iSBX 331™ FIXED/FLOATING POINT MATH MULTIMODULE™ BOARD HARDWARE REFERENCE MANUAL

Manual Order Number: 142668-002

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

REV.	REVISION HISTORY	PRINT DATE
-001	Original Issue	8/80
-002	16-bit Baseboard Addressing Added	7/81

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP	Intel	Megachassis
CREDIT	Intelevision	Micromap
i	Intellec	Multibus
ICE	iRMX	Multimodule
iCS	iSBC	PROMPT
im	iSBX	Promware
Insite	Library Manager	RMX/80
Intel	MCS	System 2000
		UPI
		μScope

and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, iMMX or RMX and a numerical suffix.

PREFACE

This manual provides general information, preparation for use, programming information, principles of operation, and service information for the iSBX 331 Math Multimodule Board. Supplementary information is provided in the following documents:

- *Intel MCS-85 User's Manual,* Order No. 9800366.
- *Intel Peripheral Design Handbook,* Order No. 9800676.
- *Intel Multibus Specification,* Order No. 9800683.
- *Intel iSBX Bus Specification,* Order No. 142686.

CONTENTS

TABLES

ILLUSTRATIONS

CHAPTER 1
GENERAL INFORMATION

1-1. INTRODUCTION

The iSBX 331 Fixed/Floating Point Math Multimodule Board is a member of Intel's growing line of expansion boards designed to augment the iSBC microcomputers. In performing high-speed mathematic functions, the iSBX 331 Math Multimodule Board (hereafter referred to as the Multimodule board) accepts data and commands from an iSBC microprocessor and performs a repertoire of 43 floating point and fixed point commands an order of magnitude faster than is possible through conventional programming routines.

1-2. DESCRIPTION

The Multimodule board, shown in figure 1-1, is designed to plug onto any iSBC microcomputer that contains an iSBX bus connector.

The board contains an 8231 Arithmetic Processing Unit (APU) that provides high performance single or double precision floating or fixed point arithmetic operations. Some of the distinctive characteristics of the APU are:

- Fixed point 16 and 32 bits operation.
- Floating point 32 bit operation.
- 18 programmable data manipulation commands.
- Square root, Logarithm and Exponentiation functions.
- Add, Subtract, Multiply and Divide functions.
- 4 MHz on-board clock generator.
- Trigonometric and inverse trigonometric functions.
- Stack oriented operand storage.
- Programmed data transfer mode.
- Floating to fixed or fixed to floating conversions.
- Binary data formats (Input and Output).
- End of operation signal.
- +12 volt and +5 volt power requirement.
- Software Reset capability.

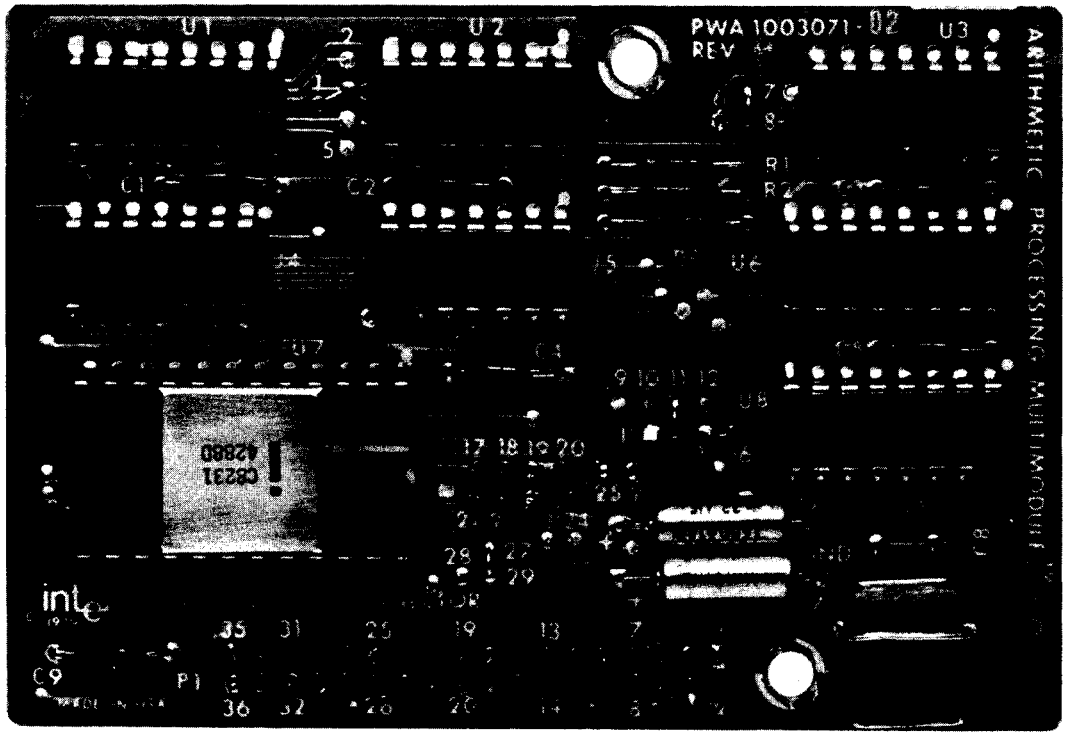


Figure 1-1. iSBX 331™ Fixed/Floating Point Math Multimodule™ Board

The Multimodule board may be conveniently divided into two functional sections; an arithmetic processor and an iSBC microcomputer interface. Each of these is detailed in the following text.

The arithmetic processing unit (APU) consists of an MOS LSI math chip (the 8231) and a clock generator chip (the 8224). The APU is designed to provide high performance operation at a maximum of 4 MHz. Control of the APU is exercised through an iSBC microcomputer via three basic types of commands: the floating point, the fixed point, and the data manipulation commands. The floating and fixed point commands perform the arithmetic operations, and the data manipulation commands access the APU data stack for storage, retrieval, and manipulation of data and/or the results of an operation.

The interface between the iSBX 331 Math Multimodule Board and the host iSBC microcomputer allows programmed data transfer. The speed of the Multimodule board will vary according to the status of the APU. The interface between the Multimodule board and the microcomputer includes a signal that provides an interrupt on completion of an operation. If the APU cannot conclude an operation at full speed, the interface passes an MWAIT signal to the host iSBC microcomputer to indicate that one or more WAIT-states are required in the microprocessor.

1-3. EQUIPMENT SUPPLIED

Since the Multimodule board plugs directly onto the host iSBC microcomputer, no interface cables are required between the two.

The following items are supplied with the iSBX 331 Math Multimodule Board:

a. Schematic Diagram, drawing number 2003073.
b. 1 plastic spacer, ½ inch x 6/32.
c. 2 plastic screws, ¼ inch x 6/32.

1-4. COMPATIBLE EQUIPMENT

The Multimodule board must be used with a host iSBC microcomputer that includes an iSBX bus connector.

The Multimodule boards cannot directly access the Multibus bus structure. Multibus interfacing is provided indirectly via the host iSBC microcomputer.

Signals from the Multimodule board are accessible to an external device by means of the serial and/or parallel output connectors (J1, J2, J3) on the host iSBC microcomputer.

1-5. SPECIFICATIONS

The specificaitons of the iSBX 331 Math Multimodule Board are listed in table 1-1.

Table 1-1. Specifications

PHYSICAL CHARACTERISTICS	
Width:	6.35 cm (2.50 inches).
Length:	9.40 cm (3.70 inches).
Height:	1.40 cm (0.56 inch) Multimodule board only. 2.82 cm (1.13 inches) Multimodule and iSBC board.
Weight:	41 gm (1.44 ounces).
ENVIRONMENTAL REQUIREMENTS	
Operating Temperature:	0° to 55°C (32° to 131°F).
Relative Humidity:	To 90% without condensation.
POWER REQUIREMENTS	
	V_{cc} = +5 ±5% I_{cc} = 365 mA max. V_{dd} = +12 ±5% I_{dd} = 75 mA max.
INTERFACE COMPATIBILITY	
iSBX Bus:	Compatible with Intel iSBX Bus specifications.
I/O ADDRESSING	
	Addressing is contingent on the host iSBC microcomputer. Refer to Table 3-1 for specific addresses.

Table 1-1. Specifications (Continued)

TYPICAL COMMAND EXECUTION TIMES

Command Mnemonic	Time (μs)	Clock Cycles	Command Mnemonic	Time (μs)	Clock Cycles
ACOS	1800	7200	LOG	1400	5600
ASIN	1800	7200	LN	1400	5600
ATAN	1425	5700	NOP	1	4
CHSD	7	28	POPD	3	12
CHSF	5	20	POPF	3	12
CHSS	6	24	POPS	2	10
COS	1100	4300	PTOD	5	20
DADD	5	22	PTOF	5	20
DDIV	50	200	PTOS	4	16
DMUL	50	200	PUPI	4	16
DMUU	50	200	PWR	2500	10000
DSUB	10	40	SADD	4	18
EXP	1050	4200	SDIV	22	90
FADD	50	200	SIN	1050	4200
FDIV	44	170	SMUL	22	90
FIXD	50	200	SMUU	22	90
FIXS	35	140	SQRT	206	830
FLTD	50	200	SSUB	8	32
FLTS	30	120	TAN	1325	5300
FMUL	40	160	XCHD	6	26
FSUB	50	200	XCHF	6	26
			XCHS	5	18

NOTE: Total execution times may require allowances for operand transfer into the APU, command execution, and result retrieval from the APU. Except for command execution, these times will be heavily influenced by the nature of the data, the control interface used, the speed of memory, the CPU used, the priority allotted to DMA and Interrupt operations, the size and number of operands to be transferred, and the use of chained calculations, etc.

CHAPTER 2
PREPARATION FOR USE

2-1. INTRODUCTION

This chapter provides instructions for preparing and installing the iSBX 331 Math Multimodule Board. The instructions cover unpacking and inspection; installation considerations such as physical, power, cooling, and mounting requirements; jumper configurations; dc characteristics; connector assignments; and installation procedure.

2-2. UNPACKING AND INSPECTION

Inspect the shipping carton immediately upon receipt for evidence of mishandling during transit. If the shipping carton is severely damaged or water-stained, request that the carrier's agent be present when the carton is opened. If the carrier's agent is not present when the carton is opened and the contents of the carton are damaged, keep the carton and packing material for the agent's inspection.

For repairs to a product damaged in shipment, contact the Intel Technical Support Center to obtain a Return Authorization Number and further instructions. A purchase order will be required to complete the repair. A copy of the purchase order should be submitted to the carrier with your claim.

It is suggested that salvageable shipping cartons and packing material be saved for future use in the event the product must be shipped.

2-3. INSTALLATION CONSIDERATIONS

The Multimodule board is designed to interface with Intel iSBC Single Board Computers that contain an iSBX bus connector. Other installation considerations, such as power, cooling, mounting, and physical size requirements, are outlined in the following paragraphs.

2-4. POWER REQUIREMENTS

The board requires +5V (±0.25V) at 365 mA maximum, +12V (±0.6V) at 75 mA maximum, and ground. All power is drawn from the host iSBC microcomputer via the iSBX bus connector (P1).

NOTE

If modification of the Multimodule board is required, ensure that none of the iSBX bus specifications and standards are violated in doing so.

2-5. COOLING REQUIREMENTS

The Multimodule board dissipates 39.0 gram-calories/minute (0.16 BTU/minute) and adequate circulation of air must be provided to prevent a temperature rise above 55°C (131°F).

2-6. MOUNTING REQUIREMENTS

Figure 2-1 shows the iSBX bus connector and stand-off locations. The Multimodule board will mount onto any iSBC microcomputer containing an iSBX bus connector and the required stand-off hole. The mounting hardware supplied as part of the Multimodule board includes:

- 2 plastic screws, 6/32 x ¼ inch, separate from the board.
- 1 plastic stand-off, 6/32 x ½ inch, separate from the board.
- 36-pin connector P1, factory-installed onto the board.

NOTE

The Multimodule board, when installed onto a host iSBC microcomputer, occupies an additional card slot adjacent to the component side of the host microcomputer in an iSBC 604/614 Cardcage.

2-7. PHYSICAL DIMENSIONS

Physical dimensions of the Multimodule board are as follows:

- Width: 6.35 cm (2.50 inches).
- Length: 9.40 cm (3.70 inches).
- Height: 1.40 cm (0.56 inch)
  Multimodule board only.
  2.82 cm (1.13 inches)
  Multimodule with iSBC board.

Figure 2-1 shows the physical dimensions and figure 2-2 shows clearances for a Multimodule board mounted onto a host iSBC microcomputer. The dimensions shown in figure 2-2 are maximum heights.

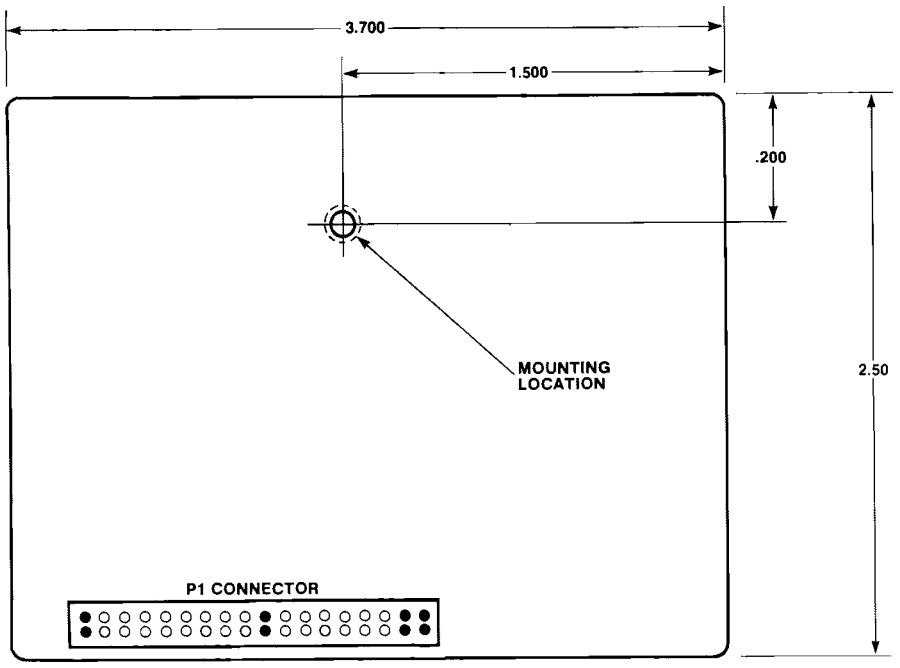


Figure 2-1. Board Dimensions (Inches)

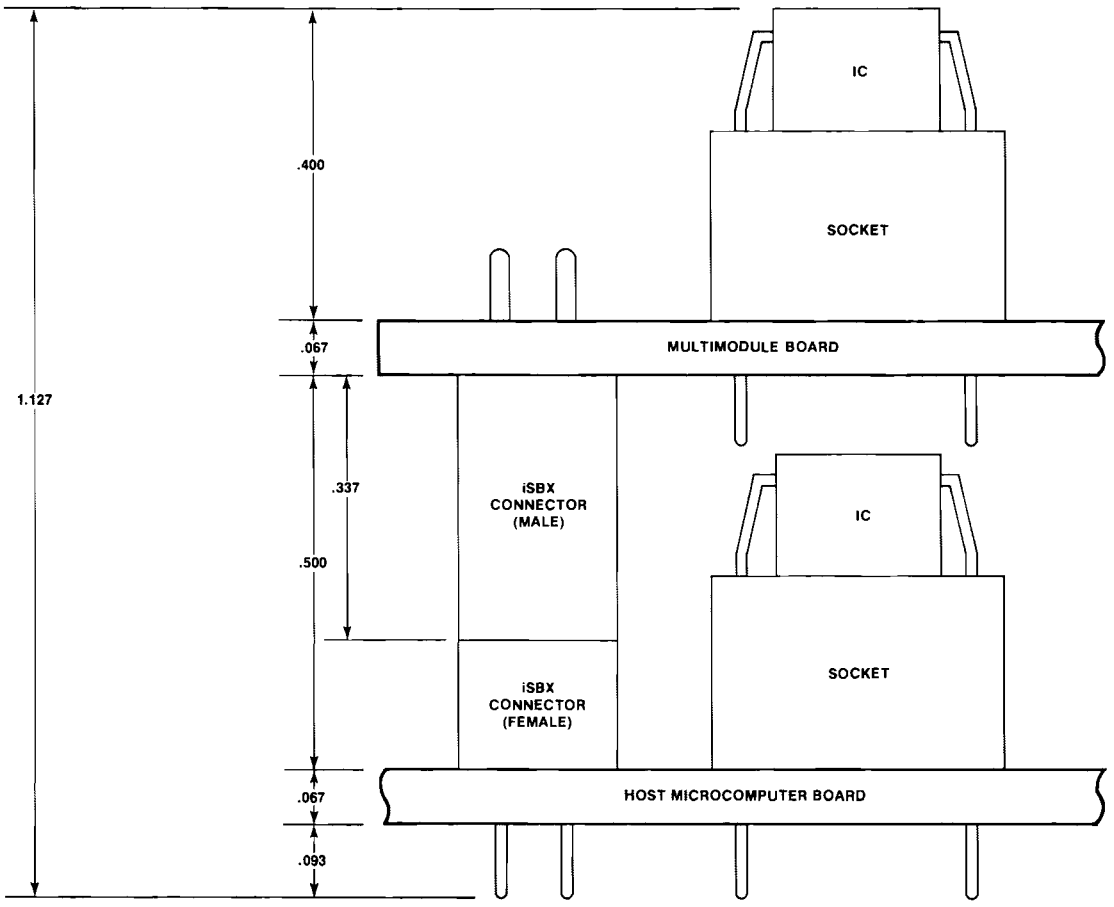


Figure 2-2. Mounting Clearances (Inches)

2-8. CONNECTOR CONFIGURATION

Connector P1 interfaces all input and output signals on the Multimodule board. The signals found on each pin of the P1 connector are listed in table 2-1.

2-9. JUMPER CONFIGURATION

The Multimodule board contains twenty-one jumper pads labeled E1 through E16 and E25 through E29. The functions of the user-configurable jumpers are outlined in the following paragraphs and in table 2-2.

The configuration of jumper pads E1 through E9 and E25 through E29 is performed before shipment and should not be modified by the user.

Jumper pads E11, E12, E14, E15, and E16 are configured with soldered-wire jumpers and jumper post E10 and E13 are wirewrapped to control the clock generation circuitry on the Multimodule board. The factory installed jumper from E10 to E13 is required for operation with the 16 MHz on-board clock. Jumper pads E11, E12, E14, E15, and E16 control four clock frequencies that may be connected so as to vary the clock rate of the APU. As shipped from the factory, the Multimodule board contains jumpers from E11 to E14, E15 to E16, and, as mentioned earlier, E10 to E13. This configures the board for 4 MHz operation.

The Multimodule board contains eight user-configurable wirewrap jumper posts labeled E17 through E24. The functions of each are outlined in the following paragraphs and in table 2-2.

Jumper posts E17 and E21 gate the service request signal from the APU when a jumper is installed; as shipped, the board does not include a jumper from E17 to E21.

Jumper posts E18 and E22 provide user control of the service acknowledge signal from the host iSBC microcomputer. The "as-shipped" configuration does not include a jumper connecting E18 to E22.

Jumper posts E19, E20, E23, and E24 control the end acknowledge (EACK) signal to the APU; either +5V (E19 to E23 connected) or ground (E20 to E24 connected). The factory configuration includes the jumper between E19 and E23.

2-10. INSTALLATION PROCEDURE

The Multimodule board mounts onto the host iSBC microcomputer. Install the board as follows:

a. With a nylon ¼ inch x 6/32 screw, secure the ½ inch spacer to the host iSBC microcomputer as shown in figure 2-3.

b. Locate pin 1 on the iSBX bus connector (P1) and align it with pin 1 of the iSBX bus connector on the host iSBC microcomputer.

c. Align the Multimodule board mounting hole with the spacer on the host iSBC microcomputer; reference figure 2-1 for hole location.

Table 2-1. iSBX™ Bus Pin Assignments

Pin	Mnemonic	Description	Pin	Mnemonic	Description
35	GND	SIGNAL GROUND	36	+5V	+5 Volts
33	MD0	MDATA BIT 0	34	—	Reserved
31	MD1	MDATA BIT 1	32	—	Reserved
29	MD2	MDATA BIT 2	30	OPT0	OPTION 0
27	MD3	MDATA BIT 3	28	OPT1	OPTION 1
25	MD4	MDATA BIT 4	26	—	Reserved
23	MD5	MDATA BIT 5	24	—	Reserved
21	MD6	MDATA BIT 6	22	MCS0/	M CHIP SELECT 0
19	MD7	MDATA BIT 7	20	MCS1/	M CHIP SELECT 1
17	GND	SIGNAL GROUND	18	+5V	+5 Volts
15	IORD/	IO READ COMMAND	16	MWAIT/	M WAIT
13	IOWRT/	IO WRITE COMMAND	14	MINTR0	M INTERRUPT 0
11	MA0	M ADDRESS 0	12	MINTR1	M INTERRUPT 1
9	—	Reserved	10	—	Reserved
7	—	Reserved	8	MPRT	M PRESENT
5	RESET	RESET	6	—	Reserved
3	GND	SIGNAL GROUND	4	+5V	+5 Volts
1	+12V	+12 Volts	2	—	Reserved

Table 2-2. User-Configurable Jumpers

Function	Configuration
Clock Enable	Factory-installed jumper between posts E10 and E13 allows the Multimodule board to operate with the on-board 16 MHz clock or an off-board clock via the option lines.
Clock Frequency Select	Posts E11, E12, E14, E15 and E16 configure the operating frequency of the APU as follows:
Option Enable	Posts E21 and E22 are user-configured to allow flexibility with the Multimodule board.
End Acknowledge	Posts E19, E20, E23, and E24 are user-configured to tie EACK high or low, dependent on the application. Jumper E19-E23 is the factory default.
Service Request/Acknowledge	Posts E17 and E18 are not connected during operation with the 8231 APU.

Frequency	Connections Required
4.0 MHz	*E16 to E15, E11 to E14
3.2 MHz	E12 to E15, E11 to E14
2.0 MHz	E16 to E15, E12 to E11

NOTE: * indicates factory installed jumpers.

d. Gently press the two boards together until the connector seats.

e. Secure the Multimodule board to the top of the spacer with the other ¼ inch x 6/32 screw.

NOTE

The placement of an installed Multimodule board and the host board connector number may vary according to the type of host iSBC microcomputer that is used.

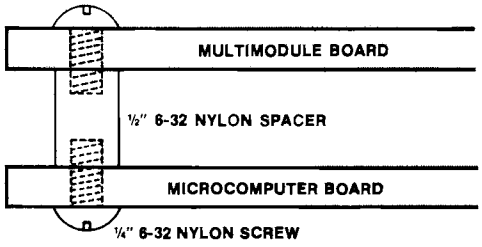


Figure 2-3. Mounting Technique

CHAPTER 3
PROGRAMMING INFORMATION

3-1. INTRODUCTION

This chapter contains information on programming the iSBX 331 Math Multimodule Board. Since the programming consists mainly of that of the 8231 APU, study this chapter of the text in close conjunction with Chapter 4 which covers the hardware of the Multimodule board. Included in this chapter are sections on addressing, command formats, status word formats, data formats, and programming examples.

3-2. ADDRESSING

The Multimodule board is addressed by the host iSBC microcomputer through use of IN and OUT instructions specifying one of the legal port addresses for the Multimodule board. These addresses are listed in table 3-1. The Multimodule board is used with a host microcomputer that may contain one or more Multimodule connectors, each of which may be accessed by 16 port addresses.

Data (operands) are transferred from an 8-bit host iSBC microcomputer to the Multimodule board and vice versa by issuing a READ or WRITE command to any one of the legal port addresses for data transfer; e.g., port address X0, X2, X4, or X6. Commands are transferred to the Multimodule board by issuing a WRITE command to any one of the legal command port addresses (X1, X3, X5, or X7). Status may be input from the Multimodule board to an 8-bit host iSBC microcomputer by issuing a READ command to the Multimodule board via any of the legal port addresses (X1, X3, X5, or X7). The APU on the Multimodule board may be reset by issuing a WRITE command (specifying any data pattern) to the Multimodule board via any of the legal reset port addresses (X8 through XF).

NOTE

Ensure that no other commands are issued to the Multimodule board within 3 μs after issuing the RESET command; the Multimodule board requires 3 μs of idle time to perform a RESET command with a 4 MHz clock frequency.

3-3. COMMAND FORMATS

Commands are issued to the APU via execution of OUT instructions in the host iSBC microcomputer. Table 3-2 lists the commands that the Multimodule board can execute. Each of these is described in detail in Appendix A.

Each command entered into the APU consists of a single 8-bit byte having the format illustrated in figure 3-1. Bits 0 through 4 select the operation to be performed, as shown in table 3-2. Bits 5 and 6 select the data format for the operation. If bit 5 is *high*, a fixed point format is specified. If bit 5 is *low*, the floating point format is specified. Bit 6 selects the precision of the data to be operated on by the fixed point commands (if bit 5 is *low*, bit 6 must be *low*). If bit 6 is *high*, single-precision (16-bit) operands are indicated; if bit 6 is *low*, double-precision (32-bit) operands are indicated. Results are undefined for all illegal combinations of bits in the command byte. Bit 7 indicates whether a service request is to be issued after the command is executed. If bit 7 is *high*, the service request output (SVREQ) becomes *high* at the conclusion of the command and will remain *high* until reset by a *low* on the service acknowledge pin (SVACK/) or until completion of execution of a subsequent command in which bit 7 is *low*. Each command issued to the APU requests post execution service dependent upon the state of bit 7 of the command byte. When bit 7 is *low*, the service request output (SVREQ) remains *low*.

Table 3-1. Multimodule Port Addresses

Function	Type of Operation	Connector Port Address (8-bit host)[1]	Connector Port Address (16-bit host)[2]
DATA/OPERAND TRANSFER	READ OR WRITE	X0, X2, X4, or X6	X0, X4, X8, or XC
COMMAND TRANSFER	WRITE	X1, X3, X5, or X7	X2, X6, XA, or XE
STATUS TRANSFER	READ	X1, X3, X5, or X7	X2, X6, XA, or XE
RESET	WRITE	X8 through XF	Y0, Y2, Y4, Y6, Y8, YA, YC, YE

NOTE:
1. The high order port address (X) for the Multimodule board is determined by the host iSBC microcomputer; refer to the respective Hardware Reference Manual to determine the upper address constraints.
2. Y is the additional chip select term required for a 16-bit interface.

The APU commands provide a method of doing high performance fixed and floating point arithmetic and a variety of floating point trigonometric and mathematical functions. When issued a command, the APU assumes that the required operands are located at the top of the stack (TOS) and next on the stack (NOS). The result of an operation is always returned to the TOS. The result of an operation is always the same precision and format as the operands.

The execution times of the APU commands are all data-dependent. Table 1-1 lists a typical execution time for each of the APU commands.

Command chaining may be performed by using a result (from a previous operation) as one of the operands for the next operation. This procedure reduces the amount of time required for the overall operation by eliminating the need to load one of the operands, for the next operation, into the stack; the result from the previous operation is already there.

Table 3-2 contains an abbreviated description of the commands that are performed by the Multimodule board. A more detailed description of the commands is contained in Appendix A.

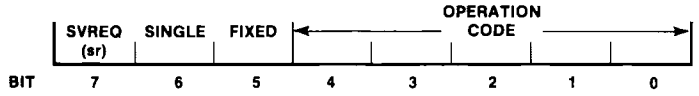


Figure 3-1. Command Format

Table 3-2. APU Commands

Command Mnemonic	Hex Code (sr = 1)	Hex Code (sr = 0)	Summary Description
			16-BIT FIXED-POINT OPERATIONS
SADD	EC	6C	Add TOS to NOS. Result to NOS. Pop Stack.
SSUB	ED	6D	Subtract TOS from NOS. Result to NOS. Pop Stack.
SMUL	EE	6E	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
SMUU	F6	76	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
SDIV	EF	6F	Divide NOS by TOS. Result to NOS. Pop Stack.
			32-BIT FIXED-POINT OPERATIONS
DADD	AC	2C	Add TOS to NOS. Result to NOS. Pop Stack.
DSUB	AD	2D	Subtract TOS from NOS. Result to NOS. Pop Stack.
DMUL	AE	2E	Multiply NOS by TOS. Lower result to NOS. Pop Stack.
DMUU	B6	36	Multiply NOS by TOS. Upper result to NOS. Pop Stack.
DDIV	AF	2F	Divide NOS by TOS. Result to NOS. Pop Stack.
			32-BIT FLOATING-POINT PRIMARY OPERATIONS
FADD	90	10	Add TOS to NOS. Result to NOS. Pop Stack.
FSUB	91	11	Subtract TOS from NOS. Result to NOS. Pop Stack.
FMUL	92	12	Multiply NOS by TOS. Result to NOS. Pop Stack.
FDIV	93	13	Divide NOS by TOS. Result to NOS. Pop Stack.
			32-BIT FLOATING-POINT DERIVED OPERATIONS
SQRT	81	01	Square Root of TOS. Result to TOS.
SIN	82	02	Sine of TOS. Result to TOS.
COS	83	03	Cosine of TOS. Result to TOS.
TAN	84	04	Tangent of TOS. Result to TOS.
ASIN	85	05	Inverse Sine of TOS. Result to TOS.
ACOS	86	06	Inverse Cosine of TOS. Result to TOS.
ATAN	87	07	Inverse Tangent of TOS. Result to TOS.
LOG	88	08	Common Logarithm of TOS. Result to TOS.
LN	89	09	Natural Logarithm of TOS. Result to TOS.
EXP	8A	0A	e raised to power in TOS. Result to TOS.
PWR	8B	0B	NOS raised to power in TOS. Result to NOS. Pop Stack.

Table 3-2. APU Commands (Continued)

Command Mnemonic	Hex Code (sr = 1)	Hex Code (sr = 0)	Summary Description
			DATA AND STACK MANIPULATION OPERATIONS
NOP	80	00	No Operation. Clear or set SVREQ.
FIXS	9F	1F	Convert TOS from floating point format to fixed point format (16-bit).
FIXD	9E	1E	Convert TOS from floating point format to fixed point format (32-bit).
FLTS	9D	1D	Convert TOS from fixed point format to floating point format (16-bit).
FLTD	9C	1C	Convert TOS from fixed point format to floating point format (32-bit).
CHSS	F4	74	Change sign of fixed point operand on TOS (16-bit).
CHSD	B4	34	Change sign of fixed point operand on TOS (32-bit).
CHSF	95	15	Change sign of floating point operand on TOS.
PTOS	F7	77	Push stack. Duplicate NOS in TOS (16-bit).
PTOD	B7	37	Push stack. Duplicate NOS in TOS (32-bit).
PTOF	97	17	Push stack. Duplicate NOS in TOS (floating point).
POPS	F8	78	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom (16-bit).
POPD	B8	38	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom (32-bit).
POPF	98	18	Pop stack. Old NOS becomes new TOS. Old TOS rotates to bottom (floating point).
XCHS	F9	79	Exchange TOS and NOS (16-bit).
XCHD	B9	39	Exchange TOS and NOS (32-bit).
XCHF	99	19	Exchange TOS and NOS (floating point).
PUPI	9A	1A	Push floating point constant π onto TOS. Previous TOS becomes NOS.

NOTES:
1. TOS mean Top of Stack. NOS means Next on Stack.
2. Appendix A provides detailed descriptions of each command function, including data ranges, accuracies, stack configurations, etc.
3. Many commands destroy one stack location (bottom of stack) during development of the result. The derived functions may destroy several stack locations. See Appendix A.
4. The trigonometric functions handle angles in radians, not degrees.
5. No remainder is available for the fixed-point divide functions.
6. Results will be undefined for any combination of command coding bits not specified in this table.

3-4. DATA FORMATS

Data operands for the APU must be loaded into the stack (in the APU) before a command is issued to the Multimodule board. The APU accepts operands in both floating and fixed point formats. Each is explained in detail in the following text.

3-5. FIXED POINT OPERANDS

Fixed point operands (shown in figures 3-2 and 3-3) may be represented in either single (16-bit) or double (32-bit) precision. Both, however, require that the operands be represented in a binary two's complement form.

The sign (positive or negative) of the operand is located in the most significant bit (MSB) of the operand. Positive values are represented with a sign (S) bit of "ZERO". Negative values are represented with a two's complement of the corresponding positive value with a sign bit of "ONE". The range of values that can be accommodated by each of these fixed point formats is −32,768 to +32,767 for single precision and −2,147,483,648 to +2,147,483,647 for double precision.

3-6. FLOATING POINT OPERANDS

The format for floating point operands in the APU is shown in figure 3-4. The mantissa is expressed as a 24-bit (fractional) value; the exponent is expressed as an unbiased two's complement 7-bit value with a range of −64 to +63. The most significant bit (bit 31) is the sign of the mantissa (0 = positive). The binary point is assumed to be to the left of the most significant bit (bit 23) of the mantissa. All floating point data operands must be normalized; bit 23 must be one for all operands except zero. The range of values that can be expressed in this format is $\pm(2.7 \times 10^{-20}$ to $9.2 \times 10^{+18})$ and zero.

Table 3-3 contains several numbers shown in decimal, hexadecimal floating point, and hexadecimal fixed point representations.

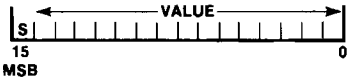


Figure 3-2. 16-Bit Fixed Point Format

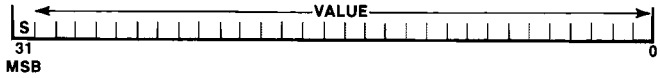


Figure 3-3. 32-Bit Fixed Point Format

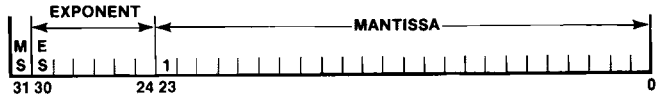


Figure 3-4. Floating Point Data Format

Table 3-3. Number Conversions

Decimal	16-Bit Fixed Point	32-Bit Floating Point
0	0000	00000000
1	0001	01800000
−1	FFFF	81800000
255	00FF	08FF0000
−255	FF01	88FF0000
2	0002	02800000
5	0005	03A00000
1000	03E8	0AFA0000
2377	0949	0C949000
1.537	0001	01C4BC6A
10.9825	000A	04AFB852
0.01234	0000	7ACA2DB6

3-7. STATUS BYTE FORMAT

APU status is provided by means of a status register in the APU. The format of the status byte is shown in figure 3-5.

If the BUSY bit in the status register is a "ONE", then the other status bits are not defined; if "ZERO" (indicating not busy), then the operation is completed and the other status bits are defined as follows:

BUSY: Indicates that APU is currently executing a command (1 = Busy).

SIGN: Indicates that the operand on the top of stack is negative (1 = Negative).

ZERO: Indicates that the operand on the top of stack is zero (1 = Value is zero).

ERROR CODE: This field contains an indication of the validity of the result of the last operation. The error codes are:

Bit 4321

- 0000 - No error.
- 1000 - Divide by zero.
- 0100 - Square root or log of negative number.
- 1100 - Argument of inverse sine, cosine, or e too large.
- XX10 - Underflow.
- XX01 - Overflow.

CARRY: Previous operation resulted in carry or borrow from most significant bit. (1 = Carry/Borrow, 0 = No Carry/No Borrow).

3-8. INTERRUPT

There is one interrupt line from the APU that may be used to generate an interrupt to the host microcomputer; END (MINTR1). The End interrupt line from the APU goes HIGH on command completion to indicate that command execution is completed. END (MINTR1) stays HIGH until status is read or until End Acknowledge (EACK) is asserted.

3-9. APU PROGRAMMING

The following text outlines some of the internal operations of the APU, including data stack control, data entry, and data removal. More information on the operation of the APU may be found in Chapter 4.

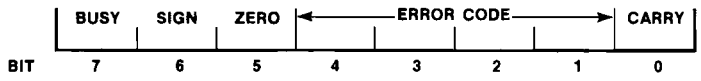


Figure 3-5. Status Byte Format

3-10. STACK CONTROL

The user interface to the APU provides access to a 16 byte data stack (within the APU) that may be used in two configurations as shown in figures 3-6 and 3-7.

Since single precision fixed point operands are 16 bits in length, eight operands may be maintained in the stack. When using double precision fixed point or floating point formats four operands may be stored. The stack, in these two configurations, can be visualized as shown in figures 3-6 and 3-7.

Operands are written onto the stack, eight bits at a time, in the order shown (B1, B2, B3, ...). The stack operates as a true push-down stack or LIFO stack. That is, the data last written in will be the data first read out. Within each stack entry, the least significant byte is entered first and retrieved last.

NOTE

To ensure proper operation of the APU when using the stack, always enter or remove one whole operand; in single precision format a "whole operand" includes 2 bytes, in double precision format it includes 4 bytes, as figures 3-6 and 3-7 show.

NOTE

Keep in mind when using the POP instruction and when retrieving data, that the stack operates as a "last-in, first-out" (LIFO) file with wrap-around capability.

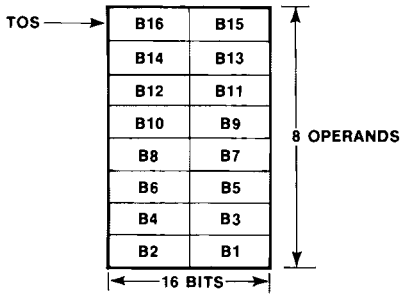


Figure 3-6. Single Precision Fixed Point Stack Format

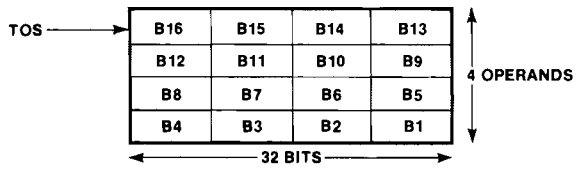


Figure 3-7. Double Precision Fixed/Floating Point Stack Format

3-11. DATA ENTRY TO STACK

When writing operands on the stack, the least significant byte must be entered first and most significant byte last. For entering operands and retrieving results, the number of operands/results must be equal to the proper number of bytes appropriate for the chosen format. Otherwise, the internal byte pointer in the APU will not be aligned properly. The APU single precision format requires 2 bytes of data per operand, and double precision and floating point formats require 4 bytes.

Each new operand entered onto the stack pushes down the previously entered operand to the "Next-On-Stack" (NOS) position. The data on the bottom of the stack before the entry is lost.

Figure 3-8 shows a typical stack loading sequence with 32-bit operands. Figure 3-8A shows the stack following entry of byte Z1 (the least significant byte of operand Z). Figure 3-8B shows the stack contents following entry of all four bytes of operand Z. After loading all bytes of operands X, Y, and Z, the stack appears as shown in figure 3-8D.

3-12. DATA REMOVAL FROM STACK

When reading the stack to retrieve the result of an operation, the most significant byte (MSB) will be available on the data bus first and the least significant byte (LSB) will be last.

The removal of data from the TOS causes the next successive entry to be redefined as the TOS. Data read from the TOS recirculates to the bottom of the stack.

Suppose that a command is issued to add operands X and Y in figure 3-9A. When the addition is completed, the APU generates a result (R) and stores it into the stack as shown in figure 3-9B. After the first byte (MSB) of R is retrieved from the stack, a shift occurs, as figure 3-9C shows. Figure 3-9D shows the stack after complete retrieval of R.

3-13. COMMAND ENTRY

After the appropriate number of bytes of data have been entered onto the stack, a command may be issued to perform an operation on that data. Commands which require two operands for execution (e.g., add) operate on the TOS and NOS values. Single operand commands operate only on the TOS. After a command is issued, the host iSBC microcomputer can continue execution of its program concurrently with the APU command execution.

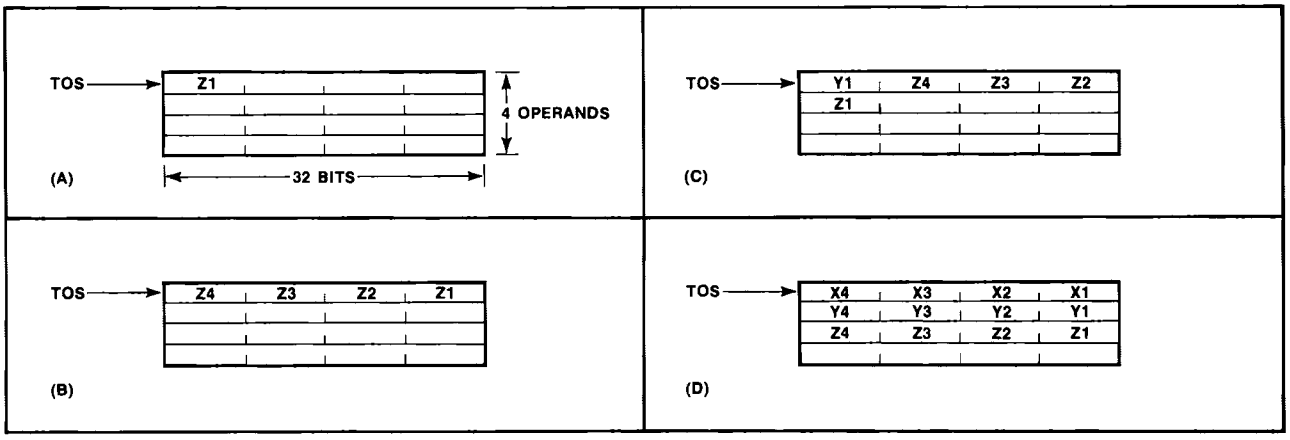


Figure 3-8. Double Precision Stack Loading Sequence

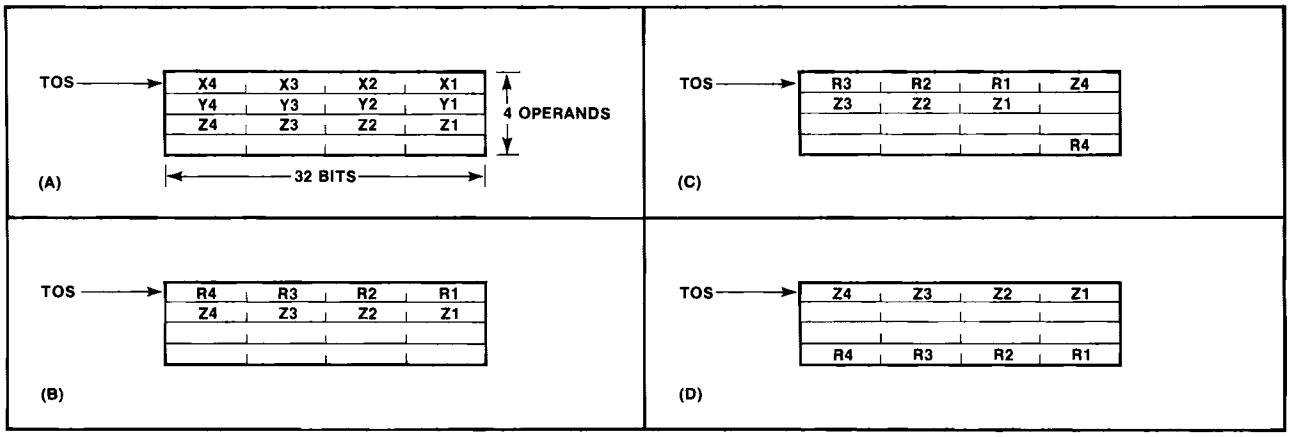


Figure 3-9. Double Precision Stack Unloading Sequence

3-14. STATUS RETRIEVAL

The APU status register can be read by the host iSBC microcomputer at any time (whether an operation is in progress or not). The status register is gated onto the data bus and may be input by the host iSBC microcomputer.

NOTE

When the BUSY bit in the status register is HIGH (indicating BUSY), the other status bits should be considered "undefined".

3-15. PROGRAMMING EXAMPLES

Five simple programming examples are listed in the following paragraphs.

3-16. RESET OPERATION

A typical RESET subroutine for the Multimodule board is given in table 3-4.

3-17. STATUS READ OPERATION

A typical Status Read Operation for the Multimodule board is given in table 3-5.

3-18. WRITE COMMAND OPERATION

A typical subroutine for writing commands to the Multimodule board is given in table 3-6.

3-19. WRITE DATA OPERATION

A typical subroutine for writing data into the APU stack on the Multimodule board is given in table 3-7.

3-20. READ DATA OPERATION

A typical subroutine for reading data from the APU stack on the Multimodule board is given in table 3-8.

Table 3-4. Typical RESET Subroutine

```
;FUNCTION-CHPRST: RESET MATH CHIP.
;USES-NOTHING; DESTROYS-NOTHING.

          CSEG
          PUBLIC    CHPRST
          EXTRN     BASAD
CHPRST:   OUT       LOW BASAD + 8   ;CONTENTS OF A IS UNIMPORTANT
          RET
          END
```

Table 3-5. Typical Status Read Subroutine

```
;FUNCTION- :READ MATH CHIP STATUS INTO A.
;USES-NOTHING; DESTROYS-A.

          CSEG
          PUBLIC    MSTAT
          EXTRN     BASAD
MSTAT:    IN        LOW BASAD + 1   ;BASE ADDRESS + 1
          RET
          END
```

Table 3-6. Typical WRITE Command Subroutine

```
;FUNCTION-WCMD: WRITE COMMAND IN C TO MATH CHIP.
;USES-C; DESTROYS-A.

          CSEG
          PUBLIC    WCMD
          EXTRN     BASAD
WCMD:     MOV       A,C
          OUT       LOW BASAD + 1   ;BASE ADDRESS + 1
          RET
          END
```

Table 3-7. Typical WRITE Data Subroutine

```
FUNCTION-WDATA: WRITE DATA TO MATH CHIP.
USES-DE: POINTS TO STARTING ADDRESS OF THE NUMBER IN MEMORY (LSB FIRST).
      C: BYTE COUNT (2 = SINGLE PRECISION, 4 = DOUBLE PRECISION).
DESTROYS-D, E, C, A, F/F.

          CSEG
          PUBLIC    WDATA
          EXTRN     BASAD
WDATA:    XCHG
LP1:      MOV       A,M             ;GET BYTE FROM MEMORY, LSB FIRST
          OUT       LOW BASAD + 0   ;SEND TO MATH CHIP (BASE ADDRESS + 0)
          INX       H               ;POINT TO NEXT BYTE IN MEMORY
          DCR       C               ;DEC LOOP COUNTER
          JNZ       LP1
          XCHG
          RET
          END
```

Table 3-8. Typical READ Data Subroutine

```
;FUNCTION-RDATA: READ DATA FROM MATH CHIP.
;USES-DE: POINTS TO STARTING ADDRESS OF THE NUMBER IN MEMORY-LSB FIRST.
        C: BYTE COUNT (4 = SINGLE PRECISION, 8 = DOUBLE PRECISION).
;DESTROYS-D, E, B, C, A, F/F.

          CSEG

          PUBLIC    RDATA
          EXTRN     BASAD

RDATA:    XCHG                        ;SINCE DATA IS READ FROM CHIP
          MVI       B,0               ;MSB FIRST AND IS STORED IN MEMORY
          DAD       B                 ;LSB FIRST, WE HAVE TO START AT THE
LP1:                                  ;END OF MEMORY AND WORK BACKWARDS.
          DCX       H                 ;DEC MEMORY POINTER
          IN        LOW BASAD + 0     ;GET DATA FROM MATH CHIP (BASE ADDRESS + 0)
          MOV       M,A               ;STORE IN MEMORY
          DCR       C                 ;DEC LOOP COUNTER
          JNZ       LP1
          XCHG
          RET
          END
```

CHAPTER 4
PRINCIPLES OF OPERATION

4-1. INTRODUCTION

This chapter provides a functional description and circuit analysis of the iSBX 331 Math Multimodule Board. The functional description includes paragraphs on the clock generator, the iSBX bus interface, the WAIT-state generator and the APU operation during execution of various commands. Each is shown in the block diagram included as figure 4-3. More detailed information on the operation of the APU may be found in Appendix A.

4-2. CLOCK GENERATOR OPERATION

The clock generation circuitry on the Multimodule board consists of a 16 MHz crystal, an 8224 clock generator chip, and a 74S163 frequency divider. Shown in figure 5-2, these elements generate a CLK input to the APU at at frequency dependent on the configuration of jumpers E11, E12, E14, E15, and E16. Reference Chapter 2 — *Jumper Configuration* for various wiring configurations of these jumpers.

4-3. iSBX™ BUS SIGNAL DESCRIPTION

Programmed control of the Multimodule board is achieved by controlling the iSBX bus interface signals to the APU chip. The interface signals to the APU are described in the following paragraphs and shown in figure 5-2.

RESET (Reset) — This active high input signal to the APU provides initialization for the chip. RESET also terminates any operation in progress, clears the status register, and places the APU into the idle state. Stack contents registers are not affected by RESET. This signal is derived from the microprocessor reset signal on the host iSBC microcomputer or generated on the Multimodule board as a result of a WRITE to any of the Reset port addresses by the host iSBC microcomputer.

MINTR1 (End Execution) — This active high output indicates that execution of the previously entered command is complete. It can be used as an interrupt request and is cleared by EACK/, RESET, or any READ or WRITE command to the APU. MINTR1, when enabled on the host iSBC microcomputer, sends an interrupt request (INT) to the microprocessor.

IORD/ (Read) — This active low input indicates that data or status is to be read from the APU onto the bus (MD0-MD7) if MCS0/ is low. The IORD/ signal is generated by the host iSBC microcomputer.

IOWRT/ (Write) — This active low input indicates that data or a command is to be written into the APU from the bus (MD0-MD7) if MCS0/ is low. The IOWRT/ signal is generated by the host iSBC microcomputer.

MCS0/ (Chip Select) — MCS0/ is an active low input signal which selects the APU chip and enables communication with the data bus (DB0-DB7) on the iSBC microcomputer. MCS0 is generated by the I/O command decode logic on the host iSBC microcomputer.

MCS1/ (Reset Select) — This active low input enables the APU Reset circuitry to be controlled by the host iSBC microcomputer. A low on IOWRT/ simultaneous with a low on MCS1/ causes the APU to be reset, allowing the programmer to return the APU to a known state.

MA0 (Command/Data) — In conjunction with the IORD/ and IOWRT/ signals, the MA0 control line input establishes the type of data exchange that is to be performed with the APU. MA0 is derived from the address bus in the host iSBC microcomputer. Table 4-1 shows the functional relationship between the signals.

EACK/ (End Acknowledge) — This active low input clears the end of execution output signal (MINTR1). If EACK/ is tied low, the MINTR1 output will be a pulse that is one clock period wide.

MD0-MD7 (Bidirectional Data Bus) — These eight bidirectional lines provide for transfer of commands, status, and data between the APU and the host iSBC microcomputer. The APU can drive the data bus only when MCS0/ and IORD/ are low (reference table 4-1).

OPT0/ (Service Acknowledge) — This active low input clears the service request output (OPT1) from the APU. OPT0/, when enabled on the Multimodule board and on the host iSBC microcomputer, generates a service acknowledge signal for the APU.

OPT1 (Service Request) — This active high output signal indicates that command execution is complete and that post execution service (bit 7 = 1) was requested in the previous command byte. OPT1 is cleared by OPT0/, by completion of a new command which does not require service (bit 7 = 0), or by

execution of RESET command. When enabled on the Multimodule board and the host iSBC microcomputer, OPT1 generates a service request signal for the APU.

MWAIT/ (Pause) — This active low output is a handshaking signal indicating that the APU is Busy executing a command and is unable to communicate with the CPU. MWAIT/ controls the READY line on the microprocessor and is active only while the APU is Busy executing a command.

Table 4-1. Control Signal Functions

MA0	IORD/	IOWRT/	MCS0/	Function
0	1	0	0	Push data byte into stack.
0	0	1	0	Pop data byte from stack.
1	1	0	0	Enter command.
1	0	1	0	Read status.

4-4. APU OPERATION

The APU is mounted in chip location U7 on the Multimodule board. In order to facilitate a better understanding of the APU, some of the internal operations must be explained. The following text outlines some of the internal operations of the APU, including data stack, data entry, data removal, command entry, command completion, and MWAIT/ control. More information on the operation of the APU may be found in the data sheet for the APU chip.

4-5. STACK CONTROL

The user interface to the APU provides access to a 16 byte data stack (within the APU) that may be used in two configurations as shown in figure 4-1 and 4-2. When operating in single precision format, the APU stores up to eight 16-bit operands in the stack. In double precision format, the APU stores four 32-bit operands. Operands are written into the stack in the order shown in figures 4-1 and 4-2. Each block (B1 through B16) in the figures represents eight bits of operand/data and is numbered in order of entry from bottom-of-stack (BOS) to top-of-stack (TOS).

NOTE

To ensure proper operation of the APU when using the stack, always enter or remove one whole operand; in single precision format a "whole operand" includes 2 bytes, in double precision format it includes 4 bytes, as figures 4-1 and 4-2 show.

NOTE

Keep in mind when using the POP instruction and when retrieving data, that the stack operates as a "last-in, first-out" (LIFO) file with wrap-around capability.

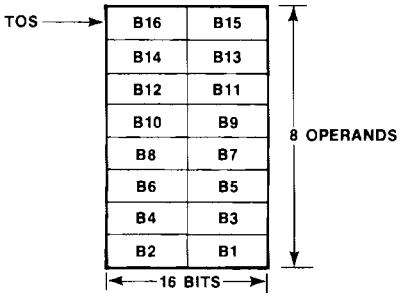


Figure 4-1. Single Precision Stack Format

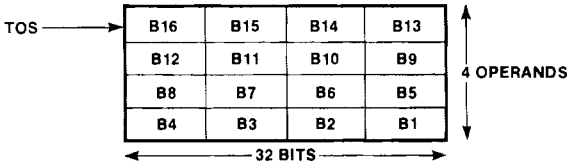


Figure 4-2. Double Precision Stack Format

4-6. DATA ENTRY TO STACK

Data is put onto the stack one byte at a time by performing an I/O WRITE. When writing operands on the stack, the least significant byte must be entered first and most significant byte last. For reading operands and retrieving results, the number of transactions must be equal to the proper number of bytes appropriate for the chosen format. Otherwise, the internal byte pointer will not be aligned properly. The APU single precision format requires 2 bytes and double precision format requires 4 bytes.

Data entry into the stack is accomplished by bringing the chip select line (MCS0/), the command/data line (MA0), and the write line (IOWRT/) *low*. The entry of each new operand "pushes down" the previously entered data and places the new data on the top of the stack (TOS). Data on the bottom of the stack (BOS) prior to the entry is over-written as the BOS recirculates to become TOS. In figures 4-1 and 4-2, the first operand entered onto the stack is B1 through B2 (B1 through B4 with double precision).

4-7. DATA REMOVAL FROM STACK

Data is removed from the stack one byte at a time by performing an I/O READ to the APU. When reading the stack to retrieve the result of an operand, the most significant byte will be available on the data bus first and the least significant byte will be last.

Data is removed from the APU stack by bringing the chip select line (MCS0/), the command/data line (MA0), and the read line (IORD/) *low*. The removal of each word redefines the TOS to be the next word on the stack (NOS). The TOS position recirculates to become BOS. The first operand removed from a full stack (reference figures 4-1 and 4-2) is B16 through B15 (B16 through B13) for double precision).

4-8. COMMAND ENTRY TO APU

After the appropriate number of operands are entered onto the stack, a command may be issued to perform an operation with the operands. Commands are issued via the bidirectional data bus (MD0-MD7) to the APU by bringing the chip select line (MCS0/) *low*, the command/data line (MA0) *high*, and the write line (IOWRT/) *low*. After issuing the command to the APU, the microprocessor can continue execution of its program concurrent with the command execution in the APU, unless MWAIT/ is asserted by the APU. Reference the programming examples contained in paragraph 3-15.

4-9. COMMAND COMPLETION

The APU signals the completion of command execution by raising the end execution line (MINTR1). Simultaneously, the busy bit in the status register is cleared, and the service request bit of the command register is evaluated; if the service request bit is high, then SVREQ = 1. The end execution line (MINTR1) is cleared by a low level on the end acknowledge (EACK) line or by any access of the APU by the host microcomputer. The service request line is cleared by a low level on the service acknowledge line (OPT0), or by completion of a subsequent command which does not request service.

4-10. WAIT-STATE REQUEST

The WAIT-state Request Generator circuitry includes logic elements U1, U3, U4, and U5. These elements allow generation of a WAIT-state request on cue from the APU (via the PAUSE output) during each READ and WRITE (via U1).

In the idle state, MCS0/, IORD/, and IOWRT/ are inactive (*high*). The *low* output from U4 pin 3 clears U1 and holds pin 4 of U4 *low*. This forces U4 pin 6 *high* and U3 pin 12 *low*. When MCS0/ goes active (*low*), it holds U5 pin 6 *low*, asserting MWAIT/. MWAIT/ will remain *low* until a command is given to the Multimodule board.

When either IORD/ or IOWRT/ goes *low*, the output of U4 pin 3 becomes *high*, removing the clear from U1. U1 then begins to shift-in ones. Approximately 200 ns after a command is received, U1 pin 10 goes *high*. If PAUSE/ is *low*, the output of U4 pin 6 won't change and MWAIT/ will remain *low* until PAUSE/ goes *high*. If PAUSE/ doesn't go *low* within 200 ns after the command is received, then MWAIT/ goes *high* when U1 pin 10 goes *high*.

PAUSE/ is pulled *low* by the APU under the following conditions:

1. A previously initiated operation is in progress (device busy) and Command Entry has been attempted. In this case, the PAUSE/ line will be pulled *low* and remain *low* until completion of the current command execution. It will then go *high* permitting entry of the new command.
2. A previously initiated operation is in progress and stack access has been attempted. In this case, the PAUSE/ line will be pulled *low*, will remain in that state until execution is complete, and will then be raised to permit completion of the stack access.
3. The APU is not busy, and data removal has been requested. PAUSE/ will be pulled *low* for the length of time necessary to transfer the byte from the top of stack to the interface latch, and will then go *high*, indicating availability of the data.
4. The APU is not busy, and a data entry has been requested. PAUSE/ will be pulled *low* for the length of time required to ascertain if the preceding data byte, if any, has been written to the stack. If so, PAUSE/ will immediately go *high*. If not, PAUSE/ will remain *low* until the interface latch is free and will then go *high*.
5. A status read has been requested. PAUSE/ will be pulled *low* for the length of time necessary to transfer the status to the interface latch, and will then be raised to permit completion of the status read. Status may be read whether or not the APU is busy.

NOTE

When MWAIT/ goes *low*, the APU expects the bus control signals present at the time to remain stable until MWAIT/ goes *high*.

4-11. RESET OPERATION

The RESET generation circuitry includes logic elements U3, U4, U5, and U2 on the Multimodule board. These elements allow generation of a RESET signal in one of two ways: either synchronous with the host iSBC microcomputer reset, or an execution of a RESET command.

When the MCS1/ and IOWRT/ are both low, the RESET command is recognized by logic element U5. Flip-flop U4 sets, yielding a *HIGH* at pin 11 which resets the APU. The *HIGH* also removes the clear from U2, allowing it to count to overflow while the APU is held reset. The counter-generated delay allows the APU the required time to reset (8 APU clock cycles). As the counter (U2) overflows, it resets flip-flop U4, removes the reset from the APU, and allows restart of normal operation on the Multimodule board.

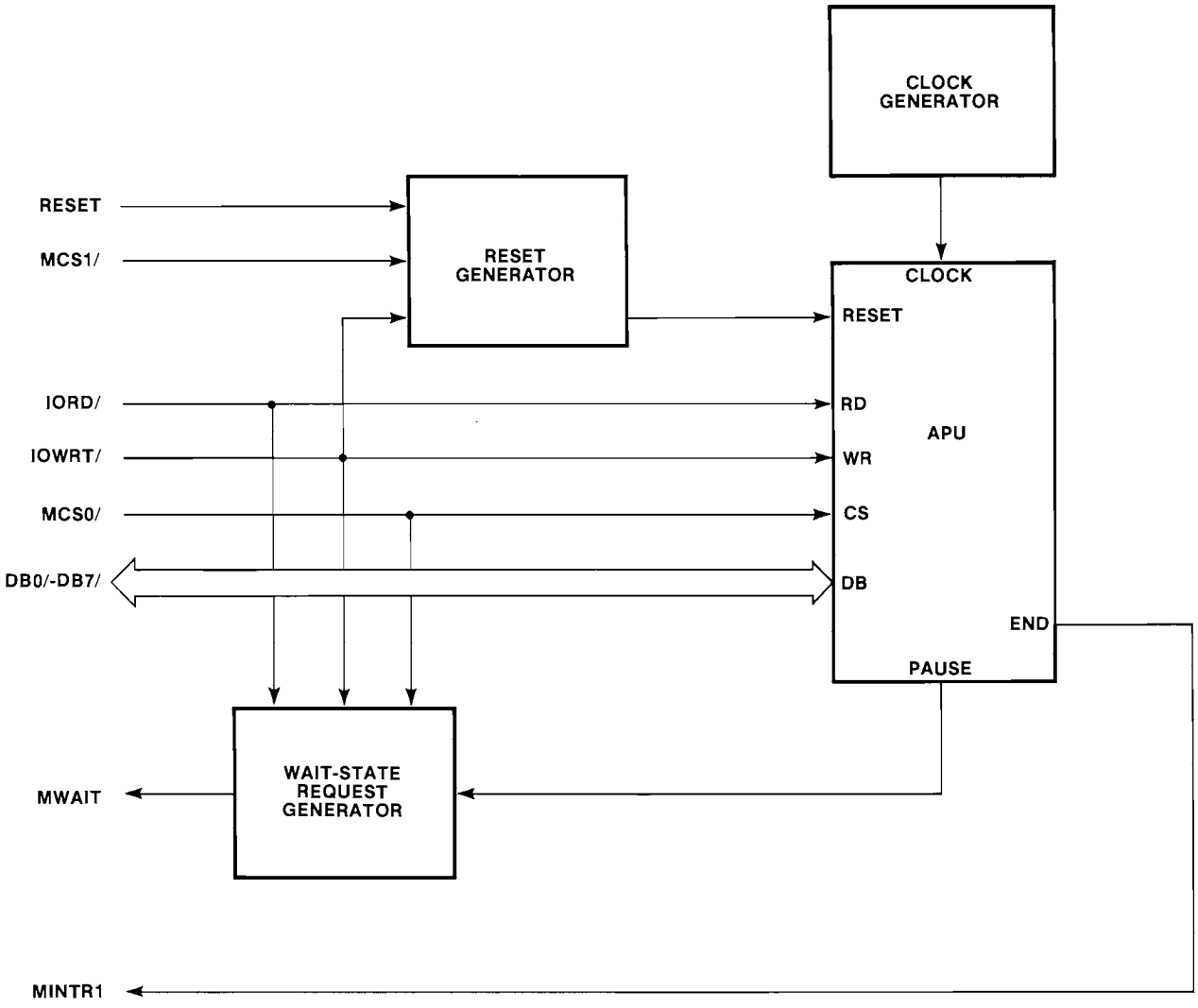


Figure 4-3. iSBX 331™ Functional Block Diagram

CHAPTER 5
SERVICE INFORMATION

5-1. INTRODUCTION

This chapter provides a list of replaceable parts, service diagrams, and service and repair assistance instructions for the iSBX 331 Math Multimodule Board.

5-2. REPLACEABLE PARTS

Table 5-1 provides a list of replaceable parts for the Multimodule board. Table 5-2 identifies and locates the manufacturers specified in the MFR CODE column in table 5-1. Intel parts that are available on the open market are listed in the MFR CODE column as "COML"; every effort should be made to procure these parts from a local (commercial) distributor.

5-3. SERVICE DIAGRAMS

The parts location diagram and schematic diagram are provided in figures 5-1 and 5-2, respectively. On the schematic diagram, a signal mnemonic that ends with a slash (e.g., MCS0/) is active low. Conversely, a signal mnemonic without a slash (e.g., OPT0) is active high.

Table 5-2. Manufacturers Codes

Mfr. Code	Manufacturer	Address
Intel	Intel	Santa Clara, CA
TI	Texas Instruments	Dallas, TX
AMP	AMP, Inc.	Harrisburg, PA
COML	Any commercial source. Order by description (OBD).	

5-4. SERVICE AND REPAIR ASSISTANCE

United States customers can obtain service and repair assistance by contacting the Intel Product Service Hotline in Phoenix, Arizona. Customers outside the United States should contact their sales source (Intel Sales Office or Authorized Distributor) for service information and repair assistance.

Before calling the Product Service Hotline, your should have the following information available:

a. Date you received the product.

b. Complete part number of the product (including dash number). On boards, this number is

Table 5-1. Replaceable Parts

Reference Designation	Description	Mfr. Part No.	Mfr. Code	Qty.
U1	IC, 74LS174, QUAD D-Type Flip-flop	SN74LS174	TI	1
U2	IC, 74LS164, 8-bit Shift Register	SN74LS164	TI	1
U3	IC, 74LS04, HEX Inverter	SN74LS04	TI	1
U4	IC, 74LS00, QUAD 2 Input NAND	SN74LS00	TI	1
U5	IC, 74LS32, QUAD 2 Input OR	SN74LS32	TI	1
U6	IC, 74S163, Synchronous 4-bit Counter	SN74S163	TI	1
U7	IC, 8231, Arithmetic Processing Unit	8231	Intel	1
U8	IC, 8224, Clock Generator/Driver	8224	Intel	1
Y1	Crystal, 16.0 MHz	OBD	COML	1
R1, 3	Resistor, 1K ohm	OBD	COML	2
R2, 4	Resistor, 10K ohm	OBD	COML	2
C1, 2, 3, 4, 5, 9	Capacitor, 0.1 μf, +80% −20%, 50V	OBD	COML	4
C6, 7	Capacitor, 22 μf	OBD	COML	2
C8	Capacitor, 10 pf, ±5%, 500V	OBD	COML	1
E10, 13, 17-24	Wire wrap stakes	OBD	COML	10
P1	Connector, 36-pin, female	103109-001	Intel	1
	Spacer, Nylon 6/32 by .50	OBD	COML	1
	Screw, Nylon 6/32 by .25	OBD	COML	2

usually silk-screened onto the board. On other MCSD products, it is usually stamped on a label.

c. Serial number of product. On boards, this number is usually stamped on the board. On other MCSD products, the serial number is usually stamped on a label.

d. Shipping and billing addresses.

e. If your Intel product warranty has expired, you must provide a purchase order number for billing purposes.

f. If you have an extended warranty agreement, be sure to advise the Hotline personnel of this agreement.

Use the following numbers for contacting the Intel Product Service Hotline:

Telephone

All U.S. locations,
except Alaska, Arizona, & Hawaii:

(800) 528-0595

All other locations: (602) 869-4600

TWX Number

910 - 951 - 1330

Always contact the Product Service Hotline before returning a product to Intel for repair. You will be given a repair authorization number, shipping instructions, and other important information which will help Intel provide you with fact, efficient service. If you are returning the product because of damage sustained during shipment or if the product is out of warranty, a purchase order is required before Intel can initiate the repair.

In preparing the product for shipment to the Repair Center, use the original factory packing material, if possible. If this material is not available, wrap the product in a cushioning material such as Air Cap TH-240, manufactured by the Sealed Air Corporation, Hawthorne, N.J. Then enclose in a heavy duty corrugated shipping carton, and label "FRAGILE" to ensure careful handling. Ship only to the address specified by Product Service Hotline personnel.

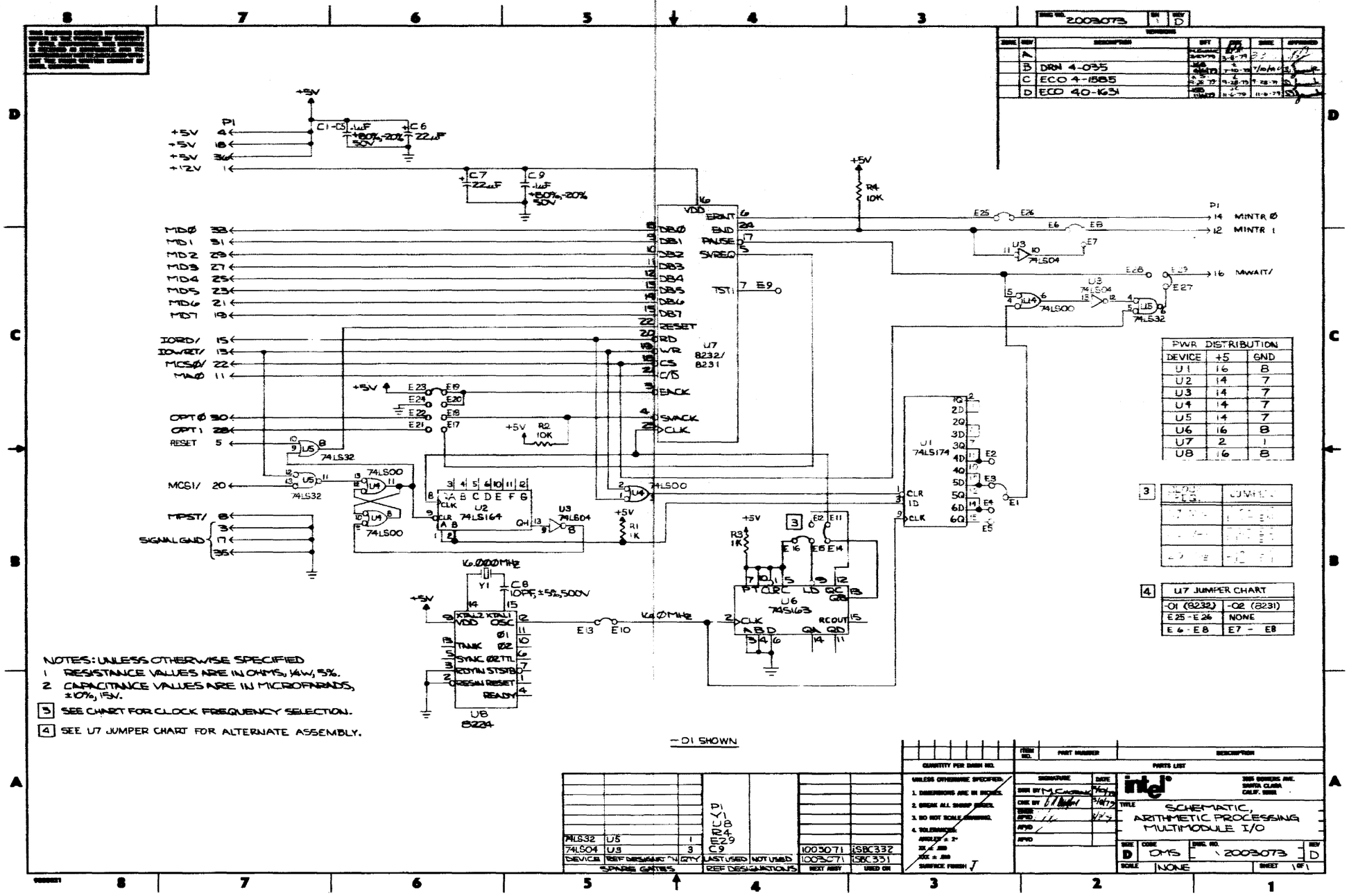


Figure 5-2. iSBX 331™ Math Multimodule ™ Board Schematic Diagram

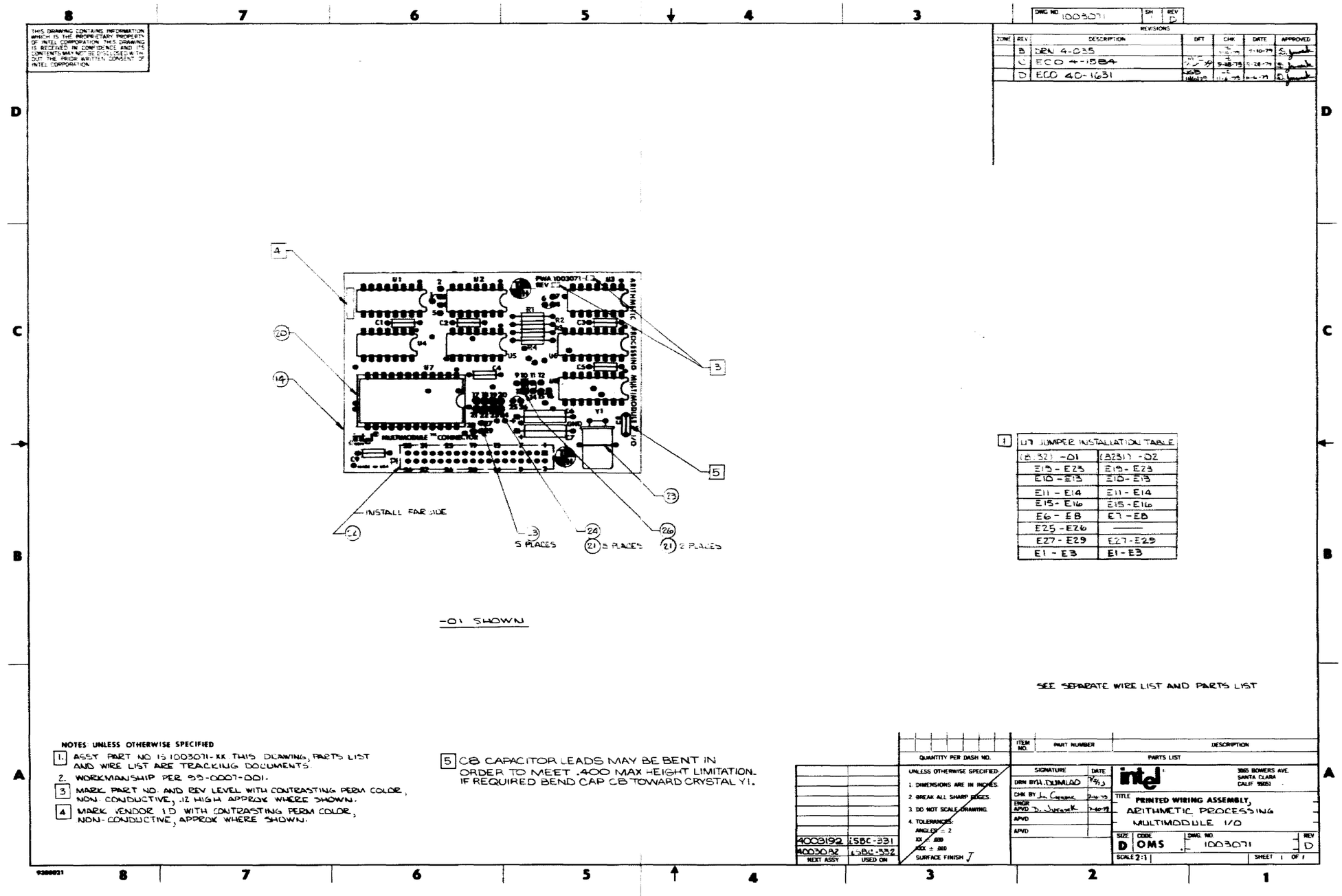


Figure 5-1. iSBX 331™ Math Multimodule ™ Board Parts Location Diagram

APPENDIX A
8231 APU COMMAND DESCRIPTION

A-1. COMMAND DESCRIPTIONS

This appendix contains detailed descriptions of the 8231 APU commands. They are arranged in alphabetical order by command mnemonic. In the descriptions, TOS means Top Of Stack and NOS means Next On Stack. SR marks the Service Request Enable bit of each command byte.

In some operations exponent overflow or underflow may be possible. When this occurs, the exponent returned in the result will be 128 greater or smaller than its true value.

Many of the functions use portions of the data stack as scratch storage during development of the results. Thus previous values in those stack locations will be lost. Scratch locations destroyed are listed in the command descriptions and shown by the crossed-out locations in the Stack Contents "After" diagram.

Execution times are listed in terms of clock cycles and may be converted into time values by multiplying by the clock period used. Table A-1 lists the command mnemonics in alphabetical order. A detailed explanation of the operation of each command is contained in the following text.

Table A-1. Command Mnemonic

ACOS	ARCCOSINE	LOG	COMMON LOGARITHM
ASIN	ARCSINE	LN	NATURAL LOGARITHM
ATAN	ARCTANGENT	NOP	NO OPERATION
CHSD	CHANGE SIGN DOUBLE	POPD	POP STACK DOUBLE
CHSF	CHANGE SIGN FLOATING	POPF	POP STACK FLOATING
CHSS	CHANGE SIGN SINGLE	POPS	POP STACK SINGLE
COS	COSINE	PTOD	PUSH STACK DOUBLE
DADD	DOUBLE ADD	PTOF	PUSH STACK FLOATING
DDIV	DOUBLE DIVIDE	PTOS	PUSH STACK SINGLE
DMUL	DOUBLE MULTIPLY LOWER	PUPI	PUSH π
DMUU	DOUBLE MULTIPLY UPPER	PWR	POWER (X^Y)
DSUB	DOUBLE SUBTRACT	SADD	SINGLE ADD
EXP	EXPONENTIATION (e^X)	SDIV	SINGLE DIVIDE
FADD	FLOATING ADD	SIN	SINE
FDIV	FLOATING DIVIDE	SMUL	SINGLE MULTIPLY LOWER
FIXD	FIX DOUBLE	SMUU	SINGLE MULTIPLY UPPER
FIXS	FIX SINGLE	SQRT	SQUARE ROOT
FLTD	FLOAT DOUBLE	SSUB	SINGLE SUBTRACT
FLTS	FLOAT SINGLE	TAN	TANGENT
FMUL	FLOATING MULTIPLY	XCHD	EXCHANGE OPERANDS DOUBLE
FSUB	FLOATING SUBTRACT	XCHF	EXCHANGE OPERANDS FLOATING
		XCHS	EXCHANGE OPERANDS SINGLE

ACOS

32-BIT FLOATING-POINT INVERSE COSINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	1	0

Hex Coding: 86 with sr = 1
06 with sr = 0

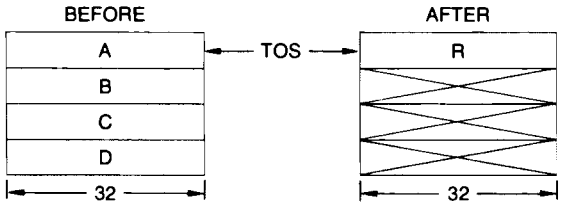
Execution Time: 6304 to 8284 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse cosine of A. The result R is a value in radians between 0 and π. Initial operands A, B, C and D are lost.
ACOS will accept all input data values within the range of −1.0 to +1.0. Values outside this range will return an error code of 1100 in the status register.

Accuracy: ACOS exhibits a maximum relative error of 2.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



ATAN

32-BIT FLOATING-POINT INVERSE TANGENT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	1	1

Hex Coding: 87 with sr = 1
07 with sr = 0

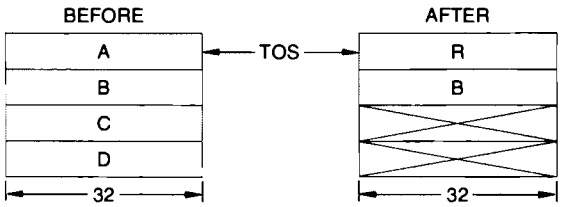
Execution Time: 4992 to 6536 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse tangent of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, C and D are lost. Operand B is unchanged.
ATAN will accept all input data values that can be represented in the floating point format.

Accuracy: ATAN exhibits a maximum relative error of 3.0×10^{-7} over the input data range.

Status Affected: Sign, Zero

STACK CONTENTS



ASIN

32-BIT FLOATING-POINT INVERSE SINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	0	1

Hex Coding: 85 with sr = 1
05 with sr = 0

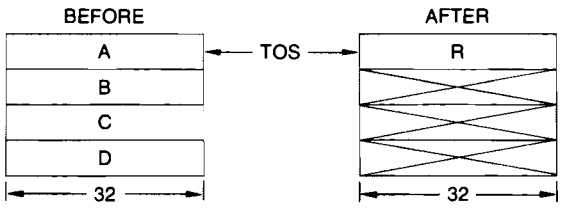
Execution Time: 6230 to 7938 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point inverse sine of A. The result R is a value in radians between $-\pi/2$ and $+\pi/2$. Initial operands A, B, C and D are lost.
ASIN will accept all input data values within the range of −1.0 to +1.0. Values outside this range will return an error code of 1100 in the status register.

Accuracy: ASIN exhibits a maximum relative error of 4.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



CHSD

32-BIT FIXED-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	0	0

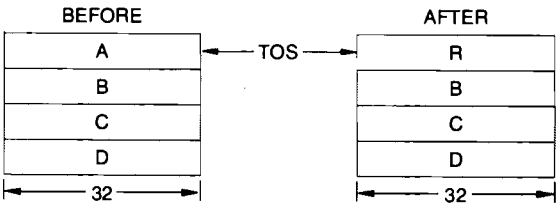
Hex Coding: B4 with sr = 1
34 with sr = 0

Execution Time: 26 to 28 clock cycles

Description:
The 32-bit fixed-point two's complement integer operand A at the TOS is subtracted from zero. The result R replaces A at the TOS. Other entries in the stack are not disturbed.
Overflow status will be set and the TOS will be returned unchanged when A is input as the most negative value possible in the format since no positive equivalent exists.

Status Affected: Sign, Zero, Error Field (overflow)

STACK CONTENTS



CHSF

32-BIT FLOATING-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	1	0	1

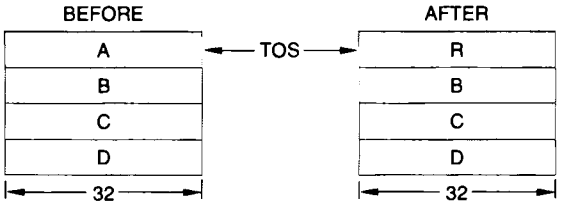
Hex Coding: 95 with sr = 1
15 with sr = 0

Execution Time: 16 to 20 clock cycles

Description:
The sign of the mantissa of the 32-bit floating-point operand A at the TOS is inverted. The result R replaces A at the TOS. Other stack entries are unchanged.
If A is input as zero (mantissa MSB = 0), no change is made.

Status Affected: Sign, Zero

STACK CONTENTS



CHSS

16-BIT FIXED-POINT SIGN CHANGE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	0	0

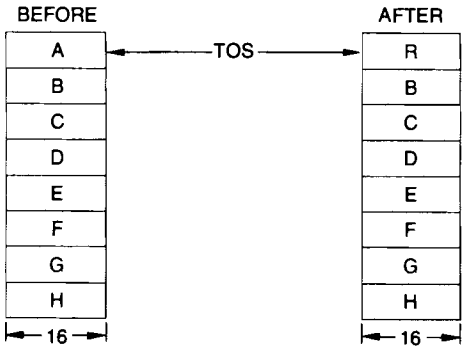
Hex Coding: F4 with sr = 1
74 with sr = 0

Execution Time: 22 to 24 clock cycles

Description:
16-bit fixed-point two's complement integer operand A at the TOS is subtracted from zero. The result R replaces A at the TOS. All other operands are unchanged.
Overflow status will be set and the TOS will be returned unchanged when A is input as the most negative value possible in the format since no positive equivalent exists.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



COS

32-BIT FLOATING-POINT COSINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	1	1

Hex Coding: 83 with sr = 1
03 with sr = 0

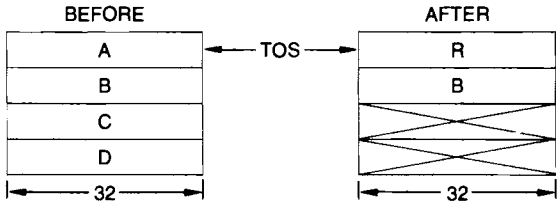
Execution Time: 3840 to 4878 clock cycles

Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point cosine of A. A is assumed to be in radians. Operands A, C and D are lost. B is unchanged.
The COS function can accept any input data value that can be represented in the data format. All input values are range reduced to fall within an interval of $-\pi/2$ to $+\pi/2$ radians.

Accuracy: COS exhibits a maximum relative error of 5.0 x 10^{-7} for all input data values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero

STACK CONTENTS



DADD

32-BIT FIXED-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	0	0

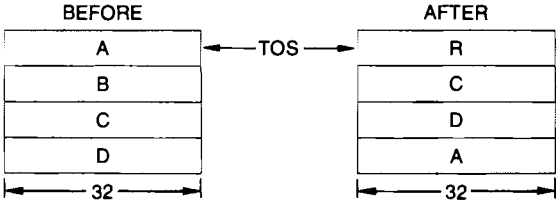
Hex Coding: AC with sr = 1
2C with sr = 0

Execution Time: 20 to 22 clock cycles

Description:
The 32-bit fixed-point two's complement integer operand A at the TOS is added to the 32-bit fixed-point two's complement integer operand B at the NOS. The result R replaces operand B and the Stack is moved up so that R occupies the TOS. Operand B is lost. Operands A, C and D are unchanged. If the addition generates a carry it is reported in the status register.
If the result is too large to be represented by the data format, the least significant 32 bits of the result are returned and overflow status is reported.

Status Affected: Sign, Zero, Carry, Error Field

STACK CONTENTS



DDIV

32-BIT FIXED-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	1	1

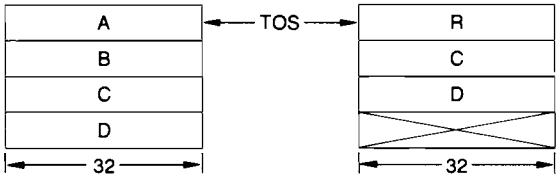
Hex Coding: AF with sr = 1
2F with sr = 0

Execution Time: 196 to 210 clock cycles when A ≠ 0
18 clock cycles when A = 0.

Description:
The 32-bit fixed-point two's complement integer operand B at NOS is divided by the 32-bit fixed-point two's complement integer operand A at the TOS. The 32-bit integer quotient R replaces B and the stack is moved up so that R occupies the TOS. No remainder is generated. Operands A and B are lost. Operands C and D are unchanged.
If A is zero, R is set equal to B and the divide-by-zero error status will be reported. If either A or B is the most negative value possible in the format, R will be meaningless and the overflow error status will be reported.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



DMUU

32-BIT FIXED-POINT MULTIPLY, UPPER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	1	0

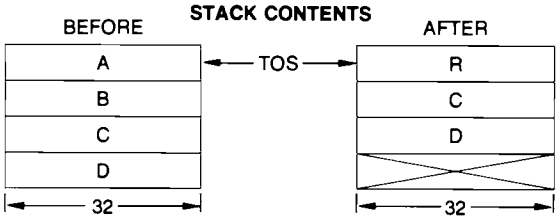
Hex Coding: B6 with sr = 1
36 with sr = 0

Execution Time: 182 to 218 clock cycles

Description:
The 32-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at the NOS. The 32-bit most significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The least significant half of the product is lost. Operands A and B are lost. Operands C and D are unchanged.
If A or B was the most negative value possible in the format, overflow status is set and R is meaningless.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



DMUL

32-BIT FIXED-POINT MULTIPLY, LOWER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	1	0

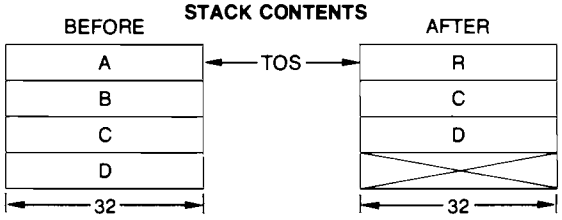
Hex Coding: AE with sr = 1
2E with sr = 0

Execution Time: 194 to 210 clock cycles

Description:
The 32-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at the NOS. The 32-bit least significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The most significant half of the product is lost. Operands A and B are lost. Operands C and D are unchanged.
The overflow status bit is set if the discarded upper half was non-zero. If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



DSUB

32-BIT FIXED-POINT SUBTRACT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	0	1	1	0	1

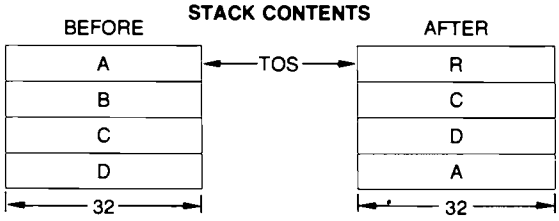
Hex Coding: AD with sr = 1
2D with sr = 0

Execution Time: 38 to 40 clock cycles

Description:
The 32-bit fixed-point two's complement operand A at the TOS is subtracted from the 32-bit fixed-point two's complement operand B at the NOS. The difference R replaces operand B and the stack is moved up so that R occupies the TOS. Operand B is lost. Operands A, C and D are unchanged.
If the subtraction generates a borrow it is reported in the carry status bit. If A is the most negative value that can be represented in the format the overflow status is set. If the result cannot be represented in the data format range, the overflow bit is set and the 32 least significant bits of the result are returned as R.

Status Affected: Sign, Zero, Carry, Overflow

STACK CONTENTS



EXP

32-BIT FLOATING-POINT e^X

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	1	0

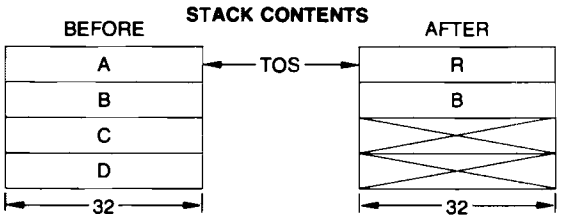
Hex Coding: 8A with sr = 1
0A with sr = 0

Execution Time: 3794 to 4878 clock cycles for $|A| \leq 1.0 \times 2^5$
34 clock cycles for $|A| > 1.0 \times 2^5$

Description:
The base of natural logarithms, e, is raised to an exponent value specified by the 32-bit floating-point operand A at the TOS. The result R of e^A replaces A. Operands A, C and D are lost. Operand B is unchanged.
EXP accepts all input data values within the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$. Input values outside this range will return a code of 1100 in the error field of the status register.

Accuracy: EXP exhibits a maximum relative error of 5.0×10^{-7} over the valid input data range.

Status Affected: Sign, Zero, Error Field



FDIV

32-BIT FLOATING-POINT DIVIDE

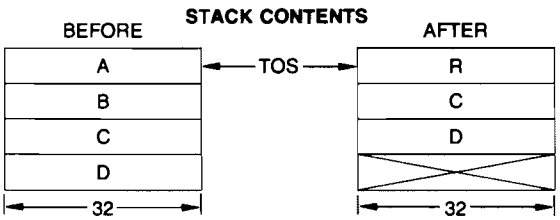
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	1	1

Hex Coding: 93 with sr = 1
13 with sr = 0

Execution Time: 154 to 184 clock cycles for A ≠ 0
22 clock cycles for A = 0

Description:
32-bit floating-point operand B at NOS is divided by 32-bit floating-point operand A at the TOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
If operand A is zero, R is set equal to B and the divide-by-zero error is reported in the status register. Exponent overflow or underflow is reported in the status register, in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field



FADD

32-BIT FLOATING-POINT ADD

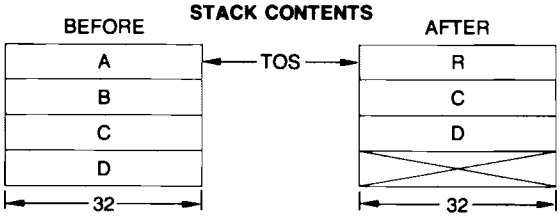
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	0

Hex Coding: 90 with sr = 1
10 with sr = 0

Execution Time: 54 to 368 clock cycles for A ≠ 0
24 clock cycles for A = 0

Description:
32-bit floating-point operand A at the TOS is added to 32-bit floating-point operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
Exponent alignment before the addition and normalization of the result accounts for the variation in execution time. Exponent overflow and underflow are reported in the status register, in which case the mantissa is correct and the exponent is offset by 128.

Status Affected: Sign, Zero, Error Field



FIXD

32-BIT FLOATING-POINT TO 32-BIT FIXED-POINT CONVERSION

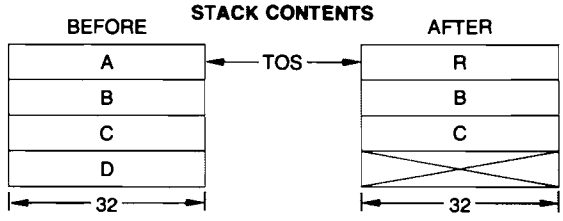
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	1	0

Hex Coding: 9E with sr = 1
1E with sr = 0

Execution Time: 90 to 336 clock cycles

Description:
32-bit floating-point operand A at the TOS is converted to a 32-bit fixed-point two's complement integer. The result R replaces A. Operands A and D are lost. Operands B and C are unchanged.
If the integer portion of A is larger than 31 bits when converted, the overflow status will be set and A will not be changed. Operand D, however, will still be lost.

Status Affected: Sign, Zero Overflow



FIXS

32-BIT FLOATING-POINT TO 16-BIT FIXED-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	1	1

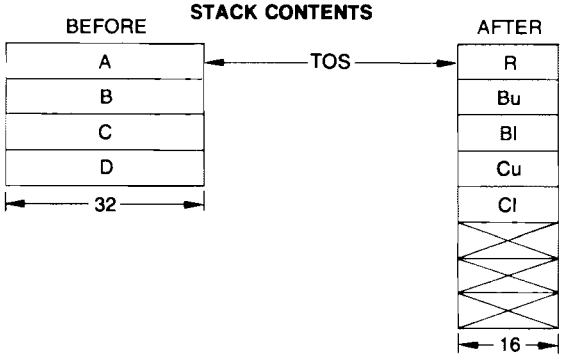
Hex Coding: 9F with sr = 1
1F with sr = 0

Execution Time: 90 to 214 clock cycles

Description:
32-bit floating-point operand A at the TOS is converted to a 16-bit fixed-point two's complement integer. The result R replaces the lower half of A and the stack is moved up by two bytes so that R occupies the TOS. Operands A and D are lost. Operands B and C are unchanged, but appear as upper (u) and lower (l) halves on the 16-bit wide stack if they are 32-bit operands.
If the integer portion of A is larger than 15 bits when converted, the overflow status will be set and A will not be changed. Operand D, however, will still be lost.

Status Affected: Sign, Zero, Overflow

STACK CONTENTS



FLTS

16-BIT FIXED-POINT TO 32-BIT FLOATING-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	0	1

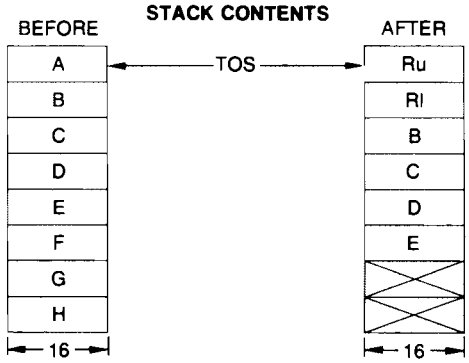
Hex Coding: 9D with sr = 1
1D with sr = 0

Execution Time: 62 to 156 clock cycles

Description:
16-bit fixed-point two's complement integer A at the TOS is converted to a 32-bit floating-point number. The lower half of the result R (Rl) replaces A, the upper half (Ru) replaces H and the stack is moved down so that Ru occupies the TOS. Operands A, F, G and H are lost. Operands B, C, D and E are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



FLTD

32-BIT FIXED-POINT TO 32-BIT FLOATING-POINT CONVERSION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	1	0	0

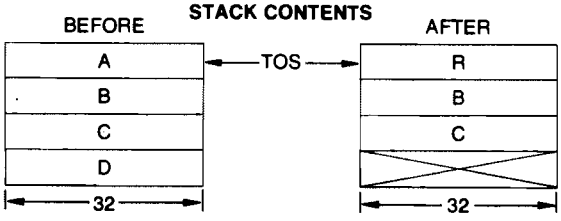
Hex Coding: 9C with sr = 1
1C with sr = 0

Execution Time: 56 to 342 clock cycles

Description:
32-bit fixed-point two's complement integer operand A at the TOS is converted to a 32-bit floating-point number. The result R replaces A at the TOS. Operands A and D are lost. Operands B and C are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS



FMUL

32-BIT FLOATING-POINT MULTIPLY

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	1	0

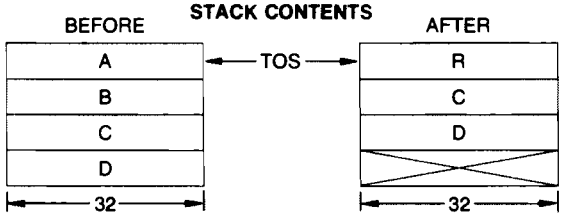
Hex Coding: 92 with sr = 1
12 with sr = 0

Execution Time: 146 to 168 clock cycles

Description:
32-bit floating-point operand A at the TOS is multiplied by the 32-bit floating-point operand B at the NOS. The normalized result R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.
Exponent overflow or underflow is reported in the status register, in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



FSUB

32-BIT FLOATING-POINT SUBTRACTION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	0	0	1

Hex Coding: 91 with sr = 1
11 with sr = 0

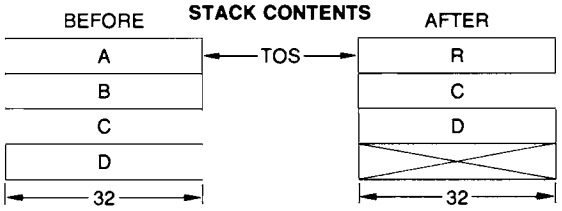
Execution Time: 70 to 370 clock cycles for A ≠ 0
26 clock cycles for A = 0

Description:
32-bit floating-point operand A at the TOS is subtracted from 32-bit floating-point operand B at the NOS. The normalized difference R replaces B and the stack is moved up so that R occupies the TOS. Operands A and B are lost. Operands C and D are unchanged.

Exponent alignment before the subtraction and normalization of the result account for the variation in execution time.

Exponent overflow or underflow is reported in the status register in which case the mantissa portion of the result is correct and the exponent portion is offset by 128.

Status Affected: Sign, Zero, Error Field (overflow)



LN

32-BIT FLOATING-POINT NATURAL LOGARITHM

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	0	1

Hex Coding: 89 with sr = 1
09 with sr = 0

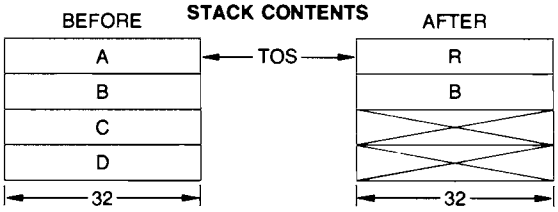
Execution Time: 4298 to 6956 clock cycles for A > 0
20 clock cycles for A ≤ 0

Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point natural logarithm (base e) of A. Operands A, C and D are lost. Operand B is unchanged.

The LN function accepts all positive input data values that can be represented by the data format. If LN of a non-positive number is attempted an error status of 0100 is returned.

Accuracy: LN exhibits a maximum absolute error of 2×10^{-7} for the input range from e^{-1} to e, and a maximum relative error of 2.0×10^{-7} for positive values less than e^{-1} or greater than e.

Status Affected: Sign, Zero, Error Field



LOG

32-BIT FLOATING-POINT COMMON LOGARITHM

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	0	0

Hex Coding: 88 with sr = 1
08 with sr = 0

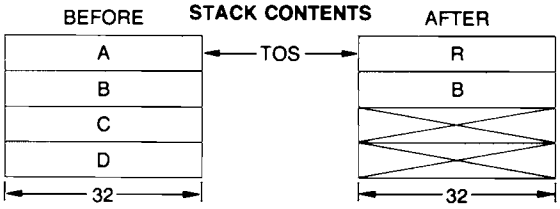
Execution Time: 4474 to 7132 clock cycles for A > 0
20 clock cycles for A ≤ 0

Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point common logarithm (base 10) of A. Operands A, C and D are lost. Operand B is unchanged.

The LOG function accepts any positive input data value that can be represented by the data format. If LOG of a non-positive value is attempted an error status of 0100 is returned.

Accuracy: LOG exhibits a maximum absolute error of 2.0×10^{-7} for the input range from 0.1 to 10, and a maximum relative error of 2.0×10^{-7} for positive values less than 0.1 or greater than 10.

Status Affected: Sign, Zero, Error Field



NOP

NO OPERATION

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	0	0

Hex Coding: 80 with sr = 1
00 with sr = 0

Execution Time: 4 clock cycles

Description:
The NOP command performs no internal data manipulations. It may be used to set or clear the service request interface line without changing the contents of the stack.

Status Affected: The status byte is cleared to all zeroes.

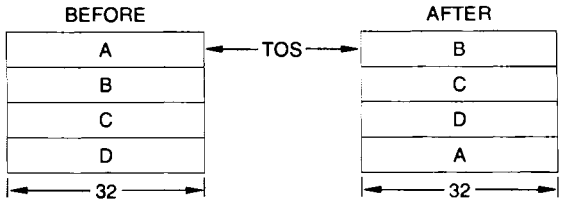
POPD

32-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	1	0	0	0

Hex Coding: B8 with sr = 1
38 with sr = 0
Execution Time: 12 clock cycles
Description:
The 32-bit stack is moved up so that the old NOS becomes the new TOS. The previous TOS rotates to the bottom of the stack. All operand values are unchanged. POPD and POPF execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



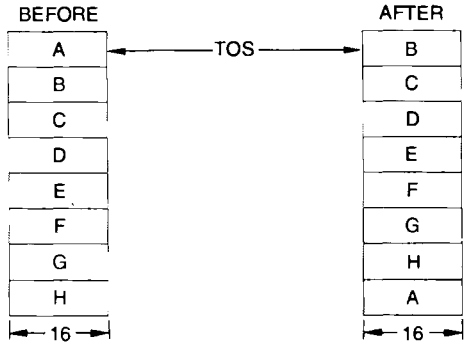
POPS

16-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	1	0	0	0

Hex Coding: F8 with sr = 1
78 with sr = 0
Execution Time: 10 clock cycles
Description:
The 16-bit stack is moved up so that the old NOS becomes the new TOS. The previous TOS rotates to the bottom of the stack. All operand values are unchanged.
Status Affected: Sign, Zero

STACK CONTENTS



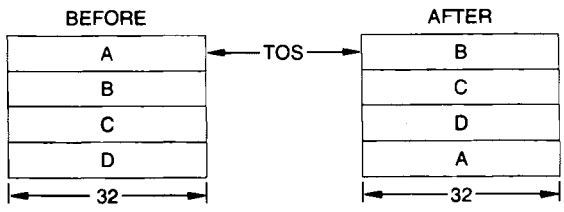
POPF

32-BIT
STACK POP

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	0	0

Hex Coding: 98 with sr = 1
18 with sr = 0
Execution Time: 12 clock cycles
Description:
The 32-bit stack is moved up so that the old NOS becomes the new TOS. The old TOS rotates to the bottom of the stack. All operand values are unchanged. POPF and POPD execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



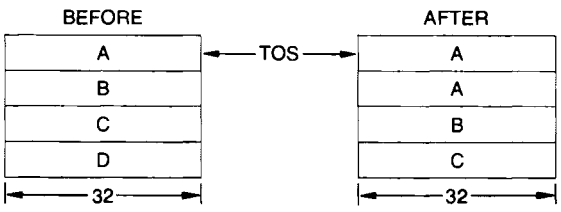
PTOD

PUSH 32-BIT
TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	0	1	1	1

Hex Coding: B7 with sr = 1
37 with sr = 0
Execution Time: 20 clock cycles
Description:
The 32-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand D is lost. All other operand values are unchanged. PTOD and PTOF execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



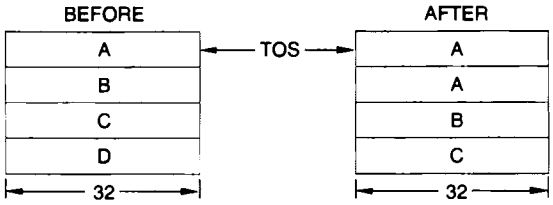
PTOF

PUSH 32-BIT TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	0	1	1	1

Hex Coding: 97 with sr = 1
17 with sr = 0
Execution Time: 20 clock cycles
Description:
The 32-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand D is lost. All other operand values are unchanged. PTOF and PTOD execute the same operation.
Status Affected: Sign, Zero

STACK CONTENTS



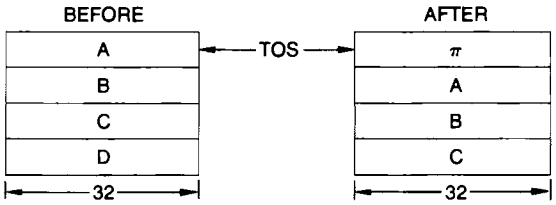
PUPI

PUSH 32-BIT FLOATING-POINT π

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	1	0

Hex Coding: 9A with sr = 1
1A with sr = 0
Execution Time: 16 clock cycles
Description:
The 32-bit stack is moved down so that the previous TOS occupies the new NOS location. 32-bit floating-point constant π is entered into the new TOS location. Operand D is lost. Operands A, B and C are unchanged.
Status Affected: Sign, Zero

STACK CONTENTS



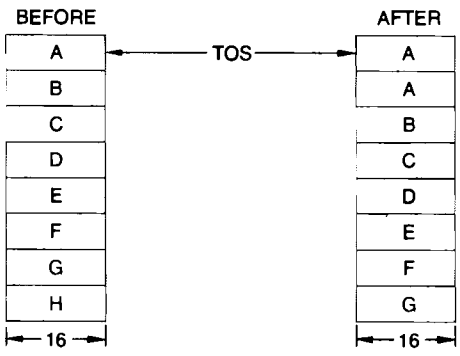
PTOS

PUSH 16-BIT TOS ONTO STACK

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	1	1

Hex Coding: F7 with sr = 1
77 with sr = 0
Execution Time: 16 clock cycles
Description:
The 16-bit stack is moved down and the previous TOS is copied into the new TOS location. Operand H is lost and all other operand values are unchanged.
Status Affected: Sign, Zero

STACK CONTENTS



PWR

32-BIT FLOATING-POINT X^Y

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	1	0	1	1

Hex Coding: 8B with sr = 1
0B with sr = 0

Execution Time: 8290 to 12032 clock cycles

Description:

32-bit floating-point operand B at the NOS is raised to the power specified by the 32-bit floating-point operand A at the TOS. The result R of B^A replaces B and the stack is moved up so that R occupies the TOS. Operands A, B, and D are lost. Operand C is unchanged.

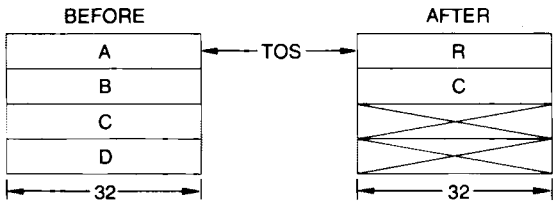
The PWR function accepts all input data values that can be represented in the data format for operand A and all positive values for operand B. If operand B is non-positive an error status of 0100 will be returned. The EXP and LN functions are used to implement PWR using the relationship $B^A = EXP\ [A(LN\ B)]$. Thus if the term $[A(LN\ B)]$ is outside the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$ an error status of 1100 will be returned. Underflow and overflow conditions can occur.

Accuracy: The error performance for PWR is a function of the LN and EXP performance as expressed by:
$|(\text{Relative Error})_{PWR}| \approx |(\text{Relative Error})_{EXP} + |A(\text{Absolute Error})_{LN}|$

The maximum relative error for PWR occurs when A is at its maximum value while $[A(LN\ B)]$ is near 1.0×2^5 and the EXP error is also at its maximum. For most practical applications the relative error for PWR will be less than 7.0×10^{-7}.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SADD

16-BIT FIXED-POINT ADD

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	0	0

Hex Coding: EC with sr = 1
6C with sr = 0

Execution Time: 16 to 18 clock cycles

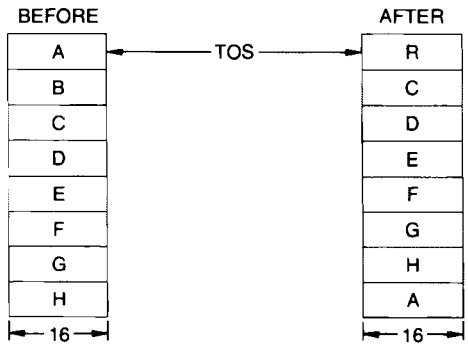
Description:

16-bit fixed-point two's complement integer operand A at the TOS is added to 16-bit fixed-point two's complement integer operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operand B is lost. All other operands are unchanged.

If the addition generates a carry bit it is reported in the status register. If an overflow occurs it is reported in the status register and the 16 least significant bits of the result are returned.

Status Affected: Sign, Zero, Carry, Error Field

STACK CONTENTS



SDIV

16-BIT FIXED-POINT DIVIDE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	1	1

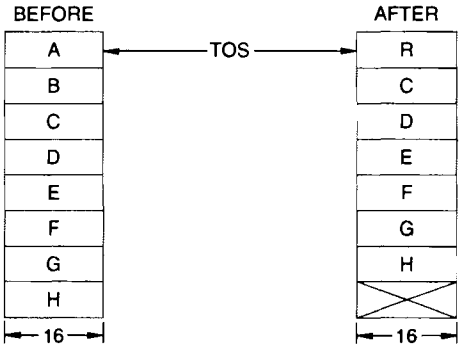
Hex Coding: EF with sr = 1
6F with sr = 0

Execution Time: 84 to 94 clock cycles for A ≠ 0
14 clock cycles for A = 0

Description:
16-bit fixed-point two's complement integer operand B at the NOS is divided by 16-bit fixed-point two's complement integer operand A at the TOS. The 16-bit integer quotient R replaces B and the stack is moved up so that R occupies the TOS. No remainder is generated. Operands A and B are lost. All other operands are unchanged.
If A is zero, R will be set equal to B and the divide-by-zero error status will be reported.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SIN

32-BIT FLOATING-POINT SINE

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	1	0

Hex Coding: 82 with sr = 1
02 with sr = 0

Execution Time: 3796 to 4808 clock cycles for $|A| > 2^{-12}$ radians
30 clock cycles for $|A| \leq 2^{-12}$ radians

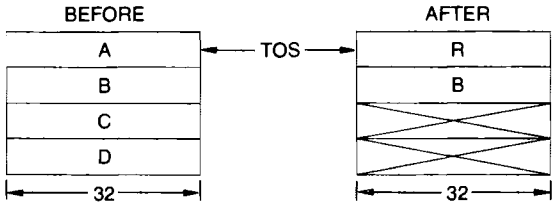
Description:
The 32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point sine of A. A is assumed to be in radians. Operands A, C and D are lost. Operand B is unchanged.

The SIN function will accept any input data value that can be represented by the data format. All input values are range reduced to fall within the interval $-\pi/2$ to $+\pi/2$ radians.

Accuracy: SIN exhibits a maximum relative error of 5.0×10^{-7} for input values in the range of -2π to $+2\pi$ radians.

Status Affected: Sign, Zero

STACK CONTENTS



SMUL

16-BIT FIXED-POINT MULTIPLY, LOWER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	1	0

Hex Coding: EE with sr = 1
6E with sr = 0

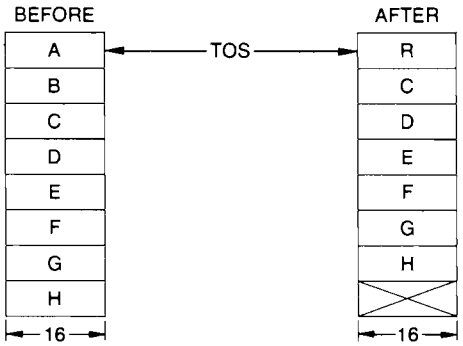
Execution Time: 84 to 94 clock cycles

Description:

16-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at the NOS. The 16-bit least significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The most significant half of the product is lost. Operands A and B are lost. All other operands are unchanged. The overflow status bit is set if the discarded upper half was non-zero. If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SMUU

16-BIT FIXED-POINT MULTIPLY, UPPER

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	0	1	1	0

Hex Coding: F6 with sr = 1
76 with sr = 0

Execution Time: 80 to 98 clock cycles

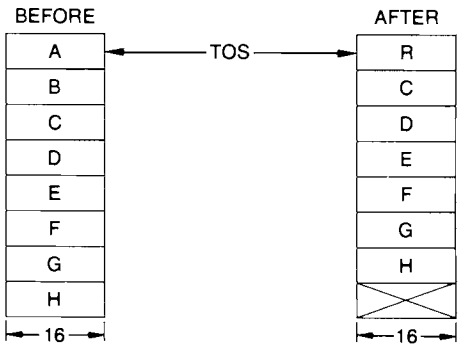
Description:

16-bit fixed-point two's complement integer operand A at the TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at the NOS. The 16-bit most significant half of the product R replaces B and the stack is moved up so that R occupies the TOS. The least significant half of the product is lost. Operands A and B are lost. All other operands are unchanged.

If either A or B is the most negative value that can be represented in the format, that value is returned as R and the overflow status is set.

Status Affected: Sign, Zero, Error Field

STACK CONTENTS



SQRT

32-BIT FLOATING-POINT SQUARE ROOT

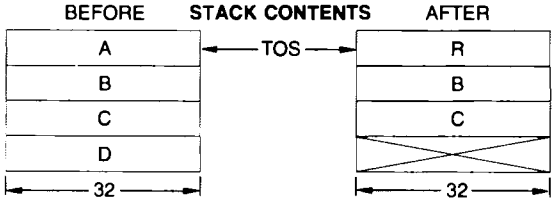
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	0	0	1

Hex Coding: 81 with sr = 1
01 with sr = 0

Execution Time: 782 to 870 clock cycles

Description:
32-bit floating-point operand A at the TOS is replaced by R, the 32-bit floating-point square root of A. Operands A and D are lost. Operands B and C are not changed.
SQRT will accept any non-negative input data value that can be represented by the data format. If A is negative an error code of 0100 will be returned in the status register.

Status Affected: Sign, Zero, Error Field



SSUB

16-BIT FIXED-POINT SUBTRACT

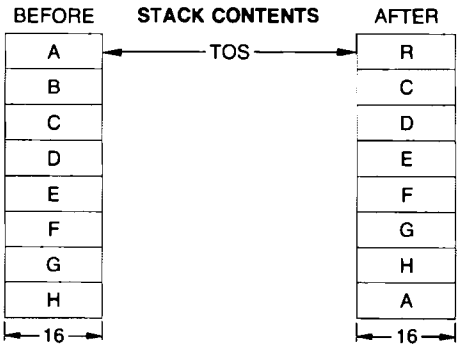
	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	0	1	1	0	1

Hex Coding: ED with sr = 1
6D with sr = 0

Execution Time: 30 to 32 clock cycles

Description:
16-bit fixed-point two's complement integer operand A at the TOS is subtracted from 16-bit fixed-point two's complement integer operand B at the NOS. The result R replaces B and the stack is moved up so that R occupies the TOS. Operand B is lost. All other operands are unchanged.
If the subtraction generates a borrow it is reported in the carry status bit. If A is the most negative value that can be represented in the format the overflow status is set. If the result cannot be represented in the format range, the overflow status is set and the 16 least significant bits of the result are returned as R.

Status Affected: Sign, Zero, Carry, Error Field



TAN

32-BIT FLOATING-POINT TANGENT

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	0	0	1	0	0

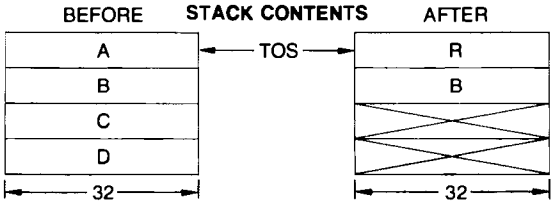
Hex Coding: 84 with sr = 1
04 with sr = 0

Execution Time: 4894 to 5886 clock cycles for $|A| > 2^{-12}$ radians
30 clock cycles for $|A| \leq 2^{-12}$ radians

Description:
The 32-bit floating-point operand A at the TOS is replaced by the 32-bit floating-point tangent of A. Operand A is assumed to be in radians. A, C and D are lost. B is unchanged.
The TAN function will accept any input data value that can be represented in the data format. All input data values are range-reduced to fall within $-\pi/4$ to $+\pi/4$ radians. TAN is unbounded for input values near odd multiples of $\pi/2$ and in such cases the overflow bit is set in the status register. For angles smaller than 2^{-12} radians, TAN returns A as the tangent of A.

Accuracy: TAN exhibits a maximum relative error of 5.0×10^{-7} for input data values in the range of -2π to $+2\pi$ radians except for data values near odd multiples of $\pi/2$.

Status Affected: Sign, Zero, Error Field (overflow)



XCHD

EXCHANGE 32-BIT STACK OPERANDS

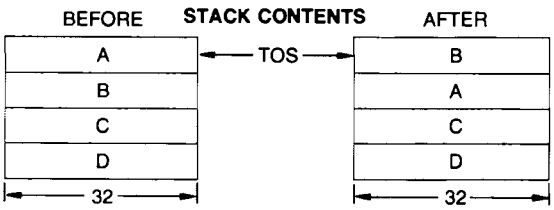
	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	1	1	1	0	0	1

Hex Coding: B9 with sr = 1
39 with sr = 0

Execution Time: 26 clock cycles

Description:
32-bit operand A at the TOS and 32-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

Status Affected: Sign, Zero



XCHF

EXCHANGE 32-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	0	0	1	1	0	0	1

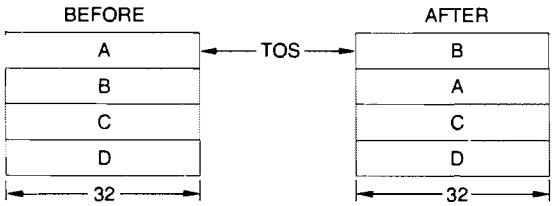
Hex Coding: 99 with sr = 1
19 with sr = 0

Execution Time: 26 clock cycles

Description:
32-bit operand A at the TOS and 32-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

Status Affected: Sign, Zero

STACK CONTENTS



XCHS

EXCHANGE 16-BIT STACK OPERANDS

	7	6	5	4	3	2	1	0
Binary Coding:	sr	1	1	1	1	0	0	1

Hex Coding: F9 with sr = 1
79 with sr = 0

Execution Time: 18 clock cycles

Description:
16-bit operand A at the TOS and 16-bit operand B at the NOS are exchanged. After execution, B is at the TOS and A is at the NOS. All operand values are unchanged.

Status Affected: Sign, Zero

STACK CONTENTS