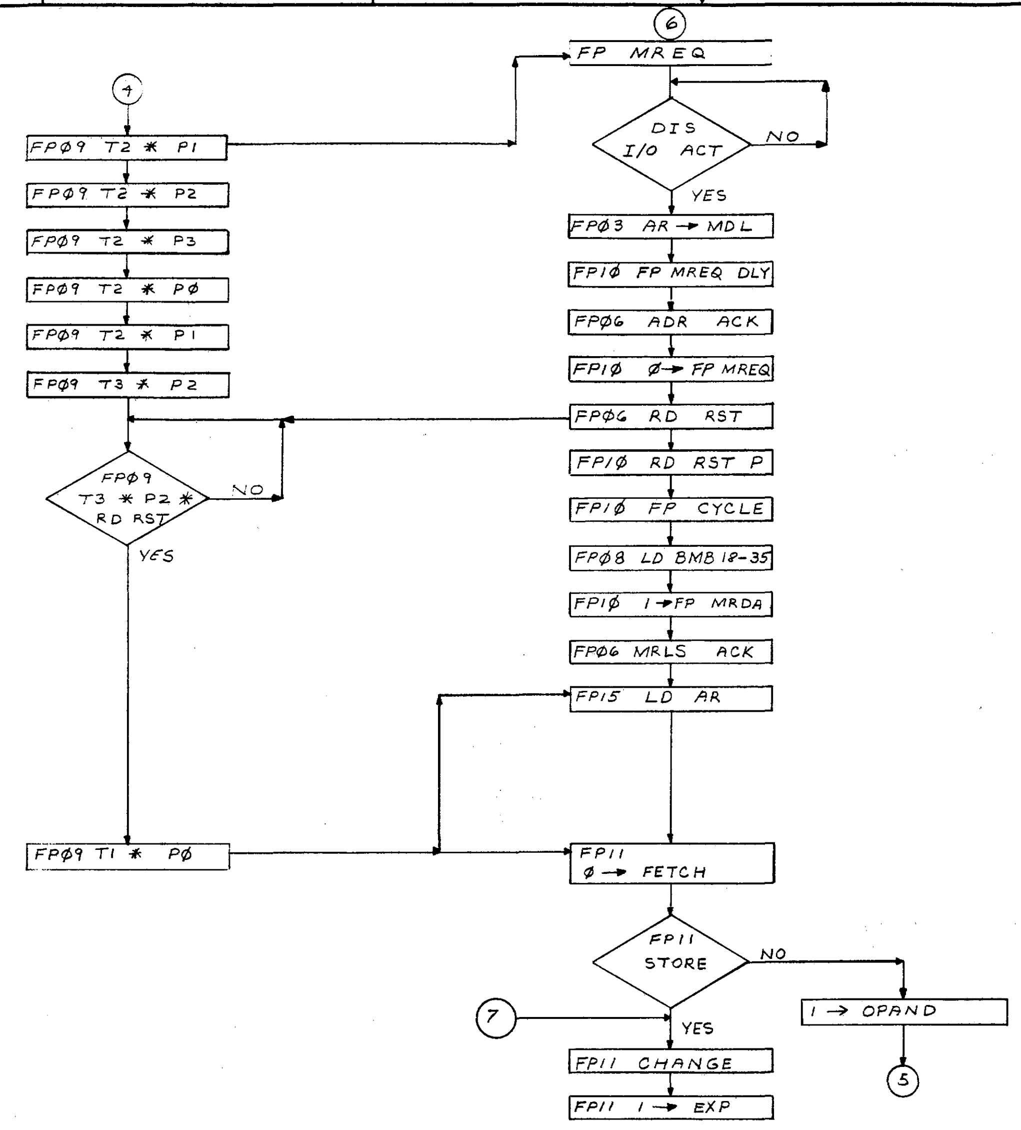
SIZE	CODE	NUMBER	REV
D	FD	FP15-0-45	

DIST.

REVISIONS REV CHANGE NO CHK

DEC FORM NO DRD 102-B

D FD FP15-0-45



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL

FINISH

		DATE
DRN.	E. Wilson	4/20/71
CHK'D.	L. B. Hottos	5/14/71
ENG.	Ed King	5/14/71
PROJ. ENG.	[illegible]	5/14/71
PROD.	[illegible]	5/14/71

NEXT HIGHER ASSY.

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

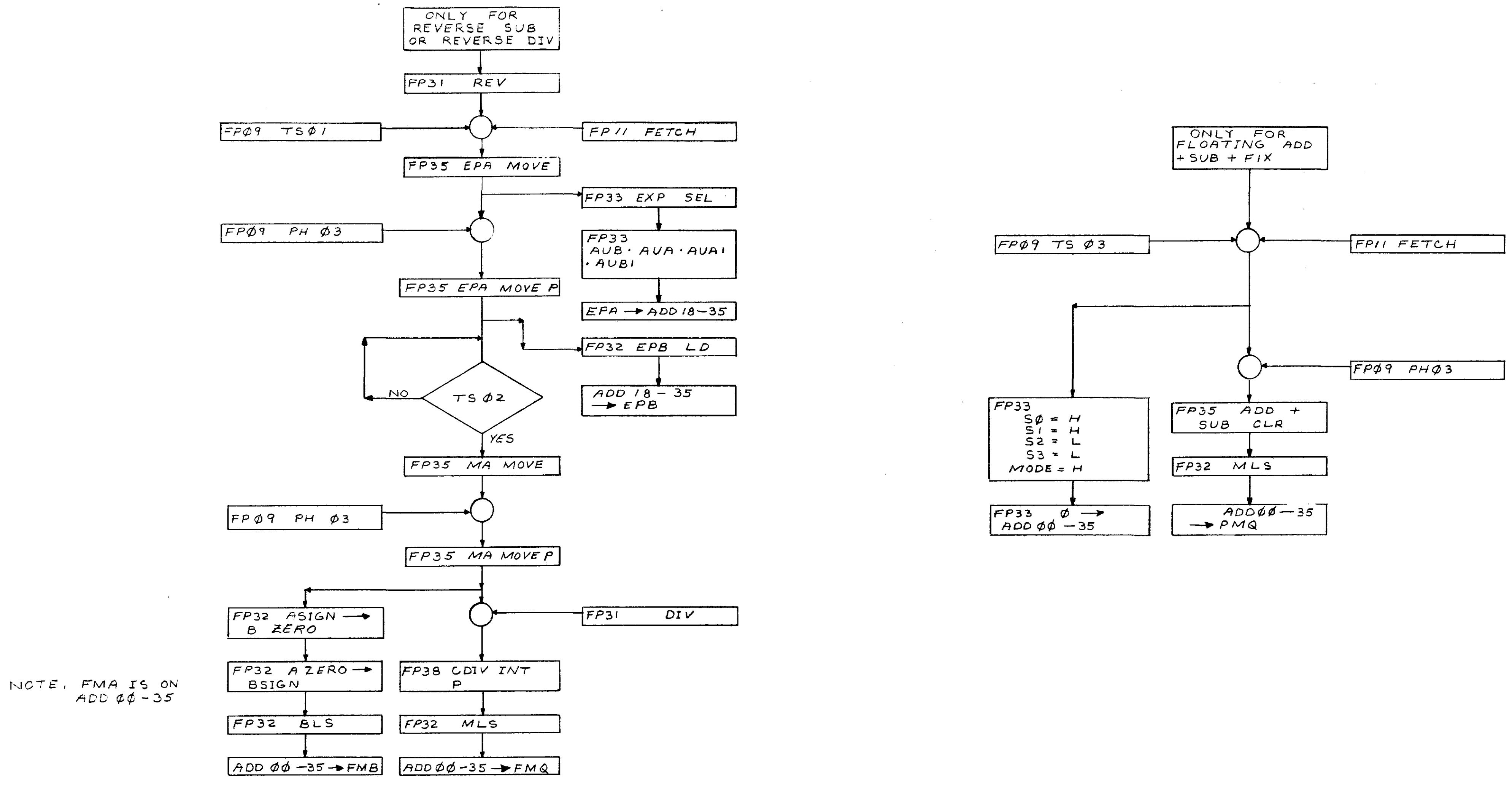
TITLE FETCH CYCLE FLOW 2

SIZE	CODE	NUMBER	REV.
D	FD	FP15-∅-46	

DIST.

REVISIONS CHANGE NO REV CHK

DEC FORM NO DRD 102-B



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES
DECIMALS .XXX = .005 .XX = .02 .X = .1 | ANGLES ±0° 30'
REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓
MATERIAL
FINISH

DRN.	[illegible]	DATE 7/14/71
CHK'D.	[illegible]	DATE 5/14/71
ENG.	[illegible]	DATE 5/14/71
PROJ. ENG.	[illegible]	DATE 5/14/71
PROD.	[illegible]	DATE 5/14/71

NEXT HIGHER ASSY.
SCALE
SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE FETCH CYCLE FLOW 3

SIZE D CODE FD NUMBER FP15-0-47 REV
DIST.

REVISIONS: CHK | CHANGE NO | REV

DEC FORM NO DRD 102-B

3.5 OPAND CYCLE INTERFACE

After the FETCH cycle, the FP15 enters the OPAND cycle. If an instruction is specified in which operands are not fetched from memory (bit 10 of the floating-point instruction word on a 1), the OPAND cycle is omitted completely and no memory reference is made. The current contents of the FMA are used as the operand.

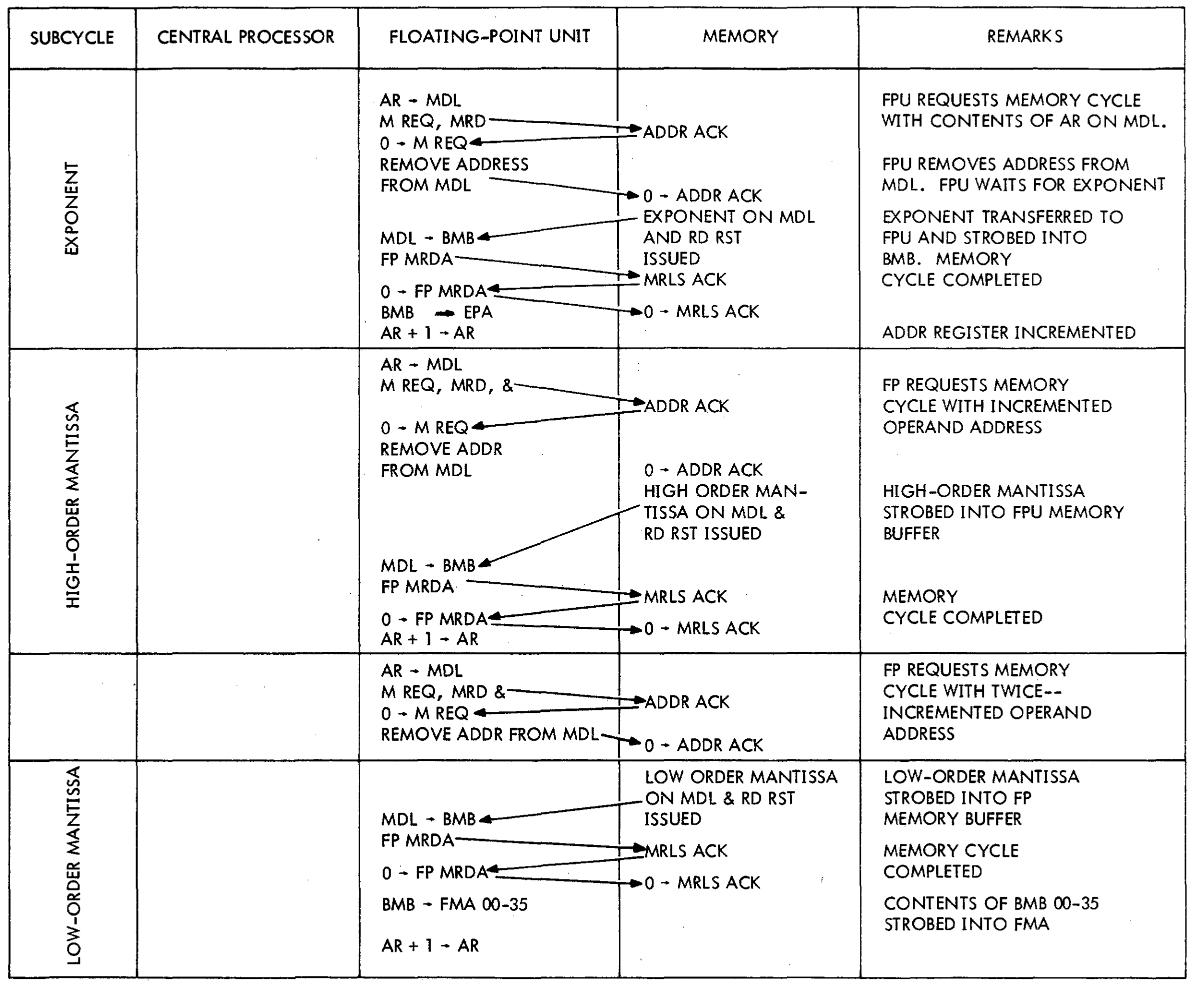
For memory reference instructions, the operand or operands from memory are transferred to the FP15 during the OPAND cycle. The number of operands is dependent on the format specified and is defined in the note associated with the OPAND cycle in Figure 3-4. This description assumes double-precision floating-point format in which the maximum number of operands (three) is transferred from memory. The first operand transferred is the exponent. The FP15 requests a memory cycle (M REQ) and transfers the effective address in the AR to memory via the MDL. Memory then places the first operand (the contents of the address specified) on the MDL and issues RD RST. The FP15 strobes the operand into the BMB and releases memory.

The next operand (high-order mantissa) to be obtained is in the next sequential location (exponent address plus one). As a result, the address in the AR is incremented so that the next memory access will transfer the second operand. The memory cycle is exactly like that described for the exponent operand.

The third operand (low-order mantissa) is in the next sequential location (high-order mantissa plus one). The address in the AR is incremented a second time to obtain the third operand address. The memory cycle is like that described for the exponent operand.

If bit 10 of first word ($71XXXX_8$) is set, the OPAND cycle is omitted completely. If the bit is 00, the OPAND cycle is performed. However, certain operations in the OPAND cycle are excluded based on the following format:

Double-Precision Floating Point	- All Operations Performed
Single-Precision Floating Point	- Omit Low-Order Mantissa
Double-Precision Integer	- Omit Exponent
Single-Precision Integer	- Omit Exponent and Low-Order Mantissa



SUBCYCLE	CENTRAL PROCESSOR	FLOATING-POINT UNIT	MEMORY	REMARKS
EXPONENT		AR → MDL M REQ, MRD → 0 → M REQ ← REMOVE ADDRESS FROM MDL → MDL → BMB ← FP MRDA → 0 → FP MRDA ← BMB → EPA AR + 1 → AR	ADDR ACK 0 → ADDR ACK EXPONENT ON MDL AND RD RST ISSUED MRLS ACK 0 → MRLS ACK	FPU REQUESTS MEMORY CYCLE WITH CONTENTS OF AR ON MDL. FPU REMOVES ADDRESS FROM MDL. FPU WAITS FOR EXPONENT EXPONENT TRANSFERRED TO FPU AND STROBED INTO BMB. MEMORY CYCLE COMPLETED ADDR REGISTER INCREMENTED
HIGH-ORDER MANTISSA		AR → MDL M REQ, MRD, & → 0 → M REQ ← REMOVE ADDR FROM MDL MDL → BMB ← FP MRDA → 0 → FP MRDA ← AR + 1 → AR	ADDR ACK 0 → ADDR ACK HIGH ORDER MANTISSA ON MDL & RD RST ISSUED MRLS ACK 0 → MRLS ACK	FP REQUESTS MEMORY CYCLE WITH INCREMENTED OPERAND ADDRESS HIGH-ORDER MANTISSA STROBED INTO FPU MEMORY BUFFER MEMORY CYCLE COMPLETED
		AR → MDL M REQ, MRD & → 0 → M REQ ← REMOVE ADDR FROM MDL →	ADDR ACK 0 → ADDR ACK	FP REQUESTS MEMORY CYCLE WITH TWICE--INCREMENTED OPERAND ADDRESS
LOW-ORDER MANTISSA		MDL → BMB ← FP MRDA → 0 → FP MRDA ← BMB → FMA 00-35 AR + 1 → AR	LOW ORDER MANTISSA ON MDL & RD RST ISSUED MRLS ACK 0 → MRLS ACK	LOW-ORDER MANTISSA STROBED INTO FP MEMORY BUFFER MEMORY CYCLE COMPLETED CONTENTS OF BMB 00-35 STROBED INTO FMA

Figure 3-4 Memory Interface--OPAND Cycle

3.6 OPAND CYCLE DESCRIPTION

During the OPAND cycle, the FMB and/or EPB is loaded from memory if the instruction specified is an arithmetic instruction (Add, Subtract, Multiply, or Divide). For other types of instructions (including Reverse Subtract and Reverse Divide), the FMA and/or EPA is loaded. For integer format, the EPA is not loaded.

Drawings D-FD-FP15-0-48 through D-FD-FP15-0-50 are flow diagrams of the OPAND cycle. The cycle is initiated when OPAND goes to 1. At TS02*PH01 of this cycle, an FP M REQ is issued.

Since a WRITE operation is inhibited (-ALL WRITE), a memory read will occur. The address of the operand, located in the AR, is gated onto the MDL via the output multiplexer (MPO). FP M REQ, after a delay to allow the MDL to settle, produces M REQ to initiate the memory cycle.

When memory receives the address, it issues ADDR ACK, which clears FP M REQ. The data (operand) and RD RST are then placed on the MDL by the memory. Before strobing the data into its memory buffer, the FP15 waits for FP CYCLE. This signal is delayed by RD RST DLY to allow time for the data to settle before it is strobed. When the data is strobed into the buffer, the FP15 issues FP MRDA and the memory responds with MRLS ACK, which clears FP MRDA to complete the memory cycle.

The data format must now be determined. For each format, the shift counter is loaded with one less than the number of operands to be transferred to the FP15, so that the shift counter will detect a borrow rather than a 0 condition. For example, in double-precision integer format the shift counter is loaded with 1. Transferring the first word to memory decrements the counter to 0; transferring the second word decrements the counter to produce a borrow indicating completion of the transfers.

3.6.1 Double-Precision Floating-Point Format

If double-precision, floating-point format is specified (IR 11 = 1, IR 12 = 1), the shift counter is loaded, during the FETCH cycle, with a count of 2 (SC 16 = 1, SC 17 = 0). A signal designated -STROBE loads the low-order bits (BMB bits 18-35) of the memory buffer with the operand. The A side of the ALU is selected. If the instruction is a Fix, Load, Float, Reverse Subtract, or Reverse Divide, an MA SEL signal is generated that causes the EPA to be loaded. If an arithmetic instruction is specified (Add, Subtract, Multiply, or Divide) MA SEL is not generated and the EPB is loaded.

The shift counter is decremented and, if no borrow is generated, the second memory reference of the OPAND cycle is initiated.

The second memory reference is similar to the first. The address in the AR has been incremented to access the next sequential memory location (high-order mantissa). The shift counter is now at a count of 1 (SC 16 = 0, SC 17 = 1). The STROBE signal loads the high-order bits of the memory buffer (BMB bits 00-17) with the second operand.

The shift counter is decremented a second time to a count of 0. The third memory reference is similar to the second except that the address is again incremented to fetch the third operand (low-order mantissa). The -STROBE signal loads the low-order mantissa into the low-order bits of the BMB. If the instruction is an arithmetic type, the B SIGN/FMB is loaded. If the instruction is a Fix, Load, Float, Reverse Subtract, or Reverse Divide, the A SIGN/FMA is loaded. The A multiplexer is again selected after the fetch of the third operand so that the A SIGN/FMA or B SIGN/FMB can be loaded as a 36-bit word from the 36-bit memory buffer.

The shift counter is decremented and now produces a borrow which indicates that all operands have been received. At this point, the OPAND cycle is cleared and the EXP cycle is enabled.

3.6.2 Single-Precision Floating-Point Format

In single-precision floating-point format (IR 11 = 0, IR 12 = 1), the shift counter is loaded in the FETCH cycle with a count of 1 (SC 16 = 0, SC 17 = 1). A memory reference is made just as for double-precision floating point and the exponent operand is strobed into the low-order bits of the BMB, as a result of -STROBE. The B side of the ALU is selected for the first word of single-precision floating-point format. The first word consists of nine bits of exponent and nine bits of mantissa. The nine bits of exponent are loaded in the EPA or EPB. The value of bit 09 (exponent sign) is extended through bit 00. The nine bits of the mantissa remain stored in bits 18 through 26 of the BMB, since the A SIGN/FMA or B SIGN/FMB are loaded 36 bits at a time. The exponent bits in the BMB are cleared (bits 27 through 35).

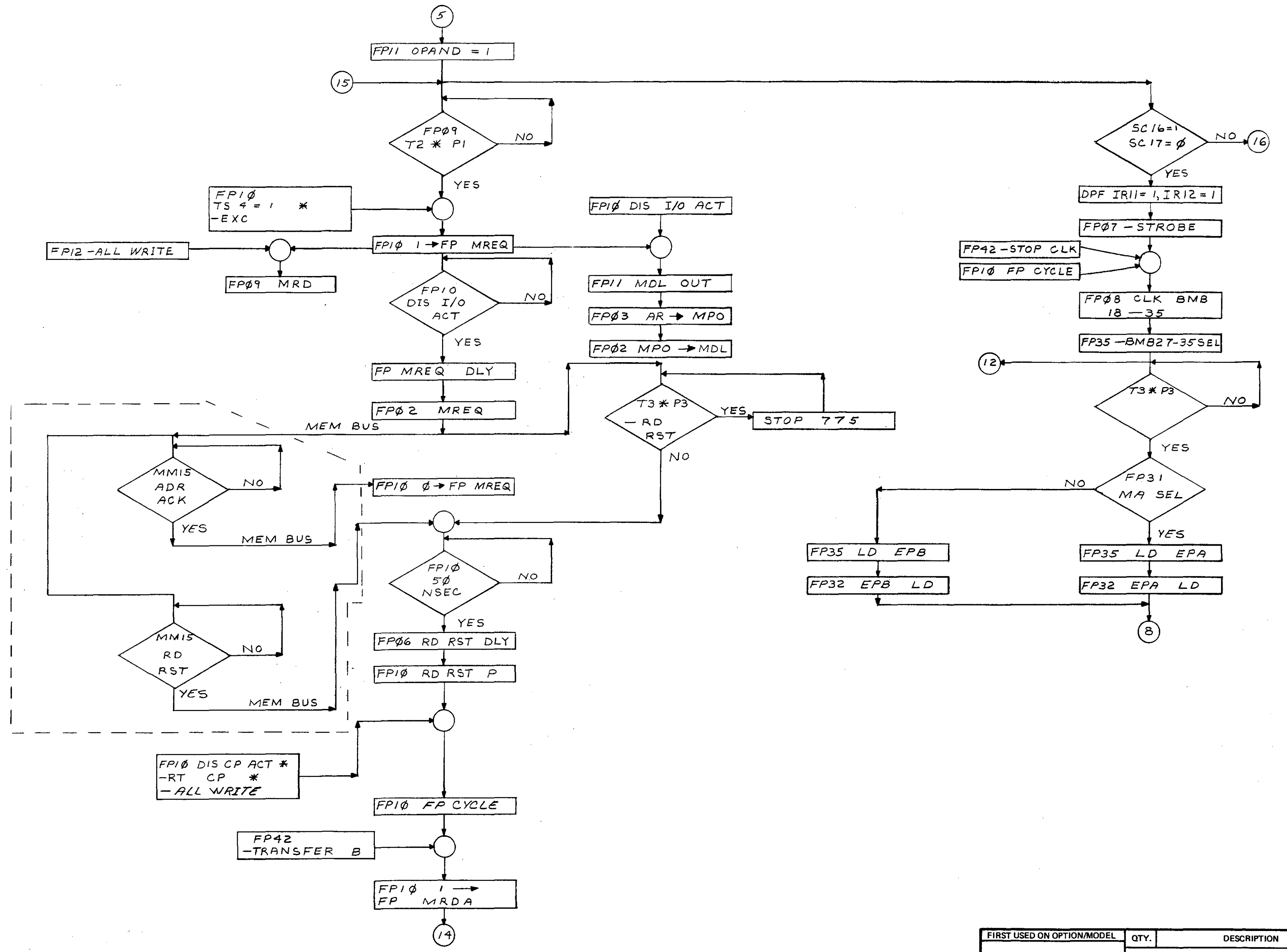
At the end of the memory reference, the shift counter is decremented to 0. Since no borrow is detected, a second memory reference is initiated to fetch the 18 bits of high-order mantissa from memory. The address in the FP15 AR is incremented to access the next sequential memory location. The 18 bits of high-order mantissa are loaded into the high-order bits of the BMB by STROBE. The A side of the ALU is selected and the A SIGN/FMA or B SIGN/FMB is loaded with the 27 bits of mantissa.

At the end of the cycle, the shift counter is decremented and produces a borrow indicating that the operation is complete. The OPAND cycle is cleared and the EXP cycle is enabled.

3.6.3 Extended Integer Format

In extended integer format (IR 11 - 1, IR 12 = 0) the shift counter is loaded with a count of 1 (SC 16 = 0, SC 17 = 1) during the FETCH cycle. The normal memory reference is made, and STROBE causes the 18 bits from memory to be loaded into the high-order bits of the BMB.

This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL

FINISH

		DATE
DRN.	E. Wilson	4/16/71
CHK'D.	[illegible]	5/14/71
ENG.	[illegible]	5/14/71
PROJ. ENG.	[illegible]	5/14/71
PROD.	[illegible]	5/14/71

NEXT HIGHER ASSY.

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE

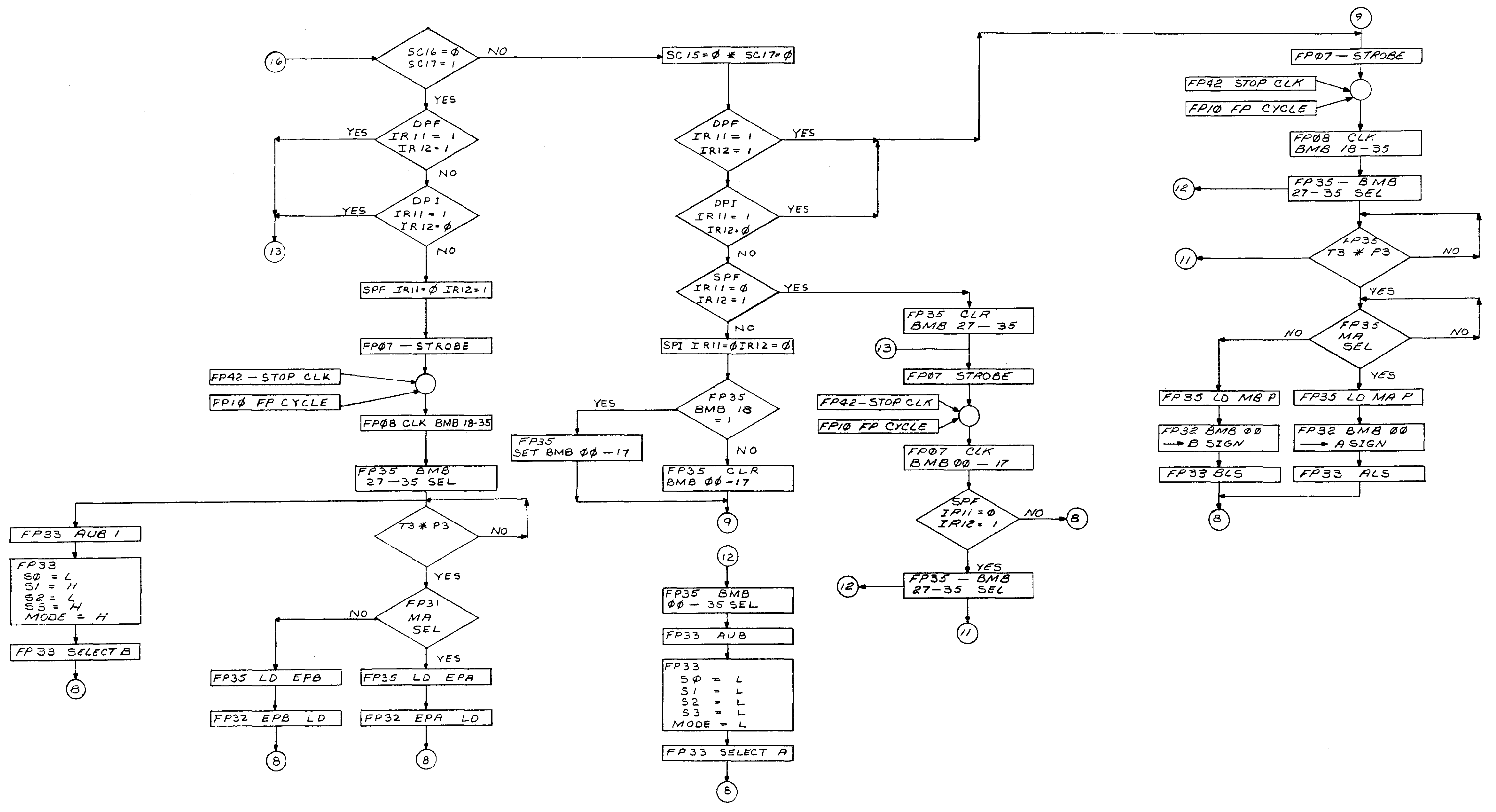
OPAND CYCLE 1

SIZE	CODE	NUMBER	REV
D	FD	FP15-Ø-48	

DIST.

REVISIONS | CHANGE NO | REV | CHK

DEC FORM NO DRD 102-B



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

		DATE
DRN.	E. Wilson	4/16/71
CHK'D	[illegible]	5/14/71
ENG	[illegible]	5/14/71
PROJ ENG	[illegible]	5/14/71
PROD	[illegible]	5/14/71

MATERIAL

FINISH

NEXT HIGHER ASSY.

SCALE

SHEET OF

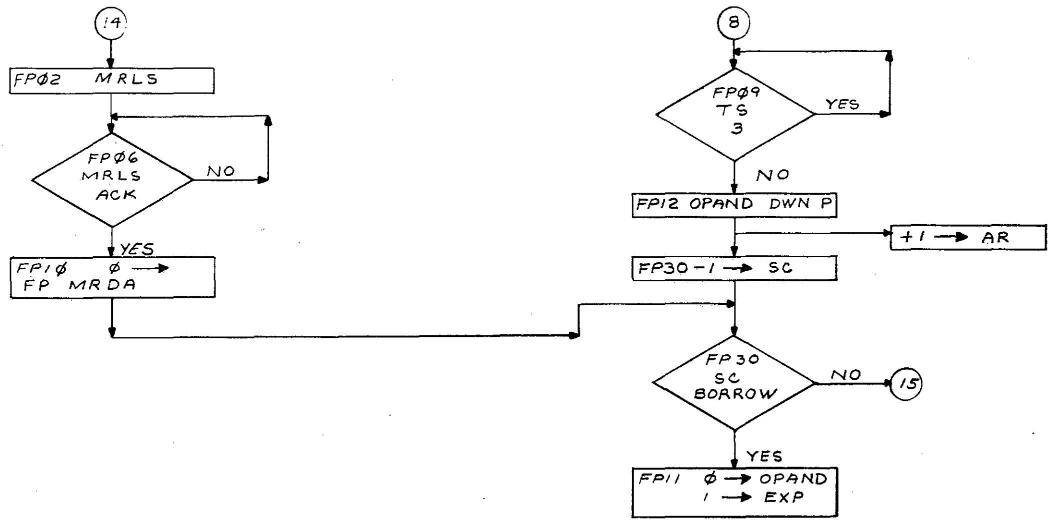
digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE OPAND CYCLE FLOW 2

SIZE	CODE	NUMBER	REV.
D	FD	FP15-Ø-49	

DIST.

DEC FORM NO DRD 102-B



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL

FINISH

DRN. E. Wilson	DATE 4/15/71
CHK'D. L.P. Hottos	DATE 5/14/71
ENG. Ed King	DATE 5/14/71
PROJ. ENG. [illegible]	DATE 5/14/71
PROD. [illegible]	DATE 5/14/71

NEXT HIGHER ASSY.

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE OPAND CYCLE FLOW 3

SIZE	CODE	NUMBER	REV
D	FD	FP15-0-50	

DIST.

REVISIONS REV CHANGE NO CHK

DEC FORM NO DRD 102-B

After completing the transfer, the shift counter is decremented to 0, no borrow is detected, the AR is incremented, and a second memory reference is initiated.

The second memory reference causes the 18 bits in the next sequential memory location to be loaded into the low-order bits of the BMB by -STROBE. The A side of the ALU is selected and the FMA or FMB is loaded with the 35-bit integer.

After the transfer of the second word, the shift counter is decremented from 0 to a borrow condition. The OPAND cycle is cleared and the EXP cycle is enabled.

3.6.4 Single-Precision Integer Format

In singl-precision integer format (IR 11 = 0, IR 12 = 0) only one memory reference is made. The shift counter is loaded in the FETCH cycle with a count of 0. A memory reference is performed to obtain the operand. The operand is loaded into the low-order bits of BMB 18-35. The value of bit 18 (sign bit) is entered through bit 00. The A side of the ALU is selected and the A SIGN/FMA or B SIGN/FMB is loaded.

At the end of the cycle, the shift counter is decremented from 0 to produce a borrow that clears the OPAND cycle and enables the EXP cycle.

3.7 WRITE CYCLE

If a Store instruction is specified, the WRITE cycle is initiated. During the WRITE cycle the contents of the desired major registers are written into memory. Drawing D-FD-FP15-0-51 is a flow diagram of the WRITE cycle. At TS02*PH03 of the NOR cycle, the shift counter is loaded with one less than the number of words to be transferred to memory.

The FP15 places the contents of the AR on the MDL and issues a delayed FP M REQ that allows for settling time. The AR contains the address where the first operand is to be stored. Memory receives the address on the MDL and issues ADR ACK indicating receipt of the address. This signal also clears FP M REQ and enables the data to be placed on the MDL.

The particular word (depending on the count in the shift counter) is strobed on the MDL. FP MRDA is delayed by ADDR ACK to allow address settling. The operand is strobed into memory by FP MRDA. Memory responds with MRLS ACK that clears FP MRDA to complete the cycle.

The number of memory references during the WRITE cycle depends on the instruction and/or data format. When the shift counter produces a borrow, the WRITE cycle is terminated. BUSY and DIS CP ACT signals are cleared and control is returned to the CPU.

The various types of store instructions are described below:

3.7.1 Store JEA

If the instruction is Store JEA, the contents of the JEA are transferred to the output multiplexer (MPO) and then to the MDL.

3.7.2 Double-Precision Floating Point

In double-precision floating-point format, the shift counter is loaded with a count of 2. The first word (contents of EPA register) is transferred to the output of the multiplexer. When the shift counter is decremented to 1, the second word (high-order mantissa ADD 00-17) is transferred to the output of the multiplexer. When the shift counter is decremented to 0, the third word (low-order mantissa ADD 18-35) is transferred to the output of the multiplexer.

3.7.3 Single-Precision Floating Point

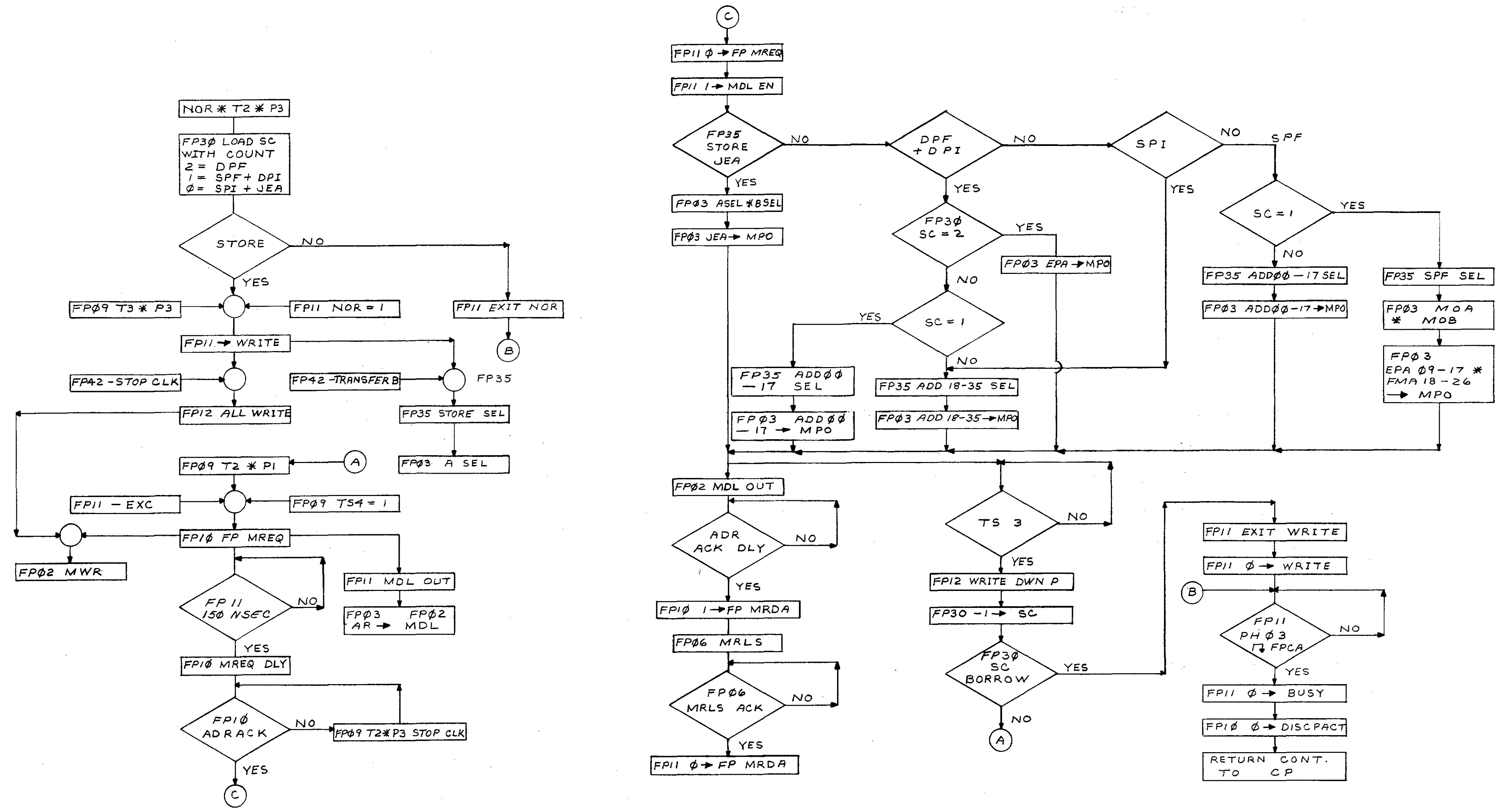
In single-precision floating-point format, the shift counter is 1; EPA bits 09 through 17 and FMA bits 18 through 26 are transferred to the output of the multiplexer. When the shift counter goes to 0, FMA bits 00 through 17 are transferred to the output of the multiplexer.

3.7.4 Extended Integer

The shift counter is loaded with a count of 1 for this format. When the shift counter is 1, the high-order bits (ADD 00-17) are transferred to the output of the multiplexer and, if the shift counter is 0, the low-order bits (ADD 18-35) are transferred to the output of the multiplexer. When an Integer Store instruction is specified, positive or negative integers are transferred from the FMA to the FMB and are 1's complemented during FUN*TS02. At NOR*TS01, the FMB is incremented so the contents of the FMB are now a 2's complement representation of the integer in the FMA. During the WRITE cycle, the sign (A SIGN) of the FMA is examined. If the sign is positive, the integer is a positive integer and the contents of the FMA are stored in memory. If A SIGN is negative, however, the contents of the FMB are stored in memory, since the FMB is the 2's complement of the FMA and negative integers are 2's complemented before being transferred to memory.

3.7.5 Single-Precision Integer

When a single-precision integer Store instruction is specified, the contents of the FMA are transferred to the FMB and 1's complemented during FUN*TS02. At NOR*TS01, the FMB is incremented and now



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

		DATE
DRN.	E. Wilson	4/15/71
CHK'D	L. P. Hottos	5/14/71
ENG.	[illegible]	5/14/71
PROJ. ENG.	[illegible]	5/14/71
PROD.	[illegible]	5/14/71

MATERIAL

FINISH

NEXT HIGHER ASSY.

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE WRITE CYCLE FLOW

SIZE	CODE	NUMBER	REV.
D	FD	FP15-Ø-51	

DIST.

DEC FORM NO DRD 102-B

represents the 2's complement of the FMA. During the WRITE cycle, A SIGN is examined. If it is positive, the contents of the FMA are stored in memory; if it is negative, the 2's complement of the negative integer are stored in memory. This 2's complement is contained in the FMB and, consequently, the contents of the FMB are stored in memory.

3.8 INTERRUPT CYCLE INTERFACE

The following conditions in the FP15 can cause an interrupt in the CPU.

a. Overflow
b. Underflow
c. Abnormal division (divide by zero)
d. Memory violations (trap)

An interrupt generated as a result of an overflow or underflow condition can occur during the FUN cycle, where the arithmetic operation is being performed, or during the NOR cycle, where the result of an arithmetic operation is being normalized. An abnormal divide interrupt can occur only during the FUN cycle; a memory violation interrupt can occur during the FETCH, OPAND, or WRITE cycles. If an interrupt should occur while an FP15/CPU cycle is in progress, the cycle is completed, the remaining sequence is aborted, and INT 1 and INT 2 interrupt cycles are initiated.

If an interrupt caused by a memory violation occurs in the OPAND cycle while the exponent is being fetched, this part of the sequence is completed, fetching of the high-order and low-order mantissas is aborted, and the interrupt occurs. If the interrupt occurs during fetching of the high-order mantissa, The FP15 completes this part of the cycle and aborts fetching of the low-order mantissa.

3.8.1 INT 1 Cycle

When a floating-point interrupt is raised, the FP15 forces a JMS*0 to the CPU by placing 120000_8 on the MDL. Figure 3-5 shows the communication between the CPU and FP15. It is assumed that a memory violation interrupt occurred during the fetching of the high-order mantissa. When the high-order mantissa has been fetched, the OPAND cycle is aborted and a dummy setup initiated. The FP15 removes DIS CP ACT and the CPU is allowed to make a memory request. DIS RD RST is raised and the FP15 completes the memory cycle. The FP15 then removes DIS RD RST, places 120000_8 (JMS*0) on the MDL, and issues FP RD RST. The 120000_8 is strobed into the MI in the CPU and then executed. The remainder of the cycle between the FP15 and CPU is completed.

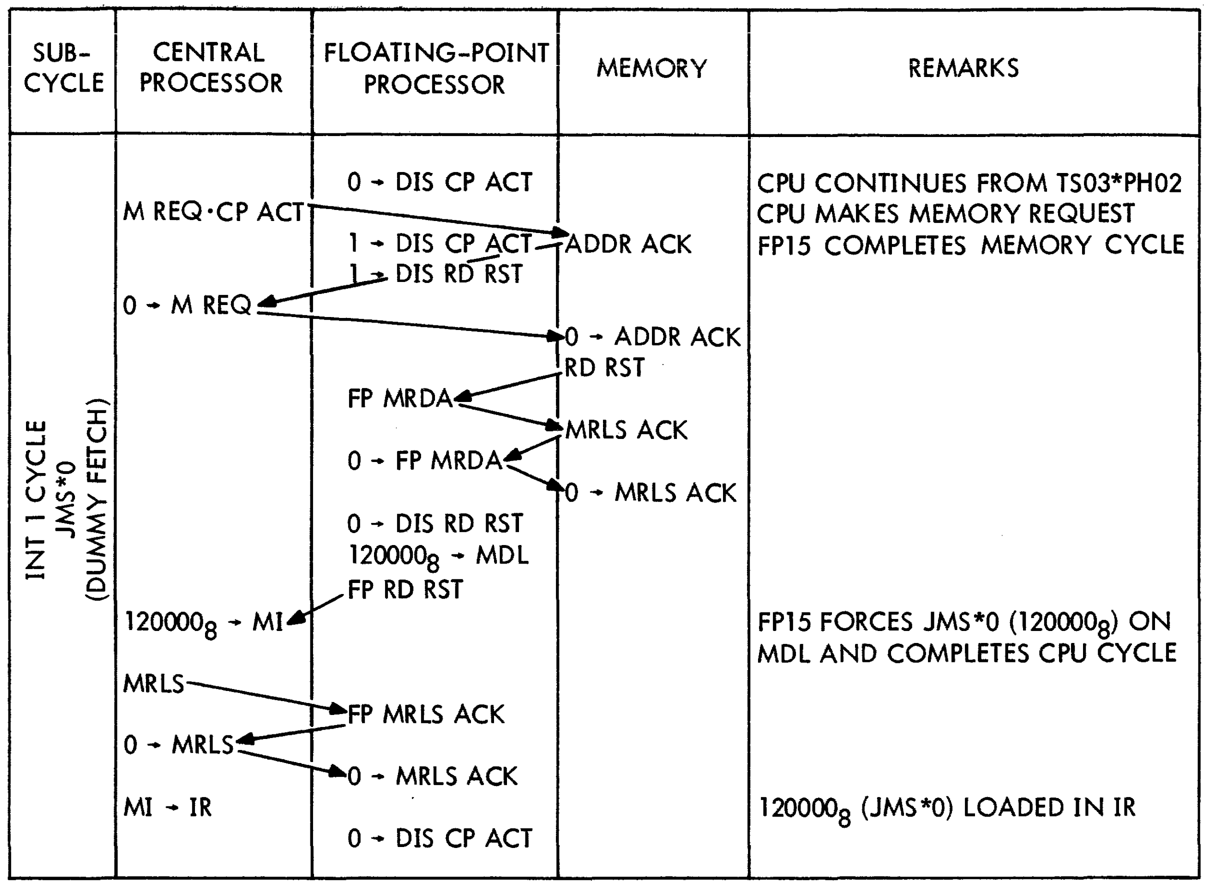


Figure 3-5 INT 1 Cycle Interface Diagram

3.8.2 INT 2 Cycle

The FP15 initiates a second dummy setup that forces the CPU to accept the JEA (JMS Exit Address) instead of the contents of location 0 (see Figure 3-6). The JEA address is under programmer control and will vary depending on the cause of the interrupt.

```
EXIT ADDRESS +0     0
             +1     JMP OVR     /GO TO OVERFLOW
             +2     0
             +3     JMP UND     /GO TO UNDERFLOW
             +4     0
             +5     JMP DIV     /GO TO DIVIDE
             +6     0
             +7     JMP TRAP    /GO TO MEMORY VIOLATION
```

In the example presented, where a memory violation caused the interrupt, the JEA address +6 will contain the address of the PC ($71XXXX_8$ instruction) +3 when the JMS is complete. JEA +7 may contain a jump instruction to an entry of a service routine associated with the interrupt.

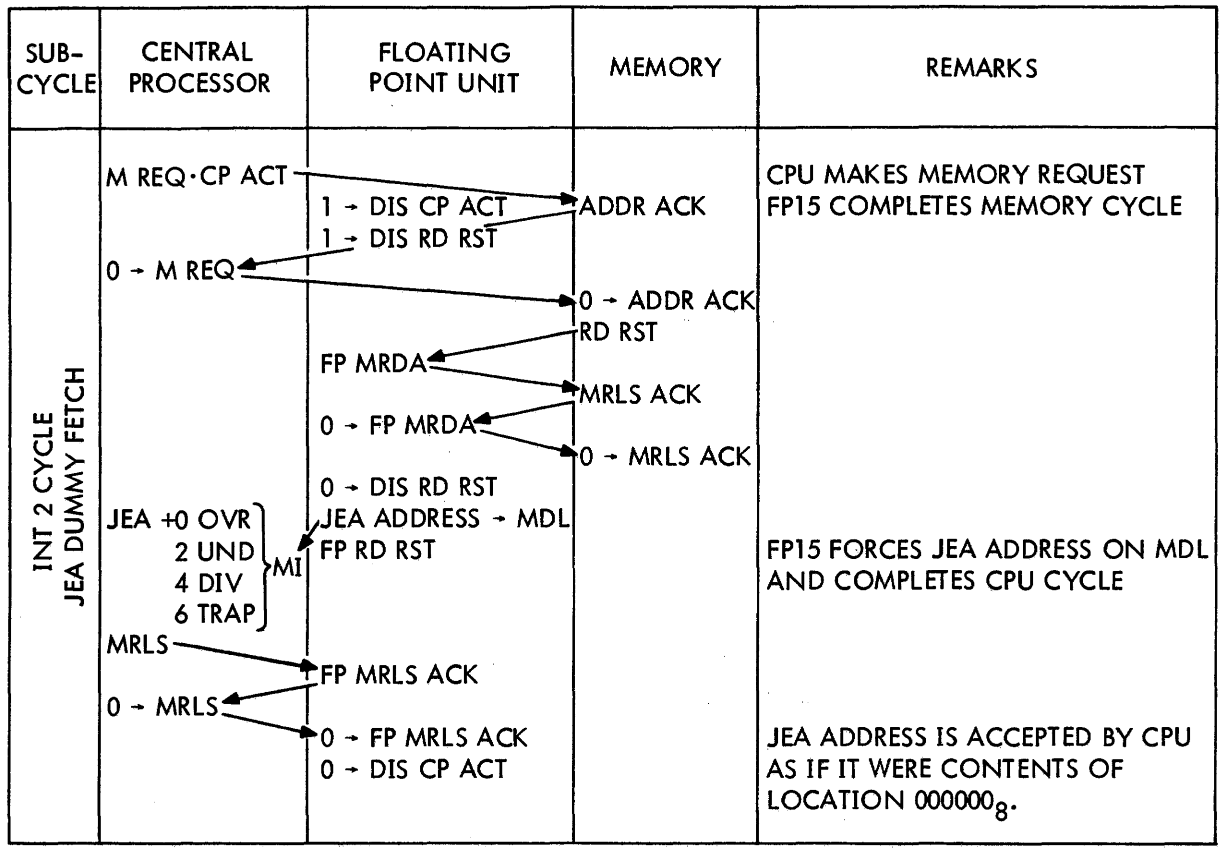
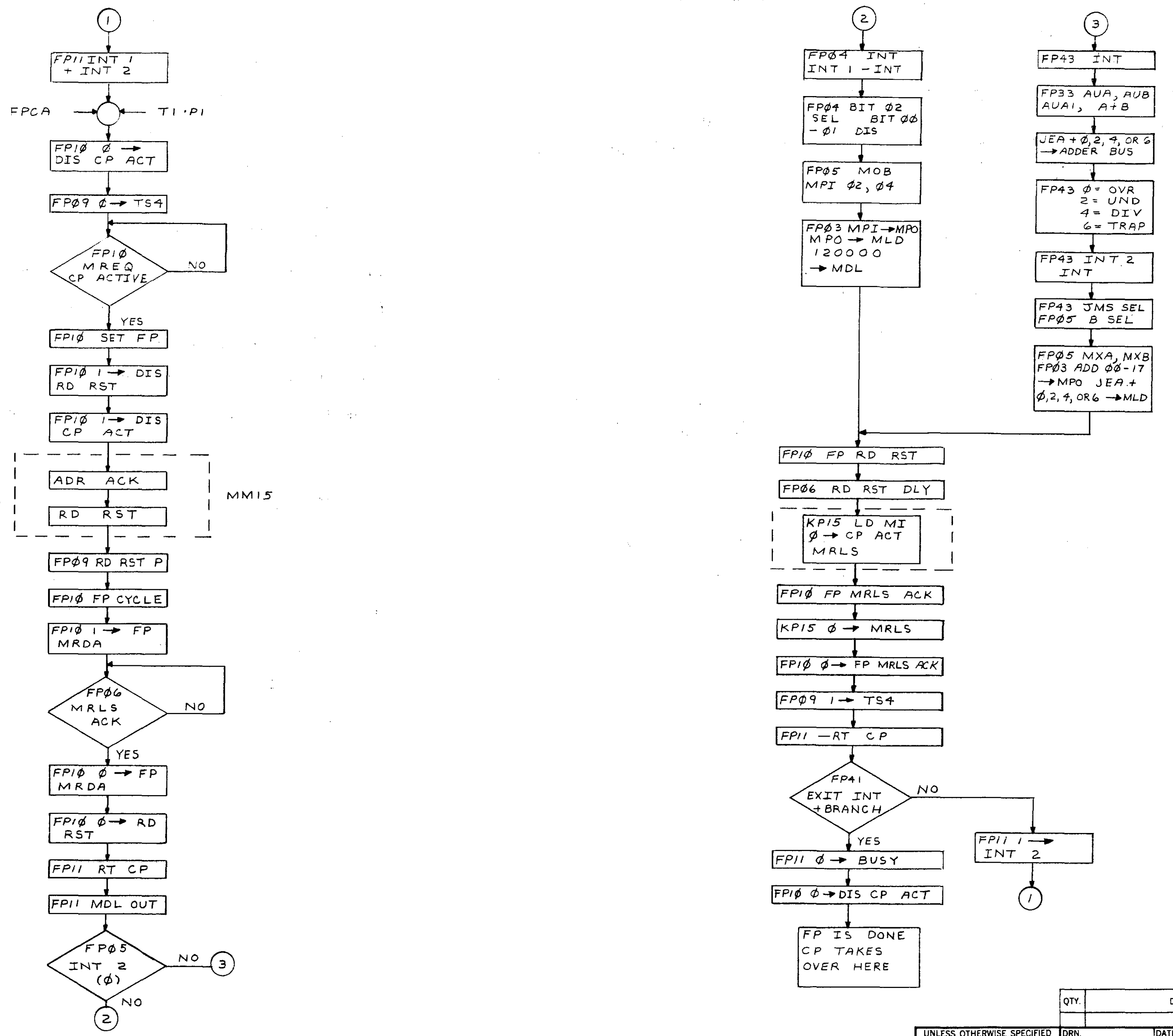


Figure 3-6 INT 2 Cycle Interface Diagram

3.9 INTERRUPT CYCLE

On entering INT 1, DIS CP ACT is removed; this allows the CPU to continue (see drawing D-FD-FP15-0-62). When CP Active is clocked high, and a M REQ is made by the CPU to obtain the next instruction, the FP15 is set up to take control over memory. In addition, DIS RD RST is raised to inhibit communication between the CPU and memory, and DIS CP ACT is raised to temporarily suspend the CPU. Memory responds to the CPU M REQ with ADDR ACK, places the contents of the specified address on the MDL, and issues RD RST. The CPU never sees the contents of the address because of DIS RD RST. The FP15 issues MRDA and the memory responds with MRLS ACK to complete the cycle. Control is returned to the CPU. The FP15 then initiates a dummy setup that places 120000_8 on the MDL via the input multiplexer (MPI) and output multiplexer (MPO). FP RD RST is also placed on the MDL. At this point, the FP15 simulates memory and communicates with the CPU to complete the cycle. The 120000_8 is loaded into the MI register in the CPU. When the CPU receives the 120000_8, it issues MRLS. The FP15 responds with FP MRLS ACK, both are then cleared and the INT 2 cycle is initiated.

The INT 2 cycle is similar to INT 1 except that the JEA address, instead of 120000_8, is placed on the MDL, the CPU executes the 120000_8 as a JMS*0 and makes a second M REQ. The FP15 again suspends the CPU with DIS RD RST, gains control of memory, and completes the memory cycle. The JEA address is placed on the MDL along with FP RD RST. At this point, the FP15 releases control to the CPU and simulates a memory so the CPU can load the JEA address into the MI register. The CPU can now complete its cycle which was initially suspended by the FP15. The action is concluded by BUSY and DIS CP ACT being cleared, thereby returning control to the CPU.



QTY.	DESCRIPTION	PART NO.	ITEM NO.

PARTS LIST

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

MATERIAL

FINISH

DRN.	E. Wilson	DATE 3/31/71
CHK'D.	L P Hottos	DATE 5/14/71
ENG.	Ed King	DATE 5/14/71
PROJ. ENG.	[illegible]	DATE 5/14/71
PROD.	[illegible]	DATE 5/14/71

FIRST USED ON

SCALE
SHEET OF

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

TITLE INTERRUPT FLOW

SIZE D CODE FD NUMBER FP15 - Ø - 62 REV.

DIST.

REVISIONS CHANGE NO REV CHK

DEC FORM NO DRD 102A

3.10 FP15/CPU CONTROL

As an aid in understanding the exchange of control between the CPU and the FP15, Figure 3-7 shows a typical program describing what instructions the CPU would see and what instructions the FP15 would see if the program were executed. The first instruction (DAC 500) is recognized by the CPU and the contents of the accumulator are deposited in location 000500. The second instruction is a floating-point instruction that is recognized by both the CPU and FP15. The next three sequential locations (000110, 000111, and 000112) are recognized by the FP15. The FP15 takes control and forces a 710000 NOP on the MDL so that the CPU does not use the floating-point operand address as an instruction. Consequently the CPU waits, since the FP15 has control of memory. When the FP15 completes the instruction, both the CPU and FP15 again monitor the next instruction fetched from core. A similar process can be traced through the remaining steps in the program.

```
DLD = Double Precision Floating Point Load
DAD = Double Precision Floating Point Add
DST = Double Precision Floating Point Store

000097    040500    DAC 500                         /CPU INSTRUCTION
000100    713150    DLD 110                         /FPU INSTRUCTION
000101    000110                                    /OPERAND ADDRESS
000102    716140    DAD 113                         /FPU INSTRUCTION
000103    000113                                    /OPERAND ADDRESS
000104    713750    DST 116                         /FPU INSTRUCTION
000105    000116                                    /SUM STORED
000106    200130    LAC 130                         /CPU INSTRUCTION
000107    740040    HLT                             /CPU INSTRUCTION
000110    XXXXXX    EXP A          }
000111    XXXXXX    HIGH MANTISSA  } AUGEND
000112    XXXXXX    LOW MANTISSA   }
000113    XXXXXX    EXP B          }
000114    XXXXXX    HIGH MANTISSA  } ADDEND
000115    XXXXXX    LOW MANTISSA   }
000116    XXXXXX    EXPONENT       }
000117    XXXXXX    HIGH MANTISSA  } SUM
000120    XXXXXX    LOW MANTISSA   }
000130    777777
000500    000000

CP SEES             FPU SEES

040500                                              /CPU INSTRUCTION
XXXXXX                                              /CPU WRITES INTO LOC 500
713150              713150                          /FPU INSTRUCTION
710000              000110                          /FPU FORCES 710000 TO CP
  :                 XXXXXX                          /FPU SEES CONTENTS
CP WAITS            XXXXXX                          /OF 000110, 000111 & 000112
  :                 XXXXXX
716140              716140                          /FPU INSTRUCTION
710000              000113                          /FPU FORCES 710000 TO CP
  :                 XXXXXX                          /FPU SEES CONTENTS
CP WAITS            XXXXXX                          /OF 000113, 000114 & 000115
  :                 XXXXXX
713750              713750                          /FPU INSTRUCTION
710000              000116                          /FPU FORCES 710000 TO CP
  :                 XXXXXX                          /FPU WRITES INTO LOC.
CPU WAITS           XXXXXX                          /000116, 000117 & 000120
  :                 XXXXXX
200130                                              /CPU INSTRUCTION
777777                                              /CONTENTS OF 000130
740040                                              /PROGRAM HALTS
```

Figure 3-7 CPU/FP15 Sample Program

CHAPTER 4
INSTRUCTION SET

4.1 INTRODUCTION

The following paragraphs describe the classes of instruction used in the FP15. Several functions are applicable to many classes; these will be described first. The flow diagrams of the instructions specify where these functions occur, if applicable. These functions include: converting negative integers to sign and magnitude format, normalizing, and rounding.

4.2 CONVERTING NEGATIVE INTEGERS TO SIGN AND MAGNITUDE FORMAT

When a 2's complement negative integer is loaded into the FMA during the OPAND cycle, it is converted to sign and magnitude format during the EXP cycle. Two's complement positive integers are already in sign and magnitude format and require no conversion. If the instruction requires no memory reference, the number in the FMA is in sign and magnitude format. Two's complementing the number again is undesirable, since it would convert the sign and magnitude number back to a 2's complement number.

For FMA conversion during TS01 of the EXP cycle, FMA is complemented as a result of COMP MA (see Figure 4-1). This signal takes the 1's complement of the integer in the FMA and puts it on the ALU bus. During PH03*TS01, the number on the ALU bus is strobed back into the FMA.

When the FP15 sequences to TS02 of the EXP cycle, INCA is generated; this puts the contents of the FMA plus one on the ALU bus. During PH03*TS02, INCA-P is generated, and the output of the ALU bus is strobed back to the FMA. The number now in the FMA is the 2's complement of the number initially contained there and is a negative number in sign and magnitude format.

For FMB conversion, during TS01 of the EXP cycle, the FMB is complemented as a result of COMP MB. This signal takes the 1's complement of the integer in the FMB and puts it on the ALU bus. In PH03*TS01, COMP MB P is generated which strobes the 1's complement integer back into the FMB.

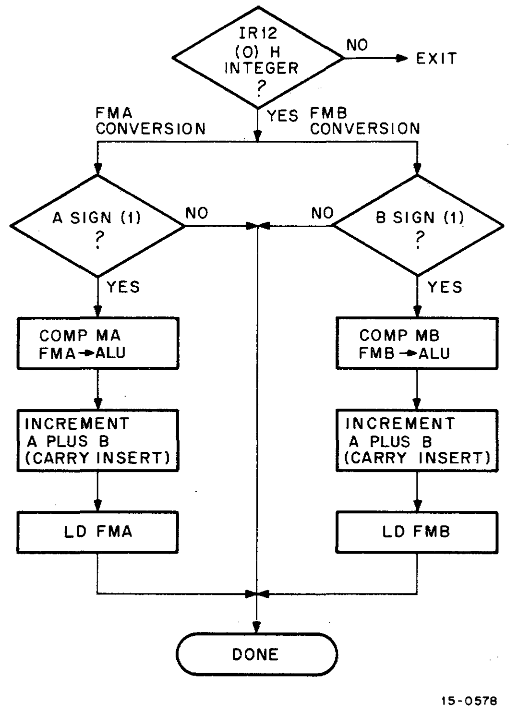


Figure 4-1 Converting Negative Integers to Sign and Magnitude

In EXP*TS02, INCB is generated which puts the contents of the FMB plus one on the ALU bus. This number is strobed back to the FMB during PH03*TS02. The number now in the FMB is the 2's complement of the number originally contained there and is a negative number in sign and magnitude format.

4.3 NORMALIZE

Normalizing a mantissa in the FMA consists of left-shifting the FMA until the most significant bit is a 1, which eliminates all leading zeros. For every left-shift of the FMA, the EPA is decremented. If the specified instruction is a Store or Divide, and normalizing is requested, the mantissa is normalized during FUN*TS01. Otherwise, the mantissa is normalized in NOR*TS01.

4.3.1 Normalization (Except Store, Divide, or Reverse Divide)

If the specified instruction is not a Store or Divide type instruction, and normalizing is requested, the normalizing process occurs in NOR*TS01. Prior to this time, the shift counter is loaded with 42_8 (at FUN*TS03*PH03 Time). The NOR SEL signal sets up the conditions for the NORM P pulses that actually cause the normalizing. For each NORM P pulse, the FMA is shifted left and the EPA and shift counters are decremented. If the instruction specified is not a Multiply, zeros are shifted into the least significant positions of the FMA. If a Multiply instruction is specified, the NORM P pulses shift the FMQ left as well as the FMA. As a result, FMQ 01 is shifted into FMA 35 and 0 is shifted into FMQ 35.

When FMA 01 goes to 1 (NORM DONE), or when the shift counter produces a borrow (SC BORROW), normalizing is terminated and the logic on FP09 is reset to allow the phases and time states to continue. A borrow indicates that normalization is not possible because the number is 0. Refer to Drawings D-FD-FP15-0-58 and D-FD-FP15-0-59 for a detailed flow of normalize.

4.3.2 Store, Divide, or Reverse Divide

When a Store or Divide instruction is specified, and normalizing is requested, a NOR SEL signal (FP40) is generated that enables NORM P to left-shift the FMA and to decrement the EPA for each left-shift (refer to Drawing D-FD-FP15-0-57). The FP15 sequences to PH03*TS01 of the FUN cycle and remains "stopped" in this state until normalizing is completed.

Before generating NORM P, the shift counter is loaded with octal 43 (35_{10}) if the specified instruction is a Divide or Reverse Divide, and is loaded with 42_8 (34_{10}) if the specified instruction is a Store.

For each NORM P pulse, the FMA is shifted left and both the EPA and shift counter are decremented. Zeros are shifted into the least significant positions of the FMA. Normalizing is complete when FMA 01 goes to a 1 (NORM DONE), or when the shift counter produces a borrow (SC BORROW). In either case, the logic on FP09 is "reset" and the phases and time states are allowed to continue.

4.4 ROUNDING

The FP15 can specify rounded or unrounded arithmetic by IR14 of the instruction word.

During alignment of the mantissas in floating-point addition, either the FMA or FMB (depending on which has the smaller exponent) is shifted right. Bits shifted out of either register are shifted into the FMQ. If rounding is requested, and FMQ 01 is a 1, +1 is added to the least significant bit of the FMA or FMB, whichever was being shifted.

A second round can occur during floating-point addition if the addition produced a carry out of the ALU (see Figure 4-2). When this occurs, the FMA is right-shifted and the EPA is incremented, putting the correct number back into the FMA. The bit shifted out of the least significant bit of the FMA is shifted into a guard bit and, if rounding is requested, +1 is added to the least significant bit of the FMA.

The following example shows two numbers being added resulting in a carry. The EPA is incremented and the FMA right-shifted. Since the least significant bit of the FMA is a 1, the guard bit is set. When rounding is requested, +1 is added to the least significant bit of the FMA.

Example:

$.101_2 + .110_2 = ?$ Three-bit registers assumed for simplicity.

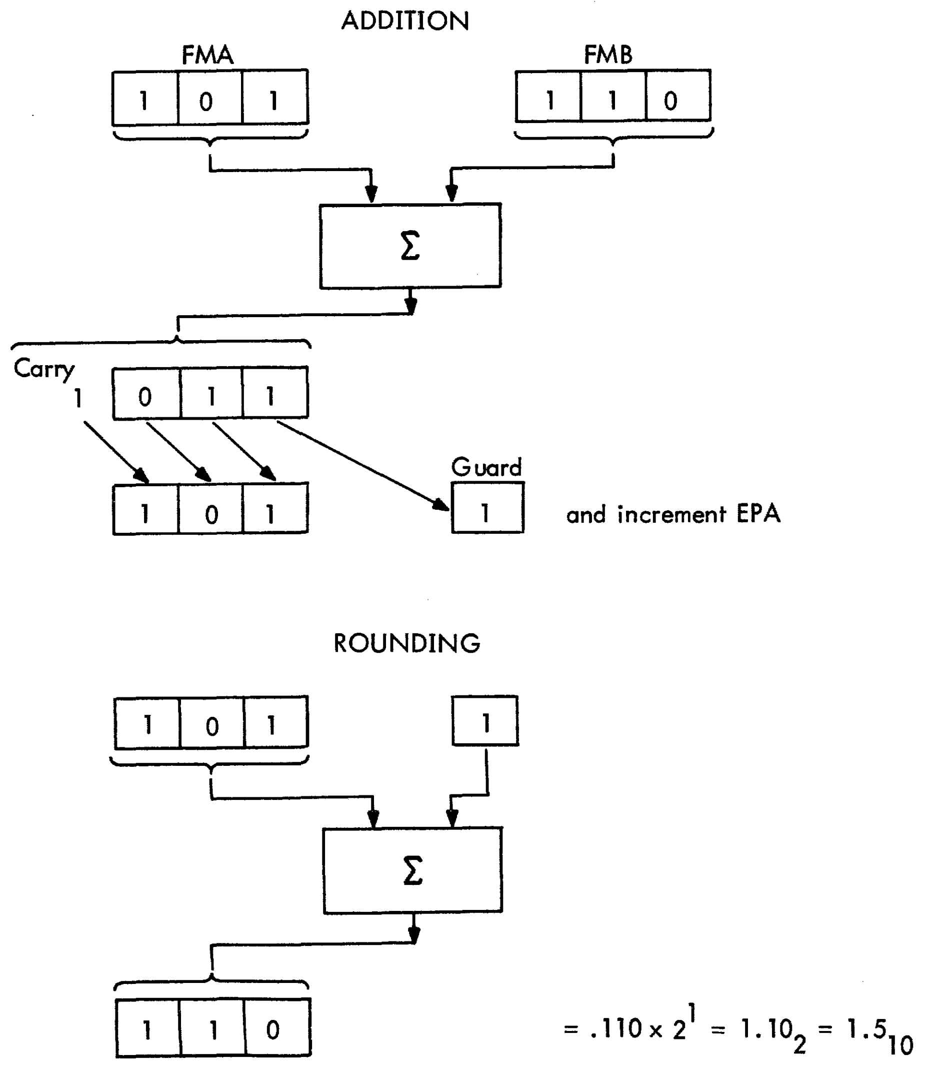
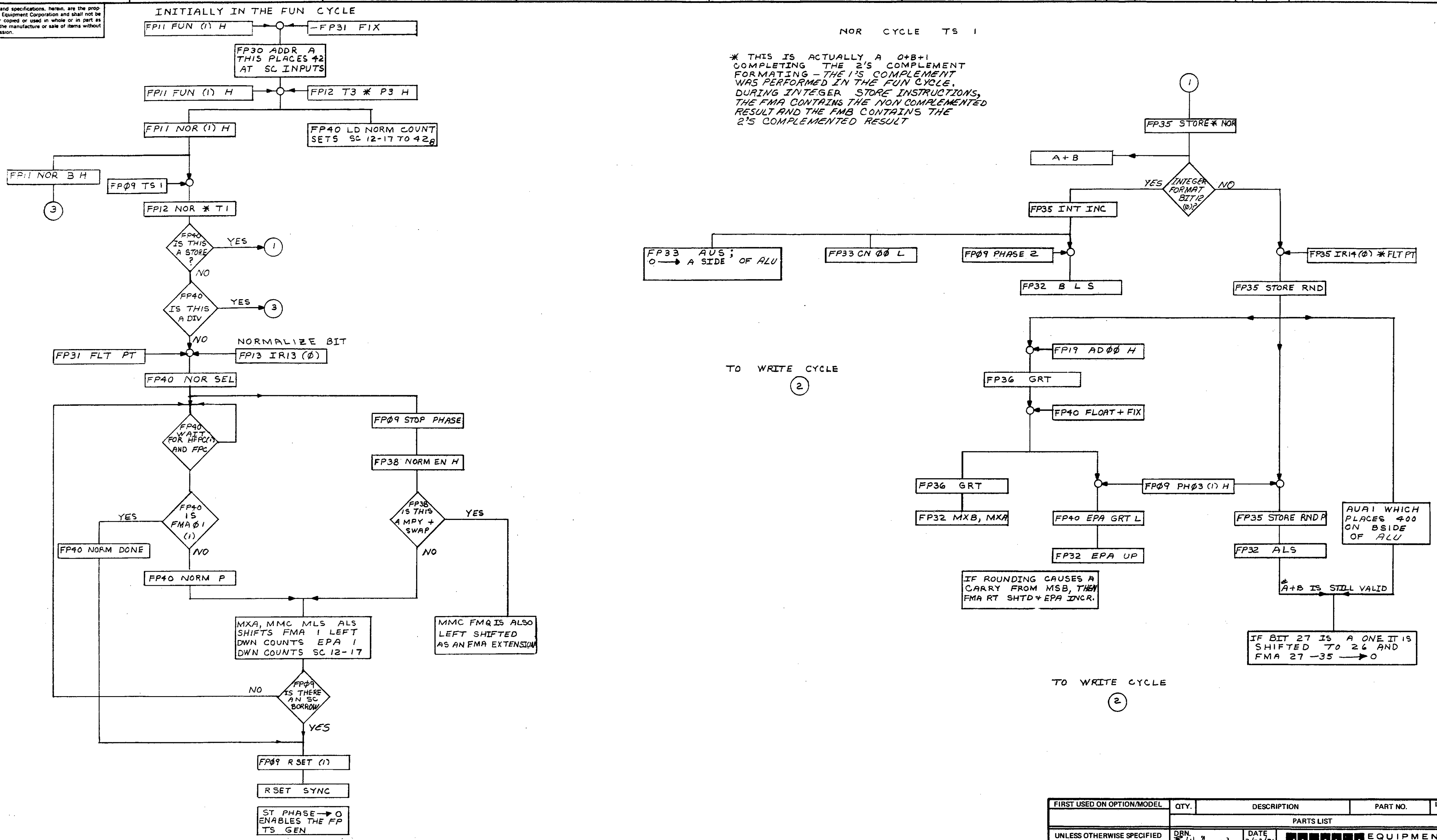


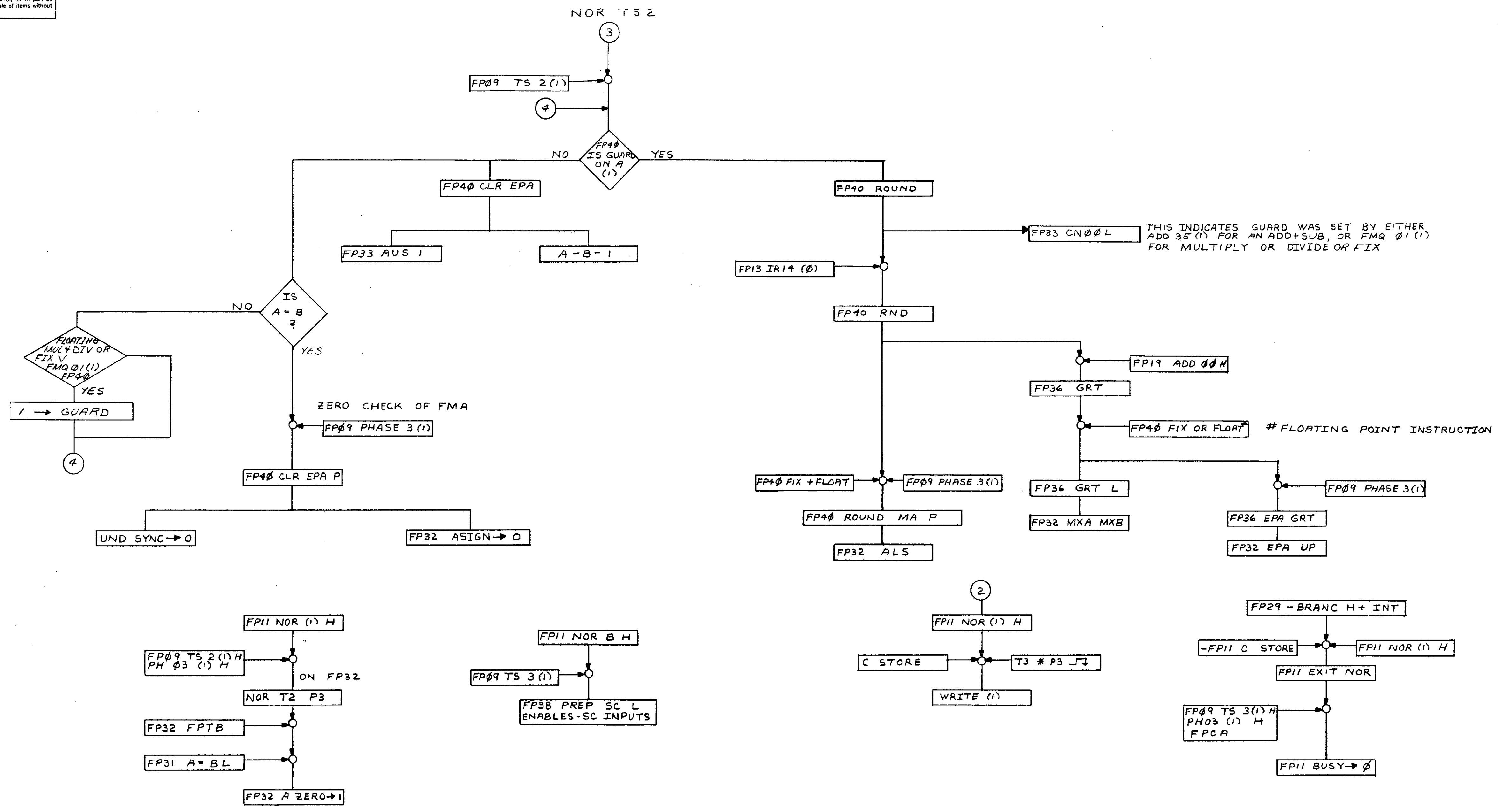
Figure 4-2 Guard Bit and Rounding


INITIALLY IN THE FUN CYCLE
FP11 FUN (1) H
-FP31 FIX
FP30 ADDR A THIS PLACES 42 AT SC INPUTS
FP11 FUN (1) H
FP12 T3 * P3 H
FP11 NOR (1) H
FP40 LD NORM COUNT SETS SC 12-17 TO 42₈
FP11 NOR 3 H
3
FP09 TS 1
FP12 NOR * T1
FP40 IS THIS A STORE ?
YES
1
NO
FP40 IS THIS A DIV
YES
3
NO
NORMALIZE BIT
FP31 FLT PT
FP13 IR13 (0)
FP40 NOR SEL
FP40 WAIT FOR HFPC(1) AND FPC
FP09 STOP PHASE
FP38 NORM EN H
FP40 IS FMA 01 (1)
YES
NO
FP40 NORM DONE
FP40 NORM P
FP38 IS THIS A MPY + SWAP
YES
NO
MXA, MMC MLS ALS SHIFTS FMA 1 LEFT DWN COUNTS EPA 1 DWN COUNTS SC 12-17
MMC FMQ IS ALSO LEFT SHIFTED AS AN FMA EXTENSION
FP09 IS THERE AN SC BORROW
NO
YES
FP09 R SET (1)
R SET SYNC
ST PHASE → 0 ENABLES THE FP TS GEN
NOR CYCLE TS 1
* THIS IS ACTUALLY A 0+B+1 COMPLETING THE 2'S COMPLEMENT FORMATING - THE 1'S COMPLEMENT WAS PERFORMED IN THE FUN CYCLE. DURING INTEGER STORE INSTRUCTIONS, THE FMA CONTAINS THE NON COMPLEMENTED RESULT AND THE FMB CONTAINS THE 2'S COMPLEMENTED RESULT
1
FP35 STORE * NOR
A + B
INTEGER FORMAT BIT 12 (0)?
YES
NO
FP35 INT INC
FP33 AUS; 0 → A SIDE OF ALU
FP33 CN 00 L
FP09 PHASE 2
FP32 B L S
FP35 IR14 (0) * FLT PT
FP35 STORE RND
TO WRITE CYCLE
2
FP19 AD00 H
FP36 GRT
FP40 FLOAT + FIX
FP36 GRT
FP32 MXB, MXA
FP40 EPA GRT L
FP32 EPA UP
FP09 PH03 (1) H
FP35 STORE RND P
FP32 ALS
AUA1 WHICH PLACES 400 ON BSIDE OF ALU
IF ROUNDING CAUSES A CARRY FROM MSB, THEN FMA RT SHTD + EPA INCR.
# A+B IS STILL VALID
IF BIT 27 IS A ONE IT IS SHIFTED TO 26 AND FMA 27 -35 → 0
TO WRITE CYCLE
2

FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
	PARTS LIST			

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES	DRN. E Wilson	DATE 3/23/71	digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS
DECIMALS .XXX = .005 .XX = .02 .X = .1 / ANGLES ±0° 30'	CHK'D. L.P. HoHos	DATE 5/14/71	TITLE NOR TSI CYCLE FLOW
	ENG. Ed King	DATE 5/14/71	
REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓	PROJ. ENG. [illegible]	DATE 5/14/71	
	PROD. [illegible]	DATE 5/14/71	
MATERIAL	NEXT HIGHER ASSY.		SIZE D CODE FD NUMBER FP15-0-58 REV.
FINISH	SCALE		
	SHEET OF		DIST.

DEC FORM NO DRD 102-B



QTY.	DESCRIPTION	PART NO.	ITEM NO.

PARTS LIST

FIRST USED ON OPTION/MODEL

DO NOT SCALE DRAWING
UNLESS OTHERWISE SPECIFIED
DIMENSION IN INCHES
TOLERANCES
DECIMALS ± .005 FRACTIONS ± 1/64 ANGLES ± 0°30'
FINAL SURFACE QUALITY ✓
REMOVE BURRS AND BREAK SHARP CORNERS
MATERIAL
FINISH

DRN. E. Wilson	DATE 3/23/71	
CHK'D. [illegible]	DATE 5/14/71	
ENG. [illegible]	DATE 5/14/71	
PROJ. ENG. [illegible]	DATE 5/14/71	
PROD. [illegible]	DATE 5/14/71	

NEXT HIGHER ASSY
SCALE
SHEET OF

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

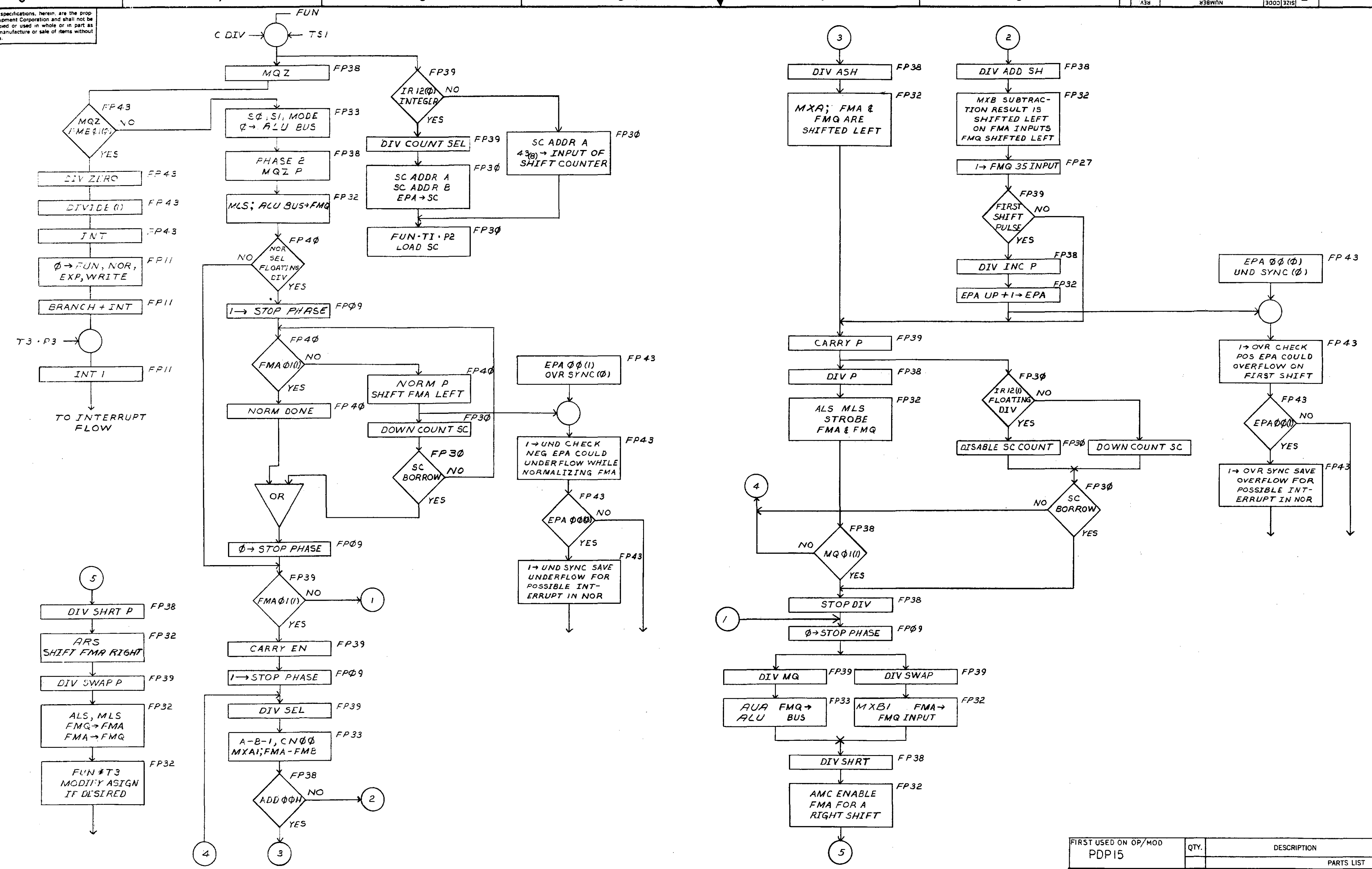
TITLE
NOR TS2 CYCLE FLOW

SIZE	CODE	NUMBER	REV.
D	FD	FP15 - 0 - 59	

DIST.

REVISIONS | CHANGE NO. | REV. | CHK

DEC FORM NO. DRD 100


FUN
C DIV
TSI
MQZ FP38
IR 12(Ø) INTEGER NO YES FP39
MQZ FME$(Ø) NO YES FP43
SØ, SI, MODE Ø→ ALU BUS FP33
DIV COUNT SEL FP39
SC ADDR A 43(8) → INPUT OF SHIFT COUNTER FP3Ø
DIV ZERO FP43
PHASE 2 MQZ P FP38
SC ADDR A SC ADDR B EPA→SC FP3Ø
DIVIDE (1) FP43
INT FP43
MLS; ALU BUS→FMQ FP32
FUN·TI·P2 LOAD SC FP3Ø
Ø→ FUN, NOR, EXP, WRITE FP11
NOR SEL FLOATING DIV NO YES FP4Ø
BRANCH + INT FP11
1→ STOP PHASE FP9
T3 · P3
INT 1 FP11
TO INTERRUPT FLOW
FMA Ø1(1) NO YES FP4Ø
NORM P SHIFT FMA LEFT FP4Ø
EPA ØØ(1) OVR SYNC(Ø) FP43
NORM DONE FP4Ø
DOWN COUNT SC FP3Ø
1→ UND CHECK NEG EPA COULD UNDERFLOW WHILE NORMALIZING FMA FP43
SC BORROW NO YES FP3Ø
OR
EPA ØØ(1) NO YES FP43
Ø→ STOP PHASE FP9
1→ UND SYNC SAVE UNDERFLOW FOR POSSIBLE INTERRUPT IN NOR FP43
FMA Ø1(1) NO YES FP39
1
DIV SHRT P FP38
ARS SHIFT FMA RIGHT FP32
CARRY EN FP39
DIV SWAP P FP39
1→ STOP PHASE FP9
ALS, MLS FMQ→FMA FMA→FMQ FP32
DIV SEL FP39
A-B-1, CNØØ MXAI; FMA-FME FP33
FUN #T3 MODIFY ASIGN IF DESIRED FP32
ADD ØØH NO YES FP38
2
4
3
5
3
DIV ASH FP38
MXA; FMA & FMQ ARE SHIFTED LEFT FP32
2
DIV ADD SH FP38
MXB SUBTRACTION RESULT IS SHIFTED LEFT ON FMA INPUTS FMQ SHIFTED LEFT FP32
1→ FMQ 35 INPUT FP27
FIRST SHIFT PULSE NO YES FP39
DIV INC P FP38
EPA UP+1→EPA FP32
EPA ØØ(Ø) UND SYNC (Ø) FP43
CARRY P FP39
1→ OVR CHECK POS EPA COULD OVERFLOW ON FIRST SHIFT FP43
DIV P FP38
IR 12(1) FLOATING DIV NO YES FP3Ø
ALS MLS STROBE FMA & FMQ FP32
EPA ØØ(1) NO YES FP43
DISABLE SC COUNT FP3Ø
DOWN COUNT SC
1→ OVR SYNC SAVE OVERFLOW FOR POSSIBLE INTERRUPT IN NOR FP43
4
SC BORROW NO YES FP3Ø
MQ Ø1(1) NO YES FP38
STOP DIV FP38
1
Ø→ STOP PHASE FP9
DIV MQ FP39
DIV SWAP FP39
AUA FMQ→ ALU BUS FP33
MXBI FMA→ FMQ INPUT FP32
DIV SHRT FP38
AMC ENABLE FMA FOR A RIGHT SHIFT FP32
5

FIRST USED ON OP/MOD	QTY.	DESCRIPTION	PART NO.	ITEM NO.
PDP15				

PARTS LIST

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

MATERIAL —
FINISH —

DRN. J. McHugh DATE 5-15-71
CHK'D [illegible] DATE 5/14/71
ENG. [illegible] DATE 5/14/71
PROJ. ENG. [illegible] DATE 5/14/71
PROD. [illegible] DATE 5/14/71
NEXT HIGHER ASSY A-ML-FP15
SCALE —
SHEET OF

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

TITLE
FLOAT & INTEGER DIV FUN CYCLE

SIZE D CODE FD NUMBER FP15-Ø-57 REV.

DEC FORM NO DRD 102A

If the +1 added to the FMA causes a carry out of the ALU, the FMA is right-shifted and the EPA is incremented.

For floating-point Multiplication and Division, rounding can occur. If the multiplication or Division operation causes FMQ 01 to go to a 1, the guard bit is set. With this bit set, and rounding requested, +1 is added to the least significant bit of the FMA.

For a Fix instruction, bits in the FMA and FMQ are right-shifted. If, upon termination of the shifting process, FMQ 01 is set, the guard bit is set. A rounding request will then cause +1 to be added to the least significant bit of the FMA.

4.5 GUARD BIT

The guard bit is used to determine whether rounding should occur if rounding is requested (see Figure 4-3). This bit is set under the following conditions:

a. During floating-point Addition, when a carry is produced out of the ALU, the FMA is right-shifted and, if the least significant bit of the FMA is a 1, the guard bit is set.
b. During floating-point multiplication and division, if FMQ 01 is a 1 after the multiplication or division operation, the guard bit is set.
c. During a Fix instruction, upon completion of the shifting process, if FMQ 01 is a 1, the guard bit is set.
d. The contents of the guard bit are saved in bit 01 of the JEA word on a Store JEA instruction.
e. The Load JEA instruction restores the guard bit to a 1 if bit 01 of the JEA operand fetched from memory is set.

When the next instruction is specified (provided it is not a Floating-Point Test, Load JEA, Store JEA, or Branch), the guard bit is cleared.

4.6 FLOATING-POINT ADDITION AND SUBTRACTION

The FP15 can perform floating-point addition, subtraction, and reverse subtraction for both single- and double-precision floating-point numbers. The manner in which these arithmetic operations are implemented is similar and will be described, with differences pointed out as they occur.

In floating-point subtraction, the minuend is loaded into the EPA/A SIGN/FMA via the Load instruction and the subtrahend is loaded into the EPB/B SIGN/FMB via the subtract instruction. If, as a result of some previous computation, the proposed subtrahend for the next subtraction is in the FMA, a Reverse Subtract instruction can be issued. In this event, the contents (subtrahend) of the

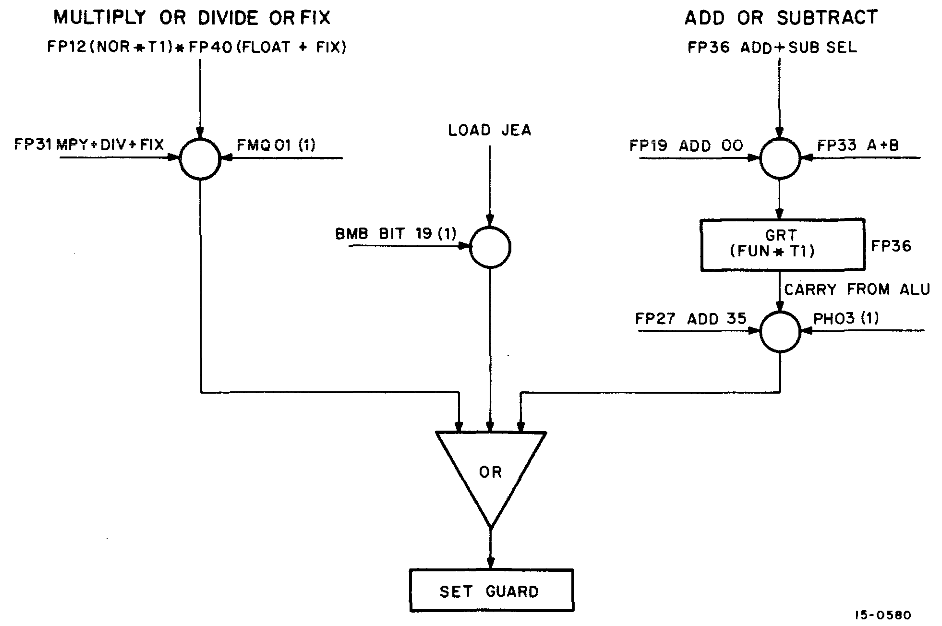


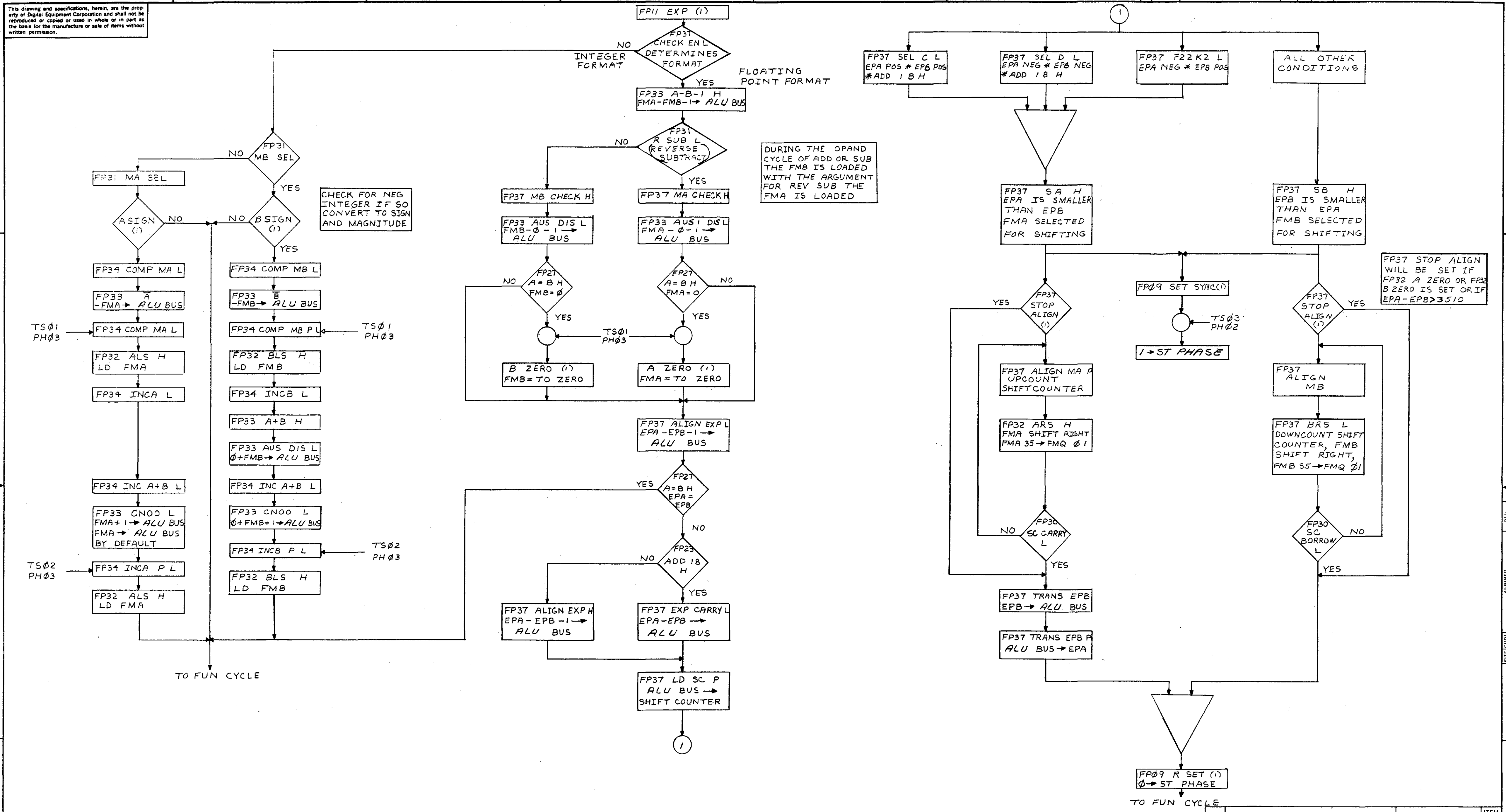
Figure 4-3 Flow Diagram for Setting Guard

EPA/A SIGN/FMA are transferred to the EPB/B SIGN/FMB during the FETCH cycle and the Reverse Subtract instruction loads the minuend into the EPA/A SIGN/FMA.

4.6.1 EXP Cycle

The first function performed in the EXP cycle for floating-point addition or subtraction is a check to determine if the specified instruction is an Add, Subtract, or Reverse Subtract (see Drawing D-FD-FP15-0-52). If it is a Reverse Subtract, A-0-1 is transferred to the ALU bus where A represents the FMA and 0 indicates that the FMB is disabled from the ALU. A test is now made to determine if A=B; if so, the FMA is known to be 0 and STOP ALIGN (1) is set. If the specified instruction is Add or Subtract, 0-B-1 is transferred to the ALU bus, where 0 indicates that the FMA is disabled from the ALU and B represents the FMB. A test is made to determine if A=B; if it does, the FMB must be equal to 0 and STOP ALIGN (1) is set. In effect, then, no alignment will occur for a zero FMA or zero FMB and the FUN cycle is initiated. Also, if the difference between the EPA and EPB is greater than 42_8, STOP ALIGN (1) is generated, no alignment occurs, and the FUN cycle is initiated. However, if the FMA and FMB are non-zero and the difference between the EPA and EPB is less than 42_8, alignment is initiated. EPA-EPB-1 is placed on the ALU bus and, if the exponents are equal, the mantissas are already aligned and the FUN cycle is initiated.

If the exponents are not equal, the sign of the result of EPA-EPB is determined. A negative sign (ADD18H) indicates that the EPB is greater than the EPA and the FMA must be aligned. A positive sign (ADD18L) indicates that the EPA is larger than the EPB and the FMB must be aligned. At this point, the shift counter is loaded with EPA-EPB-1, if the EPA is larger than the EPB or with EPA-EPB-1


FP11 EXP (1)
FP37 CHECK EN L DETERMINES FORMAT
NO
INTEGER FORMAT
YES
FLOATING POINT FORMAT
FP33 A-B-1 H
FMA-FMB-1→ ALU BUS
FP31 R SUB L (REVERSE SUBTRACT)
DURING THE OPAND CYCLE OF ADD OR SUB THE FMB IS LOADED WITH THE ARGUMENT FOR REV SUB THE FMA IS LOADED
FP37 MB CHECK H
FP37 MA CHECK H
FP33 AUS DIS L FMB-Ø-1→ ALU BUS
FP33 AUS1 DIS L FMA-Ø-1→ ALU BUS
FP27 A=B H FMB=Ø
FP27 A=B H FMA=O
TSØ1 PHØ3
B ZERO (1) FMB=TO ZERO
A ZERO (1) FMA=TO ZERO
FP37 ALIGN EXP L EPA-EPB-1→ ALU BUS
FP27 A=B H EPA=EPB
FP23 ADD 18 H
FP37 ALIGN EXP H EPA-EPB-1→ ALU BUS
FP37 EXP CARRY L EPA-EPB→ ALU BUS
FP37 LD SC P ALU BUS→ SHIFT COUNTER
1
FP31 MB SEL
FP31 MA SEL
CHECK FOR NEG INTEGER IF SO CONVERT TO SIGN AND MAGNITUDE
A SIGN (1)
B SIGN (1)
FP34 COMP MA L
FP34 COMP MB L
FP33 A -FMA→ ALU BUS
FP33 B -FMB→ ALU BUS
TSØ1 PHØ3
FP34 COMP MA L
FP34 COMP MB P L
FP32 ALS H LD FMA
FP32 BLS H LD FMB
FP34 INCA L
FP34 INCB L
FP33 A+B H
FP33 AUS DIS L Ø+FMB→ ALU BUS
FP34 INC A+B L
FP34 INC A+B L
FP33 CNOO L FMA+1→ ALU BUS FMA→ ALU BUS BY DEFAULT
FP33 CNOO L Ø+FMB+1→ ALU BUS
TSØ2 PHØ3
FP34 INCB P L
FP34 INCA P L
FP32 ALS H LD FMA
FP32 BLS H LD FMB
TO FUN CYCLE
1
FP37 SEL C L EPA POS * EPB POS *ADD 1 8 H
FP37 SEL D L EPA NEG * EPB NEG *ADD 1 8 H
FP37 F22K2 L EPA NEG * EPB POS
ALL OTHER CONDITIONS
FP37 SA H EPA IS SMALLER THAN EPB FMA SELECTED FOR SHIFTING
FP37 SB H EPB IS SMALLER THAN EPA FMB SELECTED FOR SHIFTING
FP37 STOP ALIGN WILL BE SET IF FP32 A ZERO OR FP32 B ZERO IS SET OR IF EPA-EPB>35 10
FPØ9 SET SYNC(1)
TSØ3 PHØ2
1→ST PHASE
FP37 STOP ALIGN (1)
FP37 ALIGN MA P UPCOUNT SHIFT COUNTER
FP37 ALIGN MB
FP32 ARS H FMA SHIFT RIGHT FMA 35→FMQ Ø1
FP37 BRS L DOWNCOUNT SHIFT COUNTER, FMB SHIFT RIGHT, FMB 35→FMQ Ø1
FP30 SC CARRY L
FP30 SC BORROW L
FP37 TRANS EPB EPB→ ALU BUS
FP37 TRANS EPB P ALU BUS→ EPA
FPØ9 R SET (1) Ø→ ST PHASE
TO FUN CYCLE

QTY.	DESCRIPTION	PART NO.	ITEM NO.
	PARTS LIST		

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

MATERIAL
FINISH

		DATE
DRN	[illegible]	3/21/71
CHK'D.	L.P. Hollos	5/14/71
ENG.	[illegible]	5/14/71
PROJ ENG.	[illegible]	5/14/71
PROD.	[illegible]	5/14/71

FIRST USED ON
SCALE
SHEET OF

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

TITLE ADD, SUB, REV SUB EXP CYCLE

SIZE D CODE FD NUMBER FP15—Ø—52 REV
DIST.

REVISIONS REV CHANGE NO CHK

DEC FORM NO DRD 102A

and carry insert (EPA-EPB+1-1) if the EPB is larger than the EPA. This is to set up the shift counter so the proper amount of shifts are performed to align the exponents.

To determine whether the FMA or FMB is to be selected for shifting, the signs of EPA and EPB are examined, in addition to the sign (ADD18) of the result of EPA-EPB. The three cases, in which the FMA is selected for shifting, are listed below:

a. Positive EPA, positive EPB, and a negative sign as a result of EPA-EPB. With both quantities positive and a negative result for EPA-EPB, the EPA is smaller than the EPB.

   Example:
   $$\begin{array}{rl} +3 & \text{EPA} \\ \underline{-(+5)} & \text{EPB} \\ -2 & \text{EPA-EPB} \end{array}$$

b. Negative EPA, negative EPB, and a negative sign for EPA-EPB. In this case, EPA is smaller (more negative) than EPB in order for a negative sign to occur.

   Example:
   $$\begin{array}{rl} -5 & \text{EPA} \\ \underline{-(-3)} & \text{EPB} \\ -2 & \text{EPA-EPB} \end{array}$$

c. EPA negative and EPB positive. The sign in this case is always negative indicating that the EPB is larger (more positive) than the EPA.

   Example:
   $$\begin{array}{rl} -5 & \text{EPA} \\ \underline{-(+2)} & \text{EPB} \\ -7 & \text{EPA-EPB} \end{array}$$

For all other possibilities, the FMB is selected for shifting. Up to this point, the FMA and FMB have been examined to see if either is 0; the shift counter has been loaded with EPA-EPB (if EPA < EPB) or EPA-EPB-1 (if EPA > EPB) to provide an accurate count of the number of shifts required to align exponents; and the mantissa register associated with the smaller exponent has been selected for shifting.

If STOP ALIGN is set, this indicates that mantissa alignment is not necessary as a result of one of the following conditions:

a. Zero FMA
b. Zero FMB, or
c. EPA-EPB > 35_{10}

If STOP ALIGN is not set, alignment is performed, and either the FMA or FMB is selected for shifting. The mantissa with the smaller exponent is selected for shifting. If the EPA is less than the EPB, SA H is generated and the FMA is shifted. The shift counter is loaded with EPA-EPB, which will be a negative number in this case. The counter will be incremented with each shift until an SC CARRY is detected (counter going from all ones to all zeros). For example, if the EPA contained +2 and the EPB contained +4, the shift counter is loaded with -2. The first shift of the FMA increments the counter to -1 and the second to all zeros, which is detected as an SC CARRY. This indicates termination of mantissa alignment.

If the FMB is selected for shifting as a result of EPB being smaller than EPA, SA will be low and the shift counter is loaded with EPA-EPB-1. This quantity is a positive number and the counter is decremented for each shift until an SC BORROW is detected; this is why EPA-EPB-1 is required rather than EPA-EPB. For example, assume that the EPA contains +3 and the EPB contains +1. The shift counter is loaded with EPA-EPB-1 or 1. The first shift of the FMB decrements the counter to zero, and the second shift of the FMB decrements the counter to all ones, which is detected as SC BORROW to conclude the alignment.

Since the exponent associated with the mantissa not being shifted is the true exponent of the result, it is necessary to load the EPB into the EPA, if the FMA was selected for shifting. While alignment is taking place, the time state generator is disabled. On completing the alignment process, the time state generator is restarted, and the FUN cycle is initiated.

4.6.2 FUN Cycle

In the FUN cycle, the A side of the ALU is disabled if the FMA is 0 and the B side of the ALU is disabled if the FMB is 0 (see Drawing FP15-0-53). When the EPA differs from the EPB by more than 35_{10}, the side of the ALU associated with the smaller exponent is disabled. This prevents additional shifting and is time saving. For example, if EPB is greater than EPA by 1000, EPA has to be shifted 1000 times and is, thus, a very small number compared to EPB. In fact, the number is so relatively small it can be considered 0. Consequently, the B side of the ALU is disabled, the 1000 shifts are prevented, and the time necessary to perform these shifts is saved.

The following two rules of addition and subtraction with respect to the sign are used.

a. During addition, quantities with like signs are added, while quantities with unlike signs are subtracted.

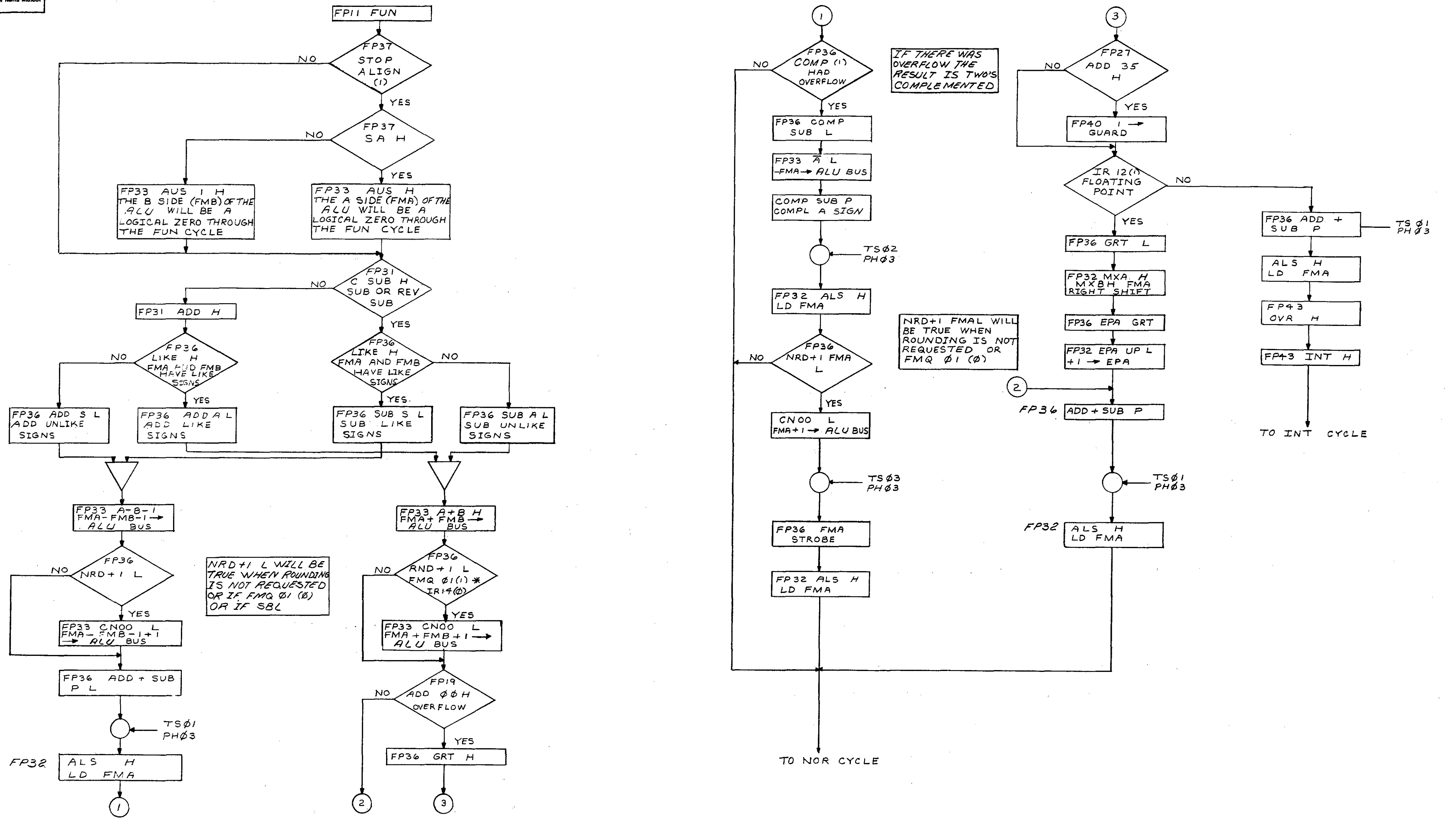
   Examples:
   $$\begin{array}{r} +5 \\ \underline{+(+2)} \\ +7 \end{array} \qquad \begin{array}{r} -5 \\ \underline{+(+2)} \\ -3 \end{array}$$

   NOTE

   In the example on the right the two quantities are subtracted although the operation specified is addition.

b. During subtraction, quantities with like signs are subtracted, while quantities with unlike signs are added.

(continued on page 4-10)



FP11 FUN
FP37 STOP ALIGN (1)
NO
YES
FP37 SA H
FP33 AUS 1 H THE B SIDE (FMB) OF THE ALU WILL BE A LOGICAL ZERO THROUGH THE FUN CYCLE
FP33 AUS H THE A SIDE (FMA) OF THE ALU WILL BE A LOGICAL ZERO THROUGH THE FUN CYCLE
FP31 C SUB H SUB OR REV SUB
FP31 ADD H
FP36 LIKE H FMA AND FMB HAVE LIKE SIGNS
FP36 ADD S L ADD UNLIKE SIGNS
FP36 ADD A L ADD LIKE SIGNS
FP36 SUB S L SUB LIKE SIGNS
FP36 SUB A L SUB UNLIKE SIGNS
FP33 A-B-1 FMA-FMB-1 → ALU BUS
FP36 NRD+1 L
NRD+1 L WILL BE TRUE WHEN ROUNDING IS NOT REQUESTED OR IF FMQ Ø1 (Ø) OR IF SBL
FP33 CN00 L FMA-FMB-1+1 → ALU BUS
FP36 ADD + SUB P L
TSØ1 PHØ3
FP32
ALS H LD FMA
1
FP33 A+B H FMA+FMB → ALU BUS
FP36 RND+1 L FMQ Ø1(1) * IR14(Ø)
FP33 CN00 L FMA+FMB+1 → ALU BUS
FP19 ADD ØØ H OVERFLOW
FP36 GRT H
2
3
1
FP36 COMP (1) HAD OVERFLOW
IF THERE WAS OVERFLOW THE RESULT IS TWO'S COMPLEMENTED
FP36 COMP SUB L
FP33 A̅ L -FMA → ALU BUS
COMP SUB P COMPL A SIGN
TSØ2 PHØ3
FP32 ALS H LD FMA
FP36 NRD+1 FMA L
NRD+1 FMAL WILL BE TRUE WHEN ROUNDING IS NOT REQUESTED OR FMQ Ø1 (Ø)
CN00 L FMA+1 → ALU BUS
TSØ3 PHØ3
FP36 FMA STROBE
FP32 ALS H LD FMA
TO NOR CYCLE
3
FP27 ADD 35 H
FP40 1 → GUARD
IR 12(1) FLOATING POINT
FP36 GRT L
FP32 MXA H MXB H FMA RIGHT SHIFT
FP36 EPA GRT
FP32 EPA UP L +1 → EPA
2
FP36
ADD + SUB P
TSØ1 PHØ3
FP32
ALS H LD FMA
FP36 ADD + SUB P
TSØ1 PHØ3
ALS H LD FMA
FP43 OVR H
FP43 INT H
TO INT CYCLE

QTY.	DESCRIPTION	PART NO.	ITEM NO.

PARTS LIST

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

MATERIAL

FINISH

DRN.	[illegible]	DATE 3/30/71
CHK'D.	L.P. Hottos	DATE 5/14/71
ENG.	[illegible]	DATE 5/14/71
PROJ. ENG.	[illegible]	DATE 5/14/71
PROD.	[illegible]	DATE 5/14/71

FIRST USED ON

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

TITLE
ADD, SUB, REV SUB FUN CYCLE

SIZE D CODE FD NUMBER FP15 – Ø – 53 REV.

DIST.

REVISIONS CHANGE NO. REV CHK

DEC FORM NO DRD 102A

```
   b.    Examples:   +5              +5
(cont)              -(+2)           -(-2)
                    -----           -----
                     +3              +7
```

NOTE

In the example on the right the two quantities are added although the operation specified is subtraction.

Referring to the flow diagram again, quantities with unlike signs during addition and like signs during subtraction are actually subtracted. Thus A-B-1 is put on the ALU bus for these cases. Conversely, quantities with like signs during addition and unlike signs during subtraction are actually added. In these cases, A+B is put on the ALU bus.

4.6.3 Processing of Subtracted Quantities

If the quantities are being subtracted and the FMB contains the mantissa with the smaller exponent, it must be determined if rounding has been requested and whether FMQ 01 is a 1. If both conditions are true, A-B-1 is put on the ALU bus. An additional 1 is subtracted to account for the rounding of the FMB (A-B-1 = A-[B+1]). This is accomplished by putting A-B-1 on the ALU bus rather than just A-B. If rounding has not been requested, or FMQ 01 is a 0, a carry insert of +1 is added and A-B-1+1, or simply A-B, is put on the ALU bus. This quantity, in both cases, represents the result that is loaded into the FMA. However, if overflow occurs, it indicates a wrong assumption was made and the result in the FMA is incorrect. This is explained in detail in the following paragraphs.

4.6.3.1 Overflow - For quantities that are actually subtracted (addition with unlike signs or subtraction with like signs), the sign of the result is assumed to be the same sign as in the FMA. If no overflow occurs, the sign of the result is correct. If overflow occurs, it indicates an incorrect sign has been assumed. If this occurs, the assumed sign is complemented and the actual result is 2's complemented. Two examples follow--the first shows that the assumed sign is correct, the second shows that the assumed sign is incorrect.

```
Example:                    Sign
(no overflow)    +6          0     0 1 1 0     FMA
                 -3          1     0 0 1 1     FMB
                 ---        ---------------
                 +3          0     0 0 1 1     = +3
                                   ↑
                                   No overflow, sign
                                   correct, result correct
```

```
Example:                -3      Sign
(with overflow)       -(-6)      1     0 0 1 1     FMA
                      -----      1     0 1 1 0     FMB
                        +3      ----------------
                                 1     1 1 0 1
                                       ↑
                                       Overflow, 2's complement
                                       result, complement sign

                                 0     0 0 1 1     = +3
                                 ↑     ↑
              Sign complemented        2's complement of result
```

If rounding is not requested or FMQ 01 is a 0, 1 is added to the FMA to compensate for the incorrect result. The result is then loaded into the FMA.

4.6.4 Processing of Added Quantities

When two quantities are to be added (addition with like signs or subtraction with unlike signs), A+B is put on the ALU bus as described previously. If FMQ 01 is a 1 and rounding is requested, +1 is added to the least significant bit of the FMA.

A check is now made for an overflow condition. A floating-point overflow causes a signal designated GRT to be issued. The FMA is right-shifted to transfer the overflowed bit back into the FMA; the EPA is incremented to compensate for the shift. ADD 35 is examined prior to the right-shift--if this bit is a 1, FMQ 01 becomes a 1 after the right-shift and the guard bit is set. The FMA is now loaded with the results of A+B on the ALU bus. If no overflow occurs, the FMA is not right-shifted, the guard bit is not set, the EPA is not incremented, and the FMA is loaded directly with A+B from the ALU bus.

4.6.4.1 Overflow Interrupt Due to Addition or Subtraction - If the addition or subtraction operation results in an exponent greater than 2^{17}-1 (377777_8), a temporary overflow occurs. The result contained in the EPA, after the overflow, is no longer the true result. However, the true result can be calculated by adding the contents of the EPA, after the overflow, to 2^{17}. The contents of A SIGN/FMA are unchanged.

4.6.4.2 Overflow Interrupt Due to Rounding - If rounding is requested, and the rounding operation produces a carry out of the ALU, the FMA is right-shifted and the EPA is incremented. If the EPA contains 377777_8 and is incremented, an overflow interrupt occurs and the interrupt cycle is initiated.

4.6.4.3 Underflow Interrupt Due to Normalizing - Normalizing is accomplished by left-shifting the FMA and decrementing the EPA for each left-shift. If, during this process, the EPA contains 400000_8 and is decremented to 377777_8, an underflow interrupt occurs. The contents of the A SIGN/FMA are correct. The EPA no longer contains the true result; however, this can be obtained by adding -2^{18} to the contents of the EPA after the underflow occurs.

Example: EPA 400000_8

$$\begin{array}{rr} \text{EPA} & 400000_8 \\ & \underline{-1} \\ \text{Result left in EPA} & 377777_8 \end{array}$$

True result = $-2^{18} + 377777_8$

It is possible for the underflow to eliminate the condition that causes the temporary overflow during the addition or subtraction. If underflow does not remove this condition, the overflow interrupt becomes a permanent interrupt and enters an interrupt cycle (see Paragraphs 3.8 and 3.9).

4.7 INTEGER ADD AND SUBTRACT

The FP15 can perform addition, subtraction, and reverse subtraction using either single-precision or extended-precision data formats. Addition, subtraction, and reverse subtraction are performed in a similar manner and will be explained using Drawings D-FD-FP15-0-52 and D-FD-FP15-0-53 for reference.

4.7.1 EXP Cycle

In the EXP cycle, negative integers (stored in memory in 2's complement format) are converted to sign and magnitude format. For example, if the specified instruction is a Load or Reverse Subtract with a negative argument, the argument is converted to sign and magnitude format and loaded into the FMA. If the instruction is an Add or Subtract, with a negative argument, the argument is converted to sign and magnitude format and loaded into the FMB. The negative integers are converted from 2's complement to sign and magnitude format by 1's complementing and incrementing the 2's complement integer. For example, the number -5_8 in 2's complement format is 1.011. One's complementing and incrementing this number yields 1.101, which represents -5_8 in sign and magnitude format.

4.7.2 FUN Cycle

In the FUN cycle, the signs of the operands are compared. If the specified operation is an integer add and the signs are unlike or an integer subtract and the signs are alike, the ALU is selected for A-B-1 operation (a straight A-B function is not possible). The -1 is compensated for by a carry insert which puts +1 in the ALU bus along with the contents of the FMA. Actually, the ALU performs an A-B-1+1 function which reduces to A-B. A represents the FMA, and B represents the FMB. The FMB is subtracted from the FMA and the result is loaded into the FMA.

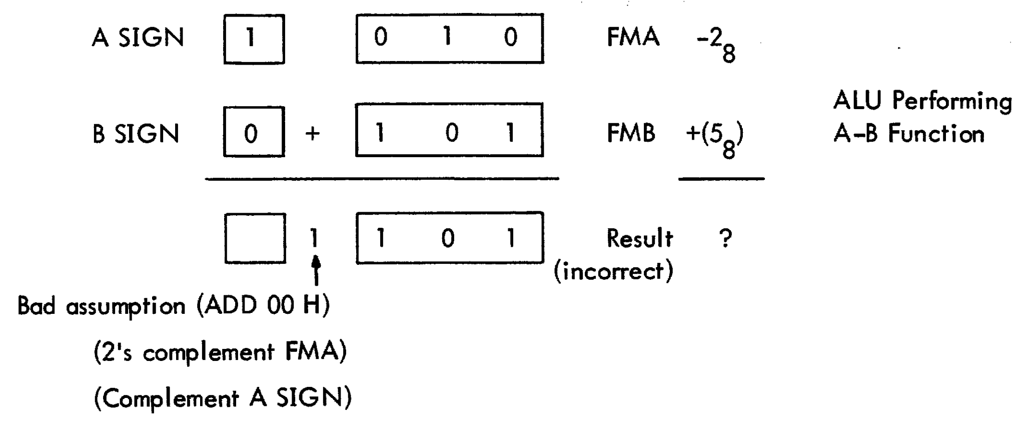
If the specified operation is an integer add and the signs are alike or an integer subtract and the signs are unlike, the ALU is selected to perform an A+B function which really adds the contents of the FMA to the FMB and puts the results into the FMA.

If the two quantities are positive and added together, it is possible for an integer overflow to occur. This is detected as a carry out of the ALU (ADD 00 high). If this occurs, the FP15 goes into an interrupt cycle.

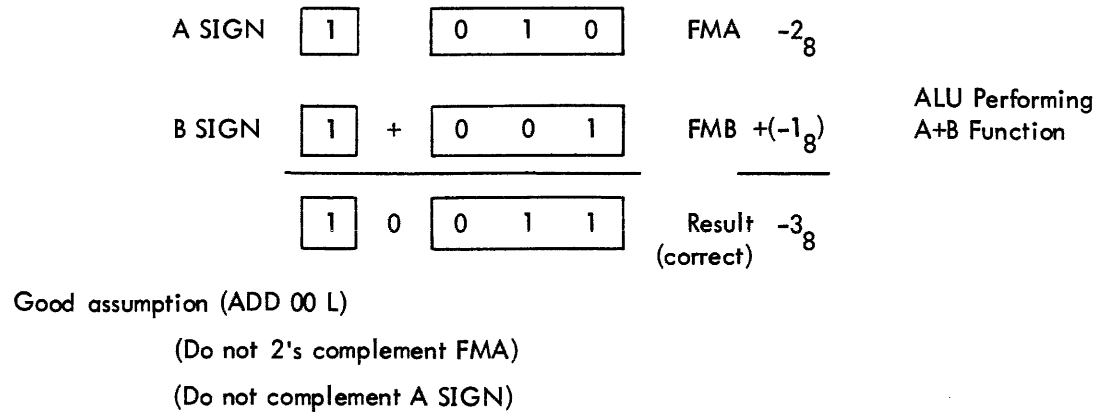
One of the last things performed in integer addition or subtraction is to determine the sign of the result. This is accomplished by assuming the previous sign of the FMA is correct. If so, there is no carry generated out of the ALU, and the addition or subtraction of the FMA or FMB is done in the normal manner. The A SIGN represents the sign of the result and the contents of the FMA yield the true number.

However, if a carry occurs out of the ALU, this indicates that the sign has been assumed incorrectly. If this is the case, the existing contents of the FMA are 2's complemented and the A SIGN is complemented. Several simplified examples follow that illustrate this concept. Note that a bad assumption can only be made when the ALU is specified to do an A-B function.

Example: Bad Assumption (Integer Add)


A SIGN 1 | 0 1 0 FMA -2_8
B SIGN 0 + 1 0 1 FMB $+(5_8)$
ALU Performing A-B Function
1 | 1 0 1 Result ? (incorrect)
Bad assumption (ADD 00 H)
(2's complement FMA)
(Complement A SIGN)
A SIGN 0 | 0 1 1 Result $+3_8$ (correct)

Example: Good Assumption (Integer Add)


A SIGN 1 | 0 1 0 FMA -2_8
B SIGN 1 + 0 0 1 FMB $+(-1_8)$
ALU Performing A+B Function
1 0 0 1 1 Result -3_8 (correct)
Good assumption (ADD 00 L)
(Do not 2's complement FMA)
(Do not complement A SIGN)

Example: Bad Assumption (Integer Subtract)

A SIGN $\boxed{0}$ $\boxed{1\ 0\ 1}$ FMA $+5_8$

B SIGN $\boxed{0}$ - $\boxed{1\ 1\ 0}$ FMB $-(+6_8)$ ALU Performing A-B Function

$\boxed{\ }$ 1 1 1 1 Result ? (incorrect)

Bad assumption (ADD 00 H)
(2's complement FMA)
(Complement A SIGN)

$\boxed{1}$ $\boxed{0\ 0\ 1}$ Result -1_8 (correct)

Example: Good Assumption (Integer Subtract)

A SIGN $\boxed{1}$ $\boxed{1\ 1\ 0}$ FMA -6_8

B SIGN $\boxed{0}$ - $\boxed{0\ 0\ 1}$ FMB $-(+1_8)$ ALU Performing A+B Function

0 $\boxed{1}$ 0 $\boxed{1\ 1\ 1}$ -7_8

Good assumption (ADD 00 L)
(Do not 2's complement FMA)
(Do not complement A SIGN)

4.7.3 Overflow

If the addition or subtraction operation results in a magnitude greater than $2^{35}-1$, an overflow interrupt will occur. The result contained in the FMA, after the overflow, is no longer the correct result.

4.7.4 Integer Reverse Subtraction

Integer reverse subtraction and integer subtraction are similar to each other except for the fact that, in integer reverse subtraction, the contents of the FMA are transferred to the FMB during the FETCH cycle and the FMA is loaded with the subtrahend when the integer reverse subtraction is specified. During the EXP and FUN cycles, operation is similar since the subtrahend is in the FMA and the minuend is in the FMB for both integer reverse subtraction and integer subtraction.

However, the correct result can be computed by adding 2^{35} to the existing contents of the FMA after the interrupt. The A SIGN remains unchanged.

Example:	A SIGN (0)	FMA 300007_8
	B SIGN (0)	FMB 077777_8
	A SIGN (1)	400006_8
	Result left in FMA	000006_8
	Correct Result = 2^{35} +	000006_8

4.8 FLOATING-POINT AND INTEGER MULTIPLY

In order to multiply two numbers in floating-point format, the following basic functions are performed: calculation of exponent, determination of the sign of the product, and multiplication of the mantissas. These are described in the following paragraphs.

4.8.1 Calculation of Exponents

During the EXP cycle, the contents of the EPA and EPB are gated into the ALU where the EPA is added to the EPB (see Figure 4-5). The sum is strobed back into the EPA. In floating-point multiplication operations, recall that the exponents are added while the mantissas are multiplied. In integer multiplication, there is no exponent calculation.

4.8.2 Determining Sign of Product

The sign of the product is determined in the EXP cycle before the mantissas are multiplied. If the multiplier and multiplicand have the same sign, the sign of the product is positive. If the signs differ, the resultant sign of the product is negative. In either case, the resultant sign is strobed into the A SIGN. Negative integers are converted to sign and magnitude format; positive integers are already in sign and magnitude format.

4.8.3 Multiplication of the Mantissas

The mantissas are multiplied by a series of additions and right-shifts of the FMA during the FUN cycle. Before the actual multiplication occurs, however, the shift counter is preloaded with a constant of 42_8 (34_{10}), the contents of the FMA are transferred to the FMQ, and the FMA is then cleared. The rules for multiplication of the mantissas are:

1. Test the least significant bit of the FMQ.
   a. If FMQ 35 is a 1, add the contents of the FMB to the contents of the FMA and shift and load the FMA and shift the FMQ right as one 70-bit register.
   b. If FMQ 35 is a 0, do not load the FMA with A+B, but merely shift the FMA and FMQ right.
2. Decrement the shift counter and test for a borrow.
   a. If a borrow is detected, the multiplication is complete.
   b. If no borrow is detected, repeat the first step.
3. After a borrow has been detected, the multiplication is complete if it is a floating-point multiply. If it is an integer multiply, the contents of the FMA and FMQ are swapped and the multiplication is complete.

Figure 4-4 shows a simplified flow diagram of the above rules. For floating-point multiplication, the most significant bits of the product are retained in the FMA. For integer multiplication, as a result of the swap, however, the most significant bits of the product are retained in the FMQ.

4.8.4 Multiply Algorithm

In order to depict the multiply algorithm, Figure 4-5 Shows a simplified example where the number 5 (101_2) is to be multiplied by the number 4 (100_2). EPA and EPB are both equal to 3, so in the final product, the binary number will be shifted six places to the right. Initially, the shift counter is loaded with 2, the FMA is transferred to the FMQ, and the FMA is cleared.

NOTE

The shift counter is loaded with one less than the number of stages in the FMQ. Since the example uses a three-stage FMQ, a count of 2 is preloaded into the shift counter. In the case of the FP15, the shift counter is loaded with 42_8 (34_{10}), actually 35_{10} shifts may occur before a borrow is produced.

In the first step, the least significant bit of the FMQ is tested. Since it is a 1, the contents of the FMB are added to the contents of the FMA and the entire FMA and FMQ are shifted right as one 6-bit register. Each time a shift occurs, the shift counter is decremented. The shift counter now contains a count of 1.

In Step 2, the least significant bit of the FMQ is tested again. Since it is a 0 in this case, the FMA and FMQ are merely shifted right. The shift counter is again decremented (this time to 00).

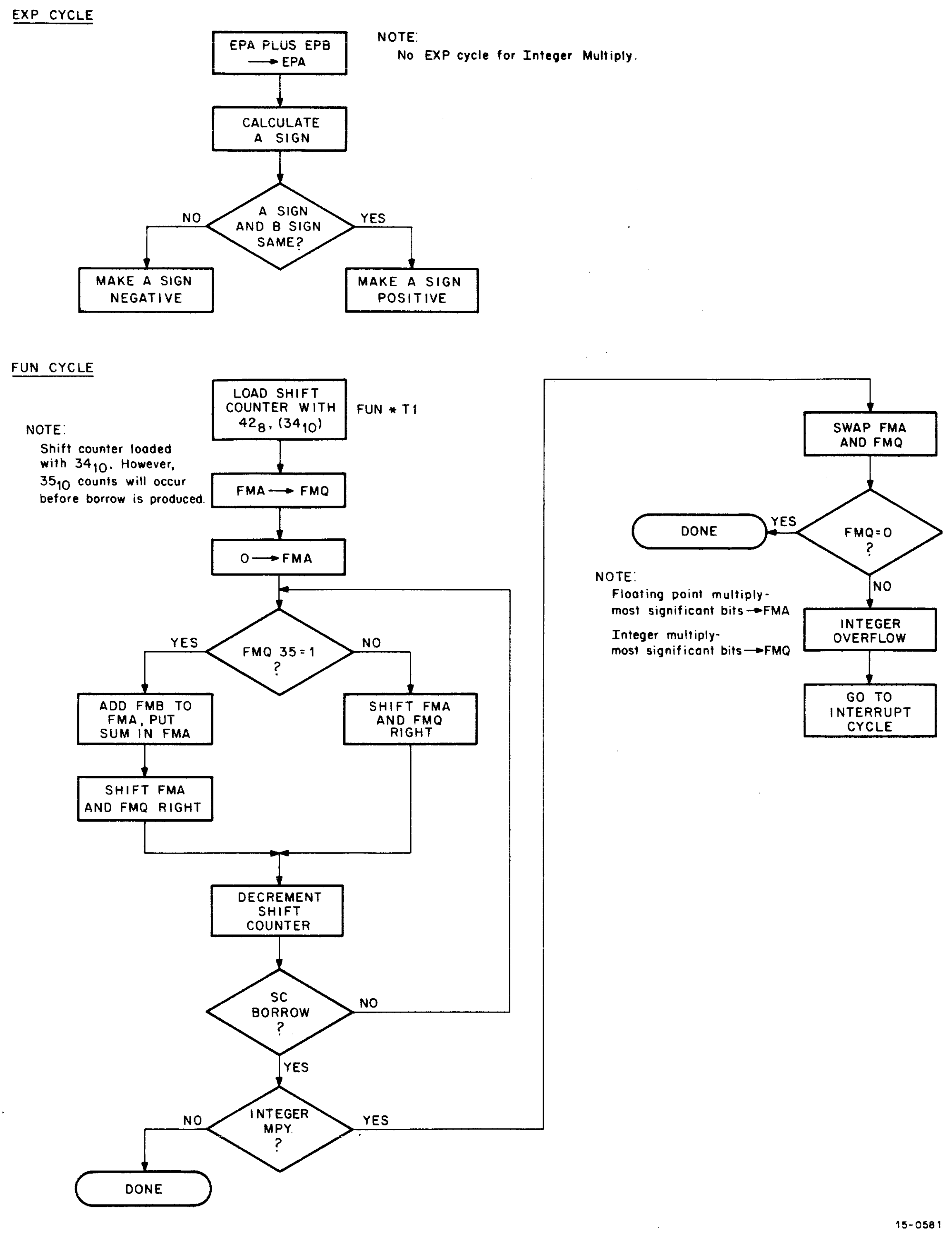


Figure 4-4 Multiply Simplified Flow Diagram

Initial Conditions:

$100_2 \times 101_2 = ?$

$4_{10} \times 5_{10} = 20_{10}$

EPA = 3
EPB = 3
FMB = 100
FMA = 101
FMQ = 000

After Swap

FMA = 000
FMQ = 101

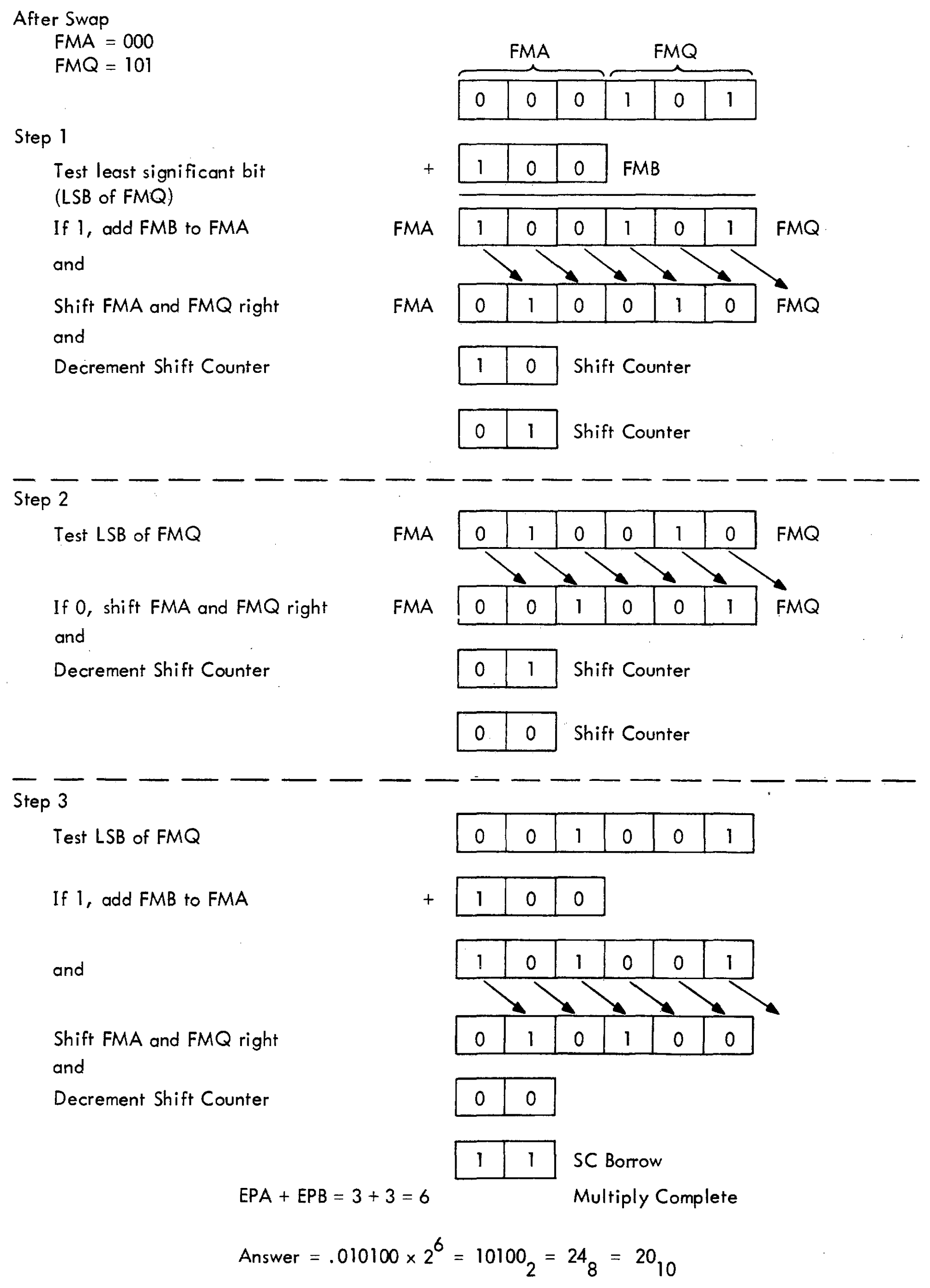


Figure 4-5 Multiply Algorithm

In Step 3, the least significant bit of the FMQ is tested again and is a 1. Consequently, the contents of the FMB are added to the contents of the FMA and the FMA and FMQ are shifted right. The shift counter is again decremented to 11, indicating a borrow condition. This signifies that the multiplication is complete and the product is $.010100 \times 2^6$. This number is 010100. in binary after the binary point has been shifted six places to the right.

If this were an integer multiply, the FMA and FMQ would be swapped. In the example presented, a 1 is contained in the FMQ after the swap. For integer multiply, any 1 contained in the FMQ after the swap results in an overflow interrupt. Therefore, a product up to a maximum of 35 bits in length (length of the FMA) is possible in the FP15 for integer Multiplication. Drawings D-FD-FP15-0-54 and D-FD-FP15-0-55 represent flow diagrams of multiplication in the EXP and FUN cycles, respectively.

4.8.5 Floating-Point Overflow

The following paragraphs describe the interrupt exceptions which can occur during floating-point multiplication. An overflow or underflow in the EXP cycle is temporary, since it can be removed by an underflow or overflow, respectively, in the FUN cycle.

4.8.5.1 Overflow Interrupt - EXP Cycle - A temporary overflow can occur if a positive EPB is added to a positive EPA with a negative result. An example of this is:

```
                           EPA 00
                          ↓ (SIGN)
377777_8 = EPA    = 0.11   111   111   111   111   111
000001_8 = EPB    = 0.00   000   000   000   000   001
----------------------------------------------------
400000   = Result = 1.00   000   000   000   000   000
                    ↑SIGN
```

The overflow condition is detected as a result of the sign bit (EPA00) going from 0 to 1.

It is possible that this temporary overflow can be eliminated during the NOR cycle if normalize is requested. Decrementing the EPA during normalize may reduce the number so that it can be contained in the EPA. If so, the temporary overflow condition is eliminated. If the condition is not removed, an overflow interrupt will occur at NOR*TS03.

4.8.5.2 Underflow Interrupt - EXP Cycle - A temporary underflow can occur if a negative EPB is added to a negative EPA with a positive result. An example of this is:

```
                           EPA 00
                          ↓ (SIGN)
 400000 = EPA    =  1.00   000   000   000   000   000
 400000 = EPB    =  1.00   000   000   000   000   000
-----------------------------------------------------
1000000 = Result = 10.00   000   000   000   000   000
                    ↑SIGN
```

It is possible that this temporary underflow can be eliminated during the NOR cycle if rounding is requested. This is possible only if the EPA just underflowed, since rounding can only increment the EPA once and only if a carry was generated out of the ALU. If the underflow condition is not removed, an underflow interrupt will occur at NOR*TS02.

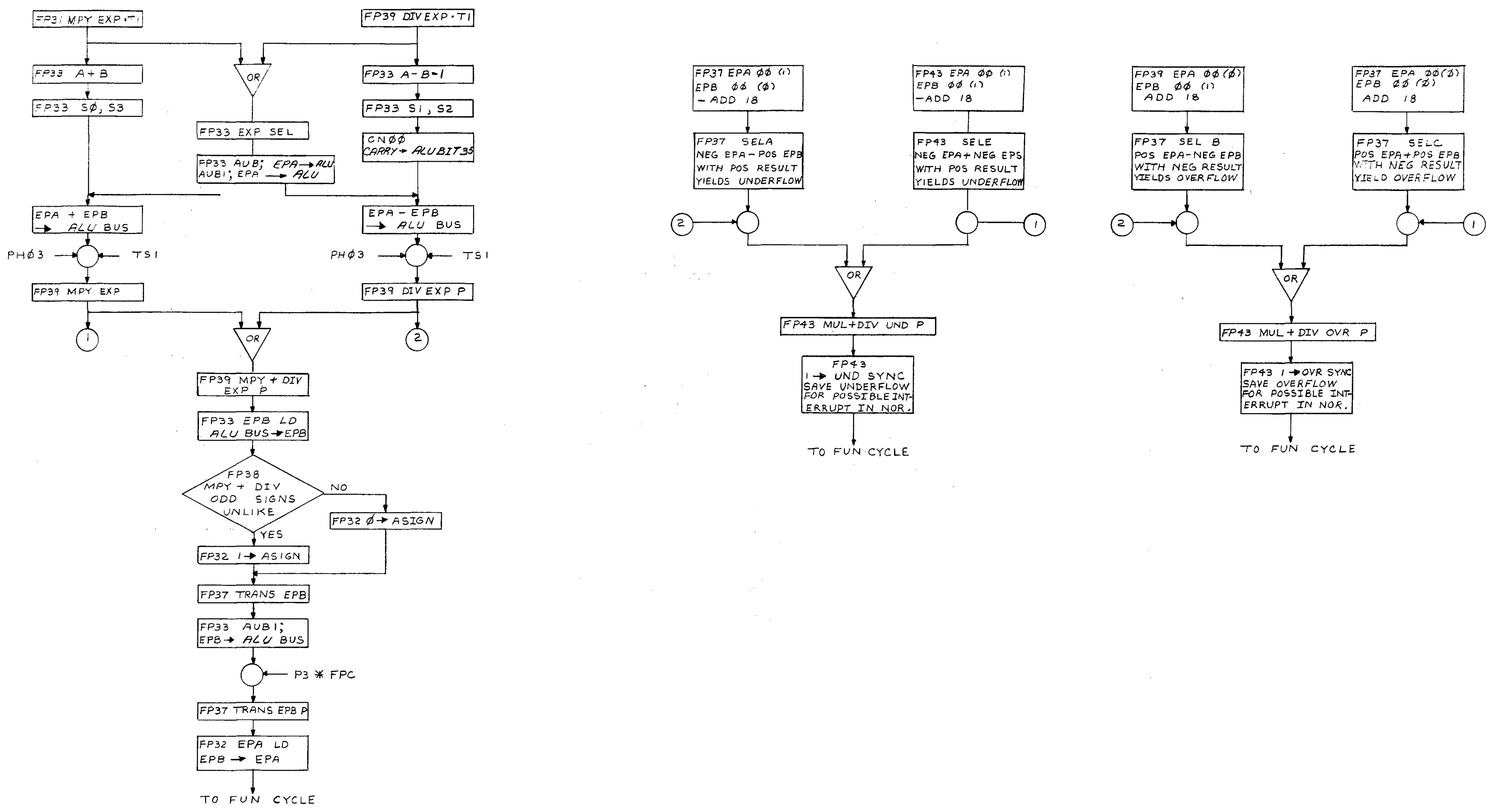
In effect, two negative quantities are added with a result too small to be shown in the register. The change of sign in the EPA from negative to positive is detected as an underflow. The bit (EPA 00 going from a 1 to a 0) is preserved until the NOR cycle, where it is possible for rounding, if requested, to eliminate the condition causing the interrupt.

4.8.5.3 Overflow Interrupt - NOR Cycle - At NOR*TS02, the guard bit is examined. If the bit is set, and rounding is requested, 1 is added to the least significant bit of the FMA. If this operation produces a carry out of the most significant stage of the ALU, the FMA is right-shifted and the EPA is incremented. If the EPA contains 377777_8 before it is incremented, an overflow interrupt will occur and the interrupt cycle is initiated.

It is possible during rounding that incrementing the EPA will remove the condition causing the temporary underflow in the EXP cycle. If the condition is not removed, the interrupt flag is raised. For example, assume that the EPA contained 377777_8 in the EXP cycle due to underflow and that a rounding request was made. The rounding caused a carry out of the ALU that necessitated right-shifting the FMA and incrementing the EPA. Incrementing the EPA to 400000 removed the temporary underflow.

4.8.5.4 Underflow Interrupt - NOR Cycle - If normalize is requested, it is performed during the NOR cycle for floating-point multiplication. As the mantissa is being left-shifted, the EPA is being decremented. During normalize, if the EPA should be decremented from 400000 to 377777, an underflow interrupt will occur at NOR*TS03 and the interrupt cycle is initiated. This is detected as a result of EPA 00 going from 1 to a 0.

It is possible during normalize that decrementing the EPA will remove the condition causing the temporary overflow in the EXP cycle. If the condition is not removed, the interrupt flag is then raised.



QTY.	DESCRIPTION	PART NO.	ITEM NO.

PARTS LIST

FIRST USED ON OPTION/MODEL

DO NOT SCALE DRAWING
UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES
TOLERANCES
DECIMALS ±.005 FRACTIONS ±1/64 ANGLES ±0°30'
FINAL SURFACE QUALITY √
REMOVE BURRS AND BREAK SHARP CORNERS

MATERIAL

FINISH

		DATE
DRN.	[illegible]	3/29/71
CHK'D.	L.P Hollos	5/14/71
ENG.	[illegible]	5/14/71
PROJ. ENG.	[illegible]	5/ /71
PROD.	[illegible]	5/14/71

NEXT HIGHER ASSY

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

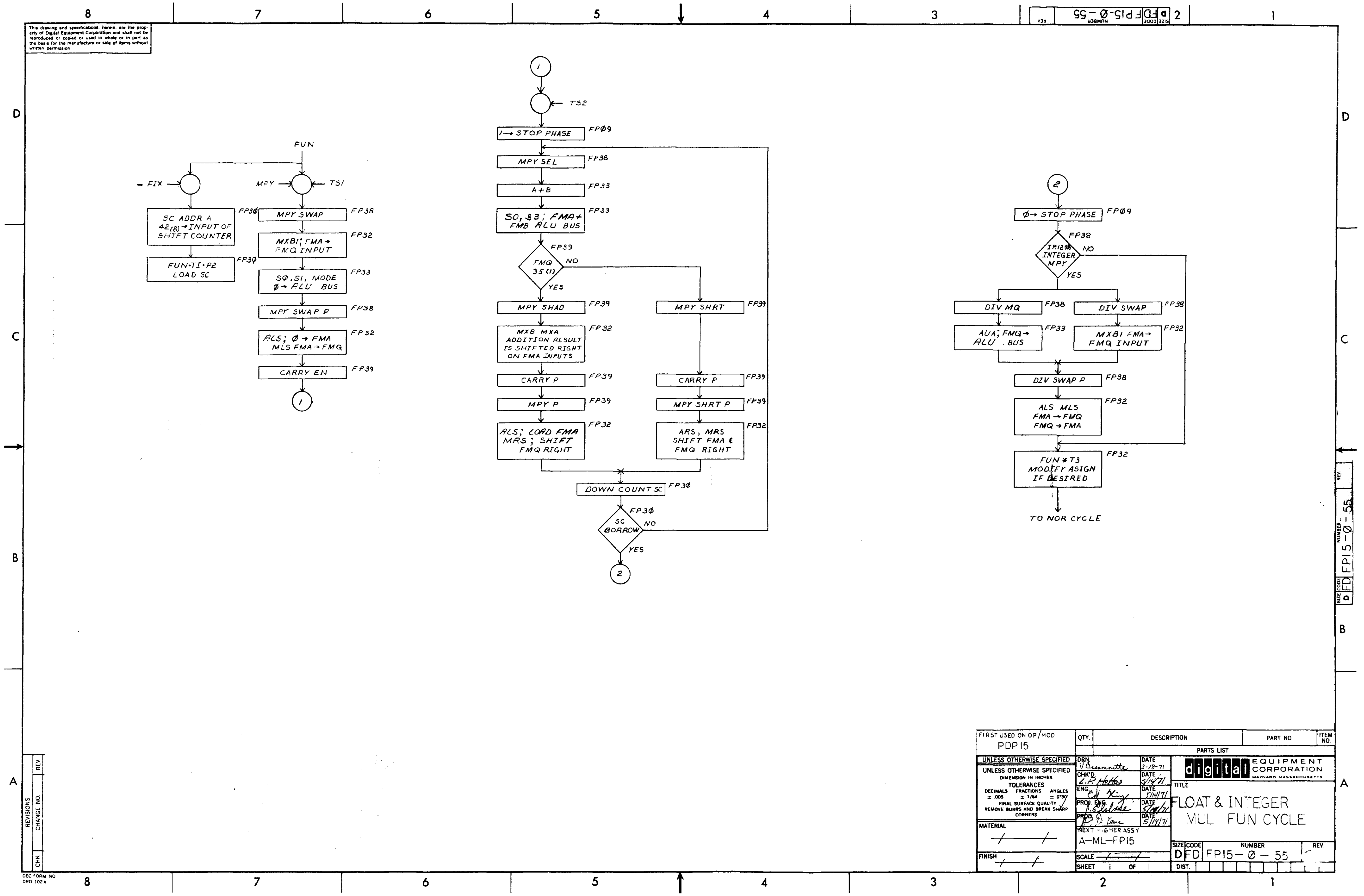
TITLE FLOATING MUL & DIV EXP CYCLE

SIZE	CODE	NUMBER	REV.
D	FD	FP15 — 0 — 54	

SCALE
SHEET OF
DIST.

REVISIONS CHK CHANGE NO. REV.

DEC FORM NO. DRD 100



This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission

FIRST USED ON OP/MOD	PDP15
DRN.	[illegible] 3-18-71
CHK'D.	L.P. Hobbs 5/14/71
ENG.	[illegible] 5/14/71
PROJ. ENG.	[illegible] 5/14/71
PROD.	[illegible] 5/14/71
NEXT HIGHER ASSY	A-ML-FP15
TITLE	FLOAT & INTEGER MUL FUN CYCLE
SIZE / CODE / NUMBER	D FD FP15-Ø-55
SHEET	1 OF 1

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

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

DEC FORM NO DRD 102A

4.8.5.5 Integer Overflow - The only interrupt possible during integer multiply is an integer overflow. After the FMA and FMQ are swapped, the FMQ is examined. If the FMQ is not zero, an overflow interrupt occurs and the interrupt cycle is initiated.

4.9 FLOATING-POINT DIVISION

To perform floating-point division in the FP15, both the dividend and divisor must be normalized. The dividend is normalized in the FUN cycle. The basic functions performed in the division process include calculation of exponents, determination of the sign of the quotient, and division of the mantissas. These are described in detail in the following paragraphs. Refer to Figure 4-6 which represents a simplified flow diagram of floating-point division.

4.9.1 Calculation of Exponents

During the EXP cycle, the contents of the EPA and EPB are gated onto the ALU where the EPB is subtracted from the EPA. The difference is loaded back into the EPA. In floating-point division, the exponent associated with the divisor is subtracted from the exponent associated with the dividend.

4.9.2 Determining Sign of Quotient

The sign of the quotient is determined in the EXP cycle before the mantissas are divided. If the dividend and divisor have the same sign, the sign of the quotient is positive. If the signs are different, the quotient is negative. In either case, the sign of the quotient is stored in A SIGN.

4.9.3 Division of the Mantissas

The dividend mantissa is divided by the divisor by a series of subtractions and left-shifts of the FMA. This process is performed in the FUN cycle and can be reduced to the following rules:

1. Normalize the dividend and divisor. If the divisor is not normalized, an abnormal divide interrupt will occur. To keep track of the number of shifts as a result of normalize, the shift counter is loaded with an octal count of 43_8 (35_{10}). Each shift decrements the counter and, on completion of normalize, the counter is disabled. If more than 35 shifts occur and the number is not normalized, the FMA is 0.

2. Subtract the FMB from the FMA and test the sign of the difference (located in ADD 00):

   a. If the sign is positive,

      - Shift a 1 into the least significant bit of the FMQ.
      - Left-shift and load the FMA with the difference just obtained.

(continued on page 4-20)

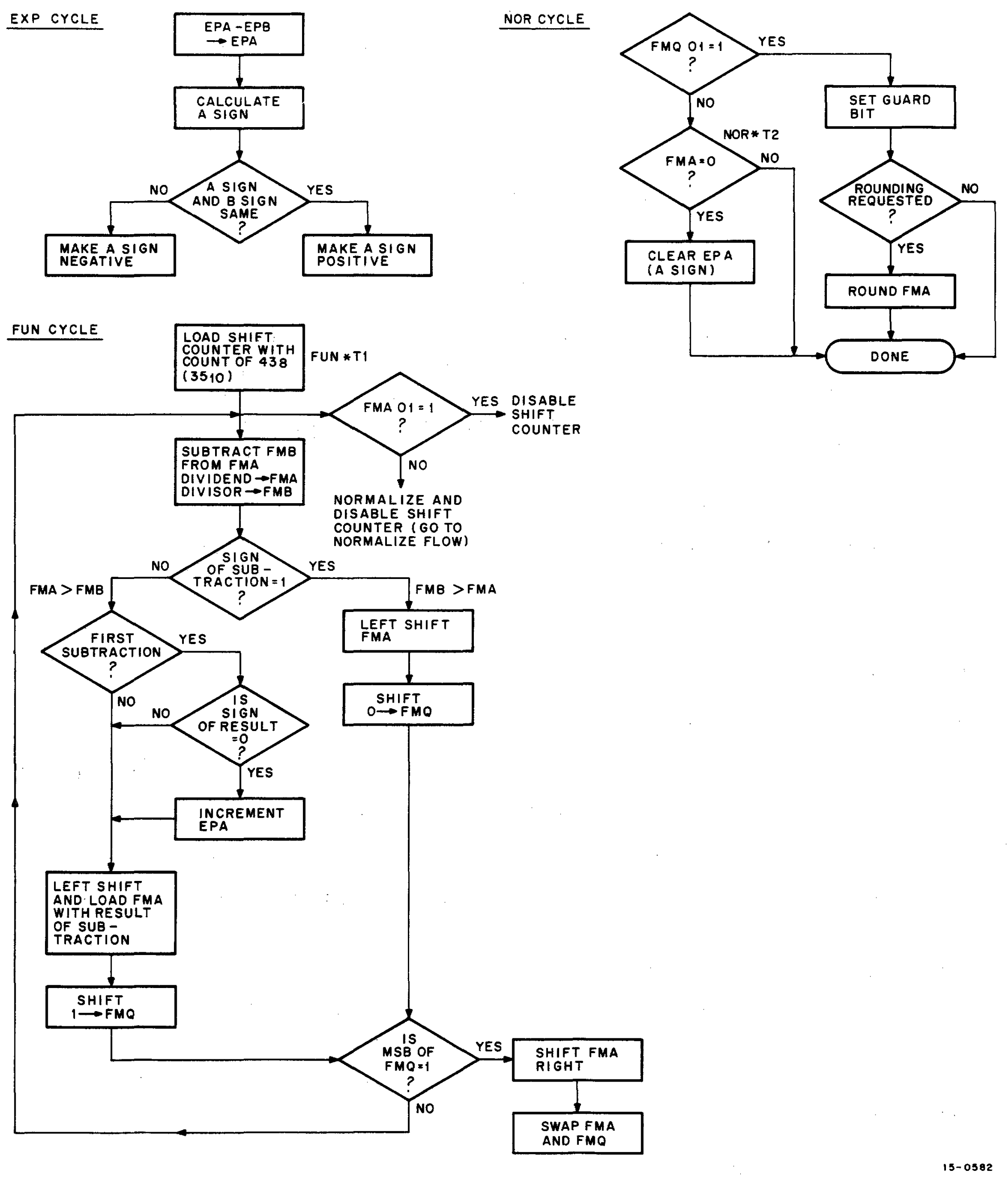


Figure 4-6 Floating-Point Divide Simplified Flow Diagram

a. (continued)

NOTE

If this is first subtraction, and a 0 sign is produced, the EPA is incremented. This condition applies only to the first subtraction.

b. If the sign is negative,

- Shift a 0 into the least significant bit of the FMQ.
- Left-shift the FMA.

3. Test whether the most significant bit of the FMQ is a 1.

   a. If the bit is 1, the division function is complete. Before this fact is detected, the FMA is left-shifted and loaded (if a negative sign) or left-shifted (if a positive sign) and should not have been. It is therefore necessary to shift the FMA right. Otherwise the bit shifted out of the MSB of the FMA will be lost.

   b. If the bit is 0, repeat Steps 2 and 3.

4. Swap the FMA and FMQ. The FMA will now contain the quotient and the FMQ will contain the remainder.

4.9.4 Divide Algorithm

Drawings D-FD-FP15-0-54 and D-FD-FP15-0-57 are flow diagrams of the EXP and FUN cycles during floating-point division. Figure 4-7 is an example of how the divide algorithm is implemented. The number 0.111_2 (0.875_{10}) is divided by 0.101_2 (0.625_{10}). These numbers are loaded in the FMA and FMB, respectively. According to the rules just described, the first step is to subtract the FMB from the FMA, since both numbers are already normalized. The first subtraction produces a 0 sign which causes:

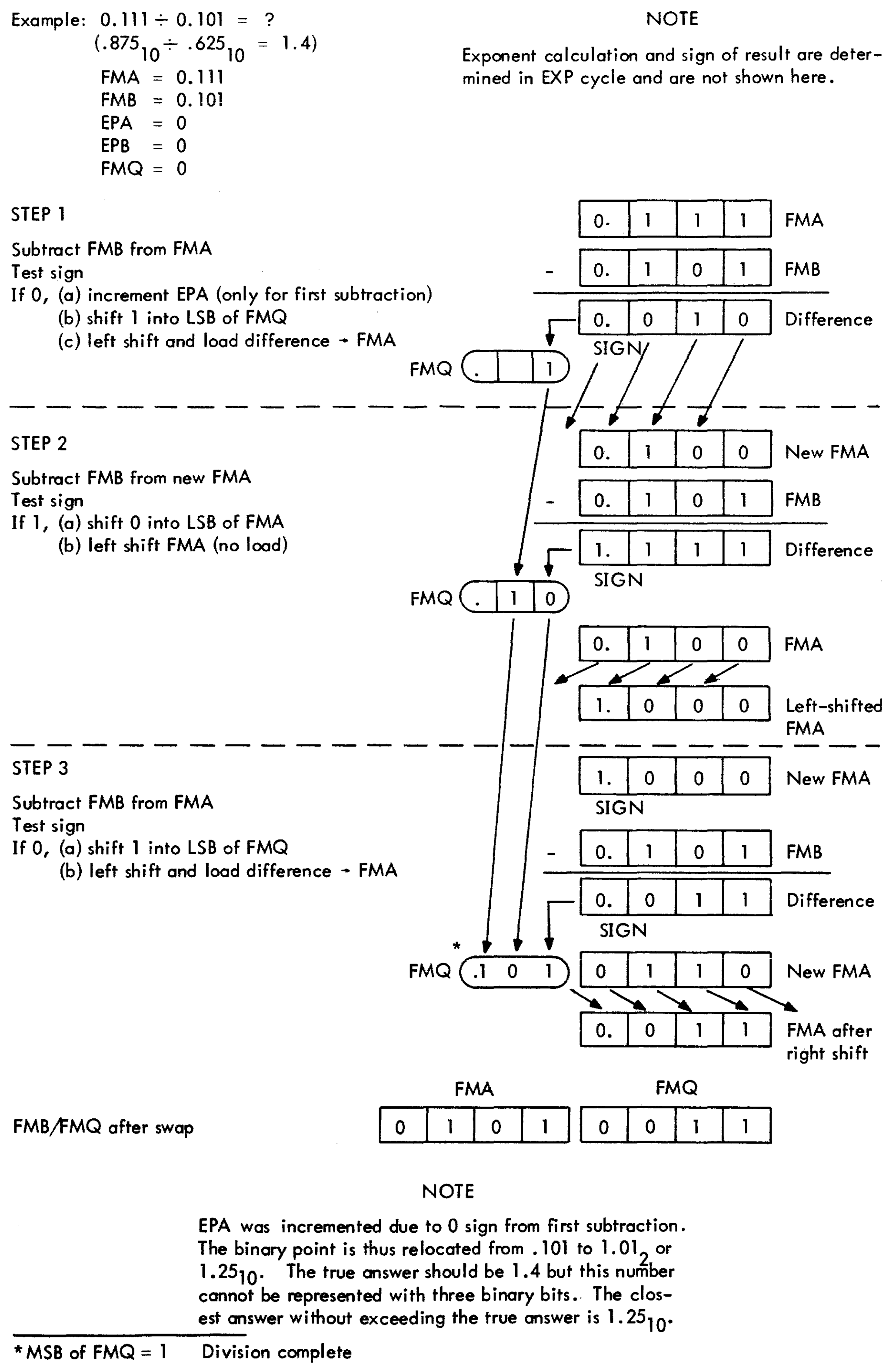
a. the EPA to be incremented,
b. a 1 to be shifted into the FMQ, and
c. the result of the subtraction to be left-shifted and loaded into the FMA.

The most significant bit of the FMQ is not a 1, so the process continues. The second subtraction (FMA-FMB) produces a sign of 1 which causes:

a. a 0 to be shifted into the FMQ, and
b. the FMA to be shifted left.

The most significant bit of the FMQ is still not a 1, so the process continues. The third subtraction produces a 0 sign which causes:

a. a 1 to be shifted into the FMQ, and
b. the result of the subtraction to be left-shifted and loaded into the FMA.



Example: $0.111 \div 0.101 = ?$
$(.875_{10} \div .625_{10} = 1.4)$
FMA = 0.111
FMB = 0.101
EPA = 0
EPB = 0
FMQ = 0

NOTE

Exponent calculation and sign of result are determined in EXP cycle and are not shown here.

STEP 1

Subtract FMB from FMA
Test sign
If 0, (a) increment EPA (only for first subtraction)
(b) shift 1 into LSB of FMQ
(c) left shift and load difference → FMA

STEP 2

Subtract FMB from new FMA
Test sign
If 1, (a) shift 0 into LSB of FMA
(b) left shift FMA (no load)

STEP 3

Subtract FMB from FMA
Test sign
If 0, (a) shift 1 into LSB of FMQ
(b) left shift and load difference → FMA

FMB/FMQ after swap

NOTE

EPA was incremented due to 0 sign from first subtraction. The binary point is thus relocated from .101 to 1.01_2 or 1.25_{10}. The true answer should be 1.4 but this number cannot be represented with three binary bits. The closest answer without exceeding the true answer is 1.25_{10}.

*MSB of FMQ = 1 Division complete

Figure 4-7 Floating Point Divide Algorithm

This condition causes a 1 to appear in the most significant bit of the FMQ indicating the division is complete. However, the FMA has been left-shifted and loaded with the result of the last subtraction. This occurred before it was detected that the divide was complete. As a result, a bit was shifted out of the MSB erroneously. Consequently, the FMA is right-shifted to restore the bit and then the contents of the FMA and FMQ are swapped. The FMA now contains the quotient and the FMQ contains the remainder.

Since the EPA was incremented in the first step, the final answer of $.101_2$ if the FMA is adjusted to 1.01_2. This yields a decimal number of 1.25, whereas the true answer should be 1.4. However, with three bits it is impossible to represent 1.4 in binary form; the closest approximation to this number without exceeding it is 1.25. Much greater accuracy is obtained in the FP15 which uses 36-bit mantissas.

During the NOR cycle, several additional events happen (refer to Drawing D-FD-FP15-0-59). If the MSB of the FMQ is a 1 after the FMA and FMQ are swapped, the guard bit is set, and rounding is requested, +1 is added to the least significant bit of the FMA. If the guard bit is 0, the FMA is checked at NOR*TS02 to see if the FMA is 0. This is done by selecting the ALU for A-B-1 operation, where A represents the FMA, and B = 0 (by being disabled from the ALU). If A = B is true, FMA = 0. In this case, EPA/A SIGN is cleared. With the guard bit set, the zero check of the FMA is not performed.

4.9.5 Interrupts

Five possible interrupt exceptions can occur during floating-point Division: EXP cycle overflow and underflow and FUN cycle overflow, underflow, and abnormal divide. The conditions causing each type are described below.

4.9.5.1 Overflow Interrupt - EXP Cycle

4.9.5.1 Overflow Interrupt - EXP Cycle - An overflow interrupt can occur if a negative EPB is subtracted from a positive EPA with a negative result. An example of this is:

$$\begin{array}{lclclllll}
 & & & & \downarrow\text{SIGN} & & & & \\
300002_8 & = & \text{EPA} & = & 0.11 & 000 & 000 & 000 & 000 \;\; 010 \\
\underline{400000_8} & = & \text{EPB} & = & 1.00 & 000 & 000 & 000 & 000 \;\; 000 \\
\hline
700002_8 & = & \text{Result} & = & 1.11 & 000 & 000 & 000 & 000 \;\; 010
\end{array}$$

The sign bit (EPA 00) going from 0 to 1 is preserved until the FUN cycle. If normalize is requested, it is possible that decrementing the EPA during normalize will remove the overflow condition. If so, an overflow interrupt will not occur. If the overflow condition is not removed, an overflow interrupt will occur at NOR*TS03.

4.9.5.2 Underflow Interrupt - EXP Cycle - An underflow interrupt can occur if a positive EPB is subtracted from a negative EPA with a positive result. An example of this is:

$$\begin{array}{lclcllllll}
 & & & & \text{(EPA 00)} & & & & & \\
 & & & & \downarrow\text{SIGN} & & & & & \\
477777 & = & \text{EPA} & = & 100 & 111 & 111 & 111 & 111 & 111 \\
\underline{377777} & = & \text{EPB} & = & 011 & 111 & 111 & 111 & 111 & 111 \\
\hline
1\ 077776 & & & & 1\ 000 & 111 & 111 & 111 & 111 & 110 \\
 & & & & \uparrow\text{SIGN} & & & & &
\end{array}$$

In effect, two negative quantities are added with a result too small to be shown in the register. The change of sign in the EPA from negative to positive is detected as an underflow. The sign bit (EPA 00 going from a 1 to a 0) is preserved until the NOR cycle where it is possible (if rounding is requested) to eliminate the condition causing the underflow. This is possible only if the EPA underflowed by 1 since rounding only increments the EPA once and only if there was a carry generated out of the ALU. If the underflow condition is not removed, an underflow interrupt will occur at NOR*TS02.

4.9.5.3 Overflow Interrupt - FUN Cycle - It is possible to get an overflow interrupt during the first shift of the divide operation. If the first subtraction produced a 0 SIGN, the EPA is incremented. If the EPA contained 377777_8 and is incremented to 400000_8, an overflow interrupt will occur at NOR*TS03. This is detected as a result of the sign bit (EPA 00) going from a 0 to 1 condition.

4.9.5.4 Underflow Interrupt - FUN Cycle - If normalize is requested, it is performed during the FUN cycle for floating-point Division. As the mantissa is left-shifted, the EPA is decremented. During normalize, if the EPA should be decremented from 400000 to 377777, an underflow interrupt will occur at NOR*TS03. This is detected as a result of EPA 00 going from a 1 to a 0.

4.9.5.5 Abnormal Divide - FUN Cycle - If the most significant bit of the divisor (FMB) is not a 1, an abnormal divide interrupt is initiated indicating an unnormalized or 0 FMB. This interrupt is not delayed until NOR*TS03 as is the case with overflow and underflow interrupts. The interrupt is raised immediately at FUN*TS01.

4.10 FLOATING POINT REVERSE DIVIDE

In a Divide instruction, the dividend is loaded into the FMA by a Load instruction and the divisor is loaded into the FMB by the Divide instruction. However, assume that as a result of some previous operation, a number which is to be used as a divisor is left in the FMA. In this case, a Reverse Divide instruction can be issued that gates the divisor from the EPA/A SIGN/FMA to the EPB/B SIGN/FMB during the FETCH cycle and loads the dividend into the EPA/A SIGN/FMA.

4.11 INTEGER DIVISION

Integer division in the FP15 is accomplished during the EXP and FUN cycles. The most significant bits of the dividend and divisor must be 1s (normalized) before the actual division can be performed. Because of the integer divide algorithm, the dividend must be larger than the divisor for integer division; otherwise, the quotient is fractional and the FMA is ultimately zeroed.

The dividend is loaded into the A SIGN/FMA as a result of the Load instruction; the divisor is loaded into the B SIGN/FMB as a result of the Integer Divide (IDV or EDV) instruction. If the divisor is negative, it is converted to sign and magnitude format.

4.11.1 EXP Cycle

Normalization of the dividend and divisor is performed in the EXP cycle. The FMA contains the dividend and the FMB contains the divisor; the contents of the FMB are then transferred to the FMQ.

If the most significant bits of the FMA and FMQ are 1s, nothing further occurs during the EXP cycle except that the contents of the FMQ are transferred back to the FMB. Three other possible conditions that can occur are:

a. If the MSB of the FMA is a 1 and the MSB of the FMQ is not a 1, the FMQ is shifted left. Each left-shift causes the EPA to be incremented. The process is terminated when the MSB of the FMQ becomes a 1.

   Example:

EPA	FMA	FMQ
0 0 0	1 0 0	0 0 1
0 0 1	1 0 0	0 1 0
0 1 0	1 0 0	1 0 0

b. If the MSB of the FMQ is a 1 and the MSB of the FMA is not a 1 the FMA will be cleared since the divisor is larger than the dividend. No integer divide will occur.

c. If neither the MSB of the FMA nor FMQ is a 1 both are shifted left. If the MSB of the FMQ becomes a 1 before the MSB of the FMA, this relates back to Step 2 and no integer divide can occur. If the MSB of the FMA becomes a 1 before the FMQ, the FMQ will continue to be shifted left; however, the EPA is incremented for each left-shift of the FMQ not accompanied by a left-shift of the FMA.

   When both the MSB of the FMA and FMQ are 1s, the contents of the FMQ are transferred back to the FMB and the EXP cycle is concluded.

4.11.2 FUN Cycle

In the FUN cycle, the actual division process consists of a series of subtractions which, depending on the sign of the difference, cause the FMA to be (1) left-shifted or (2) left-shifted and loaded with the difference just obtained. Again, depending on the sign, FMQ 35 is set temporarily storing the quotient. The algorithm can be reduced to the following set of rules:

a. Load the shift counter with the value of the EPA obtained during the EXP cycle.

b. Clear the FMQ.

c. Subtract the FMB from the FMA.
   1. If the sign of the difference (AD 00) is positive:
      - Transfer 1 to the LSB of the FMQ
      - Left-shift and load the FMA with the difference obtained
      - Increment the EPA, if this is the first subtraction. This increment of the EPA is performed merely as a matter of routine for integer divide, and is primarily used for floating-point Division.
   2. If the sign is negative:
      - Transfer a 0 to the LSB of the FMQ
      - Left-shift the FMA

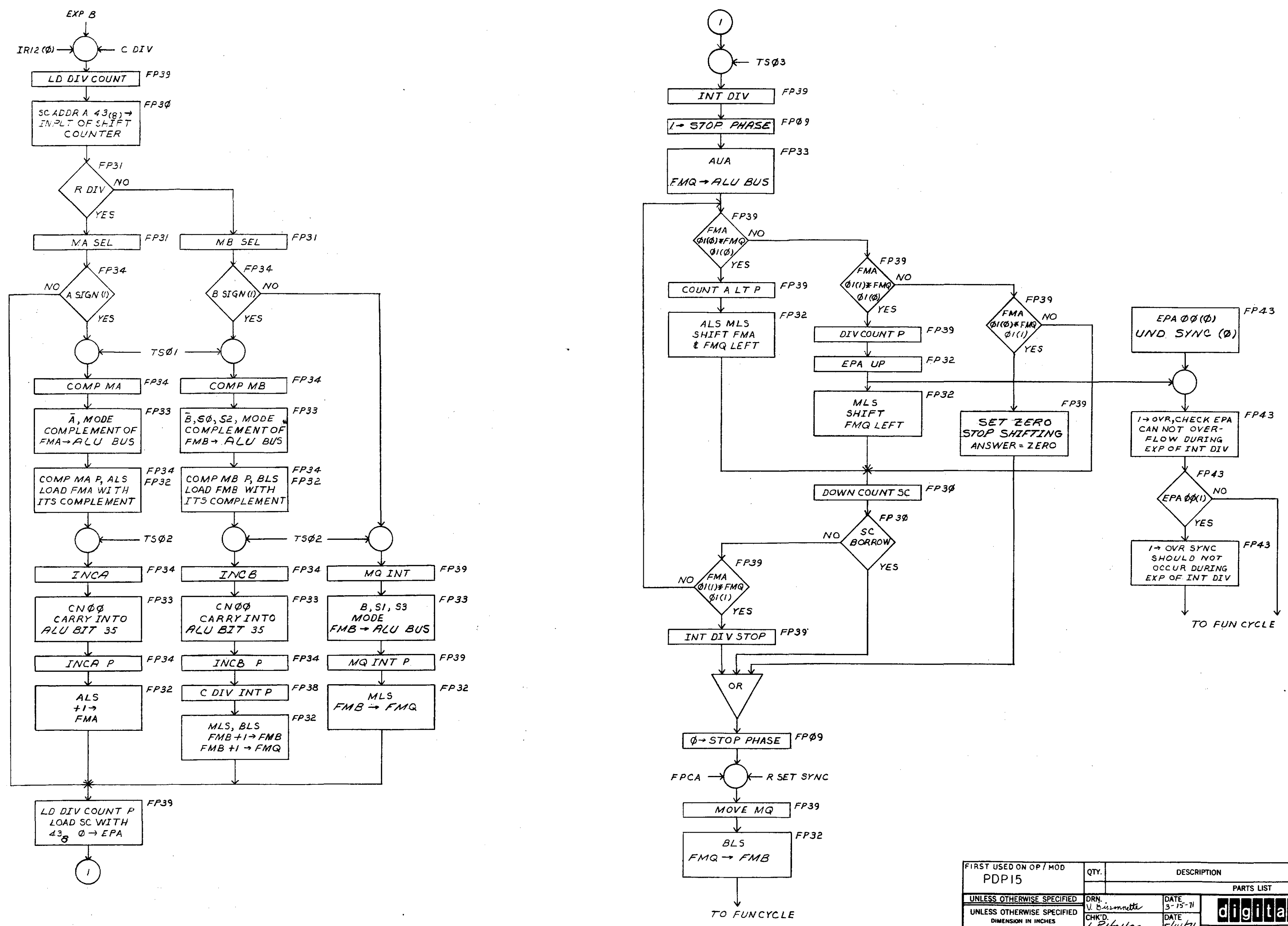
d. If the division is integer, decrement the shift counter and check for a borrow.
   1. If no borrow occurs, go back to Step 3 and repeat the process.
   2. If a borrow is generated, the divide function is completed. However, the last left-shift or left-shift and load was performed before the borrow was detected; this causes the MSB to be shifted out of the FMA and an erroneous remainder would result. The FMA is shifted right to correct the condition.

e. Swap the contents of the FMA and FMQ. The quotient is now in the FMA and the remainder in the FMQ.

4.11.3 Divide Algorithm

Drawing D-FD-FP15-0-56 is a flow diagram of integer divide during the EXP cycle and Drawing D-FD-FP15-0-57 shows the flow during the FUN cycle. A better understanding of integer divide can be obtained by reviewing the rules just described using the flow diagram for reference. An example of integer divide using two 3-bit numbers is shown in Figure 4-8. For clarity, only those registers that change as a result of a particular action are shown.

This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.



EXP B
IR12 (Ø)
C DIV
LD DIV COUNT FP39
SC ADDR A 43(8) → INPUT OF SHIFT COUNTER FP3Ø
FP31
R DIV
NO
YES
MA SEL FP31
MB SEL FP31
FP34
A SIGN (1)
B SIGN (1)
TSØ1
COMP MA FP34
COMP MB FP34
Ā, MODE COMPLEMENT OF FMA → ALU BUS FP33
B, SØ, S2, MODE COMPLEMENT OF FMB → ALU BUS FP33
COMP MA P, ALS LOAD FMA WITH ITS COMPLEMENT FP34 FP32
COMP MB P, BLS LOAD FMB WITH ITS COMPLEMENT FP34 FP32
TSØ2
INCA FP34
INCB FP34
MQ INT FP39
CNØØ CARRY INTO ALU BIT 35 FP33
B, S1, S3 MODE FMB → ALU BUS FP33
INCA P FP34
INCB P FP34
MQ INT P FP39
ALS +1 → FMA FP32
C DIV INT P FP38
MLS FMB → FMQ FP32
MLS, BLS FMB +1 → FMB FMB +1 → FMQ FP32
LD DIV COUNT P LOAD SC WITH 43(8) Ø → EPA FP39
1
TSØ3
INT DIV FP39
1 → STOP PHASE FPØ9
AUA FMQ → ALU BUS FP33
FMA Ø1(Ø) ≠ FMQ Ø1(Ø) FP39
COUNT A LT P FP39
ALS MLS SHIFT FMA & FMQ LEFT FP32
FMA Ø1(1) ≠ FMQ Ø1(Ø) FP39
DIV COUNT P FP39
EPA UP FP32
MLS SHIFT FMQ LEFT FP32
FMA Ø1(Ø) ≠ FMQ Ø1(1) FP39
SET ZERO STOP SHIFTING ANSWER = ZERO FP39
EPA ØØ(Ø) UND SYNC (Ø) FP43
1 → OVR, CHECK EPA CAN NOT OVERFLOW DURING EXP OF INT DIV FP43
EPA ØØ(1) FP43
1 → OVR SYNC SHOULD NOT OCCUR DURING EXP OF INT DIV FP43
TO FUN CYCLE
DOWN COUNT SC FP3Ø
SC BORROW FP3Ø
FMA Ø1(1) ≠ FMQ Ø1(1) FP39
INT DIV STOP FP39
OR
Ø → STOP PHASE FPØ9
FPCA
R SET SYNC
MOVE MQ FP39
BLS FMQ → FMB FP32
TO FUN CYCLE

FIRST USED ON OP/MOD	QTY.	DESCRIPTION	PART NO.	ITEM NO.
PDP15				

PARTS LIST

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

DRN.	V. Bissonnette	DATE 3-15-71
CHK'D.	L. P. Hottos	DATE 5/14/71
ENG.	[illegible]	DATE 5/14/71
PROJ. ENG.	[illegible]	DATE 5/14/71
PROD.	[illegible]	DATE 5/19/71

MATERIAL —/—/—
FINISH —/—/—
FIRST USED ON A-ML-FP15
SCALE —/—/— SHEET OF

digital EQUIPMENT CORPORATION MAYNARD, MASSACHUSETTS

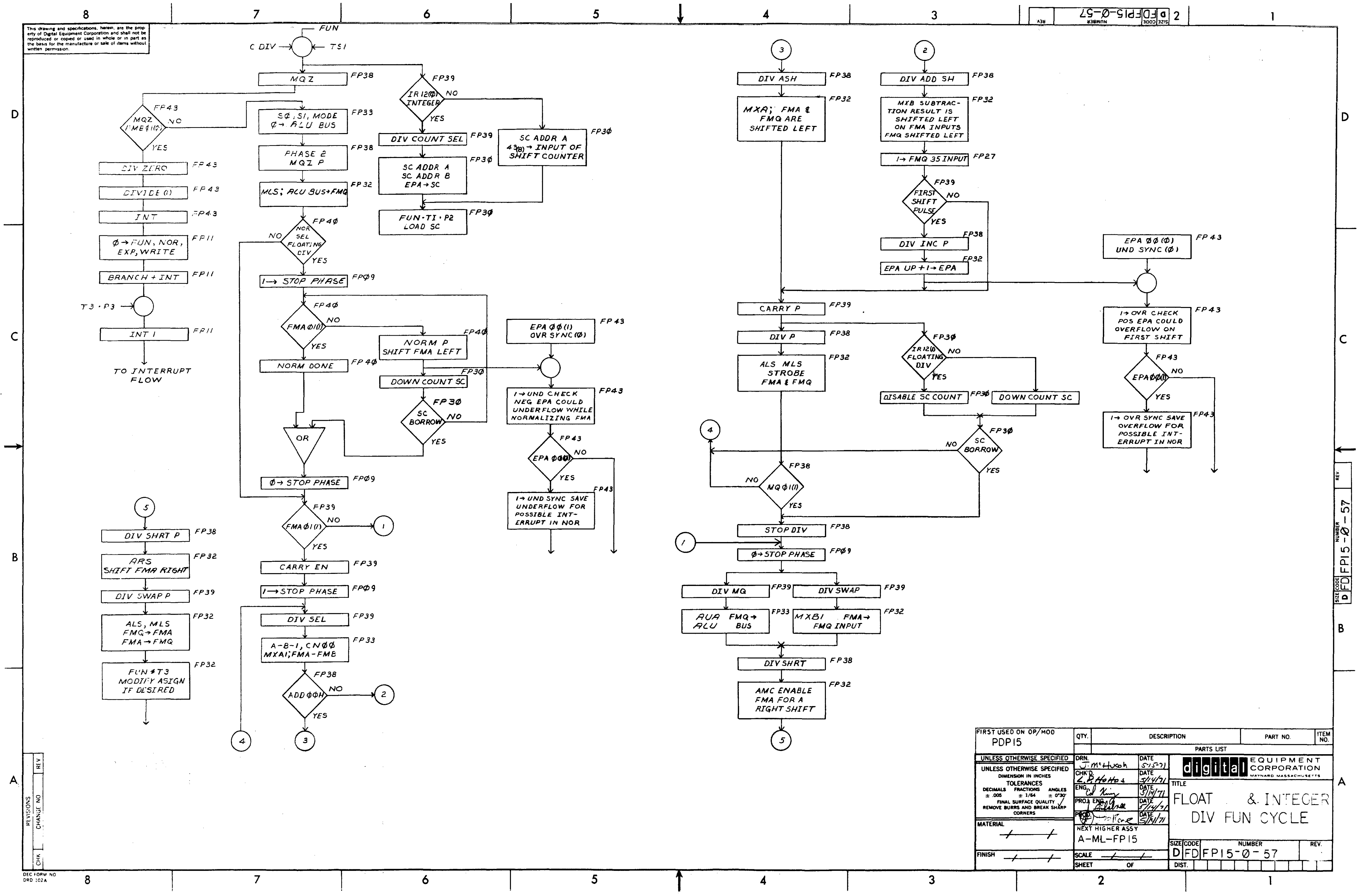
TITLE INTEGER DIVIDE EXP CYCLE

SIZE D CODE FD NUMBER FP15-Ø-56 REV.

DIST.

REVISIONS REV CHANGE NO CHK

DEC FORM NO DRD 102A



C DIV
FUN
TSI
MQZ FP38
MQZ I'ME Φ1(Φ) FP43 NO YES
DIV ZERO FP43
DIVIDE (I) FP43
INT FP43
Φ→ FUN, NOR, EXP, WRITE FP11
BRANCH + INT FP11
T3 · P3
INT I FP11
TO INTERRUPT FLOW
SΦ, SI, MODE Φ→ ALU BUS FP33
PHASE 2 MQZ P FP38
MLS; ALU BUS→FMQ FP32
NOR SEL FLOATING DIV FP4Φ NO YES
I→ STOP PHASE FPΦ9
FMA Φ1(Φ) FP4Φ NO YES
NORM P SHIFT FMA LEFT FP4Φ
NORM DONE FP4Φ
DOWN COUNT SC FP3Φ
SC BORROW FP 3Φ NO YES
OR
Φ→ STOP PHASE FPΦ9
FMAΦ1(I) FP39 NO YES
1
CARRY EN FP39
I→ STOP PHASE FPΦ9
DIV SEL FP39
A-B-I, CNΦΦ MXAI;FMA-FMB FP33
ADD ΦΦH FP38 NO YES
2
4
3
IR 12(Φ) INTEGER FP39 NO YES
DIV COUNT SEL FP39
SC ADDR A SC ADDR B EPA→SC FP3Φ
SC ADDR A 43(8)→ INPUT OF SHIFT COUNTER FP3Φ
FUN·TI·P2 LOAD SC FP3Φ
EPA ΦΦ(I) OVR SYNC (Φ) FP43
I→ UND CHECK NEG EPA COULD UNDERFLOW WHILE NORMALIZING FMA FP43
EPA ΦΦ(Φ) FP43 NO YES
I→ UND SYNC SAVE UNDERFLOW FOR POSSIBLE INT-ERRUPT IN NOR FP43
5
DIV SHRT P FP38
ARS SHIFT FMA RIGHT FP32
DIV SWAP P FP39
ALS, MLS FMQ→FMA FMA→FMQ FP32
FUN * T3 MODIFY ASIGN IF DESIRED FP32
3
DIV ASH FP38
MXA; FMA & FMQ ARE SHIFTED LEFT FP32
2
DIV ADD SH FP38
MXB SUBTRACTION RESULT IS SHIFTED LEFT ON FMA INPUTS FMQ SHIFTED LEFT FP32
I→ FMQ 35 INPUT FP27
FIRST SHIFT PULSE FP39 NO YES
DIV INC P FP38
EPA UP+I→EPA FP32
CARRY P FP39
DIV P FP38
ALS MLS STROBE FMA & FMQ FP32
IR 12(Φ) FLOATING DIV FP3Φ NO YES
DISABLE SC COUNT FP3Φ
DOWN COUNT SC
SC BORROW FP3Φ NO YES
4
MQ Φ1(I) FP38 NO YES
STOP DIV FP38
1
Φ→STOP PHASE FPΦ9
DIV MQ FP39
DIV SWAP FP39
AUA FMQ→ ALU BUS FP33
MXBI FMA→ FMQ INPUT FP32
DIV SHRT FP38
AMC ENABLE FMA FOR A RIGHT SHIFT FP32
5
EPA ΦΦ(Φ) UND SYNC (Φ) FP43
I→ OVR CHECK POS EPA COULD OVERFLOW ON FIRST SHIFT FP43
EPAΦΦ(I) FP43 NO YES
I→ OVR SYNC SAVE OVERFLOW FOR POSSIBLE INT-ERRUPT IN NOR FP43

This drawing and specifications, herein, are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.

FIRST USED ON OP/MOD	PDP15
DRN.	J. McHugh — DATE 5/5/71
CHK'D	[illegible] — DATE 5/14/71
ENG.	[illegible] — DATE 5/17/71
PROJ. ENG.	[illegible] — DATE 5/14/71
NEXT HIGHER ASSY	A-ML-FP15
TITLE	FLOAT & INTEGER DIV FUN CYCLE
SIZE CODE NUMBER	D FD FP15-Φ-57

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

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

DEC FORM NO DRD 102A

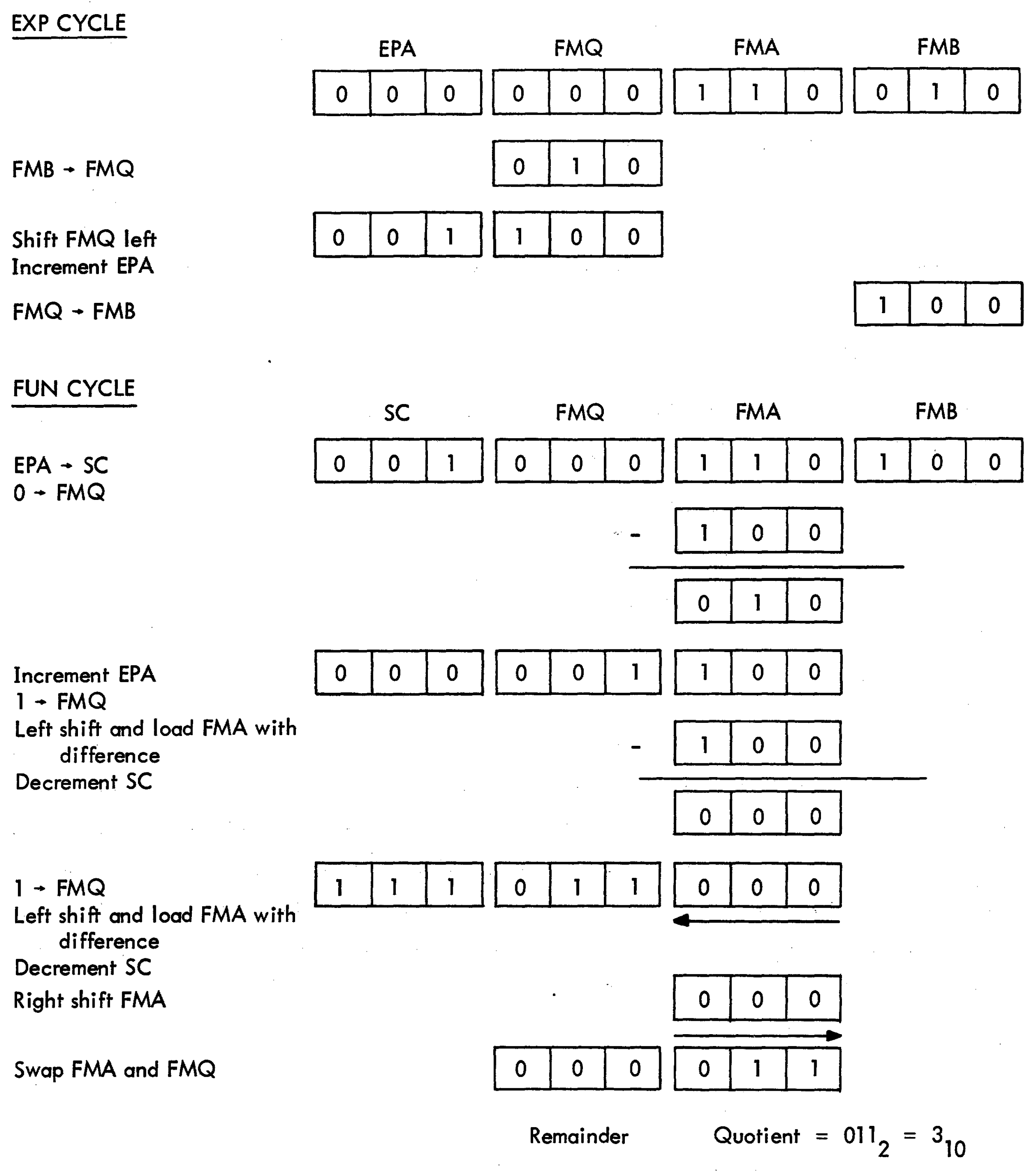


Figure 4-8 Integer Divide Algorithm

4.11.4 Interrupt Exception - Abnormal Divide

The only interrupt that can occur as a result of performing integer division is abnormal divide. Abnormal divide occurs if the most significant bit of the FMB is 0 (FMB 01=0). The abnormal divide interrupt flag is raised immediately at FUN*TS01.

4.12 INTEGER REVERSE DIVISION

In a Divide instruction the dividend is loaded into the FMA with a Load instruction and the divisor is loaded into the FMB by a Divide instruction. If, as a result of some previous computation, the proposed divisor is in the FMA, a Reverse Divide instruction can be issued. This instruction causes the divisor to be gated from the FMA to the FMB during the FETCH cycle and causes the dividend to be loaded into the FMB.

4.13 INTEGER STORE

For single-precision Integer Store instructions, A SIGN and bits 19 through 35 of the FMA are stored in 2's complement format at the argument address (refer to Drawing D-FD-FP15-0-58). For extended integer Store, A SIGN and bits 01 through 35 of the FMA are stored in 2's complement format in two locations starting at the argument address. If the result of an arithmetic operation resulted in a negative answer, the answer is converted to 2's complement format prior to being written into memory. Two's complementing is accomplished by 1's complementing the negative answer in FUN*TS02 and incrementing this value in NOR*TS01. No operands are fetched from memory during a Store instruction.

4.13.1 Overflow Interrupt

If any of the high-order bits (bits 00-18) are a 1 during a single Integer Store, an overflow interrupt is initiated at FUN*TS01. No interrupts are possible with double Integer Store instructions.

4.14 FLOATING-POINT STORE

For single-precision floating-point Store instructions, the first word is stored in 2's complement format at the argument address and consists of bits 09 through 17 of the EPA register and bits 18 through 26 of the FMA. The second word consists of A SIGN and bits 01 through 17 of the FMA and is stored in the argument address plus one. For double-precision floating-point instructions, the first word is stored in the argument address and consists of bits 00 through 17 in the EPA register. The second word is stored in the argument address plus one and consists of A SIGN and bits 01 through 17 of the FMA. The third word is stored in the argument address plus two and consists of bits 18 through 35 of the FMA. Floating-point Store instructions require no fetch from memory.

Normalize, if requested, occurs at FUN*TS01 and rounding, if requested, occurs at NOR*TS01. Rounding of double-precision floating-point Store instructions cannot be specified. If rounding is requested for a single-precision floating-point Store instruction, bit 27 of the FMA is examined. If it is a 1, 1 is added to the FMA, bit 26. If bit 27 is a 0, no rounding occurs. Bits 27 through 35 are then zeroed. The following interrupt exceptions can occur during a single-precision floating-point Store instruction. The only interrupt exception that can occur during a double-precision floating-point Store instruction is an underflow interrupt due to normalize, which occurs at NOR*TS03.

4.14.1 EPA Underflow or Overflow Interrupt

During a single-precision floating-point Store instruction, either an EPA overflow or EPA underflow interrupt can occur at NOR*TS02*PH03. If the EPA is positive, the high-order bits (bits 01 through 08) of the EPA are checked. A 1 in any of these bit positions initiates a temporary overflow. If the EPA is negative, then the high-order bits of the EPA are checked for 0s. A 0 in any one of these bit positions initiates a temporary underflow.

4.14.2 Underflow Interrupt Due to Normalize

If normalize is requested, the FMA is left-shifted and the EPA is decremented. If the EPA contains 400000_8 and is decremented to 377777_8, an underflow interrupt occurs at NOR*TS03. It is possible that the condition causing the EPA overflow interrupt at NOR*TS03*PH03 is eliminated when the EPA is decremented during normalize. If so, no interrupt is raised. If not, the temporary EPA overflow interrupt becomes permanent and is raised at NOR*TS03. The normalize underflow interrupt can occur for both single- and double-precision floating-point Store instructions.

4.14.3 Overflow Interrupt Due to Rounding

If rounding is requested, for a single-precision floating-point Store instruction, FMA bit 27 is examined. If it is a 1, 1 is added to FMA bit 26. Should a carry occur out of the ALU as a result of this operation, the FMA is right-shifted and the EPA is incremented. If the EPA contained 000377_8 and is incremented to 000400_8, an overflow interrupt is raised at NOR*TS03.

It is possible that the condition causing the EPA underflow interrupt at NOR*TS02*PH03 can be eliminated if the EPA is incremented during a rounding request. This condition can occur only if the EPA just underflowed as the EPA can only be incremented once due to rounding.

4.15 SWAP, LOAD AND SWAP

The Swap instruction swaps the contents of the FMA and the FMQ. If the instruction is a Load and Swap, the operand from memory is loaded into the FMA and then the contents of the FMA and FMQ are swapped.

Drawing D-FD-FP15-0-60 is a flow diagram of the Swap instruction. The swap occurs at FUN*TS01*PH03. The contents of the FMQ are gated to the A side of the ALU bus, and the contents of the FMA are gated into the FMQ. The A side of the ALU bus is enabled through the ALU by default (nothing specified), and the ALU output is strobed into the FMA completing the swap.

4.15.1 Underflow Interrupt

If, as a result of normalize, the EPA is decremented from 400000_8 to 377777_8, an underflow interrupt will occur and the interrupt cycle is initiated.

4.16 FLOAT, LOAD AND FLOAT FMA

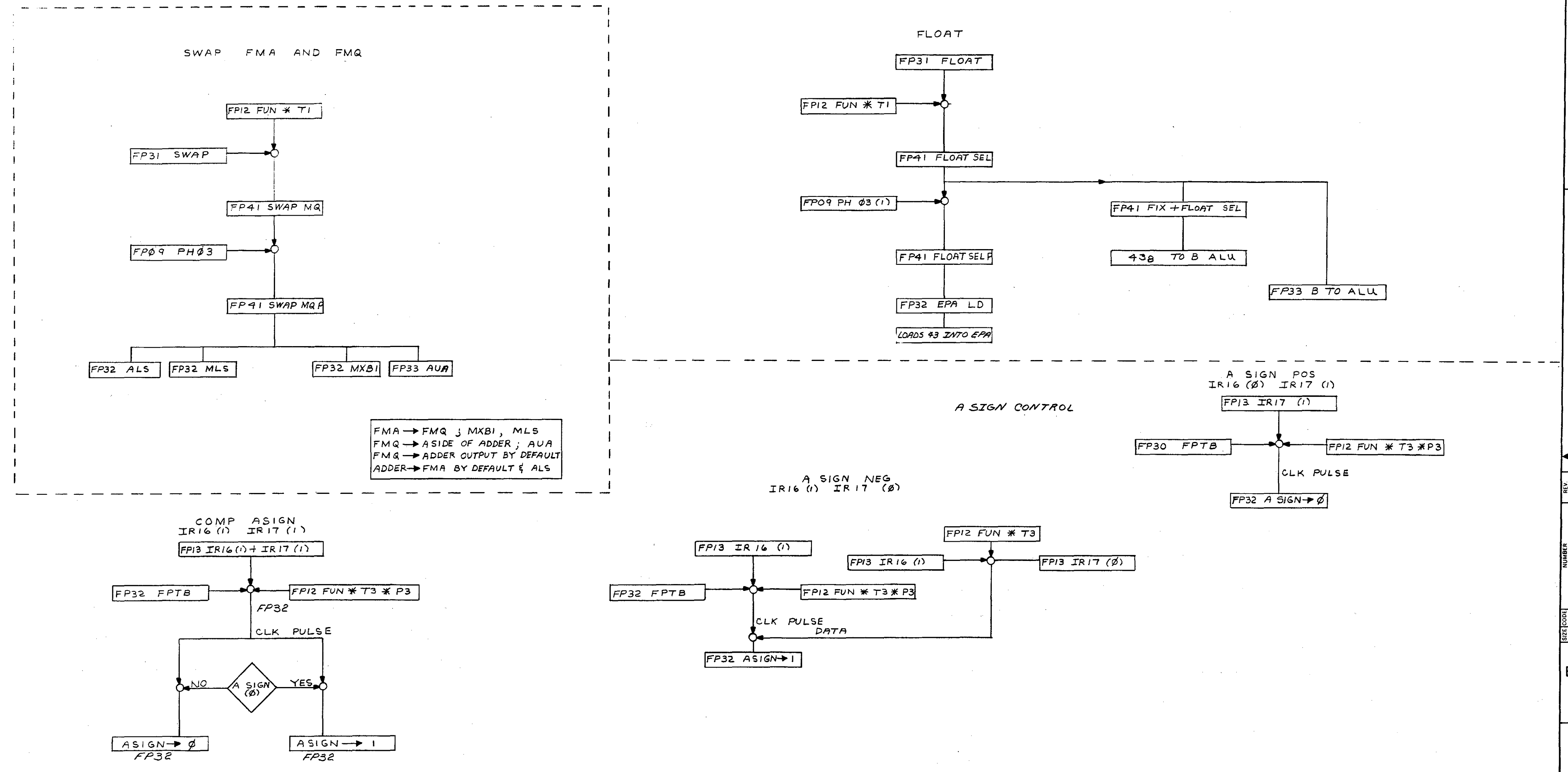
The two basic types of Float instructions are:

a. Load and Float FMA, and
b. Float FMA

The Float class of instructions convert integer format to floating-point format. The Load and Float instructions require a memory reference cycle(s) to fetch an operand(s) from memory. The Float FMA instruction merely floats the existing contents of the FMA with no operand fetch involved. Floating an integer is accomplished simply by loading the EPA with 43_8, which effectively relocates the binary point to the left of the number. The integer is thus converted to a floating-point number--the mantissa is contained in the FMA and the exponent of 35_{10} is contained in the EPA. The following example shows the integer 5_8 being converted to a floating-point number. The EPA is loaded with 3 since a 3-bit integer and 3-bit EPA and FMA have been shown for simplicity.

$$\text{EPA} = \boxed{011} \qquad \text{FMA} = \boxed{0.101} = 0.101_2 \times 2^3 = 5_8$$

Drawing D-FD-FP15-0-60 is a flow diagram of the Float instruction. If a Float instruction is specified, a signal called FLOAT SELECT is generated at FUN*TS01. At FUN*TS01*PH03, a Float Select P signal causes the EPA to be loaded from the ALU bus with a constant of 43_8 (35_{10}).



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL

FINISH

DRN.	E. Wilson	DATE 3/22/71
CHK'D.	L. P. Holtos	DATE 5/14/71
ENG.	[illegible]	DATE 5/14/71
PROJ. ENG.	[illegible]	DATE 5/14/71
PROD.	[illegible]	DATE 5/14/71

NEXT HIGHER ASSY.

SCALE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE FLOAT, SWAP AND ASIGN CONTROL

SIZE	CODE	NUMBER	REV.
D	FD	FP15 - Ø - 6Ø	

DIST.

REVISIONS: REV | CHANGE NO | CHK

DEC FORM NO DRD 102-B

In order to load the constant into the EPA, it is first specified at the input to the B multiplexer by a FIX or FLOAT SEL signal. An AUA1 signal enables the 43_8 to the output of the ALU. This is accomplished by forcing S0 and S2 low and S1, S3 and MODE high.

4.17 FIX, LOAD AND FIX

The Fix or Load and Fix instructions convert floating-point format to integer format. If the instruction is a Fix, no memory reference is required. An example of this is the FIX EPA (FMA) instruction that converts the existing contents of the FMA to integer format. If the instruction is a Load and Fix, a memory reference is required to load the FMA with the operand from memory.

Drawing D-FD-FP15-0-61 is a flow diagram for the Fix type instruction. The diagram is divided into two major branches--one for a positive EPA and one for a negative EPA. If the EPA contains a negative number, the floating-point number is a fraction that cannot be converted to an integer and the FMA is cleared.

NOTE

At FUN*TS01, FIX ZERO is generated if the EPA is negtive. This signal forces a logical zero on the ALU bus and at Phase 2, a FIX ZERO P signal strobes the ALU output (zero) to the FMA resulting in a zero FMA.

If the EPA is positive, the floating-point number can be converted to an integer and the Fix operation is initiated. The ALU is selected for A-B-1 operation during the FUN cycle. "A" represents the EPA, and "B" represents a special constant that is 35_{10} for a Fix instruction. At this point a test is made to determine if the EPA is equal to 43_8. If so, the Fix operation is completed. If not, 43_8 is subtracted from the EPA and the difference is loaded into the shift counter from the ALU bus. If the difference is positive (EPA 43_8), the number cannot be fixed since 35_{10} or more shifts would shift the number completely out of the FMA; in this case, a Fix Overflow is generated and the Fix operation ceases. An interrupt sequence is initiated due to the overflow resulting from EPA 43_8. The interrupt sequence consists of INT 1 and INT 2 cycles that lead to a service routine in the CPU associated with the overflow. The interrupt sequence is described more fully in Chapter 3.

If the difference between the EPA and 43_8 is negative, the operand can be converted from floating-point to integer and the Fix operation proceeds. The shift counter is loaded with the negative quantity that results from EPA-43_8 (where EPA < 43_8). Logic on FP09 causes the FP15 to stop in TS02*PH03 of the FUN cycle. At this time, the FMA and FMQ right-shifting process is initiated. Shifting is accomplished by the FIX SHMA P signal that is generated for each shift.

The shift counter is incremented each time a shift occurs. The counter is tested after each shift to see if a carry is generated. If not, the FMA and FMQ are shifted until a carry is generated. At this point, the FMA and FMQ have been shifted the required number of places to fix the floating-point number. The operation is concluded by "resetting" the logic on FP09 to allow continuation of the phase and time states.

4.18 LOAD JEA AND STORE JEA

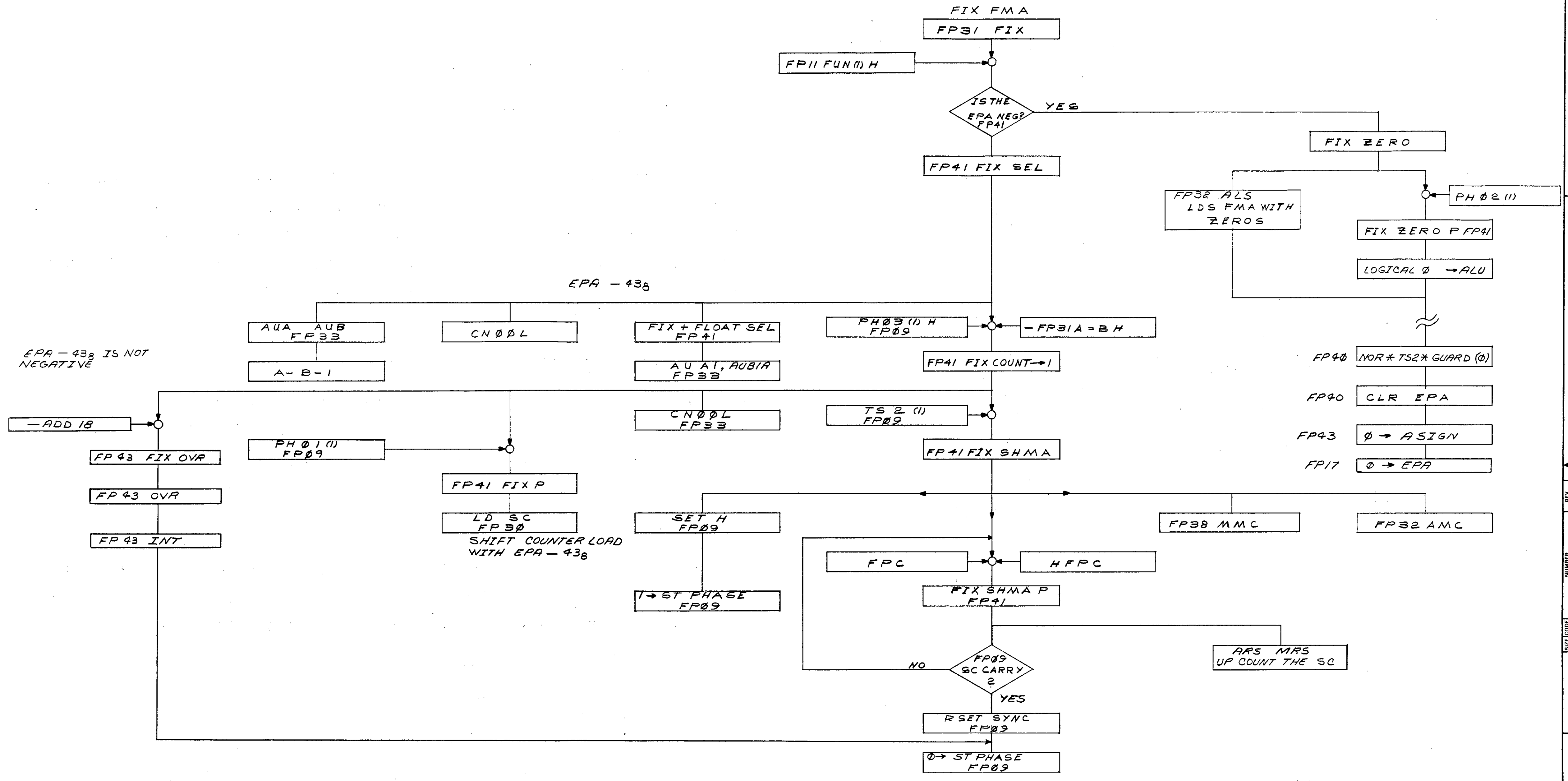
The Load JEA instruction loads the JEA register (bits 07-17) from bits 21-35 of the BMB. The guard bit is loaded from BMB 19.

The JEA register is loaded by a LD JMS P signal that occurs during FUN*TS01.

The Store JEA instruction occurs during the WRITE cycle where the operand is written into memory (see Paragraph 2.10). JEA bits 03-17 are gated to MPO bits 03-17, and the A SIGN and guard bits are gated to MPO bits 00 and 01, respectively.

4.19 BRANCH

The Branch instruction provides the programmer with a means of altering the program sequence. Bits 13-17 of the instruction word are used as a mask to test for certain conditions such as zero or non-zero FMA, positive or negative A SIGN, and FMA carry. Figure 4-9 is a simplified flow diagram of the instruction. As an example, assume the programmer wishes to test for FMA = 0 and to branch if it is. The test mask would have bit 17 on a 1 to test the FMA. If the FMA is 0 and a Branch instruction has been specified, the Branch test is successful. If indirection has been specified, the indirect cycle must be completed. This is indicated by CHANGE H which occurs when no indirection is specified or when indirection is specified and has been completed. The FP15 enters an INT 1 cycle that forces the CPU to begin execution of a JMP*0 instruction. The INT 2 cycle is initiated and the FP15 forces the CPU to accept the contents of the address register that contains the address specified by the Branch instruction. If the Branch is not successful, the instruction is exited, and is cleared at the end of FETCH*TS03*PH03. Indirection, if specified, must be completed before BUSY is cleared. The INT 2 cycle is completed at INT 2*TS03*PH03 to complete the instruction. The Branch instruction can be microprogrammed on an inclusive OR basis.



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED
DIMENSION IN INCHES.
TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL
FINISH

		DATE
DRN.	[illegible]	4-20-71
CHK'D.	L.P. Hollos	5/14/71
ENG.	[illegible]	5/14/71
PROJ. ENG.	[illegible]	5/14/71
PROD.	[illegible]	5/14/71

NEXT HIGHER ASSY.

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE
FIX FLOW

SIZE	CODE	NUMBER	REV.
D	FD	FP15-Ø-61	

SCALE NONE
SHEET 1 OF 1
DIST.

REVISIONS: REV | CHANGE NO. | CHK

DEC FORM NO DRD 102-B

D FD FP15-Ø-61

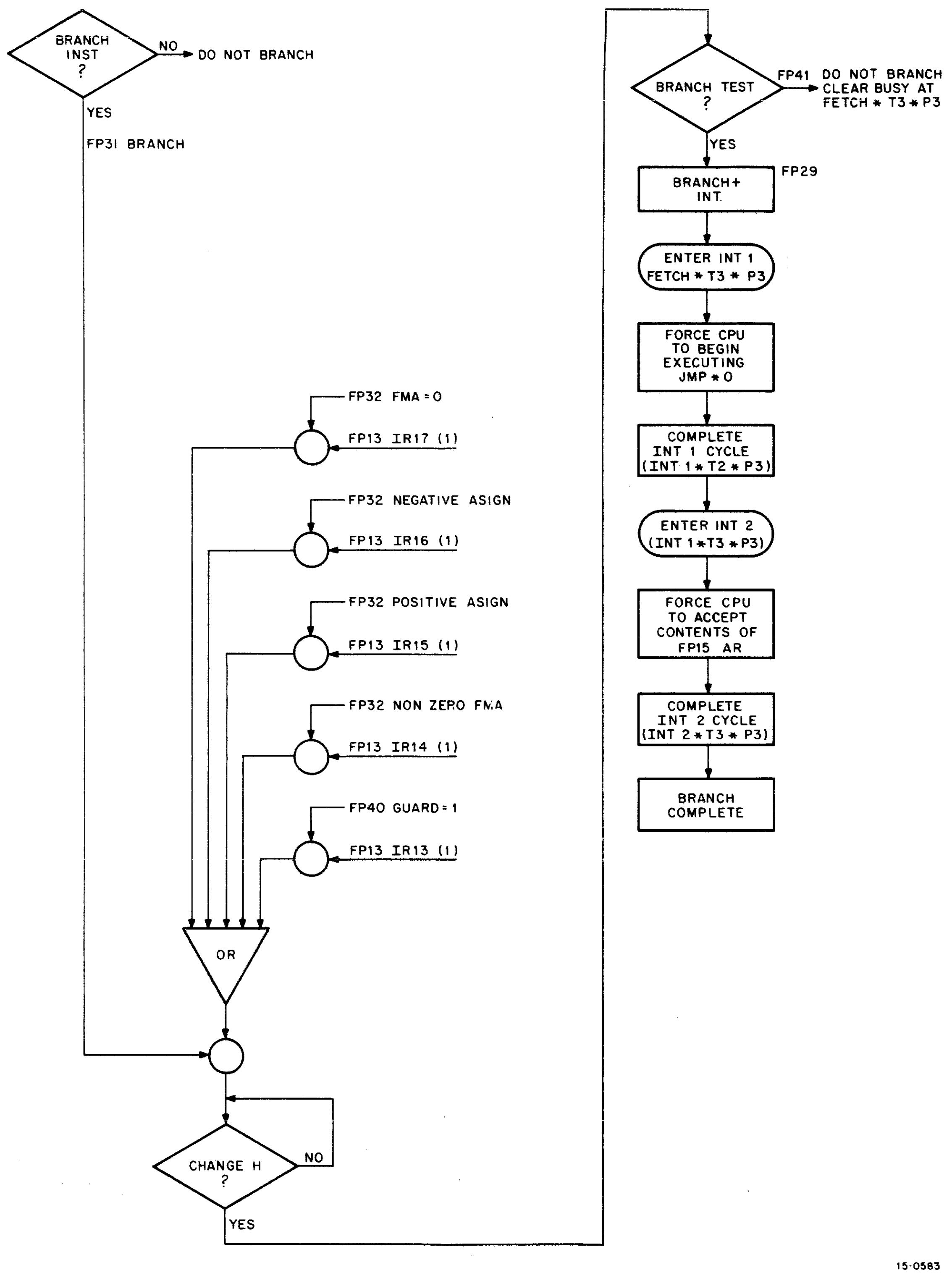


Figure 4-9 Branch Instruction Flow Diagram

4.20 MODIFY FMA

The class of instructions used to modify the FMA are:

a. Zero EPA (A SIGN) FMA
b. Normalize EPA (A SIGN) FMA
c. Make A SIGN positive
d. Make A SIGN negative
e. Complement A SIGN

The flow diagram for control of A SIGN is shown in Drawing D-FD-FP15-0-58. This diagram is applicable to making the A SIGN positive or negative, or complementing the A SIGN. If IR16 and IR17 of the instruction word are a 0 and 1, respectively, the A SIGN becomes 0 (positive) at FUN*TS03*PH03. If IR16 is a 1 and IR17 is a 0, the A SIGN becomes a 1. If both IR16 and IR17 are 1s, the A SIGN is examined and complemented at FUN*TS03*PH03.

4.20.1 Underflow Interrupt Due to Normalization

The only possible interrupt for this class of instructions is an underflow interrupt as a result of normalize EPA/A SIGN/FMA. If the exponent of the result is less than 400000_8 (-2^{17}), an underflow interrupt occurs since the resultant exponent cannot be correctly represented in the EPA.

4.21 DIAGNOSTIC INSTRUCTIONS

The FP15 maintenance mode provides the user with the capability of sequencing through any floating-point instruction step by step. Each instruction contains a number of steps determined by the format, type of instruction, and operand values. One step is counted at each of the following times.

FETCH * TS03 * PH03	
FETCH * TS03 * PH03	(if indirection)
OPAND * TS03 * PH03	(if not immediate)
OPAND * TS03 * PH03	Depends on data format (1, 2, or 3 words)
OPAND * TS03 * PH03	
EXP * TS01 * PH03	
EXP * TS02 * PH03	(FMA and FMB aligned - 1 step count for every align shift.)
EXP * TS03 * PH03	
FUN * TS01 * PH03	
FUN * TS02 * PH03	(FMA and FMB are multiplied or divided here-- 1 step count per shift. FMA also fixed here-- 1 step count per every fix shift.)
FUN * TS03 * PH03	
NOR * TS01 * PH03	(FMA normalized here--1 step count per every normalize shift.)
NOR * TS02 * PH03	
NOR * TS03 * PH03	

(continued on page 4-31)

WRITE * TS03 * PH03	(if a Store type)	Depends on data format (1, 2, or 3 words)	
WRITE * TS03 * PH03	(if a Store type)		
WRITE * TS03 * PH03	(if a Store type)		

For example, if a single-precision floating-point Add instruction was specified, a step is counted at the following times:

		No. of Steps
FETCH * TS03 * PH03		1
OPAND * TS03 * PH03	Two OPAND cycles	1
OPAND * TS03 * PH03		1
EXP * TS01 * PH03		
EXP * TS02 * PH03	(1 step count for every align shift)	1 to 35*
EXP * TS03 * PH03		1
FUN * TS01 * PH03		1
FUN * TS02 * PH03		1
FUN * TS03 * PH03		1
NOR * TS01 * PH03		1 to 35*
NOR * TS02 * PH03		1
NOR * TS03 * PH03		1

In the preceding example, the number of steps ranges from 11 to 79 and, depending on how many align shifts and normalize shifts, must be performed.

The FP15 maintenance mode is initiated by a DMN (Diagnostic Mode On) instruction. CPU instructions are handled in the normal manner and are not affected by the FP15 at this point.

Drawings D-FD-FP15-0-63 and D-FD-FP15-0-64 are flow diagrams of the events occurring during maintenance mode. The first floating-point instruction received after the FP15 is in maintenance mode is handled in a manner similar to that described in the memory interface; in other words, the instruction is loaded into the CPU instruction register and the FP15 instruction register. The next word (operand address) is loaded into the FP15 BMB; a dummy cycle is initiated to prevent the CPU from sensing the operand address as an instruction. DIS RD RST prevents the CPU from accepting the operand address and the CPU is idle waiting for RD RST. The FP15 forces a 710000 NOP on the MDL; the FP15 now simulates memory to complete the CPU/memory reference. The operand address is then strobed into the FP15 address register. The FP15 executed instruction stops in TS03*PH03 of the FETCH cycle. When the dummy cycle is complete and stop clock is present, the signals that were previously inhibiting the CPU are cleared and control is returned to the CPU. At this time, BUSY is a 1, the instruction has stopped executing at TS03*PH03 of the FETCH cycle, stop clock is present, and maintenance mode is enabled.

The next floating-point instruction fetched from core should logically be a maintenance instruction, such as a Diagnostic Read or Diagnostic Step and Read. Since BUSY is a 1, any floating-point instruction will be treated as a maintenance instruction. The instruction from core is now loaded into the DIR and the next word is loaded into the DAR. The CPU is again disabled by DIS CP RD RST and waits in TS03*PH03 for the next RD RST to occur.

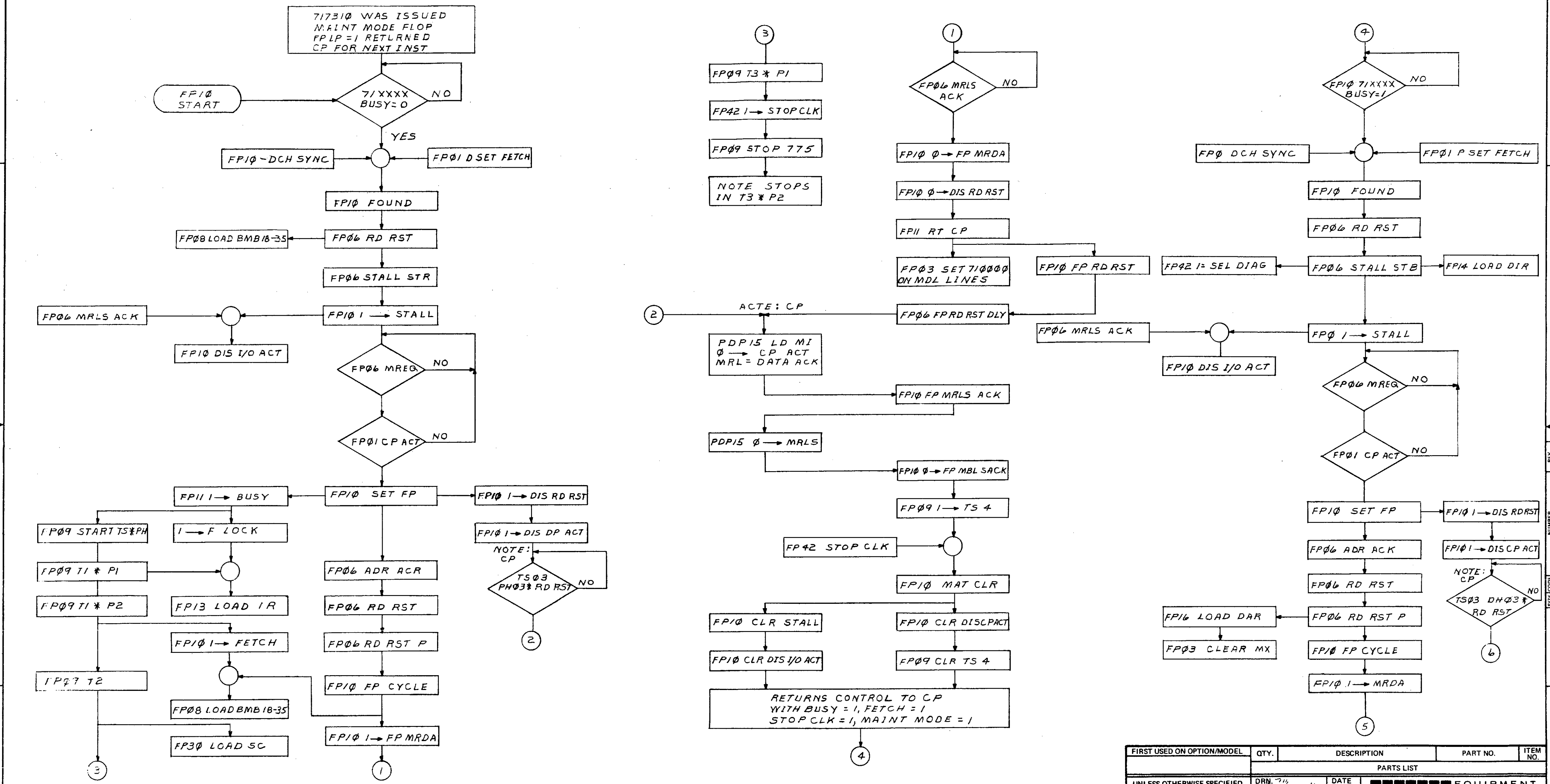
The CPU/memory reference cycle is completed, DIS CP RD RST is removed, the FP15 places a 710000 NOP on the MDL, the CPU strobes the NOP in the MI register, and the memory cycle is completed. Upon completion of the memory cycle, the FP15 goes into diagnostic operation.

4.21.1 Diagnostic Read

If bit 11 in the DIR is a 0, the instruction in the DIR is interpreted as a Diagnostic Read instruction. The FP15 instruction is only partially complete at this point; the contents of sixteen 18-bit words are transferred one at a time from the FP15 to memory starting at the argument address. The words are transferred in the following order:

1. BMB 00-17 (Buffered Memory Buffer)
2. BMB 18-35
3. SC 12-17 and IR 06-17 (Shift Counter and Instruction Register)
4. EPA 00-17
5. A SIGN and FMA 01-17
6. FMA 18-35
7. EPB 00-17
8. B SIGN and FMB 01-17
9. FMB 18-35
10. B SIGN and FMQ 01-17
11. FMQ 18-35

*Depends on operand values.



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL —

FINISH —

		DATE
DRN	[illegible]	5/13/71
CHK'D	L P Hohos	5/14/71
ENG	[illegible]	5/14/71
PROJ. ENG	[illegible]	5/14/71
PROD	[illegible]	5/14/71

NEXT HIGHER ASSY.

SCALE

SHEET OF

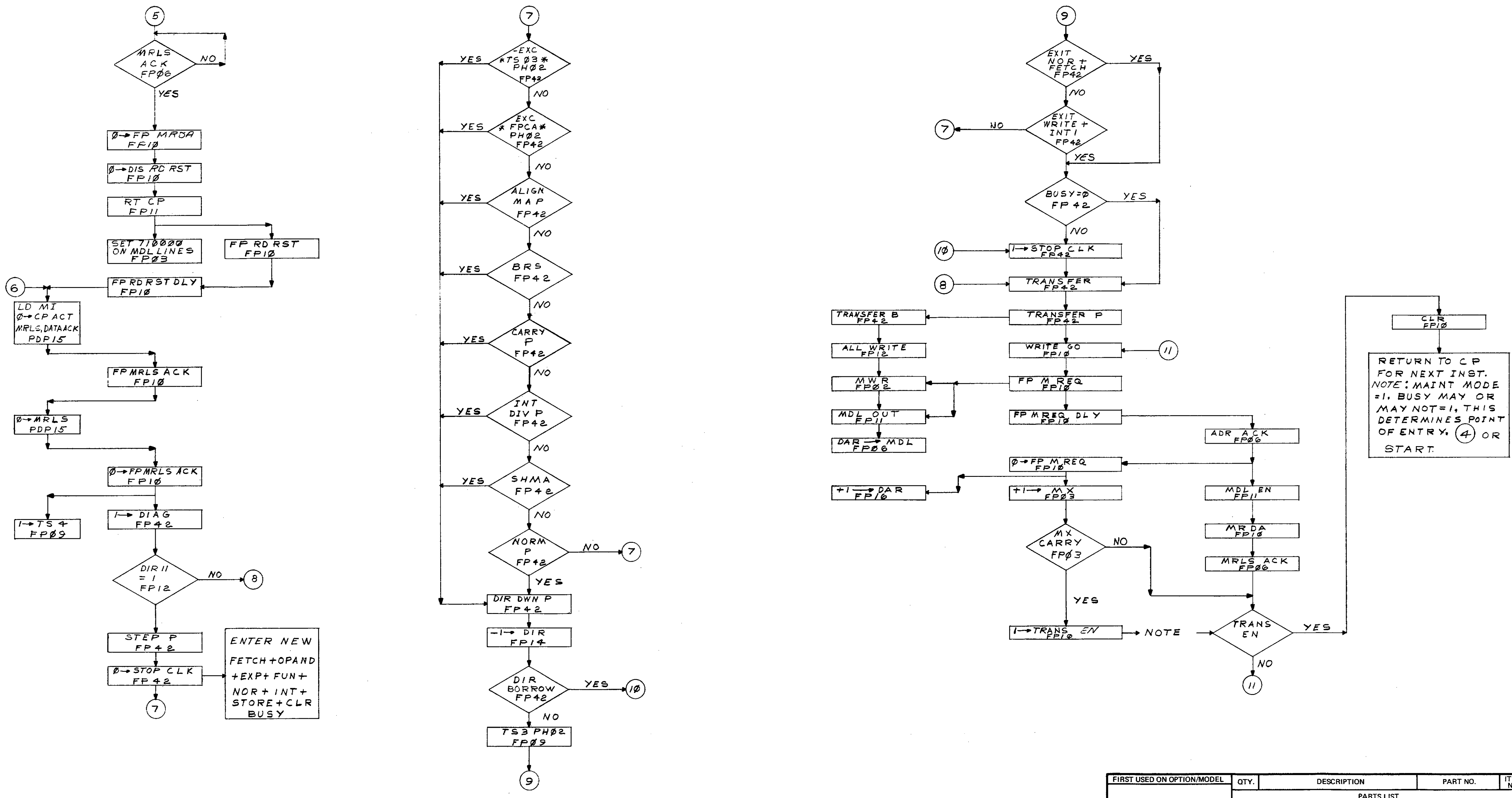
digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE MAINT FLOW 1

SIZE	CODE	NUMBER	REV.
D	FD	FP15-0-63	

DIST.

DEC FORM NO DRD 102-B



FIRST USED ON OPTION/MODEL	QTY.	DESCRIPTION	PART NO.	ITEM NO.
		PARTS LIST		

UNLESS OTHERWISE SPECIFIED DIMENSION IN INCHES. TOLERANCES

DECIMALS	ANGLES
.XXX = .005 .XX = .02 .X = .1	±0° 30'

REMOVE BURRS AND BREAK SHARP CORNERS SURFACE QUALITY ✓

MATERIAL —/—/—

FINISH —/—/—

		DATE
DRN.	[illegible]	4-15-71
CHK'D.	L. P. Hatlos	5/14/71
ENG.	[illegible] King	5/14/71
PROJ. ENG.	[illegible]	5/14/71
PROD.	[illegible]	

NEXT HIGHER ASSY.

SCALE NONE

SHEET OF

digital EQUIPMENT CORPORATION MAYNARD MASSACHUSETTS

TITLE MAINT FLOW 2

SIZE	CODE	NUMBER	REV.
D	FD	FP15-C-64	

DIST.

REVISIONS CHK CHANGE NO REV

DEC FORM NO DRD 102-B

D FD FP15-C-64 P00

12. ADD 00-17 (ALU)
13. ADD 18-35
14. JEA 00-17 (JMS Exit Address)
15. STA 00-17 (see following Note)
16. AR 00-17 (Address Register)

A memory cycle is initiated for each transfer. Each time a word is transferred, the MPO counter is incremented.

NOTE

The STA 00-17 is a status word comprised of the following information:

STA 00	FP15 BUSY
STA 01	FETCH CYCLE
STA 02	OPAND CYCLE
STA 03	EXP CYCLE
STA 04	FUN CYCLE
STA 05	NOR CYCLE
STA 06	WRITE CYCLE
STA 07	INT 1
STA 08	INT 2
STA 09	TIME STATE 1
STA 10	TIME STATE 2
STA 11	TIME STATE 3
STA 12-17	DIR 12-17

The DAR is also incremented; thus, the sixteen 18-bit words are transferred to 16 sequential memory locations starting at the argument address. When a count of 16 is reached, the MPO counter generates a carry that sets TRANS EN. TRANS EN clears the FP15 and control is returned to the CPU for the next instruction. The Diagnostic Read instruction may be executed indefinitely without affecting the partially completed instruction.

4.21.2 Diagnostic Step and Read

If bit 11 of the word in the DIR is a 1, the instruction is handled as a Diagnostic Step and Read. The instruction is sequenced through one or more steps and, depending on instruction type, format, and operand values, a new cycle may be entered. For example, if indirection is specified, the instruction is sequenced through another FETCH cycle; if a non-memory reference instruction is specified with no indirection, the OPAND cycle is bypassed; if the instruction is integer, the EXP cycle is bypassed, etc. The FP clock, which was halted at TS03*PH03, is restarted (see D-FD-FP15-0-64). At this point, the flow sequences through a decision network that determines whether a step has occurred.

The FP15 is stopped if any of the following conditions occur:

-EXC*TS03*PH02	When FP15 is in TS03*PH02 and is not in the EXP, NOR, or FUN cycle.
EXC*FPCA*PH02	When FP15 is in PH02, FP clock is present and an EXP, NOR, or FUN cycle is specified.
ALIGN MA P	When FP15 is doing an alignment to align mantissas.
BRS	When the FMB is doing a right shift.
CARRY P	During each shift that occurs in a multiply or divide operation.
INT DIVIDE P	During an integer divide operation.
SHMA P	When the FMA is being shifted during a Fix instruction.
NORM P	When a normalize operation is taking place.

For each of the preceeding steps that occurs, the DIR is decremented. The Diagnostic Step and Read is initially loaded with a value 710100+n, where n is the desired number of steps. If the number of steps completed is less than n, the logic determines whether the FP15 is at the end of the NOR, or WRITE cycle, or in an interrupt sequence. If the FP15 is not in any of these states, the Diagnostic Step and Read causes another step to be performed.

If the FP15 is at the end of a NOR or WRITE cycle or in an interrupt sequence, and the instruction is not completed (BUSY=1), the clock is stopped and the current contents of the registers are transferred to memory. If the FP15 is at the end of a NOR or WRITE cycle or in an interrupt, and the instruction has been completed, the clock is not stopped and the current contents of the registers are transferred to memory. When the 16 words have been transferred, an MX CARRY is generated, the memory cycle is completed, the FP15 cleared, and control is returned to the CPU for the next instruction. If the FP15 instruction is not completed (BUSY=1), the point of entry is via the diagnostic instruction path. If the instruction has been completed, (BUSY=0) the point of entry is through the initial path.

CHAPTER 5
INSTALLATION AND MAINTENANCE

5.1 INSTALLATION

The FP15 Floating-Point Processor is installed in the H963E Cabinet (Bay 1R) of the PDP-15/20/30/40 Systems. This cabinet contains the PC15 and BA15 and may also include the BB15. When the FP15 is included in a new system, it is completely installed and tested at the factory before the system is shipped. The following paragraphs describe how to install, interconnect, and test an FP15 that is to be installed in an existing PDP-15 System. Table 5-1 summarizes the major components supplied as part of the FP15 Floating-Point Processor. A complete list is provided on drawing D-UA-FP15-0-0. Figure 5-1 shows the general location of the major components installed in the H963E Cabinet.

Table 5-1
FP15 Floating-Point Processor Major Components

Quantity	Item	Part Number
1	FP15 Wired Assembly	D-AD-7007243-0-0
1	FP15 Indicator Panel	D-UA-7006331-0-0
1	H721 Power Supply	H721
1	716 Indicator Power Supply	716

NOTE

If the FP15 is to be installed in early PDP-15 Systems with 783 Power Supplies mounted on the rear door of the H963E cabinet, an H950-C 19-in. mounting panel door will be included and substituted for the original rear door of the H963E Cabinet.

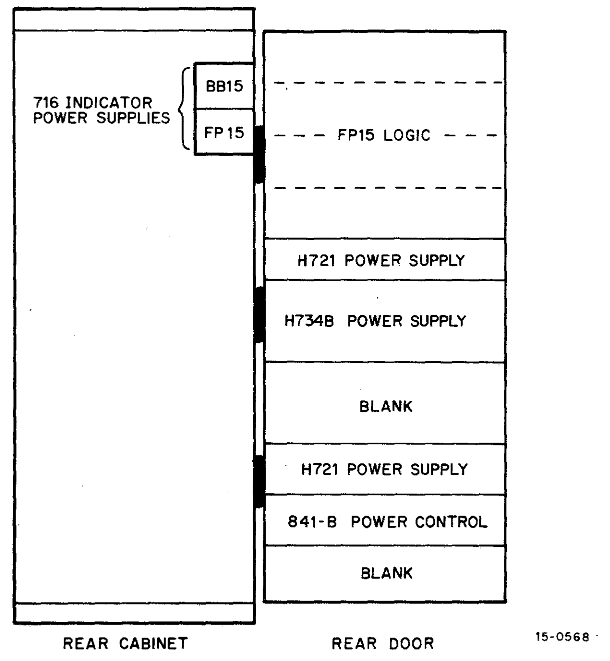


Figure 5-1 H963E Cabinet (Bay 1R), Rear View with Mounting Panel Door Open

5.1.1 Field Installation Procedures

Step	Procedure
1	Remove the H950-P (5-1/4 in.) Cover Panel below the BB15 Indicator Panel. Install the FP15 Indicator Panel in this location.
2	Install the 716 Indicator Power Supply on the inside right wall of the cabinet (as viewed from the rear). Mount the 716 directly below the existing 716 that provides power to the BB15 Indicator Panel.
3	Install the H721 Power Supply on the rear door of the cabinet directly above the existing 734D Variable Power Supply.
4	Locate the FP15 logic wired assembly directly above the H721 Power Supply on the rear door of the cabinet. Fasten securely to the rear door with the mounting hardware supplied. Be sure to use the spacers.

5.1.2 Indicator Panel/Power Supply Wiring

Connect the FP15 Indicator Panel and associated 716 Indicator Power Supply as follows:

Step	Procedure
1	Connect black wire between the FP15 Indicator Panel ground tab and the cabinet chassis ground.
2	Connect orange wire between the +6.5V tab on the FP15 Indicator Panel and the orange tab on the 716 Power Supply.
3	Connect both 716 Power Supplies to cabinet chassis ground.
4	Connect a red and white twisted pair between the AC tabs on the 716 Power Supplies.

5.1.3 H721 Power Supply Wiring

Step	Procedure
1	Connect the red and white twisted pair from the 841B Power Control to the H721 Power Supply ac input terminals (TB2-1 and 2). Refer to D-CS-H721-0-1 for internal connections.
2	Connect a black wire from TB2-8 to cabinet chassis ground.
3	Disconnect the console power switch lead from the existing H721 Power Supply (TB2-6) and connect it to the added 721 Power Supply at TB2-6. Connect a wire from TB2-6 on the original H721 to TB2-7 on the added H721. These connections will connect both H721 thermostat circuits in series with the console power switch.

5.1.4 Signal Cable Connections

Table 5-2 is a signal cable connection chart that indicates how to connect the FP15 into an existing PDP-15 System.

NOTE

The connections place the FP15 between the KP15 and the BB15.

When the system does not include certain BB15 options (KM, KT, or MP), ignore the BB15 cable connections and connect the FP15 directly to the MM15A as indicated in the table.

5.1.5 Indicator Bus Cable Connections

Connect the FP15 Indicator Bus cables to the FP15 wired assembly indicator cable connector card locations (J03, J04, J05, and J06) as designated in Figure 5-2. Dress the indicator bus cables between the FP15 wired assembly and the H721 Power Supply.

Table 5-2
Signal Cable Connections

Cable Function	Connector Locations						Remarks
	KP15	FP15		BB15		MM15A	
	OUT	IN	OUT	IN	OUT	IN	
Memory Data Lines	J02	H29	J29	B02	A02	B02	If BB15 does not contain KM15, KT15, or MP15, connect FP15-J29 to MM15A-B03.
Memory Control Lines	J03	H30	J30	B03	A03	B03	Under conditions listed above, connect FP15-J30 to MM15A-B02.
API Control	H03	H31	J31	B05	--	--	This cable is not required to be connected to memory.

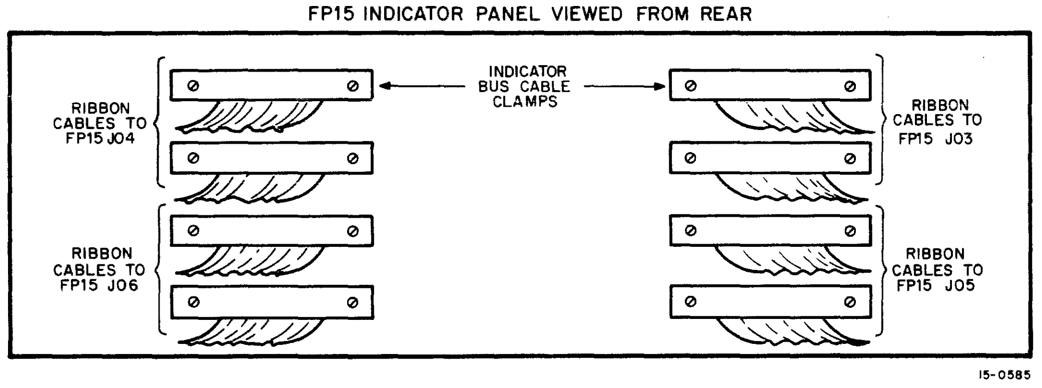

FP15 INDICATOR PANEL VIEWED FROM REAR
INDICATOR BUS CABLE CLAMPS
RIBBON CABLES TO FP15 J04
RIBBON CABLES TO FP15 J03
RIBBON CABLES TO FP15 J06
RIBBON CABLES TO FP15 J05
15-0585

Figure 5-2 FP15 Indicator Bus Connections

5.1.6 Handwire List

The KP15 must be modified per handwire list supplied in the FP15 Installation Kit.

5.1.7 Postinstallation Checks and Tests

Make a final check of the completed installation to ensure that:

a. All modules are correctly installed in the FP15 wired assembly.

b. Major components are securely mounted in the cabinet.

c. Cable and wired connections are correct, and cables and harnesses are dressed and fastened within the cabinet.

Apply primary power to the cabinet by closing the circuit breaker on the 841B Power Control. Test for +5V at any of the G829 modules.

Run the FPU 01 Random Exerciser diagnostic program to test FP15 Floating-Point Processor operation. As a further test to ensure that the FP15 is correctly installed and operational, load and run the Instruction Test diagnostic program MAINDEC-15-DOTA.

5.2 MAINTENANCE

The FP15 Floating-Point hardware includes built-in diagnostic hardware that allows any floating-point instruction to be sequenced through step-by-step and allows the user to obtain a printout of each register as each step of an instruction is performed. An indicator panel, also supplied with the FP15, providing a visual display of the major registers. The stepping of the instruction and the printout is accomplished under software control. The diagnostic programs assume that the CPU and memory are functioning and operating properly, and are designed to minimize actual troubleshooting since malfunctions can be isolated before troubleshooting techniques have to be used. The following paragraphs describe the FP15 Indicator Panel and the diagnostic programs used.

5.2.1 FP15 Indicator Panel

The FP15 Indicator Panel is used as a maintenance aid and is located directly above the BB15 Option Panel. The indicator panel consists of the following indicators.

EPA	Denotes the state of the 18 bits in the EPA register.
JEA	Bit 00 denotes the state of A SIGN, bit 01 denotes the state of the GUARD bit; bit 02 is not used; bits 03 through 17 denote the JEA exit address in memory.
A SIGN, FMA	A SIGN denotes sign of operand in the FMA; FMA 1 through 35 represents the value of the operand in the FMA.
B SIGN, FMQ	B SIGN denotes the sign of the FMB; FMQ 1 through 35 denotes the value of the quantity stored in the FMQ.
MAJOR STATE, TIME STATE	Denotes the current major state and time state of the FP15. The FP15 could be in the FETCH, OPAND, EXP, FUN, NOR, WRITE, INT 1, or INT 2 major states and in TS01, TS02, or TS03. The BUSY indicator indicates that the FP15 is in the process of performing some function which it has not yet completed. For example, the FP15 may be sequenced through an instruction in Diagnostic Mode.
DIR	The DIR indicators denote the number of steps to be sequenced through for an instruction in Diagnostic Mode. The value represented by the indicators is decremented for each step which occurs.
STAL	The STAL indicator denotes that a $71XXXX_8$ floating-point instruction has been detected by the FP15.
TS4	The TS4 indicator, when on, denotes that the FP15 has control of memory and, when off, indicates that the FP15 is simulating a memory.
ST PHAS	This indicator denotes that the FP15 is temporarily halted and is not advancing through the various phases, time states, and major states.
MDL EN	This indicator denotes that the MDL lines are enabled and that data is about to be placed on these lines.
L MIT	This indicator denotes that the FP15 is in the second FETCH cycle (indirection).
MAINTENANCE	The indicator panel has five maintenance indicators that perform the following functions:
MAT	This indicator denotes that a Maintenance (Diagnostic) instruction has been decoded.
MANT MODE	This indicator denotes that the FP15 is in Maintenance (Diagnostic) mode.
SEL DIAG	This indicator denotes that a Diagnostic instruction has been selected.
DIAG	This indicator denotes that a Diagnostic instruction is being executed.
TRNS EN	This indicator denotes that the sixteen 18-bit words representing the contents of the various registers have been written into memory.

(continued on page 5-4)

DISABLES	The FP15 indicator panel is equipped with the following three disable indicators:
RD RST	This indicator denotes that the CPU is inhibited from using the RD RST from memory.
CP ACT	This indicator denotes that the CPU is temporarily suspended from sequencing through phases and time states.
I/O ACT	This indicator denotes that the FP15 is doing a memory reference cycle (FETCH, OPAND, or WRITE).
FP MEMORY CONTROL	The FP indicator panel has five indicators associated with the FP15 Memory Interface. These indicators are described below.
COND	This indicator denotes that an FP memory request is being made. The indicator remains on during the memory cycle.
M REQ	This indicator denotes that an FP memory request is initiated.
RD RST	This indicator denotes that the FP15 is simulating memory and has placed data on the MDL.
MRDA	This indicator denotes that the FP15 has received data from memory and is releasing memory for additional requests.
MRLS ACK	This indicator denotes that memory is free to accept additional memory requests.

5.2.2 Diagnostic Programs

In addition to the built-in diagnostic hardware and indicator panel, the following test programs are available.

Instruction Test - Part 1	FPIT 01	MAINDEC-15-D0TA
Instruction Test - Part 2	FPIT 02	MAINDEC-15-D0UA
Instruction Test - Part 3	FPIT 03	MAINDEC-15-D0VA
Floating-Point Diagnostic Random Exerciser		MAINDEC-15-D0WA
Diagnostic Mode Stepping	FP STEP	MAINDEC-15-D0SA

These test programs are described in the following paragraphs. Before these programs are run, the System Exerciser should be run on a daily basis for preventive maintenance.

5.2.3 Instruction Tests

The instruction tests perform the following major functions:

a. Verify that the diagnostic instructions are operating correctly.
b. Provide loop information for debugging.
c. Check whether all FP15 registers can be cleared and then set to all 1s.
d. Exercise the FP15 instructions in Diagnostic Mode in a general fashion.
e. Run automatically until an error is detected.

The error is identified at a 6-digit location (address of program listing). A copy of the contents of the major registers can be obtained at the time of the error. For further isolation of a malfunction causing the error condition, a scope loop is utilized.

In order to run the instruction test, the program FPSTEP, which is a separate independent program, must be preloaded in core. The FPSTEP program allows diagnostic mode stepping of any FPU instruction. The operator must specify the instruction to be stepped and must specify either an argument or data to be used with the instruction.

The FP STEP program can perform the following major functions:

a. Scope loop any FP15 instruction at any step rate.
b. Automatically step any FPU instruction to completion using a pre-set step rate, with or without typeouts of the FPU registers.
c. Step any FP15 instruction with complete control over step rate and register typeouts between steps.
d. Restart at any time without affecting the program.

5.2.4 Random Exerciser

The FP15 Floating-Point Processor Random Exerciser is a test program to simulate system usage for preventive maintenance. A PDP-15 Computer with 8,192 word memory and an FP15 Floating-Point Processor are necessary to run the program. The complete FP15 Instruction Test Hardware Diagnostic series should be run prior to running the random exerciser. The following system parameters are selected:

a. 50 or 60 Hz power
b. API or no API
c. The amount of memory to be initialized

After system parameter selection, the instruction and data format are selected. Hardware operations and software calculations can then be performed on specified operands or on randomly selected functions.

The random exerciser contains a real-time clock (RTC) routine to keep track of time and uses a 24 hour clock (for example, 2:00 p.m. is 14:00). The program will print:

```
Disable RTC
Type in time
When time is reached enable RTC
Time
```

Errors are detected in the random exerciser program by comparing a software calculated arithmetic result to the actual FPU completed result. Occurrence of an error condition causes an error typeout format to be printed.

5.3 ENGINEERING DRAWINGS

Engineering drawings pertinent to the FP15 Floating-Point Processor are listed in Table 5-3 and included in a separate volume entitled FP15 Floating Point Processor, Engineering Drawings.

Table 5-3
FP15 Floating-Point Processor Engineering Drawings

Drawing No.	No. of Sheets	Title
D-UA-FP15-0-0	2	Floating Point Processor
A-PL-FP15-0-0	1	Floating Point Processor
D-DI-FP15-0-67	1	Drawing Index List (FP15)
D-AD-7007243-0-0	2	Wired Assy (FP15)
A-PL-7007243-0-0	1	Wired Assy (FP15)
D-MU-FP15-0-66	2	Module Utilization
A-PL-FP15-0-66	2	Module Utilization
D-BS-FP15-0-01	1	Memory Interface Cables
D-BS-FP15-0-02	1	Memory Drivers
D-BS-FP15-0-03	1	Output Multiplexer (MPO)
D-BS-FP15-0-04	1	Multiplexer Inputs (MPI)
D-BS-FP15-0-05	1	Multiplexer Control
D-BS-FP15-0-06	1	Memory Receivers
D-BS-FP15-0-07	1	Buffered Mem Bits 00-17
D-BS-FP15-0-08	1	Buffered Mem Bits 18-35
D-BS-FP15-0-09	1	Time State Generator
D-BS-FP15-0-10	1	Memory Interface Ctrl 1
D-BS-FP15-0-11	1	Memory Interface Ctrl 2
D-BS-FP15-0-12	1	Memory Interface Ctrl 3
D-BS-FP15-0-13	1	Instruction Register (IR)
D-BS-FP15-0-14	1	Diagnostic Inst Reg (DIR)
D-BS-FP15-0-15	1	Address Register (AE)
D-BS-FP15-0-16	1	Diagnostic Address Reg (DAR)
D-BS-FP15-0-17	1	A Exponent Register (EPA)
D-BS-FP15-0-18	1	B Exponent Register (EPB)
D-BS-FP15-0-19	1	Arith Logic Unit 00-03
D-BS-FP15-0-20	1	Arith Logic Unit 04-07
D-BS-FP15-0-21	1	Arith Logic Unit 08-11
D-BS-FP15-0-22	1	Arith Logic Unit 12-15
D-BS-FP15-0-23	1	Arith Logic Unit 16-19
D-BS-FP15-0-24	1	Arith Logic Unit 20-23
D-BS-FP15-0-25	1	Arith Logic Unit 24-27
D-BS-FP15-0-26	1	Arith Logic Unit 28-31
D-BS-FP15-0-27	1	Arith Logic Unit 32-35
D-BS-FP15-0-28	1	Carry Look Ahead
D-BS-FP12-0-29	1	JMS Exit Address Reg (JEA)
D-BS-FP15-0-30	1	Shift Counter (SC)
D-BS-FP15-0-31	1	Instruction Decoder
D-BS-FP15-0-32	1	Mantissa & Exponent Ctrl
D-BS-FP15-0-33	1	Adder Control
D-BS-FP15-0-34	1	Load & Store Control 1
D-BS-FP15-0-35	1	Load & Store Control 2
D-BS-FP15-0-36	1	Add & Subtract Ctrl 1
D-BS-FP15-0-37	1	Add & Subtract Ctrl 2
D-BS-FP15-0-38	1	Multiply & Divide Ctrl 1
D-BS-FP15-0-39	1	Multiply & Divide Ctrl 2
D-BS-FP15-0-40	1	Normalize Control
D-BS-FP15-0-41	1	Misc Inst Control
D-BS-FP15-0-42	1	Diagnostic Control
D-BS-FP15-0-43	1	Error Check
D-BS-FP15-0-44	1	Indicator Cables
A-SP-FP15-0-70	10	Acceptance Specification
A-SP-FP15-0-71	12	Installation Specification
A-SP-FP15-0-72	1	FP15 Hand Wire List
D-CS-H721-0-1	1	H721 Power Supply
C-CS-716-0-1	1	716 Power Supply
D-FD-FP15-0-45	1	Fetch Cycle Flow 1
D-FD-FP15-0-46	1	Fetch Cycle Flow 2
D-FD-FP15-0-47	1	Fetch Cycle Flow 3
D-FD-FP15-0-48	1	Opand Cycle Flow 1
D-FD-FP15-0-49	1	Opand Cycle Flow 2
D-FD-FP15-0-50	1	Opand Cycle Flow 3
D-FD-FP15-0-51	1	Write Cycle Flow
D-FD-FP15-0-52	1	Add, Sub, Rev Sub, Exp Cycle
D-FD-FP15-0-53	1	Add, Sub, Rev Sub, Sub Cycle
D-FD-FP15-0-54	1	Floating Mul & Div Exp Cycle
D-FD-FP15-0-55	1	Float & Integer Mul Fun Cycle
D-FD-FP15-0-56	1	Integer Divide Exp Cycle
D-FD-FP15-0-57	1	Float & Integer Div Fun Cycle
D-FD-FP15-0-58	1	NOR TS1 Cycle Flow
D-FD-FP15-0-59	1	NOR TS2 Cycle Flow
D-FD-FP15-0-60	1	ASIGN Swap & Float Control
D-FD-FP15-0-61	1	Fix Flow
D-FD-FP15-0-62	1	Interrupt Flow
D-FD-FP15-0-63	1	Maint Flow 1
D-FD-FP15-0-64	1	Maint Flow 2

APPENDIX A
SIGNAL GLOSSARY

Signal Mnemonic	Logic Print	Function
AA + PC	FP15-0-05	Address Acknowledge or Power Clear.
ADD A	FP15-0-36	Indicates an addition of two quantities with like signs.
ADD 00-17 SEL	FP15-0-35	Used for selecting MPO address lines during a Store instruction.
ADD S	FP15-0-36	Indicates addition of two quantities with unlike signs (actually a subtraction).
ADD 18-35 SEL	FP15-0-35	Used for selecting MPO address lines during a Store instruction.
ADR ACK (1) B	FP15-0-01	Notifies the peripheral devices of receipt of MREQ, memory address, and mode of operation (read or write).
ALIGN MA	FP15-0-37	Indicates that FMA is to be aligned during addition or subtraction. Also indicates that the exponent associated with the FMA, in this case, is less than the exponent associated with the FMB.
ALIGN MB	FP15-0-37	Indicates that FMB is to be aligned during addition or subtraction. Also indicates that the exponent associated with the FMB, in this case, is less than the exponent associated with the FMA.
ALL WRITE	FP15-0-12	Indicates that the FP15 is in a WRITE cycle or a diagnostic routine.
-ALL ZEROS	FP15-0-37	Indicates that the difference between EPA and EPB is not greater than 35_{10}.
AR LOAD	FP15-0-15	A signal used to load the AR at FETCH*T3*P3.
A SEL, B SEL	FP15-0-05	Used to select one of four inputs to be gated through M1701 Data Selector.
A SIGN, B SIGN	FP15-0-32	The sign bits of the FMA and FMB, respectively.
AUA, AUB	FP15-0-33	Address lines for selecting the A side of ALU.
AUA1, AUB1	FP15-0-33	Address lines for selecting the B side of ALU.
AUS	FP15-0-33	Strobe line for multiplexer connected to the A side of ALU.
AUS1	FP15-0-33	Strobe line for multiplexer connected to the B side of ALU.
A ZERO, B ZERO	FP15-0-32	Used to detect whether the FMA or FMB registers, respectively, are cleared. (Equal to Zero.)
BIT 00-01 DIS	FP15-0-04	Sets bits 00 and 01 to indicate jump type instruction.
BIT 02 SEL	FP15-0-04	Sets bit 02 to indicate JMS type instruction.
BMB 00-35	FP15-0-35	Used for loading the FMA during a non-arithmetic function and for loading the FMB during an arithmetic function. This signal generates AUB on D-BS-FP15-0-33 to select the A side of the ALU.
BRANCH EN	FP15-0-41	Indicates that a successful branch test has occurred.
BMB 27-35 SEL	FP15-0-35	A signal used to load bits 27 through 35 of the BMB into the EPA or EPB when single-precision floating-point format is specified.
BRANCH TEST	FP15-0-41	Indicates a successful branch test was made and a branch is to be performed.
BRS	FP15-0-37	In EXP cycle during addition or subtraction, BRS (FMB Right Shift) causes shifting of FMB to align mantissas.
BUSY	FP15-0-11	Indicates that the FP15 is busy and sets up certain conditions for floating-point operation.
CARRY P	FP15-0-39	Generates the strobe that loads the FMA or FMQ after each shift.
C DIV (Combined Divide)	FP15-0-31	This signal represents the OR of Divide and Reverse Divide.
C DIV INT P	FP15-0-38	Used in the EXP cycle of Integer Divide for negative integers to increment the FMB containing the negative integer.
CHANGE	FP15-0-11	Indicates that the FP15 has finished the FETCH cycle.
CHECK EN	FP15-0-37	Determines whether format is floating-point or integer addition or subtraction.
CLK 00-17	FP15-0-07	A signal used to clock bits 00 through 17 of the BMB.
CLK 18-35	FP15-0-08	A signal used to clock bits 18 through 35 of the BMB.
CLR BMB 00-17	FP15-0-35	Clears bits 00-17 when a positive 2's complement single-precision integer number is loaded into the BMB.
CLR EPA P	FP15-0-40	During normalize, FMA is checked to see if it is 0. If so, CLR EPA P clears EPA and A SIGN.

Signal Mnemonic	Logic Print	Function
CLR EXC	FP15-0-29	A signal used to clear EXP, FUN, or NOR cycle upon receipt of an interrupt or Branch instruction.
COMP	FP15-0-36	Indicates that an overflow has occurred during subtraction.
COMP MA	FP15-0-34	Used during integer arithmetic when a negative 2's complement number is used. This number is converted to sign and magnitude by complementing and incrementing the FMA. COMP MA complements the FMA.
COMP MB P	FP15-0-34	Used during integer arithmetic when a negative 2's complement number from memory is used. The number is converted to sign and magnitude format by complementing and incrementing the FMB. COMP MB complements the FMB.
COMP SUB	FP15-0-36	Complements the result if an overflow occurred during a subtraction.
CN 00	FP15-0-33	Indicates a carry input to the least significant stage of the ALU.
CN 01-08	FP15-0-28	Carry inputs to each ALU from the carry look-ahead generator.
COND	FP15-0-11	Indicates that the FP15 is making a memory request.
COUNT A LT P	FP15-0-39	Shifts FMA and FMQ left during EXP cycle of Integer Divide.
CP ACT DIS	FP15-0-01	Disables CPU cycle to allow FPU to communicate with memory.
CP RD RST DIS	FP15-0-01	Inhibits CPU from seeing data on MDL.
C SUB	FP15-0-31	This signal represents the OR of Subtract and Reverse Subtract.
DAR CLK	FP15-0-16	A signal used to increment the Diagnostic Address Register during Maintenance Mode.
DATA ACK L	FP15-0-01	Notifies memory that it may remove the data from the bus.
DCH SYNC	FP15-0-06	Indicates I/O Processor wants memory access.
DIAG	FP15-0-42	Indicates next instruction fetched from core will be interpreted as a Diagnostic instruction.
DIR DWN	FP15-0-42	Decrements the DIR for each step of a Diagnostic Step and Read instruction.
DIS CP ACT	FP15-0-10	Used to disable the CPU in order to allow the FP15 to gain control of memory.
DIS I/O ACT	FP15-0-10	Used to prevent I/O from gaining control of memory bus during floating-point operations.
DIS RD RST	FP15-0-10	Used to disable the CPU from seeing a RD RST signal and allowing the FP15 to gain control of memory.
DIV ADD SH	FP15-0-38	Produces MXB during Divide if subtraction produces positive result. MXB shifts subtracted result left on inputs to FMA.
DIV ASH	FP15-0-38	Produces MXA during Divide if subtraction produces negative result. MXA enables FMA to be shifted left.
DIV COUNT P	FP15-0-39	Used to increment the EPA and left-shift the FMQ in the EXP cycle of Integer Divide.
DIV COUNT SEL	FP15-0-39	Enables EPA to inputs of shift counter during FUN cycle of Integer Divide.
DIV EXP	FP15-0-39	Initiates EXP cycle during floating-point division.
DIV EXP P	FP15-0-39	Used in detecting possible overflow or underflow in the EXP cycle during division.
DIVIDE (1) H	FP15-0-43	Indicates abnormal divide has been detected.
DIV INC P	FP15-0-38	Used to produce EPA UP on first shift of divide if first subtraction result is positive.
DIV MQ	FP15-0-38	Produces AUA to enable FMQ to ALU bus for subsequent swap of the FMA and FMQ at the end of the divide.
DIV MQ SH	FP15-0-38	Produces MXA1 which enables FMQ to be shifted left in the FUN cycle during division.
DIV P	FP15-0-38	A pulse used to produce ALS and MLS during division in order to strobe the FMA and FMQ. A DIV P pulse is produced for each shift during Divide.
DIV SHRT P	FP15-0-38	Produces ARS which shifts the FMA right one place at the end of the divide process and prior to the swap.
DIV SWAP P	FP15-0-38	Produces ALS and MLS which causes the contents of the FMA and FMQ to be swapped.
DIV ZERO	FP15-0-43	A divide-by-zero operation has been attempted.
DUMMY EN	FP15-0-05	A signal used in the dummy fetch of the FETCH cycle.
EPA GRT	FP15-0-36	Increments the EPA during a floating-point or Fix instruction due to a carry out of the ALU.
EPA LD	FP15-0-32	A signal that loads the EPA.
EPA MOVE P	FP15-0-35	Used during Reverse Divide or Reverse Subtract to load the contents of the EPA into the EPB.
EPA UP	FP15-0-32	A signal (that increments the EPA).
EPB SEL	FP15-0-37	Selects EPB to be inputted to B side of ALU when calculating exponent during multiplication and division. Also used to transfer EPB to EPA if EPB > EPA during addition or subtraction.
EXC	FP15-0-11	Indicates that the FP15 is in the EXP, FUN, or NOR cycle, which are all internal cycles within the floating-point processor.
EXIT INT + BRANCH	FP15-0-41	Indicates completion of interrupt or Branch instruction.
EXP	FP15-0-11	Denotes exponent cycle which is used to align or calculate exponents of the operands.
EXP EXC (Exponent Exception)	FP15-0-37	Used during exponent alignment and indicates that difference between exponents is too large to be aligned.
EXP ONES	FP15-0-37	Indicates EPB - EPA is greater than positive 35.
EXP SEL	FP15-0-33	A signal used to enable the EPA during shifting operations.
EXP ZEROS	FP15-0-37	Indicates EPA - EPB is greater than positive 35.

Signal Mnemonic	Logic Print	Function
FETCH	FP15-0-11	Denotes FETCH cycle where the instruction is strobed into the FP15 Instruction Register.
FIX COUNT	FP15-0-41	Establishes the number of shifts required to fix the floating-point numbers.
FIX + FLOAT SEL	FP15-0-41	Indicates a Fix or Float instruction has been selected.
FIX P	FP15-0-41	Used to load the shift counter with the difference between 35_{10} and the EPA and indicates the number of shifts required to fix the number.
FIX SHMA	FP15-0-41	Upcounts the shift counter and right shifts the FMA and FMQ during a Fix instruction.
FIX SEL	FP15-0-41	Indicates a number greater than 1 which can be fixed.
FIX ZERO	FP15-0-41	Indicates a fractional number that cannot be fixed. A SIGN and EPA are cleared.
FLOAT + FIX	FP15-0-40	Designates floating-point instruction or Fix instruction.
FLOAT SEL P	FP15-0-41	Loads 43_8 in the EPA during a Float instruction.
F LOCK	FP15-0-11	Used to set up the FETCH cycle during the start of a floating point operation.
FMA STROBE	FP15-0-36	This signal causes the FMA to be reloaded if an overflow occurs out of the ALU.
FPCA, FPC	FP15-0-09	Floating-point clock outputs.
FP MRDA	FP15-0-10	Memory Release, Data Acknowledge. Used to indicate to memory that cycle is completed and data has been accepted.
FP MREQ	FP15-0-10	A memory request made by the FP15. Memory senses the request as a CPU memory request.
FP MRLS ACK	FP15-0-10	Used to simulate MRLS ACK generated by the memory to complete memory cycle.
FP RD RST	FP15-0-10	Used to simulate Central Processor in order to complete memory cycle.
FP WAIT	FP15-0-09	Locks floating-point processor in TS03*PH01 during the dummy FETCH.
FUN	FP15-0-11	Denotes function cycle which includes the actual instruction to be executed.
GG 00, GG 01	FP15-0-28	Carry generate outputs from carry look-ahead logic used to speed up carry propagation through the ALU.
G01-G07	FP15-0-20 through -26	Carry generate outputs of one of the 4-bit ALU circuits used in carry look-ahead circuitry.
GRT	FP15-0-36	Generated (greater than) when a carry occurs out of the MSB of the ALU during addition, subtraction, or rounding.
GUARD	FP15-0-40	Indicates that rounding is possible.
HFPC	FP15-0-40	A clock pulse used for normalizing numbers - two HFPC pulses (half FPC) are required per shift during normalize.
INCA P	FP15-0-34	Used during integer arithmetic when a negative 2's complement number from memory is used. This number is converted to sign and magnitude by complementing and incrementing the FMA. INCA increments the FMA.
INCB P	FP15-0-34	Used during integer arithmetic when a negative 2's complement number from memory is used. This number is converted to sign and magnitude by complementing and incrementing the FMB. INCB increments the FMB.
INT	FP15-0-43	Indicates an interrupt has been detected.
INT + API ST	FP15-0-05	Indicates that a Trap has been found.
INT CHECK 1	FP15-0-43	Check for overflow of negative integer during single-precision Integer Store instruction.
INT CHECK 2 P	FP15-0-43	Checks for overflow of positive integer during single-precision Integer Store instruction.
INT COMP P	FP15-0-35	Loads the complement of the FMA into the FMB during Integer Store.
INT DIV P	FP15-0-39	Used during integer divide to generate signals indicating whether FMA, or FMQ, or both, are to be left-shifted.
INT DIV STOP	FP15-0-39	Generated when both FMA and FMQ are normalized during Integer Divide.
INT INC P	FP15-0-35	Used for incrementing the FMB during an Integer Store.
INT MPY OVR	FP15-0-43	Indicates an overflow has occurred during Integer Multiply.
INTRP SYNC (1) H	FP15-0-10	Used to disable program interrupt and API when STALL is set.
INT 1, INT 2	FP15-0-11	This signal is raised during an overflow, underflow, or divide by zero condition to indicate entry to a Service routine in the CPU.
I/O ACT DIS	FP15-0-01	Disables I/O processor to allow FP15 to communicate with memory.
IR CLK	FP15-0-13	A signal used to clock the IR.
JMS SEL	FP15-0-43	Forces JMS exit address onto MDL lines.
LD DIV COUNT	FP15-0-39	Causes shift counter to be loaded with 43_8 in the EXP cycle during Integer Divide.
LD EPA, LD EPB	FP15-0-35	Used to load the EPA or EPB register, respectively, during the OPAND cycle.
LD IR	FP15-0-08	A strobe signal used to load the DIR when a floating-point instruction has been detected.
LD JMS P	FP15-0-41	A pulse used to load JMS during FUN*TS1.
LD MA	FP15-0-35	Used to load the FMA during the OPAND cycle when a non-arithmetic or reverse arithmetic instruction is issued.
LD MB	FP15-0-35	Used to load the FMB during the OPAND cycle when an arithmetic instruction (except for Reverse Subtract or Reverse Divide) is issued.

Signal Mnemonic	Logic Print	Function
LD NORM COUNT	FP15-0-40	Used to load the shift counter with 43_8 to limit the number of shifts during normalize.
LD SC P	FP15-0-37	A pulse used to load the shift counter to check the number of shifts needed for alignment of mantissas.
LIKE	FP15-0-36	Indicates A SIGN and B SIGN are both positive or both negative.
LIMIT	FP15-0-11	Allows FP15 to perform only one level of indirection.
MA CHECK	FP15-0-37	Check to see if FMA is equal to 0.
MB CHECK	FP15-0-37	Checks to see if FMB is equal to 0.
MAINT MODE (1) H	FP15-0-41	When set, this signal indicates maintenance instructions are to be performed.
MA MOVE P	FP15-0-35	Used during Reverse Divide or Reverse Subtract to load the contents of the FMA into the FMB.
MAT CLR	FP15-0-10	Indicates that the maintenance instruction is complete and the register contents have been written into memory.
M CLR	FP15-0-41	Indicates a Debreak instruction or a Power Clear condition.
MDL EN	FP15-0-11	Enables data from the FP15 to be placed on the MDL.
MDL 00-MDL 17	FP15-0-01	18 memory data lines providing bidirectional transfer of address and/or data from memory.
MPI 00-17	FP15-0-04	Each MPI line can receive one of four different input signals. Data on the output line is determined by select signals MOA and MOB.
MPO 00-17	FP15-0-03	18 output multiplexer lines that transfer one of sixteen 18-bit words to memory.
M PWR-OK	FP15-01	Memory power is applied to the memory circuits.
MPY + DIV EXP P	FP15-0-39	Used to produce EPB LD which strobes ALU contents into EPB in the EXP cycle of Multiply or Divide.
MPY + DIV ODD	FP15-0-38	This signal indicates negative quotient.
MPY + DIV OVR P	FP15-0-43	Indicates an overflow has been detected during multiplication or division.
MPY + DIV UND P	FP15-0-43	Indicates an underflow has been detected during multiplication or division.
MPY EXP P	FP15-0-39	Used in detecting possible underflow or overflow in the EXP cycle during multiplication.
MPY P	FP15-0-39	Used to produce ALS and MRS in order to load the FMA and shift FMQ right during multiplication.
MPY SEL	FP15-0-38	Produces A+B which strobes FMA + FMB into the ALU.
MPY SHAD	FP15-0-39	Produces MXA and MXB during floating-point and Integer Multiply which causes the added result to be shifted right at inputs to FMA and also enables FMQ for right-shift.

Signal Mnemonic	Logic Print	Function
MPY SHRT	FP15-0-39	During floating-point or Integer Multiply, this signal causes FMA and FMQ to be right-shifted.
MPY SWAP P	FP15-0-38	Produces ALS and MLS in order to zero the FMA and strobe the FMA into the FMQ at the beginning of the FUN cycle in multiplication.
MQ INT	FP15-0-39	Used in Integer Divide during the EXP cycle to enable the FMB to the ALU bus.
MQ INT P	FP15-0-39	Produces MLS which strobes the FMB into the FMQ.
MRD	FP15-01	Selects read/restore memory cycle.
MRDA		Memory Release, Data Acknowledge. Issued by the FP15 to indicate data has been received and to allow additional memory requests.
MREQ	FP15-01	The signal is generated by the CPU requesting start of a memory cycle.
MRLS	FP15-01	The CPU issues this signal to release memory for additional requests.
MRLS ACK (1) B	FP15-01	Notifies device that memory has accepted data and is terminating memory cycle.
MRS	FP15-0-32	A signal that causes the FMQ to be right-shifted.
MWR	FP15-01	Selects clear/write memory cycle.
MXA, MXB	FP15-0-32	Used as select signals to supply data from one of four sources to the FMA.
MXA, MXB, MXC, MXD	FP15-0-05	Select lines to select one of 16 possible inputs to MPO.
MXA1, MXB1	FP15-0-32	Used as select signals to supply data from one of four sources of the FMQ.
NOR	FP15-0-11	Denotes normalize cycle, where an operand is to be normalized.
NOR EN	FP15-0-32	A signal which causes normalize to occur when requested.
NORM DONE	FP15-0-40	Indicates FMA 01 is on a 1 and normalize is completed.
NORM P	FP15-0-40	Pulse used for normalizing FMA. A NORM P pulse is generated for each normalize shift.
NOR SEL	FP15-0-40	Indicates normalization has been requested.
ODD	FP15-0-36	Indicates sign bits (A SIGN and B SIGN) are not equal.
OPAND	FP15-0-11	Denotes OPAND cycle in which the operand(s) is fetched from memory.
OPAND DWN P	FP15-0-12	Down counts the shift counter during the OPAND cycle. Up to three down counts are possible depending on number of operands required from memory.
OVR (1) H	FP15-0-43	Indicates an overflow has been detected.
P01-P07	FP15-0-20 through -26	Carry propagate outputs from the ALU where a carry is propagated at the output of a 4-bit ALU circuit.

Signal Mnemonic	Logic Print	Function
PP00, PP01	FP15-0-28	Propagate output from the carry look-ahead circuitry used to indicate a carry was propagated from previous stage.
PREP SC	FP15-0-38	Used to inhibit the stepping of the shift counter during floating-point Divide (FUN cycle). PREP SC loads the shift counter at NOR*T3.
RD RST (1) B	FP15-01	Notifies the CPU that the data from memory is on the bus and ready to be strobed into the MI register.
RND	FP15-0-40	Indicates that rounding has been requested and is about to take place.
RND+1	FP15-0-36	Occurs during addition at FUN*T1 as a result of mantissa alignment.
ROUND MA P	FP15-0-40	Indicates FMA is to be rounded if guard is set.
R SET (1) H	FP15-0-09	A signal that clears R SET SYNC which allows ST PHASE to reset in order to start the phase and time state generator.
R SET SYNC	FP15-0-09	Used to reset ST PHASE in order to reset the FP clock.
RT CP	FP15-0-11	Allows CPU to complete cycle since the FP15 simulates an NOP which is transmitted to the CPU.
SC ADDR A, SC ADDR B	FP15-0-30	Selects one of two address lines on the M1701 Data Selector which is outputted to the shift counter.
SEL A, SEL B	FP15-0-37	SEL A is generated when the absolute value of EPA-EPB is greater than 2^{17}. SEL B is generated when the absolute value of EPA-EPB is greater than $2^{17}-1$.
SEL C, SEL D	FP15-0-37	SEL C is generated when EPB is more positive than EPA and both are positive quantities. SEL D is generated when EPA is more negative than EPB and both quantities are negative.
SEL DIAG	FP15-0-42	Maintenance mode is enabled and the instruction on which maintenance is to be performed was loaded.
SET BMB 00-17	FP15-0-35	Sets bits 00 through 17 to all 1's when a negative 2's complement single-precision integer number is loaded into the BMB.
SET FP	FP15-0-10	Indicates that the CPU is fetching the address of the argument.
SET OVR	FP15-0-43	Detects overflow during multiplication or division.
SET SC 17	FP15-0-30	Used to indicate the number of operands or the number of write cycles to be performed.
SET UND	FP15-0-43	Detects underflow during multiplication or division.
SET ZERO	FP15-0-39	Indicates a zero quotient and also that shifting is halted.
SKIP ZERO	FP15-0-39	Decreases amount of time between carry pulses for multiplication when a shift rather than an add and shift is to be performed.
S0, S1, S2, S3	FP15-0-33	Address selection lines used to specify arithmetic or logical operations to be performed by ALU (see FP15-0-33).
STALL	FP15-0-10	Generated during detection of a $71XXXX_8$ op code denoting an FP instruction.

Signal Mnemonic	Logic Print	Function
STALL RESET	FP15-0-10	Used to reset the STALL flip-flop as a result of a PI or API break.
STALL STB	FP15-0-06	Monitors MDL lines and strobes data into FP15 when $71XXXX_8$ has been detected.
STEP P	FP15-0-42	Indicates first step of Diagnostic Step and Read instruction.
STOP ALIGN	FP15-0-37	Used during EXP cycle of addition or subtraction when exponent difference is greater than 35 and denotes that alignment is completed or no alignment is to be performed.
STOP CLK	FP15-0-42	Halts the FP15 Clock to allow sixteen 18-bit words to be transferred to memory during a Diagnostic Step and Read or Diagnostic Read instruction.
STOP DIV	FP15-0-38	Stops the division process when the divisor is normalized.
STORE COMP	FP15-0-35	Indicates that the contents of the FMB are written into memory. This signal is raised for a negative integer.
STORE JEA	FP15-0-41	Used to store the JEA.
STORE OVR P	FP15-0-43	Indicates that overflow has been detected during normalization of a single-precision floating-point Store instruction.
STORE RND P	FP15-0-35	Used to round on a single-precision floating-point Store instruction.
STORE SEL	FP15-0-35	Used to select inputs to the multiplexer during a WRITE cycle.
STORE UND P	FP15-0-43	Indicates that underflow has been detected during normalization of a single-precision floating-point Store instruction.
ST PHASE	FP15-0-09	Used to stop the phase during arithmetic operations.
SUB A	FP15-0-36	Indicates a subtraction of two quantities with unlike signs (actually an addition).
SUB S	FP15-0-36	Indicates a subtraction of two quantities with like signs.
SWAP MQ P	FP15-0-41	Used to swap the contents of the FMA and FMQ.
TRANSFER	FP15-0-42	Enables Transfer P which initiates transfer of sixteen 18-bit words to memory.
TRANS EN	FP15-0-10	Used during Maintenance mode to indicate completion of transfer of data from sixteen registers to memory.
TRANS EPB	FP15-0-37	Transfers contents of EPB into EPA during exponent alignment when the EPB is greater than the EPA.
TRANSFER P	FP15-0-42	Initiates transfer of sixteen 18-bit words to memory during Maintenance mode.
TS 1 (1), TS 2 (1), TS 3 (1)	FP15-0-09	Various time state of the time state generator.
UND (1) H	FP15-0-43	Indicates an underflow has been detected.
UND SYNC (1) H	FP15-0-43	Used for storage of temporary underflow condition.
WRITE	FP15-0-11	Denotes WRITE cycle in which data is written into memory.
WRITE DWN P	FP15-0-02	Down counts the shift counter during the WRITE cycle. Up to three down counts are possible.