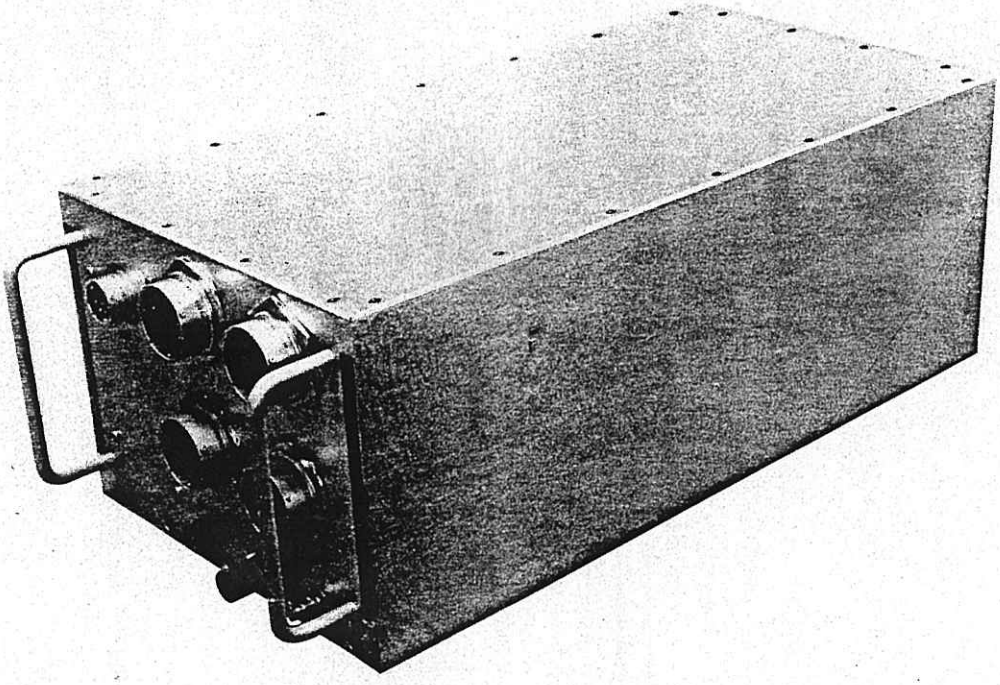
Space Shuttle Advanced System/4 Pi Model AP-101 Central Processor Unit

Technical Description



IBM

Space Shuttle Advanced System/4 Pi
Model AP-101 Central Processor Unit

Technical Description

Prepared for

Rockwell International Corporation
Space Division
12214 Lakewood Blvd
Downey, CA 90241

31 March 1975

Under

Space Shuttle GPC
M4J7XMA-483019

IBM File No. 75-A97-001

Federal Systems Division, Owego, New York 13827

TABLE OF CONTENTS

Section 1

SUMMARY OF CHARACTERISTICS

1.1 SUMMARY OF CHARACTERISTICS

The Space Shuttle Central Processor Unit is a General-Purpose Digital Computer

The Central Processor Unit (CPU) provides real-time data processing.

The CPU is microprogram-controlled and provides 32-bit, parallel data flow, floating point arithmetic, and 40,960 36-bit words of core main storage in a single LRU (line replaceable unit). The processing capability is 480,000 operations per second based on a typical distribution of instructions. The instruction repertoire includes short- and extended-precision floating point, conversion, input/output, fixed point, shifting, and logic operations.

The modular memory consists of pluggable modules containing 8192 18-bit halfwords. The failure of any module is detectable by self-test and built-in test hardware.

During periods of time when the CPU is operating below its processing capacity, power-switching places the memory in a low-power, quiescent mode when the memory is not addressed.

Support software and hardware features, such as halfword storage protect and interrupt masking, provide aids for debugging of software.

CPU CHARACTERISTICS

Machine Organization	
Type	General-purpose, stored-program, parallel, general register
Organization	Fixed-point, binary, fractional Floating-point, hexadecimal fraction
Data Flow	32-bit, parallel
Control Structure	Microprogrammed
Instruction Word Lengths	16-bit and 32-bit
Fixed-Point Data Word Lengths	16-bit and 32-bit
Floating-Point Data Word Lengths	32-bit and 64-bit
General Registers (3 sets)	Two selectable sets of eight 32-bit hardware registers for fixed-point operations; one set of eight 32-bit hardware registers for floating-point operations
Number of Instructions	154 total
Addressing Modes	Base plus displacement, indexed, indirect, relative, extended, immediate and indirect address pointer
Addressing Capability	Halfword resolution, 524,288 capacity
Interrupt Structure	35 interrupts, organized into 19 priority levels and five classes

Average Instruction Execution Times (μs)	Register-to Register	Storage-to Register
Fixed-Point:		
Add	1.2	1.8
Multiply	5.2	5.6
Divide	8.8	9.4
Floating-Point:		
Add (Short Operand)	2.8	3.0
Multiply	6.4	6.6
Divide	9.8	10.0

Main Storage	
Type	Random access, destructive readout (DRO) ferrite magnetic core, nonvolatile
Organization:	2-1/2 D
Capacity:	40,096 36-bit words, (32 data bits, 2 parity bits, 2 store protect bits). Up to 262,144 words with external main storage units
Modularity:	8,192 36-bit words (2 pluggable modules)
Cycle Time:	800-ns read/write; 1.2 μs min. read/modify/write
Access Time:	375 ns
Addressable Unit:	16-bit halfword
Features:	Power transient protection, power switching, temperature compensation, halfword parity, halfword storage protect, interchangeable modules

Input/Output	
I/O Channel	
Type:	Microprogram-controlled, multimode, fullword, half-duplex
Word Length:	32 bits + parity (2) + store protect (2)
Data Rates:	770,000 full words/s max input (burst mode) and 625,000 full words/s max input (normal mode)
Operating Modes:	Program-controlled direct I/O, externally controlled DMA I/O
External Interrupts:	5 levels
Discretes:	4 outputs
Real-Time Clocks:	Two 4295-second, with 1-μs resolution

Power System	
Primary Power:	28 VDC
Regulated DC Output:	+5, -5, +12 VDC
Features:	Overvoltage, undervoltage, and overcurrent protection: power transient protection: power sequencing

Physical Characteristics	
Weight:	57.9 lb (40K 36-bit words of main storage)
Size:	Height 7.62, width 10.12, depth 19.56 inches (full ATR)
Volume:	0.87 ft^3
Power:	350 W (50% memory duty cycle, 40K main storage, 50% ones, 2 pages operate, 2 pages standby, 6 pages quiescent) 2K x 48-bit PROM and 72% PS efficiency
Construction:	Plug-in modules
Cooling:	Indirect air conductive, optional blower available
Environment:	Designed to meet MC 615-0001 requirements

Section 2

ARCHITECTURE

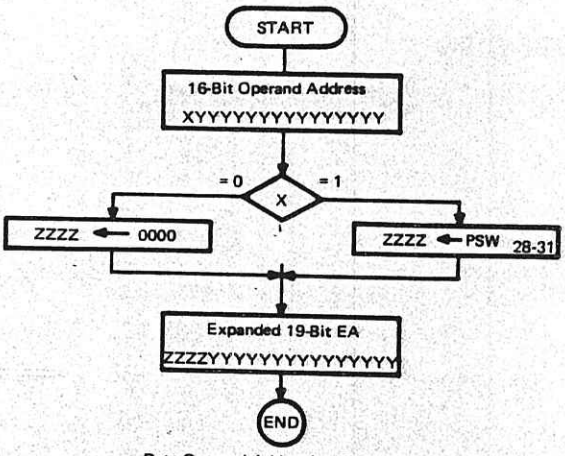
2.1 ADDRESSING

Sixteen-Bit Address with Expansion Capability

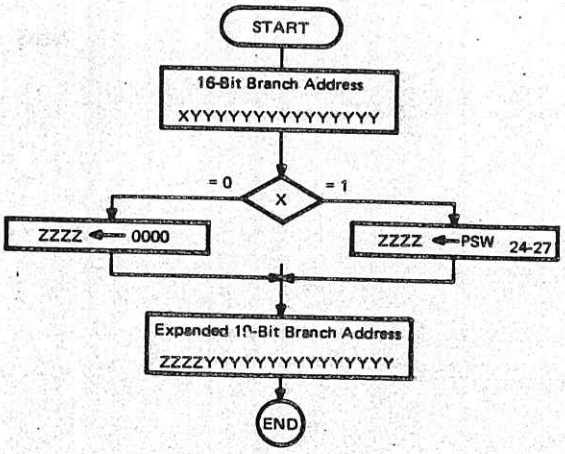
Address field provides 64K halfword addresses with expansion to 512K halfwords.

The 16-bit address provides 64K halfword addresses. The addressing range is extended to 512K halfword locations by using PSW bits.

Address expansion to 19 bits is achieved by replacing the high-order bit of a 16-bit address with 4 bits, as shown below. Data operand addresses are extended to 19 bits by specifying either a 4-bit data sector register or an implied data sector register. When the high-order bit of a 16-bit data address is 1, a 4-bit data sector register (PSW bits 28 through 31) is selected to replace the high-order bit. When the high-order bit of a 16-bit data address is a 0, an implied data sector register containing 0000 is selected. Note that indirect addressing locates the indirect address pointer as if the pointer were a data operand. Branch addresses are extended to 19 bits in an equivalent manner. When the high-order bit of a 16-bit branch address is a 1, a 4-bit branch sector register (PSW bits 24 through 27) is selected to replace the high-order bit. When the high-order bit is a 0, an implied branch sector register containing 0000 is selected.



Data Operand Addressing Expansion



Branch Addressing Expansion

2.2 DATA FORMATS

Flexible Formats Improve Storage Efficiency and Performance

Fixed-point data forms are halfword (16 bits) or fullword (32 bits), while floating-point formats are 32 bits and 64 bits, providing the equivalent of 6.2 and 15.5 decimal digits of precision, respectively.

Fixed-point data representation is two's-complement-fractional with the binary point immediately to the right of the sign bit. A halfword is 15 bits plus sign; a fullword is 31 bits plus sign. In fractional data representation, bit position 1 represents the fraction 2^{-1}, bit position 2 represents 2^{-2}, and bit position 31 represents 2^{-31}. Fixed-point operands may be located on any halfword boundary.

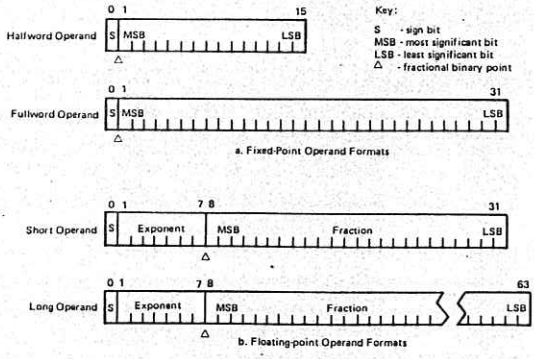
The floating-point data formats are 32 bits and 64 bits long, differing only in the size of the fraction. The 32-bit length provides sign plus 24 bits of fraction (6 hexadecimal digits), equivalent to 6.2 decimal digits. The 64-bit length provides sign plus 56 bits of fraction (14 hexadecimal digits), equivalent to 15.5 decimal digits. In both lengths, bit position 0 is the sign of the fraction. Bit positions 1 through 7 of both formats constitute the exponent, which represents a hexadecimal (16) power and is expressed in excess 64 notation.

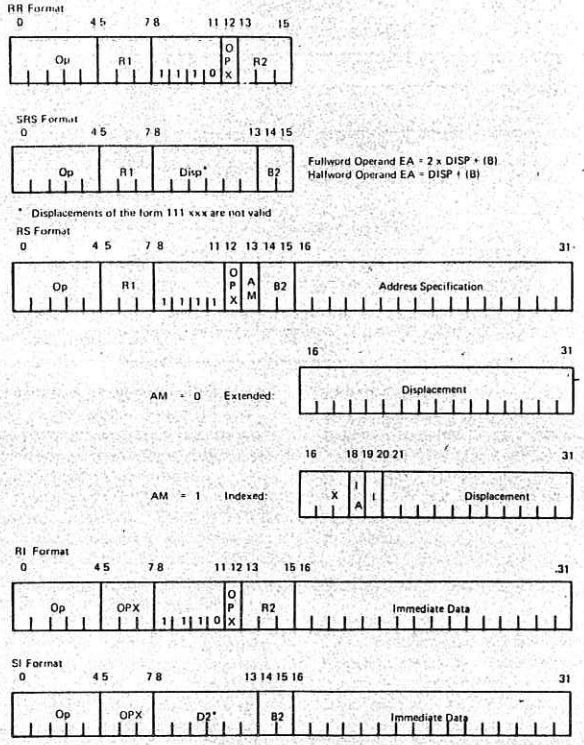
The value of a floating-point number is given by the product of the fraction times the power of 16 exponent. Since the exponent represents a hexadecimal base, the floating-point fractions are hexadecimally normalized; i.e., a normalized fraction has a nonzero hexadecimal digit in the first four bits of the fraction. Maximum precision is preserved in addition, subtraction, multiplication, and division, because all results are normalized.

The exponent field represents numbers from 0 through 127. To accommodate positive and negative exponents, the true exponent is determined by subtracting 64 from the value shown in the exponent field. This use of excess 64 notation permits an exponent range of -64 through +63 and a range of magnitude for normalized floating-point data of

- In short precision
  $16^{-65} \leq M \leq (1-16^{-6}) \times 16^{63}$
- In long precision
  $16^{-65} \leq M \leq (1 - 16^{-10}) \times 16^{63}$
- Thus, giving a range of approximately $5.4 \times 10^{-79} \leq M \leq 7.2 \times 10^{75}$

The hexadecimal notation provides improved performance through shifting pre- and post-normalization operations four bits at a time.





Instruction Field Definition:

- Op – 5-bit field defines an operation, or the class of an operation, to be performed by the CPU
- OPX – an extension of Op field
- R1 – 3-bit field designates the register containing the first operand; except for operations which alter main storage, the result usually replaces the first operand
- R2 – 3-bit field appears in the RR or RI format; used to specify a general register containing the second operand.
- B2 – 2-bit field specifies the register containing the base address
- X – 3-bit field specifies one of seven general registers that contains the index value; the number of the general register specified is the same as the value in the X-field except, when X = 000, an index value of zero is used
- IA – indirect addressing bit, when a logical "1," specifies indirect addressing; a logical "0" specifies direct addressing
- I – format bit, in conjunction with X and IA, specifies various addressing modes

Instruction Formats

2.3 INSTRUCTION FORMATS

Five Formats Provide Storage Efficiency and Speed

Two 16-bit formats permit storage effiency while three 32-bit formats enhance execution speed and provide extensive addressing flexibility.

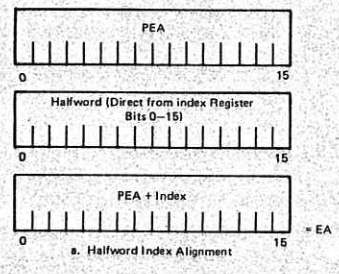
Note: The instruction formats discussed in this section are illustrated on fold-out sheets (see page 2-8) for ease of reference during the following discussions.

The short Register-to-Register (RR) and Short Register-to-Storage (SRS) formats minimize the amount of instruction storage required for a given problem. In addition, two short instructions can be read from storage with one storage access. RR format instructions execute operations when both operands are in general registers; this format has the shortest execution times since no storage access is required for operands. The SRS format executes operations when one operand is in a general register and the other is in storage. The SRS format is limited to addressing storage locations that are 56 or less fullword or halfword locations from the address in a base register. The assembler automatically uses the halfword length SRS format unless the fullword format is specified by the programmer.

Indexed Addressing

Indexed addressing is specified by RS format bit 13 equal to 1. This addressing mode contains three additional instruction fields which contribute to address generation as follows:

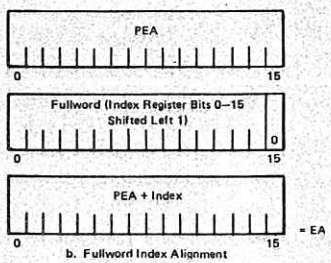
X This 3-bit field specifies one of seven general registers containing the index. Indexing is not performed when X is equal to 000. An index is contained in the upper halfword of a general register. The index is automatically aligned with the preliminary effective address (PEA).
General register contents can be used interchangeably as either a base or an index or both. When indirect addressing is specified, indexing follows indirect addressing.



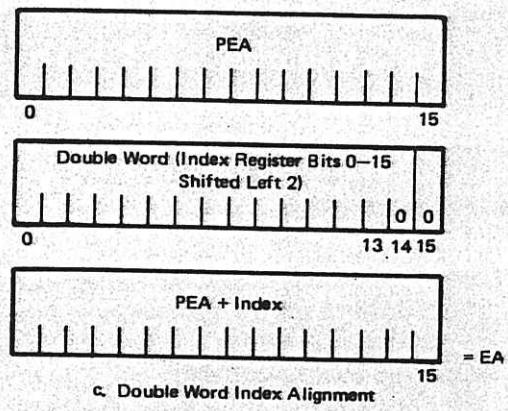
a. Halfword Index Alignment

The fullword Register-to-Storage (RS) format has two modes of operation — extended addressing (RS bit 13 = 0) and indexed addressing (RS bit 13 = 1). Extended addressing provides a full 16-bit address displacement, either used as the absolute operand address or added to the address in a base register. Indexed addressing provides an 11-bit address displacement.

The Register Immediate (RI) and Storage Immediate (SI) instruction formats execute operations where the first operand is contained in the instruction. The second operand is in a general register for the RI format and in main storage for the SI format. An operand fetch operation is eliminated by using immediate instructions.



b. Fullword Index Alignment



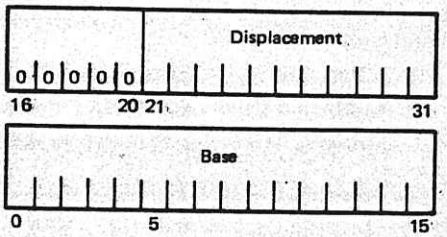
c. Double Word Index Alignment

IA This format bit, when a one, specifies indirect addressing. Indirect addressing is not performed when this bit is zero.

I This format bit, in conjunction with X and IA, specifies various address modes which are explained below.

The development of the EA for the indexed mode of operand addressing is explained in detail in the subsequent steps:

1) Indexed addressing is specified by RS format bit 13 (AM) equal to 1. This addressing mode provides an 11-bit displacement. The base and displacement are aligned when indexed addressing is performed.



Displacement Alignment for Indexed Addressing

The displacement is aligned so that bit 31 corresponds to base or index bit 15 and displacement bit 21 corresponds to base or index bit 5. The displacement is expanded to 16 bits by appending five leading zeros.

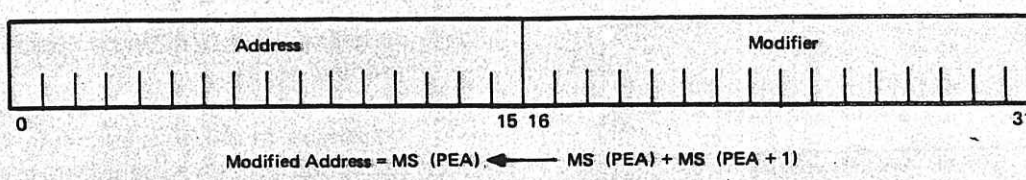
2) If B is not equal to 11, the 16-bit base, contained in the high order half of the specified register, is added to the aligned displacement. This results in a preliminary effective address (PEA) whereby the PEA = (B) + Displacement.
If B = 11, the aligned displacement is added to zero. This result is the preliminary effective address (PEA), whereby the PEA = Displacement.

3) If the X field is all zeros, IA (bit 19) is a zero and I (bit 20) is a zero, then the 16-bit result of Step 2 is added to the contents of the instruction counter (IC) to form the 16-bit EA whereby EA = IC + PEA. (This EA is then expanded to a 19-bit EA.)

4) If the X field is all zeros, IA (bit 19) is a zero and I (bit 20) is a one, the 16-bit result of Step 2 is subtracted from the contents of the IC to form the 16-bit EA whereby EA = IC - PEA. (This EA is then expanded to a 19-bit EA.)

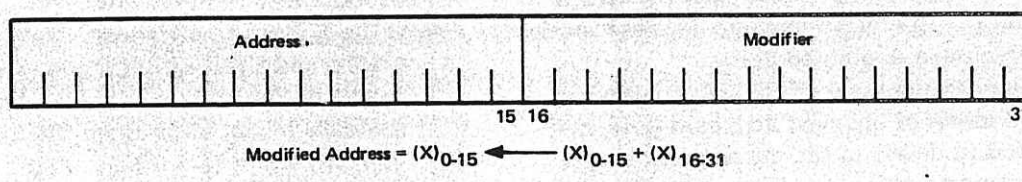
5) If the X field is all zeros, IA (bit 19) is a one and I (bit 20) is a zero, then Indirect Addressing is performed. The 16-bit result of Step 2 is expanded to a 19-bit address and is used as the address of a main-storage halfword. This halfword is then fetched and expanded to 19-bits by using expanded addressing to form the EA. EA = MS (PEA). Functional equivalency to preindexing capability can be obtained through modification of the base.

6) If the X field is all zeros, IA (bit 19) is a one and I (bit 20) is a one, Indirect Addressing is performed as described in Step 5. Then, storage modification is automatically performed. The indirect address is contained in a full word and must have an even address. A modifier is contained in bits 16 through 31. An address is contained in bits 0 through 15. The modifier is added to the address and the resulting modified address replaces bits 0 through 15.



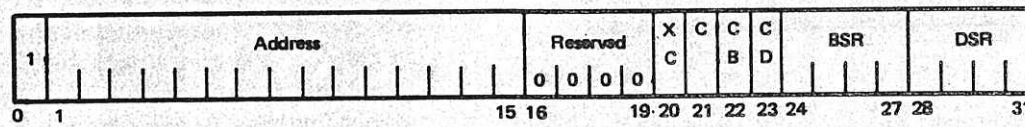
The Contents of Storage Modification Word

7) If the X field is not all zeros, IA (bit 19) is a zero and I (bit 20) is a zero, the most significant 16-bits of the general registers specified by the X field are aligned, and then added to the 16-bit result of Step 2 to form the 16-bit EA. (This EA is then expanded to a 19-bit EA.)
8) If the X field is not all zeros, IA (bit 19) is a zero and I (bit 20) is a one, the most significant 16 bits of the general register specified by the X field are aligned, and then added to the 16-bit result of Step 2 to form the 16-bit EA. (This EA is then expanded to a 19-bit EA.)
The figure below illustrates the address and modifier format in the index register. The modifier is added to the address and the resulting modified address replaces bits 0 through 15 of index register after the EA is determined.



The Contents of Index Register X

9) If the X field is not all zeros, IA (bit 19) is a one and I (bit 20) is a zero, Indirect Addressing (IA) with post-indexing is performed. The 16-bit result of Step 2 is expanded to a 19-bit address and is used to fetch a main-storage halfword. The index contained in the general register by X is aligned and then added to the fetched halfword to form the 16-bit EA. This EA is then expanded to a 19-bit EA by using expanded addressing. Functional equivalency to preindexing capability can be obtained through modification of the base.
10) If the X field is not all zeros, IA (bit 19) is a one and I (bit 20) is a one, a direct addressing mode is defined using a 32-bit fullword indirect pointer as follows:
    a) First, the PEA from Step 2 must locate a fullword indirect address pointer, with the format as follows:



Field	Function
XC	Index Control
C	Control
CB	Control BSR Usage
CD	Control DSR Usage

Fullword Indirect Address Pointer

b) If C (bit 21) equals 0, XC (bit 20) equals 1, and the instruction is not a branch type instruction, the 19-bit EA equals the 4-bit DSR with the 15-bit address field appended. When C (bit 21) equals 0, XC (bit 20) equals 0, and the instruction is not a branch type instruction, the 19-bit EA equals the 4-bit DSR with the 15-bit address field appended, with the result added to the index value in indexing register X. The current PSW's DSR is not changed. If C (bit 21) equals 0 and the instruction is a branch type instruction, the current PSW's BSR in conjunction with IA pointer bits 0 through 15 will be used to form the BA. If XC = 0, postindexing will occur.

c) If C (bit 21) equals 1 and the instruction is a branch type instruction, the BSR and DSR fields may selectively replace the corresponding fields in the current PSW, based on the CB and CD bit values as follows:

CB	CD	Result
0	0	Use current PSW's BSR to form the BA.
0	1	Replace the current PSW's DSR with this DSR. Form the BA normally.
1	0	Replace the current PSW's BSR with this BSR before forming the BA.
1	1	First, replace the current PSW's DSR with this DSR. Then, replace the current PSW's BSR with this BSR before forming the BA.

d) When C (bit 21) equals 1 and XC (bit 20) equals 1, postindexing is not performed. When C (bit 21) equals 1 and XC (bit 20) equals 0, the BA calculation includes a final addition of the index value in index registers X. If C (bit 21) equals 1 and the instruction is not a branch, the 19-bit EA equals the current PSW's DSR and the 15-bit field appended. If XC = 0, postindexing will occur.

The results of indexed mode RS operations normally replace the first operand except for store operation where the first operand replaces the second operand. The second operand is unaltered for nonstore operations, and the first operand is unaltered for store operation.

Effective address generations for the various instruction formats are summarized in the EA summary chart.

EFFECTIVE ADDRESS GENERATION SUMMARY CHART

	SRS, SI Formats	RS Format				
		Extended Addressing (AM=0)	Indexed Addressing (AM=1)			
			IA	I	X=000	X=$\overline{000}$
					PEA=(B)+DISP	
B≠11	EA=(B)+DISP	EA=(B)+DISP	0	0	EA=IC+PEA	EA=$(X)_{0-15}$+PEA
			0	1	EA=IC-PEA	EA=$(X)^{*}_{0-15}$+PEA
			1	0	EA=MS(PEA)	EA=MS(PEA)+$(X)_{0-15}$
					EA=MS(PEA)**	EA=MS(PEA)***+$(X)_{0-15}$
					PEA=DISP	
B=11	EA=(B)+DISP	EA=DISP	0	0	EA=IC+PEA	EA=$(X)_{0-15}$+PEA
			0	1	EA=IC-PEA	EA=$(X)^{*}_{0-15}$+PEA
			1	0	EA=MS(PEA)	EA=MS(PEA)+$(X)_{0-15}$
			1	1	EA=MS(PEA)**	EA=MS(PEA)***+$(X)_{0-15}$

Definitions

EA	Effective address, main storage address of second operand
PEA	Preliminary effective address
(RN)	Contents of bits 0-15 of general register specified by B or X
RN	General register "N", where N = 0 to 7
(B)	Contents of bits 0-15 of general register specified by register B
B	B field of SRS, SI, or RS format instruction
MS( )	Contents of the main storage location specified by the contents of the parenthesis
DISP	Displacement field of instruction
X	X field of RS format instruction with indexed mode of addressing
$(X)_{0-15}$	Most significant halfword (bits 0-15) of the content of index register X automatically aligned.
AM	AM (addressing mode) field of RS format instruction
IA	IA (indirect address) field of RS format instruction with the indexed mode of addressing
I	I bit of RS format instruction with indexed mode of addressing
IC	Instruction counter
*	Automatic Index Modification
**	Automatic Storage Modification
***	Direct Storage Addressing with/without Post Indexing

X	INDEX VALUE	X	INDEX VALUE
000	Zero	100	(R4)
001	(R1)	101	(R5)
010	(R2)	110	(R6)
011	(R3)	111	(R7)

GENERAL REGISTER FUNCTION ASSIGNMENTS

General Register Number	Register Function		
	Operand	Base	Index
0	000	000	None
1	001	001	001
2	010	010	010
3	011	011 or None	011
4	100		100
5	101		101
6	110		110
7	111		111

2.4 INSTRUCTION REPERTOIRE

Comprehensive Instruction Repertoire Enhances Programmability and Performance

The instruction repertoire consists of 154 operations in seven classes. A total of 480,000 fixed-point operations/second can be achieved using a typical instruction mix.

The instruction set was selected to enhance programmability and offers a balanced and comprehensive instruction repertoire. Since the CPU uses a microprogram control structure, the machine personality need not be permanent; that is, the instruction set can be modified or changed as necessary.

The instruction set along with typical execution times in microseconds is as follows:

Fixed-Point Operations	Format	Execution Time (μs)				
		Even	Odd NOK	Odd / NOK	Even 100	Even 200 EMU
Add	RR	1.2	0.8	1.2	1.3	1.4
Add	SRS	1.8	1.4	1.8	2.0	2.2
Add	RS	1.8	2.0	2.8	2.1	2.3
Add Halfword	SRS	1.8	1.4	1.8	2.0	2.2
Add Halfword	RS	1.8	2.0	2.8	2.1	2.3
Add Halfword Immediate	RI	1.8	2.0	2.8	1.9	2.0
Add to Storage	RS	2.3	2.4	3.2	2.6	2.9
Compare	RR	1.6	1.2	1.6	1.7	1.8
Compare	SRS	2.2	1.8	2.2	2.4	2.6
Compare	RS	2.2	2.4	3.2	2.4	2.6
Compare Between Limits (Note 1)	RR	5.4	5.0	5.4	5.7	5.8
Compare Halfword	SRS	2.2	1.8	2.2	2.4	2.6
Compare Halfword	RS	2.2	2.4	3.2	2.4	2.6
Compare Halfword Immediate	RI	2.2	2.4	3.2	2.3	2.4
Compare Immediate with Storage	SI	2.6	2.8	3.6	2.8	3.0
Divide	RR	8.8	8.4	8.8	8.9	9.0
Divide	SRS	9.4	9.0	9.4	9.6	10.0
Divide	RS	9.4	9.6	10.4	9.6	10.0
Exchange Upper and Lower Halfwords	RR	2.8	2.4	2.8	2.9	3.0
Insert Address Low	SRS	2.2	1.8	2.2	2.4	2.6
Insert Address Low	RS	2.0	2.2	3.0	2.2	2.4
Insert Halfword Low	RS	2.8	3.0	3.8	3.0	3.2
Load	RR	1.2	0.8	1.2	1.3	1.4
Load	SRS	1.8	1.4	1.8	2.0	2.2
Load	RS	1.8	2.0	2.8	2.1	2.3
Load Address	SRS	1.6	1.2	1.6	1.8	2.0
Load Address	RS	1.6	1.8	2.6	1.8	2.0
Load Arithmetic Complement	RR	1.4	1.0	1.4	1.5	1.6
Load Fixed Immediate	RR	2.8	2.4	2.8	2.9	3.0
Load Halfword	SRS	1.8	1.4	1.8	2.0	2.2
Load Halfword	RS	1.8	2.0	2.8	2.0	2.2
Load Multiple	RS	10.6	10.8	11.6	11.6	12.6
Modify Storage Halfword	SI	2.5	2.6	3.4	2.7	2.9
Multiply	RR	5.2	4.8	5.2	5.3	5.4
Multiply	SRS	5.6	5.2	5.6	5.8	6.0
Multiply	RS	5.6	5.8	6.6	5.8	6.0
Multiply Halfword	SRS	5.0	4.6	5.0	5.2	5.4
Multiply Halfword	RS	5.0	5.2	6.0	5.2	5.4
Multiply Halfword Immediate	RI	6.2	6.4	7.2	6.3	6.4
Multiply Integer Halfword	RS	6.0	6.2	7.0	6.2	6.4
Store	SRS	2.0	1.6	2.0	2.2	2.4
Store	RS	2.0	2.2	3.0	2.3	2.6
Store Halfword	SRS	2.8	2.4	2.8	3.0	3.2
Store Halfword	RS	2.8	3.0	3.8	3.1	3.4
Store Multiple	RS	11.8	12.0	12.8	12.7	13.6
Subtract	RR	1.2	0.8	1.2	1.3	1.4
Subtract	SRS	1.8	1.4	1.8	2.0	2.2
Subtract	RS	1.8	2.0	2.8	2.0	2.2

Fixed Point Operations	Format	Execution Time (μs)				
		Even	Odd NOK	Odd NOK	Even 100	Even 200 EMU
Subtract from Storage	RS	2.3	2.4	3.2	2.6	2.9
Subtract Halfword	SRS	1.8	1.4	1.8	2.0	2.2
Subtract Halfword	RS	1.8	2.0	2.8	2.0	2.2
Tally Down	SRS	3.2	2.8	3.2	3.5	3.8
Tally Down	RS	3.2	3.4	4.2	3.5	3.8
Branch						
Branch and Link	RR	1.8	1.4	1.8	1.9	2.0
Branch and Link	RS	1.6	1.8	2.6	1.7	1.8
Branch on Condition	RR	1.8	1.4	1.8	1.9	2.0
Branch on Condition	RS	1.8	2.0	2.8	1.9	2.0
Branch on Condition Backwards	SRS	2.0	1.6	2.0	2.1	2.3
Branch on Condition (Extended)	RR	2.8	2.4	2.8	2.9	3.0
Branch on Condition Forward	SRS	2.0	1.6	2.0	2.1	2.2
Branch on Count	RR	2.2	1.8	2.2	2.3	2.4
Branch on Count	RS	2.2	2.4	3.2	2.3	2.4
Branch on Count Backward	SRS	2.4	2.0	2.4	2.5	2.6
Branch on Overflow and Carry	RR	2.0	1.6	2.0	2.1	2.3
Branch on Overflow and Carry	RS	2.0	2.2	3.0	2.1	2.2
Branch on Overflow and Carry Forward	SRS	2.2	1.8	2.2	2.3	2.4
Shift (based on 3 shifts)						
Normalize and Count	RR	3.6	3.2	3.6	3.7	3.8
Shift Left Logical	SRS	2.4	2.0	2.4	2.5	2.6
Shift Left Double, Logical	SRS	2.8	2.4	2.8	2.9	3.0
Shift Right Arithmetic	SRS	2.0	1.6	2.0	2.1	2.2
Shift Right Double Arithmetic	SRS	2.4	2.0	2.4	2.5	2.6
Shift Right Logical	SRS	2.0	1.6	2.0	2.1	2.2
Shift Right Double Logical	SRS	2.4	2.0	2.4	2.5	2.6
Shift Right and Rotate	SRS	2.2	1.8	2.2	2.3	2.5
Shift Right Double and Rotate	SRS	2.4	2.0	2.4	2.5	2.6
Logical Operations						
AND	RR	1.2	0.8	1.2	1.3	1.4
AND	SRS	1.8	1.4	1.8	1.9	2.1
AND	RS	1.8	2.0	2.8	2.0	2.2
AND Halfword Immediate	RI	1.8	2.0	2.8	1.9	2.0
AND Immediate with Storage	SI	2.5	2.6	3.4	2.7	2.0
AND to Storage	RS	2.3	2.4	3.2	2.6	2.0
Exclusive-OR	RR	1.2	0.8	1.2	1.3	1.4
Exclusive-OR	SRS	1.8	1.4	1.8	2.0	2.2
Exclusive-OR	RS	1.8	2.0	2.8	2.0	2.2
Exclusive-OR Halfword Immediate	RI	1.8	2.0	2.8	1.9	2.0
Exclusive-OR Immediate with Storage	SI	2.5	2.6	3.4	2.7	2.9
Exclusive-OR to Storage	RS	2.3	2.4	3.2	2.6	2.9

Logical Operations	Format	Execution Time (μs) Even	Odd NOK	Odd NOK	Even 100	Even 200 EMU
OR	RR	1.2	0.8	1.2	1.3	1.4
OR	SRS	1.8	1.4	1.8	2.0	2.2
OR	RS	1.8	2.0	2.8	2.0	2.2
OR Halfword Immediate	RI	1.8	2.0	2.8	1.9	2.0
OR to Storage	RS	2.3	2.4	3.2	2.6	2.9
Search Under Mask (Note 2)	RR	6.4	6.0	6.4	6.6	6.8
Set Bits	SI	2.5	2.6	3.4	2.7	2.8
Set Halfword	SRS	3.2	2.8	3.2	3.5	3.8
Set Halfword	RS	3.2	3.4	4.2	3.5	3.8
Test Bits	SI	2.8	3.0	3.8	3.0	3.2
Test Register Bits	RI	2.2	2.4	3.2	2.3	2.4
Test Halfword	SRS	2.6	2.2	2.6	2.8	3.0
Test Halfword	RS	2.6	2.8	3.6	2.8	3.0
Zero Bits	SI	2.7	2.8	3.6	2.9	3.1
Zero Register Bits	RI	1.8	2.0	2.8	1.9	2.0
Zero Halfword	SRS	3.2	2.8	3.2	3.5	3.8
Zero Halfword	RS	3.2	3.4	4.2	3.6	4.0
Floating-Point Operations (Note 3)						
Add (Long Operand)	RR	5.6	5.2	5.6	5.7	5.8
Add (Long Operand)	RS	7.0	7.2	8.0	7.3	7.6
Add (Short Operand)	RR	2.8	2.4	2.8	2.9	3.0
Add (Short Operand)	SRS	3.0	2.6	3.0	3.2	3.4
Add (Short Operand)	RS	3.0	3.2	4.0	3.2	3.4
Compare (Long Operand)	RR	7.6	7.2	7.6	7.7	7.8
Compare (Long Operand)	RS	8.4	8.6	9.4	8.6	8.8
Compare (Short Operand)	RR	4.0	3.6	4.0	4.1	4.2
Compare (Short Operand)	RS	4.2	4.4	5.2	4.4	4.6
Convert to Fixed Point	RR	4.0	3.6	4.4	4.1	4.2
Convert to Floating Point	RR	3.4	3.0	4.0	3.5	3.6
Divide (Long Operand)	RR	115.6	115.2	115.6	115.7	115.8
Divide (Long Operand)	RS	116.6	116.8	117.6	116.8	117.0
Divide (Short Operand)	RR	9.8	9.4	9.8	9.9	10.0
Divide (Short Operand)	SRS	10.0	9.6	10.0	10.2	10.4
Divide (Short Operand)	RS	10.0	10.2	11.0	10.2	10.4
Load (Long Operand)	RS	3.0	3.2	4.0	3.3	3.6
Load (Short Operand)	RR	1.4	1.0	1.4	1.5	1.6
Load (Short Operand)	SRS	1.8	1.4	1.8	2.0	2.2
Load (Short Operand)	RS	1.8	2.0	2.8	2.1	2.3
Load Complement (Short Operand)	RR	1.4	1.0	1.4	1.5	1.6
Load Fixed Register	RR	1.4	1.0	1.4	1.5	1.6
Load Floating Immediate	RR	2.6	2.2	2.6	2.7	2.8
Load Floating Register	RR	1.2	0.8	1.2	1.3	1.4
Multiply (Long Operand)	RR	24.0	23.6	24.0	24.1	24.2
Multiply (Long Operand)	RS	25.2	25.4	26.2	25.4	25.6
Multiply (Short Operand)	RR	6.4	6.0	6.4	6.5	6.6
Multiply (Short Operand)	SRS	6.6	6.2	6.6	6.8	7.0
Multiply (Short Operand)	RS	6.6	6.8	7.6	6.8	7.0
Subtract (Long Operand)	RR	6.2	5.8	6.2	6.3	6.4
Subtract (Long Operand)	RS	7.6	7.8	8.6	7.8	8.0

Floating-Point Operations (Note 3)	Format	Execution Time (μs)				
		Even	Odd NOK	Odd $\overline{\text{NOK}}$	Even 100	Even 200 EMU
Subtract (Short Operand)	RR	3.8	3.4	3.8	3.9	4.0
Subtract (Short Operand)	SRS	4.0	3.6	4.0	4.2	4.4
Subtract (Short Operand)	RS	4.0	4.2	5.0	4.2	4.4
Store (Long Operand)	RS	4.0	4.2	5.0	4.2	4.4
Store (Short Operand)	SRS	2.8	2.4	2.8	3.0	3.2
Store (Short Operand)	RS	2.8	3.0	3.8	3.0	3.2
Special Operations						
Insert Store Protect Bits (Privileged Instruction)	RS	3.2	3.4	4.2	3.5	3.8
Load Program Status (Privileged Instruction)	RS	3.8	4.0	4.8	4.0	4.3
Set Program Mask	RR	2.4	2.0	2.4	2.6	2.8
Set System Mask (Privileged Instruction)	RS	3.4	3.6	4.4	3.6	3.8
Supervisor Call	RS	7.6	7.8	8.6	8.2	8.8
Test and Set Bits	SI	3.4	3.6	4.4	3.6	3.8
Test and Set	RS	3.0	3.2	4.0	3.2	3.4
I/O and Internal Control						
Input/Output (Privileged Instruction) (IOR) (Note 5)	RR	4.4	4.0	4.4	4.5	4.6
Internal Control (Privileged Instruction) (ICR)						
a) Read Counters	RR	3.4	3.0	3.4	3.5	3.6
b) Write Counters	RR	3.2	2.8	3.2	3.3	3.4
c) AGE Read (Privileged Instruction)	RR	28.8	28.4	28.8	28.9	29.0
d) AGE Write (Privileged Instruction)	RR	5.4	5.0	5.4	5.5	5.6
e) Load Discretes Out	RR	3.6	3.2	3.6	3.7	3.8

NOTE 1 The operand is between the limits and has the same sign as the limits.

NOTE 2 The search count = 1. Each additional count will increase execution time by 2.6 μs for internal, 2.7 μs for even 100, and 2.8 μs for even 200.

NOTE 3 The Floating-Point Add and Subtract times are based on operands requiring one shift to equalize characteristics and no normalization. The Floating-Point Multiply and Divide times are based on operands requiring no normalization. Each additional shift to equalize characteristics adds 0.2 μs. Each additional normalize shift adds 0.4 μs.

NOTE 4 For index operations, add 0.4 μs.
For index modification operations, add 1.2 μs.
For indirect operations, add 1.2 μs.
For indirect modification operations, add 2.8 μs.

NOTE 5 Based on CPU not being held by I/O.

DEFINITIONS

EVEN - Instruction counter contains an even address.
ODD - Instruction counter contains an odd address.
NOK - Next instruction register contains either an RR/SRS instruction or one-half of an RS instruction.
$\overline{\text{NOK}}$ - Next instruction register contains neither an RR/SRS instruction nor one-half of an RS instruction. This condition only occurs following branch operations.
RR - Register to Register
RS - Register to Storage
SRS - Short register to Storage
RI - Register immediate
SI - Storage immediate
Even 100 - Execution time for 24K External Memory
Even 200 - Execution time for an additional 65K EMU

2.5 SYSTEM STATUS AND PROGRAM STATES

Program Status Word Determines System Status and Enhances Interrupt Transparency

Features are provided to accommodate change of the CPU status, so that program operation can be resumed automatically after an interrupt or abnormal system function.

The program status word (PSW) contains all information not contained in storage or registers but required for proper program execution. The 64-bit PSW includes the next instruction address, the current condition code, the system mask for interrupts, and other fields significant to CPU operations. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. The active or controlling PSW is called the "current PSW." Storing the current PSW during an interruption allows preserving the status of the CPU for subsequent use. Loading a new PSW, or part of the PSW, permits initializing or changing the state of the CPU. The overall status of the CPU is preserved in the current PSW and the contents of the general registers. The PSW is automatically retained upon taking an interrupt. The programmer has the responsibility for preserving the contents of the general registers and any interrupts, real-time clocks, or I/O system status, if required.

The overall CPU status is determined by several program state alternatives, each of which can be changed independently to its opposite and most of which are indicated by a bit in the PSW. The following paragraphs describe the four types of program state alternatives.

Supervisor or Problem State

In the Supervisor state (PSW bit 47 equal 0), all instructions are valid. In the Problem state (PSW bit 47 equal 1), Insert Storage Protect Bits, Load Program Status, Set System Mask, and Input/Output instructions are illegal, and their use produces a program interrupt. These instructions are reserved for use only in the Supervisor state. This provides one common centralized system control function for interrupt handling, system status switching, and I/O handling; thus eliminating unrestricted system control operations by the various problem programs. This state is controlled by a bit in the PSW. The SVC instruction is provided to switch from problem to supervisor state. The load PSW instruction is used to switch from supervisor to problem state.

Masked or Interruptable State

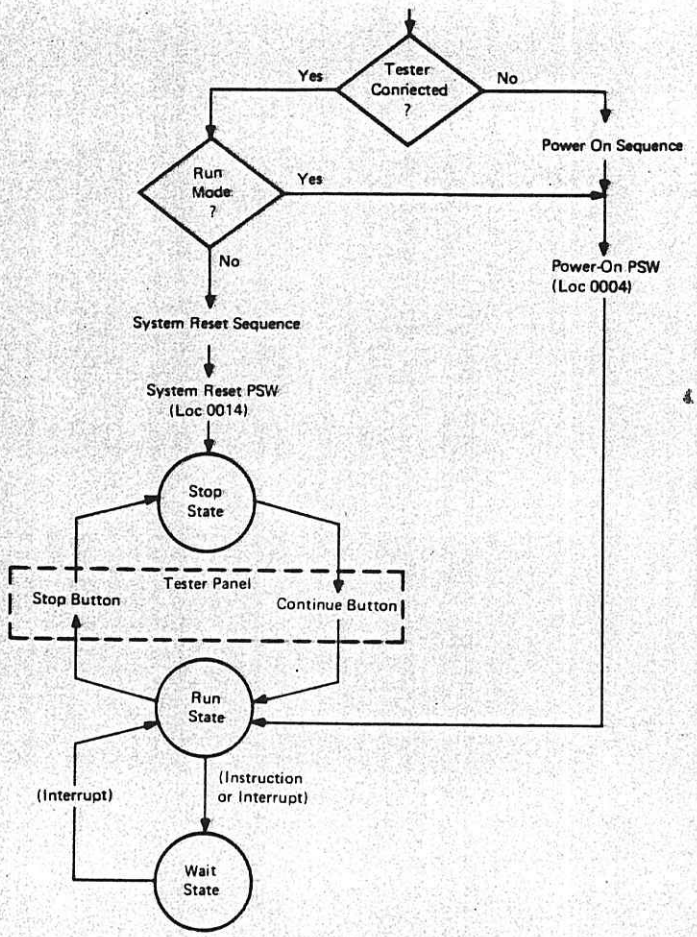
The computer may be masked for certain interrupt conditions at any given time. These conditions generally remain pending within the system until the masked condition is changed by the program. Certain error conditions cannot be masked off, while other error conditions, such as program checks, are ignored when specifically masked.

General Register Selection

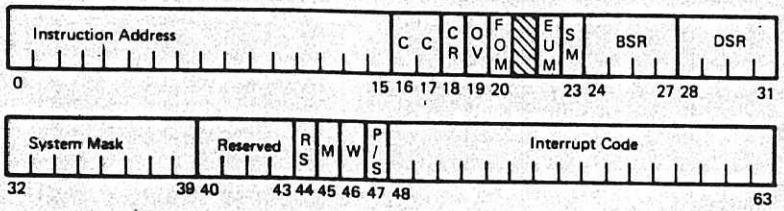
Bit 44 in the current PSW selects the set of general registers in current use.

Stopped or Operating State

The run state and wait state are collectively termed the operating state for the system. When the computer is in the run state, instructions are executed in the normal manner. An instruction may be encountered or an interrupt processed that forces the computer into the wait state. The computer does not execute instructions in the wait state, but it is interruptable when not masked. Real-time clock timers are updated and I/O operations continue in the wait state. When the CPU is in the stopped state, instructions and interruptions are not executed, and clocks are not updated. The stopped or operating state can be changed only by manual intervention.



Program State Sequence



0-15 — Next instruction Address
16-17 — Condition code for certain arithmetic, logical, and I/O instructions
18 — Carry indicator
19 — Overflow indicator
20 — Fixed Point Arithmetic Overflow Mask
21 — Reserved
22 — Exponent Underflow Mask
23 — Significance Mask
24-27 — Branch sector register replaces the high-order bit of a branch address when that bit equals
28-31 — Data sector register replaces the high-order bit of a data address when that bit equals 1
32-39 — System mask for external interrupts
40 — P08 XDSDCN
41 — P09 BDSDCN
42 — P10 SPR1N
43 — P11 SPR2N
44 — Register set controls which of two sets of general registers is in current use
45 — Machine check mask
46 — Wait state bit
47 — Problem/supervisor state control bit
48-63 — Interrupt code for supervisor call instruction machine check program and certain external interrupts

Program Status Word

Table 1
MICRO INTERRUPT STRUCTURE IN HARDWARE

Priority	Interrupt Number	Interrupt	Initial Prom Address (Hex)	Mask Bit
Highest	1	Power On & System Reset	03F	None
	2	Main Store Data Parity Error *1	037	Current PSW Bit 45
	3	I/O Interface Address Parity Error *1	03B	Current PSW Bit 45
	4	Extended Memory Address Parity Error	033	Current PSW Bit 45
	5	I/O Interface Data Parity Error *1	03D	Current PSW Bit 45
	6	Extended Memory Data Parity Error *1	035	Current PSW Bit 45
	7	Address Specification *1	031	None
	8	Store Protect Violation *1	03E	Current PSW Bit 45
	9	ROS Parity Error	036	Current PSW Bit 45
	10	Fixed Point Overflow	03A	Current PSW Bit 20
Lowest	11	Instruction Monitor	032	Current PSW Bit 33

*1 Enabled only when ROS Data Register Bit 45 = 1 and is not masked

Table 2
MACRO INTERRUPT STRUCTURE IN HARDWARE

Priority	Interrupt Number	Interrupt	Initial Prom Address (Hex)	Mask Bit
Highest	1	Single Instruction Cycling/Put-away	00F, 01F, 02F	None
	2	Counter 1	00C, 01C, 02C	Current PSW Bit 32
	3	Counter 2	004, 014, 024	Current PSW Bit 33
	4	External 0	008, 018, 028	Current PSW Bit 34
	5	AGE Interrupt	000, 010, 020	None
	6	External 1	00E, 01E, 02E	Current PSW Bit 35
	7	External 2	006, 016, 026	Current PSW Bit 36
	8	External 3	00A, 01A, 02A	Current PSW Bit 37
Lowest	9	External 4	002, 012, 022	Current PSW Bit 38

Table 3
MICROPROGRAMMED INTERRUPT LIST

Interrupt Number	Interrupt Type	Mask Bit
1	Privileged Instruction	See Note *1
2	Illegal Operation	None
3	Significance	Current PSW Bit 23
4	Exponent Underflow (Floating Point)	Current PSW Bit 22
5	Unnormalized Inputs (Floating Point Divide)	None
6	Exponent Overflow (convert)	None
7	Exponent Overflow (Floating Point)	None
8	Divide Exception	None

*1 Interrupt taken when current PSW Bit 47 = 1

2.6 CPU INTERRUPT STRUCTURES

Interrupt Structures are Implemented in Hardware and Microcode Operations

Microinterrupt, macrointerrupt, and microprogrammed interrupt structure operation is basically transparent to the macroprogrammer until a PSW swap results.

Each of the interrupt structures discussed in this section are listed in applicable tables on pages 2-28 and 2-29 for ease of reference during the following description discussions.

Table 1 lists the microinterrupt structure, its associated priority level, the mask, and the PROM address (in hex) of the first microinstruction in the microinterrupt subroutine. Microinterrupts are interrupts that can occur at any time during macroinstruction operations if the interrupts are not disabled or masked. The interrupt occurs when the hardware forces the microprogram to branch to the specified microinstruction address. The highest priority interrupt will be taken if more than one interrupt occurs simultaneously, the others will remain pending and wait their turn to be taken. The microinterrupts noted by *1 are enabled only when ROS data register bit 45 is a one. When this bit is a "0", the specific interrupt is held pending. There are micro-order sequences where microinterrupts would disrupt computer operations, such as in the middle of memory cycles. Therefore, bit 45 is used to enable these interrupts at specific locations in the microcode sequencing. When microinterrupts are masked by the current PSW mask bit, these interrupts do not remain pending.

The macrointerrupts are listed in Table 2. This interrupt structure is identical in operation to the microinterrupts with the exception that the macrointerrupts are processed by the microprogram only at the end of the current instruction. When the interrupt occurs, the hardware forces the microprogram to branch, at the end of the instruction, to one of the three specified microinstruction addresses. The reason there are three possible locations to branch to is because storebacks (storing results into the general registers) are done in the first microinstruction of the following macroinstruction. This means that three possible microbranch addresses are needed to indicate whether no storeback, storeback into general registers or storeback into floating point registers are to be performed. Table 2 also lists the mask bit associated with each interrupt. If the mask bit equals "0", the macrointerrupt will remain pending until the mask bit is set to a "1".

The third type of interrupt structure is simply a microcoded interrupt and not a hardware forced interrupt. Table 3 lists these interrupts with their associated mask bits. During macroinstruction sequencing, a microbranch may occur within specific macroinstructions which may lead directly to the interrupt microroutine, if it is not masked. An example of this is the significance interrupt. This interrupt is to occur if the current PSW bit 23 equals "1" and if the resulting fraction of a floating point add or subtract is zero. Within the floating point add and subtract macroinstruction, a branch is used to test the resulting answer for a zero fraction. If the microbranch indicates a zero floating point fraction, the microbranch leads directly to the interrupt microroutine. An exception to this is the privileged instruction interrupt. This interrupt is a microprogrammed branch within all privileged instructions which tests the current PSW bit 47. If PSW bit 47 is a zero, the CPU is in the supervisor state and the microbranch allows completion of the privileged instruction. If PSW bit 47 is a one, the CPU is in the problem state and the microbranch leads to the microinterrupt routine bypassing the privileged instruction execution.

In general, an interrupt subroutine at the micro level performs the following functions:

- Disables all interrupts.
- Resets the hardware interrupt latch that was set
- Generates an interrupt code if applicable
- Performs the PSW swap microroutine.

- Enables interrupts
- Begins the macrointerrupt routine located by the new PSW or goes into the wait state if the current PSW bit 46 equals a zero.

The above sequencing is true for both microinterrupt and macrointerrupt structures. For the microprogramed interrupt structure, the microroutine is similar to the above with the exclusion of item 2. Since this interrupt structure is in microcode there is no hardware latch to reset. The above microroutine will be referred to as the "typical" microroutine for interrupts in later discussions. There are exceptions to the above routines which are described in the applicable section.

Table 4 lists the interrupts as seen by the macroprogrammer. The interrupts in Table 4 are a combination of the three interrupt structures listed in Tables 1, 2, and 3. Table 4 lists the interrupt priority, the class, the interrupt code if applicable, the PSW location and the masks. The table also gives an indication as to whether the old PSW, that is stored away when the interrupt occurs, is pointing correctly to the next macroinstruction address. Imprecise old PSWs occur for interrupts using the microinterrupt structure (Table 1). These interrupts can occur during macroinstruction operation which means that the instruction counter may or may not be updated when the interrupt is taken. The following paragraphs describe each interrupt in Table 4.

Power On/System Reset

The two highest priority interrupts in Table 4, Interrupts 1 and 2, are microinterrupts (Table 1, Interrupt 1). When either of these interrupts occur, the CPU is forced by hardware to branch to the PROM address 03F. Figure 1 shows the microroutine starting at PROM address 03F. The very first thing that occurs is that all interrupts and Direct Memory Access (DMA) operations are hardware disabled at 03F. The System Reset latch is set only if the 03F PROM location was entered via a System Reset operation. At the I/O interface there is a line called I/O Lockout. If this line is held active by the I/O during System Reset operations, the CPU is to perform Instruction Program Load (IPL). Microbranch A in Figure 1 tests this line. If I/O Lockout is "off", initialization of the CPU occurs. Specifically the low 16 bits of counters 1 and 2 are set to their maximum value, the status registers are reset, all the interrupt hardware latches are reset, and the store protect override latch is reset. If I/O lockout is "on", microbranch A takes the IPL path. This consists of microcode operations that set the store protect override logic and then enables DMA functions. Microbranch B is a microcode operation that loops on itself until the I/O Lockout line goes to "off". When I/O lockout is "on", the I/O will be allowed via DMAs, to do an Instruction Program Load (IPL) while the CPU loops on microbranch B. Since the store protect override logic is set, the DMAs may, at the discretion of the I/O, turn on the store protect bit during IPL. After IPL is finished, the I/O will turn "off" the I/O lockout which will cause microbranch B to cease looping (Figure 1) and continue microcode operations through the initialization. After initialization occurs, microbranch C tests the System Reset latch to see if the microroutine was entered from a Power On or a System Reset. If the System Reset latch is "on", the current PSW is taken from the mainstore location 0004 (hex) through 0007 (hex); if the latch is "off", the PSW is taken from mainstore location 0014 (hex) through 0017 (hex). After this, the next microcode operation enables DMAs and interrupts. Then the first instruction is fetched from the main store location specified by the current PSW.

Table 4
SPACE SHUTTLE CPU INTERRUPTS

Interrupt No.	Hardware Priority *2	Interrupt Type	Old PSW IC Value	Class	Interrupt Code (Hex)	PSW Location (Hex)	Current PSW Mask Ext,
1	1	Power On	N/A	Power	N/A	0004-0007	None
2	1	System Reset	N/A	Power	N/A	0014-0017	None
3	1	Power Off (POID)	N/A	Power	N/A	N/A	None
4	2	CPU Storage Parity	Imprecise	Machine	0003	0040-0047	45
5	3	IOP Storage Parity	Imprecise	Machine	0002	0040-0047	45
6	4	Extended Storage Address Parity	Imprecise	Machine	0001	0040-0047	45
7	5	Extended Storage data Parity	Imprecise	Machine	0004	0040-0047	45
8	6	CPU Address Specification	Imprecise	Program	0003	0048-004F	None
9	7	CPU Store Protect Violation	Imprecise	Program	0007	0048-004F	45
10	8	CPU ROS Parity	Imprecise	Machine	0005	0040-0047	45
11	9	Instruction Monitor	Imprecise	Program	N/A	0070-0077	33
12	10	Illegal Operation	Imprecise	Program	0000	0048-004F	None
13	10	Privileged Instruction	Next Instr.	Program	0001	0048-004F	47
14	10	Fixed Point Overflow	Next Instr.	Program	0004	0048-004F	20
15	10	Significance	Next Instr.	Program	0005	0048-004F	23
16	10	Exponent Underflow (Floating)	Next Instr.	Program	0009	0048-004F	None
17	10	Exponent Overflow (Convert)	Next Instr.	Program	000A	0048-004F	22
18	10	Exponent Overflow (Floating)	Next Instr.	Program	000B	0048-004F	None
19	10	Divide (Floating Point)	Next Instr.	Program	000C	0048-004F	None
20	10	Unorm. Inputs (Dvd. Fltg. Pt.)	Next Instr.	Program	0006	0048-004F	None
21	11	Executive (Supervisor Call)	Next Instr.	Supervisor	N/A	0058-005F	None
22	12	Initiate Putaway	Next Instr.	Power	N/A	0010-0013	None
23	13	Counter 1	Next Instr.	System	N/A	0060-0067	32
24	14	Counter 2	Next Instr.	System	N/A	0068-006F	33
25	15	External 0	Next Instr.	System	N/A	0078-007F	34
26	16	External 1	Next Instr.	System	N/A	0080-0087	35
27	16	I/O Address Specification *1	Next Instr.	System	0003	0080-0087	35
28	16	I/O Store Protect Violation *1	Next Instr.	System	0004	0080-0087	45 & 35
29	16	I/O Write Parity (Interface) *1	Next Instr.	System	0002	0080-0087	45 & 35
30	16	PCI Data Parity *1	Next Instr.	System	0001	0080-0087	45 & 35
31	16	I/O Address Parity *1	Next Instr.	System	0005	0080-0087	45 & 35
32	16	AGE Interrupt *1	Next Instr.	System	0006	0080-0087	35
33	17	External 2	Next Instr.	System	N/A	0088-008F	36
34	18	External 3	Next Instr.	System	N/A	0090-0097	37
35	19	External 4	Next Instr.	System	N/A	0098-009F	38

*1 I/O Interrupts Detected by the Space Shuttle CPU
*2 The larger the number the lower the priority

Power Off Delayed (POID)

When the CPU is powered off, a Power off signal is initiated by the power supply when the source drops below 17.5 V. After 100μs, a Power Off Delayed (POID) interrupt (Table 4, Interrupt 3) is generated which clamps memory and holds the computer at PROM address 03F until power is "off". During the 100-μs period it is expected that the Initiate Putaway interrupt will occur and allow completion of the putaway routine before the POID interrupt occurs. The Initiate Putaway interrupt will be discussed later in more detail.

Main Store Parity Interrupts (CPU & I/O)

The CPU storage parity interrupt (Table 4, interrupt 3) will occur if not masked and if a parity error is detected in the low 40K halfwords of main storage located in the computer, during CPU operations. The IOP storage parity interrupt (Table 4, Interrupt 5) will occur if not masked and a parity error is detected in main store via IOP Direct Memory Access "out" (DMA) operations. These two interrupts are microinterrupts (Table 1) sharing the same hardware. Microinterrupt 2 in Table 1 is activated by either of these interrupts and the microcode routine, beginning at PROM 037, distinguishes between the two interrupts by a microbranch which then provides the proper interrupt code. In general, these interrupts follow the "typical microroutine sequence defined earlier.

When a DMA "out" results in a main storage parity error, a parity error will be issued on CPU/IOP interface lines to allow the IOP to detect a data error more rapidly. There are two parity error lines on the CPU/IOP interface, one for each halfword.

When a main storage parity error is detected during the memory read portion of a CPU read-compute-write operation, the write operation will be inhibited to prevent loading of erroneous data.

Extended Storage Parity (data and address)

Extended storage address parity interrupt is a microinterrupt (Table 1, Interrupt 4) that occurs if not masked and if an address parity error is detected when addressing storage locations within the IOP unit. This error is detected by the extended memory unit in the IOP and is sent to the CPU via hardwire. If the error occurs during DMA operations, the two data parity error hardwires at the CPU/IOP interface will both be activated, allowing rapid error detection by the IOP. If during memory write operations

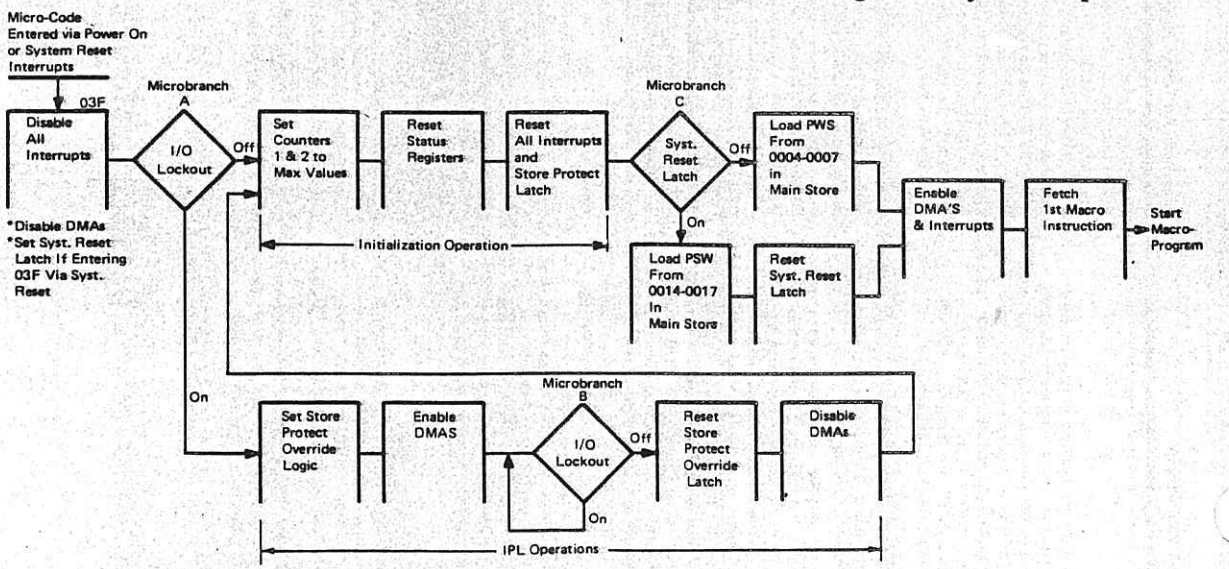


Figure 1. Initialization Microroutines used for Power On or System Reset Interrupts

an address parity error occurs, the write operations will be inhibited to prevent unwanted memory alterations. This interrupt follows the "typical" microroutine sequence defined earlier. Extended storage data parity is also a microinterrupt (Table 1, Interrupt 6) which will occur during CPU operations if not masked and if there is a data parity error when reading the main storage locations contained in the IOP. The microroutine for this interrupt begins at PROM address 035 (hex) and follows the "typical" microroutine sequences. When this data error is detected during the memory read portion of a CPU read-compute-write operation, the write operation will be inhibited.

CPU Address Specification

The CPU address specification interrupt is a microinterrupt (Table 1, Interrupt 7) that will occur if a main store operation is performed by the CPU using a storage address location that is out of bounds, that is, a non-existent main storage location. This interrupt is not maskable. The microroutine for this interrupt begins at PROM address 031 and follows the "typical" microroutine sequences.

CPU Store Protection

CPU store protection interrupt is a microinterrupt (Table 1, Interrupt 8) that will occur if not masked and if, during CPU sequencing, a memory write operation is attempted into any main store location in which the store protect bit is active. The main store write operation will be inhibited to prevent memory alterations. The microroutine for this interrupt begins at PROM address 03E (hex) and follows the "typical" microroutine sequences.

CPU ROS Parity

CPU ROS Parity interrupt is a microinterrupt (Table 1, Interrupt 9) that occurs if not masked and if the 48 bits, at the output of the ROS Data Register (ROSDR), does not have odd parity. This interrupt will occur if there is a single bit failure in accessing ROS data. The microroutine for this interrupt begins at PROM address 036 (hex) and follows the "typical" microroutine sequencing.

Instruction Monitor

Instruction Monitor interrupt is a microinterrupt (Table 1, Interrupt 11) that can occur if not masked and if the instruction that is fetched from main store during macroinstruction operations is not store protected. The hardware only looks at the store-protect bit of the first halfword of a long instruction to determine if the interrupt is to happen. For short instructions, the halfword boundary in which it is stored must be store protected. The microroutine for this interrupt begins at PROM address 032 (hex) and follows the "typical" microroutine sequencing. This interrupt does not allow this macroinstruction to execute.

Illegal Operation

Illegal operation interrupt is a microprogrammed interrupt (Table 3, Interrupt 2) that occurs when the macroprogrammer tries to perform an instruction with an illegal or unspecified operation code. This interrupt has no mask. The sequencing for this microprogrammed interrupt is the same as the "typical" microroutine described earlier.

Privileged Instruction

Privileged instruction interrupt is a microprogrammed interrupt (Table 3, Interrupt 1) that occurs when attempting to do a privileged instruction when in the problem state (current PSW bit 47 = '1'). The following macroinstructions are privileged instructions:

- Insert Store Protect bits (ISPB)
- Load Program Status (LPS)
- Set-System Mask (SSM)
- Internal Control (IC)
- Program Controlled Input (PCI)
- Program Controlled Output (PCO).

The microsequencing for this interrupt is the same as that of the "typical" microroutine described earlier.

Fixed Point Overflow

Fixed point overflow interrupt is a microinterrupt (Table 1, Interrupt 10) that can occur if not masked and if the result of fixed point arithmetic operations is too large and will not fit into the fullword format (32 bits). When this microinterrupt occurs, it begins its micro operations at PROM address 03A (hex) and then follows the "typical" microroutine sequencing.

Significance

The significance interrupt is a microprogrammed interrupt (Table 3, Interrupt 3) that will occur if not masked and when the resulting fraction of a floating point operation is zero while the characteristics is not zero. Within the microprograms of the floating point macroinstructions, the floating point result is tested by the microprogram for significance (characteristic doesn't equal zero, fraction equals zero). If significance occurs, then the microprogram tests the current PSW bit 23. If this bit = '0', the microprogram finishes the floating point instruction and provides a result of all zeros (the characteristic is zeroed). If the current PSW bit 23 = '1', the microprogram branches to perform the "typical" microroutine sequencing.

Unnormalized Inputs

The unnormalized input interrupt is a microprogrammed interrupt (Table 3, Interrupt 5) that will occur when the input data to a floating point divide is unnormalized. This interrupt is taken by microbranches in the floating-point divide microcode. There is no mask for this interrupt. Once the interrupt microbranch is taken, the rest of the sequence is similar to the "typical" microroutine.

Exponent Underflow

The exponent underflow interrupt is a microprogrammed interrupt (Table 3, Interrupt 4) that occurs during floating-point arithmetic operations if not masked and if the resulting characteristic becomes negative and cannot be expressed in the 7-bit format allocated for the characteristic. This interrupt is taken by a microbranch in the floating-point arithmetic microcode. The interrupt micro-operation once taken follows the "typical" microroutine sequencing.

Exponent Overflow for Converts

The exponent overflow interrupt is a microprogrammed interrupt (Table 3, Interrupt 6) that occurs during the convert to floating (CVFL) point macroinstruction if the resultant characteristic is greater than 127 and the fraction is not zero. There is no mask for this interrupt. The interrupt is taken by a microbranch in the CVFL microcode which leads to microroutine which is similar to the "typical" microroutine described earlier.

Exponent Overflow for Floating Point

This description is identical to the exponent overflow for converts with the exception that the interrupt (Table 3, Interrupt 7) is used to detect overflows for the other floating point operations.

Divide Floating Point

This interrupt is also a microprogrammed interrupt (Table 3, Interrupt 8) that occurs if the denominator of a floating-point divide is equal to zero. This interrupt has no mask. The microinterrupt routine followed is similar to the "typical" microroutine and is taken when the divide microcode detects a zero denominator and forces a microbranch to this routine.

Executive (Supervisor Call)

The Supervisor Call interrupt is caused by execution of the Supervisor Call macroinstruction (SVC). It basically falls under the microprogrammed interrupt structure. When the SVC instruction is executed, the instruction performs a PSW switch. This includes loading a 16-bit condition code which is equal to the 16-bit effective address generated by the SVC macroinstruction operational code.

Initiate Putaway

Initiate Putaway interrupt is a macroinstruction (Table 2, Interrupt 1) that will occur when the power supply issues a Power Off signal (POI) and when the current macroinstruction has completed. This interrupt has no mask. Figure 2 illustrates the microcode operations for the putaway routine. With the occurrence of a POI, the hardware disables DMAs. At the end of the current macroinstruction in which the POI occurred, the interrupt is forced to one of three PROM locations 00F, 01F, or 02F, depending on whether the current macroinstruction required a register store back. The microcode then disables all interrupts, and stores in main storage the general registers, the floating-point registers, the microprogrammers working register, the counters, and the operational registers. After this, the CPU clocks are stopped by a Stop Clock micro order; the computer waits for the final POID which clamps the computer and memory for the final powering off operation. As was explained under the POID interrupt description, there is 100μs between POI and POID. This means that the time to complete the current instruction plus the putaway time (38μs) must not exceed 100 μs. There are two macroinstructions which may not allow completion of the putaway routine. These macroinstructions are Search Under Mask (a variable length instruction) and Long Floating-Point Divide (approx. 123μs). The length of Search Under Mask (SUM) is controlled by the macroprogrammer, therefore, he should make sure that the count doesn't cause the SUM to execute more than 62μs. Long Floating-Point Divide on the other hand is 123μs long and is not controlled by the macroprogrammer. Therefore, this instruction is designed with a microbranch that tests for the POI interrupt. When the POI interrupt occurs the microbranch terminates the divide immediately, resets the instruction address counter, and then proceeds to perform the Putaway routine shown in Figure 2.

The Putaway interrupt is also initiated by the AGE interface when performing single instruction stepping operations. When single Instruction Stepping via AGE control the clocks are restarted by activating the AGE "continue" switch, causing the microcode shown in Figure 2 to come out of the "Stop Clocks" micro order, load the current PSW and reset the Putaway Interrupt latch. The hardware then sets the Putaway Interrupt latch during the next macroinstruction operation. The interrupt is taken at the end of the macroinstruction and sequences again from 00F, 01F, or 02F through the putaway microroutine (Figure 2) to the "Stop Clocks" micro order where the computer stops and waits for the next AGE continue operation. Using the interrupt in this fashion allows single instruction stepping via AGE controls.

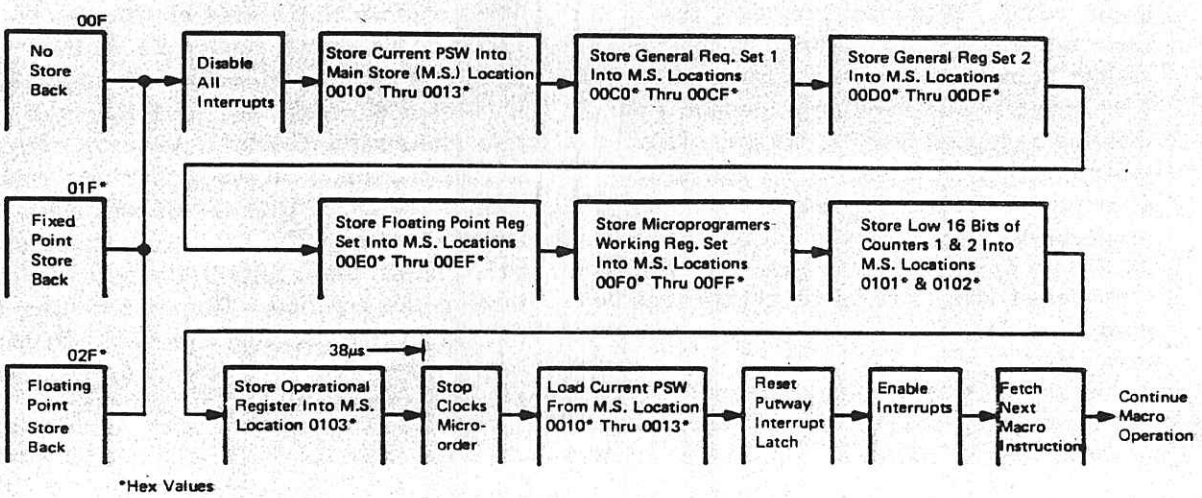


Figure 2. Putaway Microroutine

75-A97-001

Counter 1

Counter 1 in the CPU is a 32-bit interval timer in which the upper 16 bits resides in the main storage location 00B0 while the low 16 bits resides in a hardware 16-bit binary counter. The low 16-bit hardware binary counter has a basic clock of 1 μs. The hardware binary counter counts down from FFFF (hex) to 0000 (hex) in 65.536 ms and causes a macrointerrupt (Table 2, Interrupt 2) to occur, if not masked, every time the counter passes through 0000 (hex). The hardware counter counts even when masked. Figure 3 shows the microroutine that is performed every time an interrupt occurs. The microroutine disables all interrupts, resets the Counter 1 interrupt latch then fetches the high 16 bits of the counter from main store location 00B0 (hex). These high 16 bits are decremented by one via the microcode then stored back into the main store location 00B0 (hex). This gives the appearance of a full 32-bit counter. The routine also tests the undecremented high 16 counter bits; and if they are equal to zero, then a PSW swap occurs. If the undecremented high 16 counter bits are not equal to zero, then the microroutine returns to the previous macroinstruction sequencing. If the high 16 counter bits are not equal to zero, then the interrupt is transparent to the macroprogrammer. He only knows an interrupt occurs when the PSW swap occurs. If the mask is "on", the interrupt will remain pending and the high 16 bits in main store will not be decremented. The upper 16 bits may loose a count if the mask is on longer than 65.536 μs. This means that the 32 bit counter would be in error by 65.536 μs. Therefore, the macroprogrammer should always unmask every 65.536 μs or less to make sure a 65.536 μs count is not lost. The 32-bit counter can be read or loaded via an Internal Control macroinstruction.

Counter 2

Counter 2 operates in exactly the same manner as Counter 1 and requires no additional description.

External 0

External 0 interrupt is a macrointerrupt (Table 2, Interrupt 4) that occurs if unmasked, and when the IOP requests the interrupt via the CPU/IOP interface. This interrupt, if masked, will remain pending. The microroutine for this interrupt begins at PROM address 008 (hex), 018 (hex), or 028 (hex) and performs similar to the "typical" microroutine described earlier. Since this is a macrointerrupt, the microroutine doesn't begin until the current macroinstruction completes its execution.

External 1

External 1 interrupt is a macrointerrupt (Table 2, Interrupt 6) that occurs if unmasked and when the IOP requests the interrupt via the CPU/IOP interface. This interrupt, in addition, can be set by the occurrence of CPU detected CPU/IOP interface errors. Figure 4 illustrates the microroutine that occurs when this interrupt is taken. This interrupt fetches an interrupt code located in the upper byte of the halfword main store location 0087 (hex). Following this, the microsequencing performs a PSW swap which involves shifting the interrupt code right 8 binary places then storing it away, during the PSW swap, in location 0083 (hex). CPU/IOP interface microinterrupts (Table 4, Interrupts 27 and 28), when taken, load the proper interrupt code in the upper byte of halfword main store location 0087 (hex). This process will be explained more fully in the next five interrupt descriptions. The microroutine for the External 1 interrupt begins at PROM address 00E (hex), 01E (hex), or 02E (hex) and performs as shown in Figure 4.

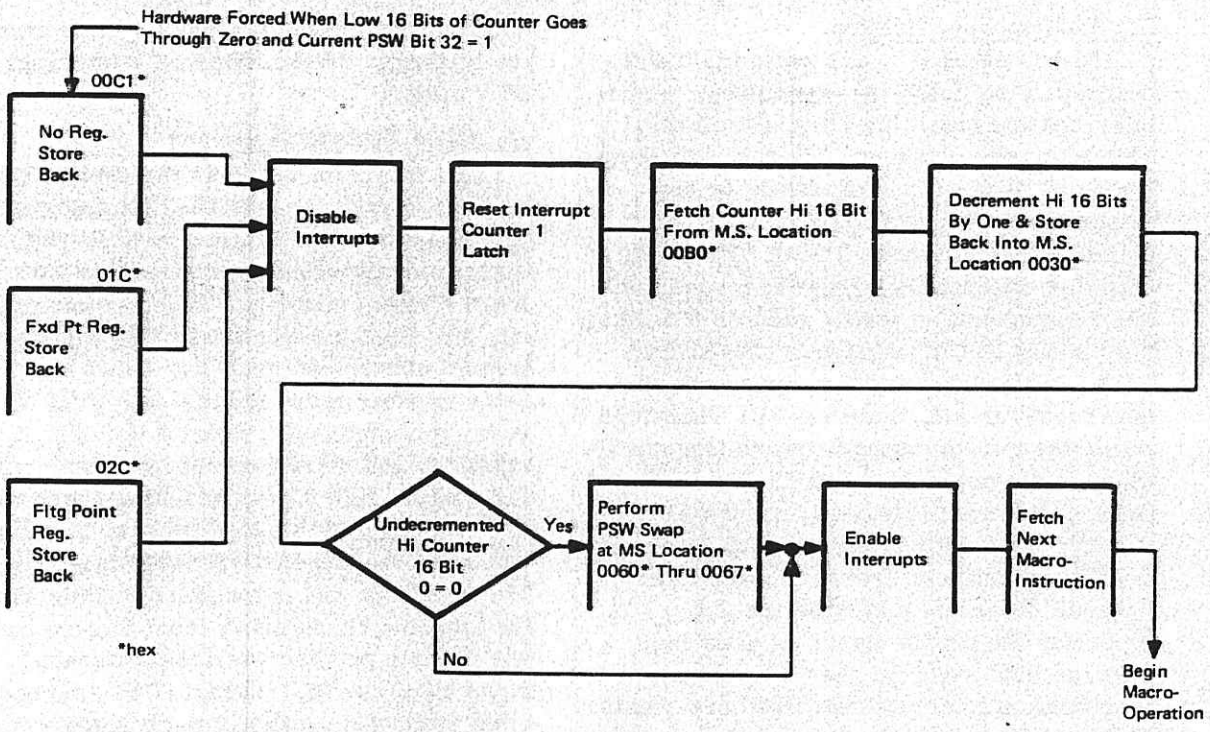


Figure 3. Counter 1 Microroutine

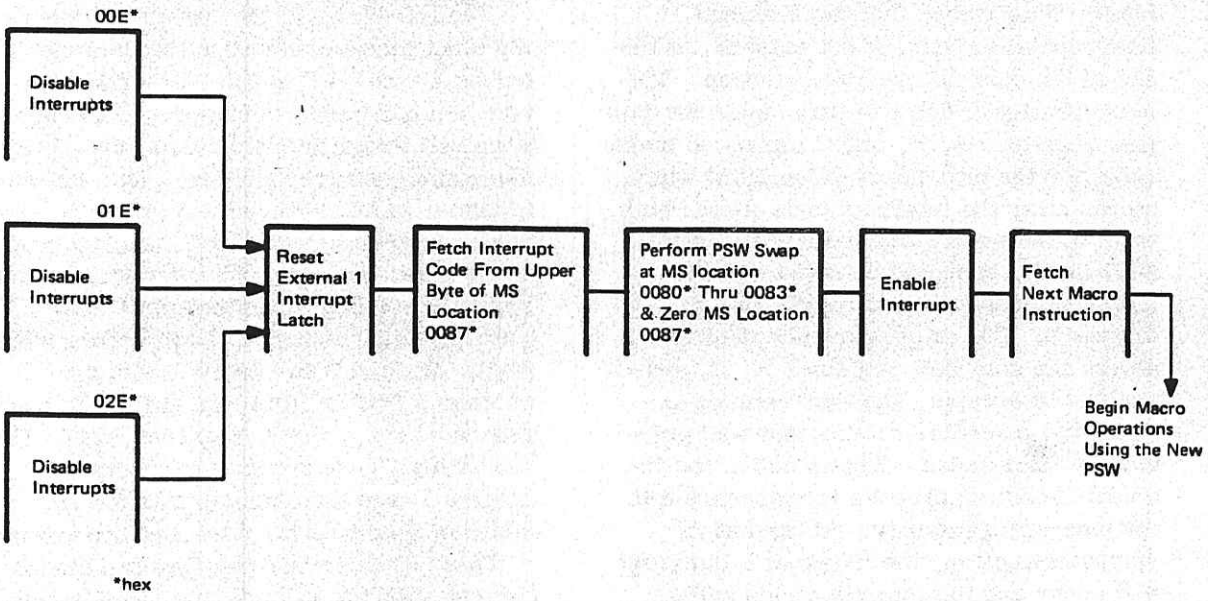


Figure 4. External 1 Interrupt Microroutine

I/O Address Specification

The I/O Address Specification is a combination of two interrupt structures, microinterrupt and macrointerrupt structures. This interrupt, when not masked, will occur if the I/O uses an out of bounds or nonexistent main storage address when doing Direct Memory Access (DMA). When this interrupt occurs, the hardware forces the microsequencing to PROM address 031 (hex). This initial PROM address is also shared by the CPU caused Address Specification interrupt (Table 1, Interrupt 7). Therefore, within the microinterrupt routine (Figure 5) which is shared by both the CPU and I/O there is a branch (Figure 5, branch A). If the address specification interrupt is caused by CPU operations, branch A leads to a "typical" interrupt microroutine that generates the proper interrupt code and does the PSW swap for the CPU Address Specification interrupt described previously (Table 4, Interrupt 8). If this interrupt is caused by I/O operations (DMA's), branch A leads to the I/O microutine operations. This path first sets the External 1 interrupt latch. This means that the External 1 interrupt will occur, if not masked, at the end of the current macroinstruction. The microroutine (Figure 5) then saves the data flow registers A, B, and C simply to make room for the next micro-operations which stores away the interrupt code (in the high byte) in the main store halfword location 0087 (hex). It should be noted that the interrupt code was generated prior to branch A. The microsequence then restores the data flow registers A, B, and C, enables interrupts, and then returns to finish the macroinstruction that was previously interrupted. This much of the interrupt handling process is transparent to the macroprogrammer. At the end of macroinstruction, the External 1 interrupt will occur and the interrupt code will be transferred from main store location 0087 (hex). The interrupt code will be shifted to the the low byte of the halfword then stored away in main store location 0083 (hex). The macroprogrammer will now be made aware of the interrupt via the External 1 interrupt PSW swap.

I/O Store Protect Violation

This interrupt works in the same manner as the I/O Address Specification interrupt. The I/O Store Protect interrupt shares the same microinterrupt operations as the CPU Store Protect interrupt (Table 1, interrupt no. 8). Figure 5 illustrates the microinterrupt sequencing process. Since the CPU and I/O share Store Protect interrupt microcode, this means the current PSW bit 45 can mask the I/O Store Protect interrupt. Both the current PSW bit 45 (machine check mask) and current PSW bit 35 (External 1 mask) can mask the I/O Store Protect interrupt. The interrupt will not remain pending with the machine check mask equal to zero but will remain pending when the External 1 mask equals zero. When a DMA "in" operation attempts to cause a main store write into a store-protected location, that write operation is inhibited.

I/O Write Parity

The I/O Write Parity interrupt occurs when not masked and when the data received by the CPU at the CPU/IOP interface has bad parity during DMA "in" operations and Program Controlled Input (PCI) macroinstruction execution. This interrupt is also a combination of two interrupt structures, microinterrupt and macrointerrupt structures. Figure 5 illustrates the microinterrupt (Table 1, Interrupt 5) sequencing beginning at PROM address 03D (hex). Branch B in Figure 5 detects whether a PCI or DMA "in" occurred and generates the correct interrupt code. The remaining I/O microroutine operations (Figure 1) are the same as that for I/O Address Specification interrupt processing.

This I/O interrupt also has two masks, current PSW bit 45 (machine check mask) and current PSW bit 35 (External 1 mask). The interrupt does not remain pending with the machine check mask but does remain pending with the External interrupt mask.

During a DMA "in" operation if the CPU/IOP interface has a data parity error, the main store write operations will be inhibited.

I/O Address Parity

The I/O Address Parity interrupt occurs when not masked and when the address received by the CPU, at the CPU/IOP interface during DMA operations, has bad parity. This interrupt uses both microinterrupt (Table 1, Interrupt 3) and the macrointerrupt structures. Figure 5 illustrates the microinterrupt portion, which begins at PROM location 03B (hex). The microinterrupt sequence enters the I/O microroutine illustrated in Figure 5 after the interrupt latch is reset and the interrupt code is generated. The remaining portion of the interrupt handling process is identical to that of I/O address specification. This interrupt has the same two masks as I/O write parity interrupt. During a DMA operation if the CPU/IOP interface has an address parity error, the main store write operation will be inhibited. This interrupt uses the same two masks as the I/O Address Specification interrupt.

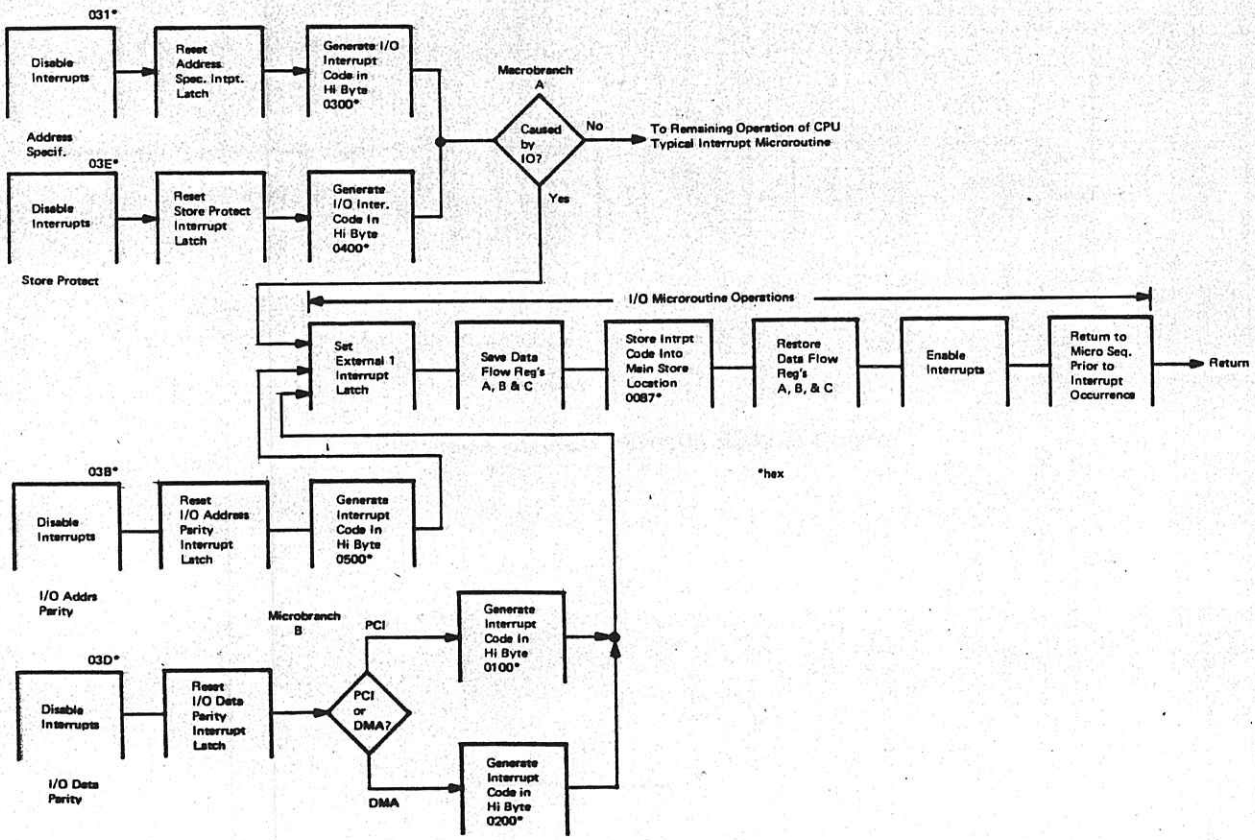


Figure 5. CPU/IO Interface Interrupt Handling Microroutines

AGE Interrupt

The AGE interrupt occurs when the AGE interrupt line at the CPU/AGE interface is activated. This interrupt uses two macrointerrupt structures to perform its task. The first macrointerrupt (Table 2, Interrupt 5) beginning at PROM address 000, 010, or 020 is illustrated in Figure 6. This macrointerrupt routine is transparent to the programmer. When taken, it occurs at the end of the current macroinstruction in which the AGE interface signal is activated. This interrupt sequence, Figure 6, basically sets the External 1 interrupt latch and, then generates and stores the interrupt code in main store location 0087. The External 1 macrointerrupt (Figure 4) will be taken after the completion of the transparent AGE interrupt routine, if a higher priority interrupt is not pending. The External 1 interrupt will then go through its operations basically moving the interrupt code from 0087 (hex) to 0083. The AGE interrupt is maskable only with the External 1 interrupt mask.

External 2, 3, and 4

The reamining three external interrupts operate in the identical manner as External 0 interrupt described earlier.

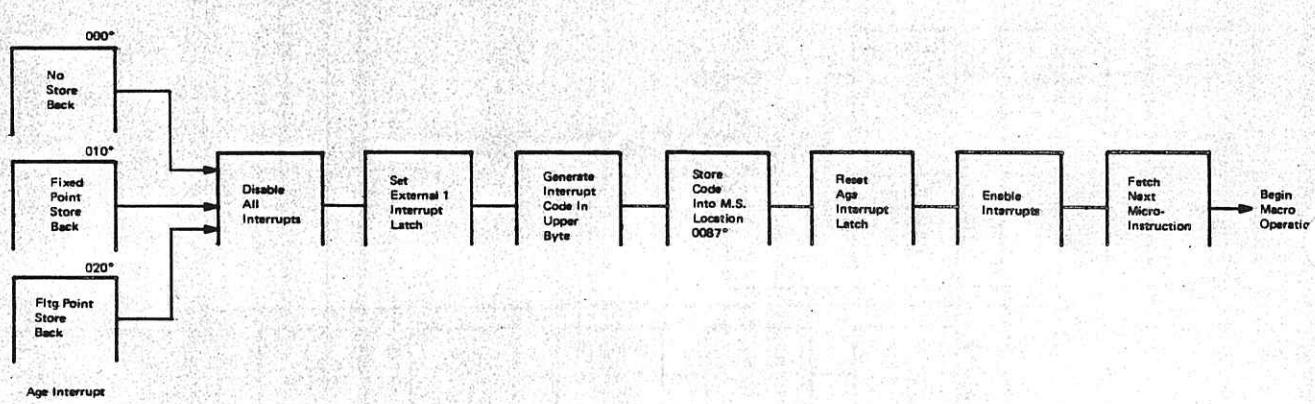


Figure 6. AGE Interrupt Handling Microroutines

Section 3

SUPPORT SOFTWARE

3.1 SYSTEM 360/370 SUPPORT PROGRAMMING SYSTEM

Extensive Background and Experience Results in Efficient, Proven Software

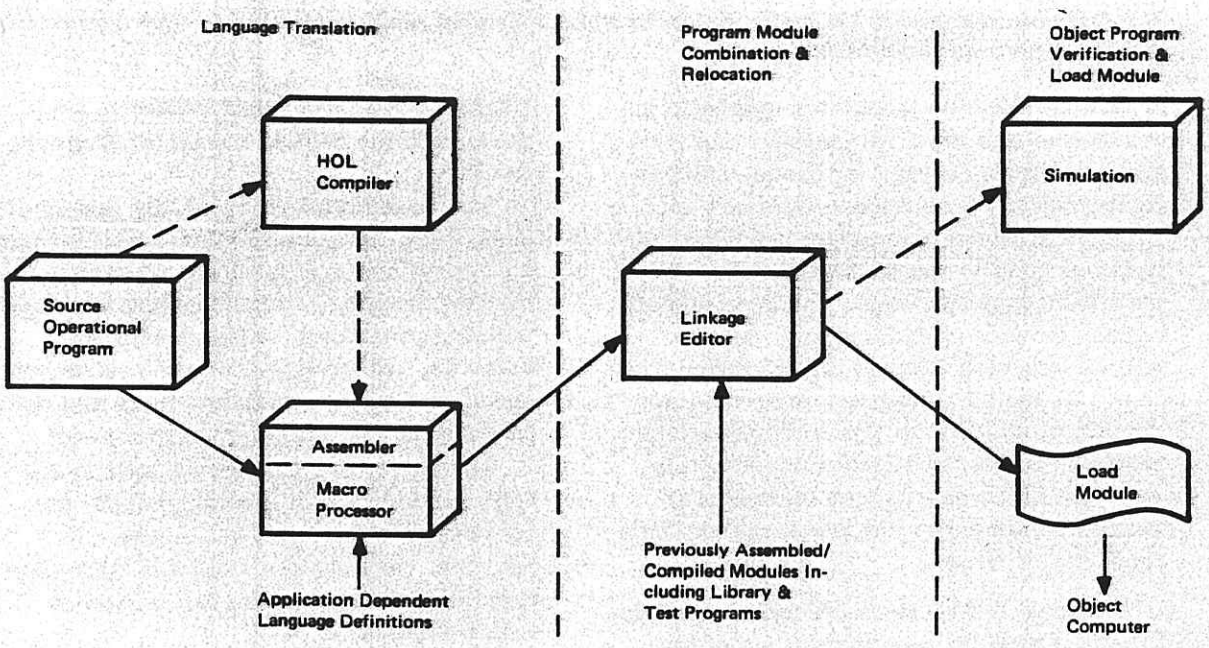
The Assembler, Linkage, and Simulator are the culmination of more than 15 years of aerospace software development. These routines incorporate important features derived from IBM System 360/370.

Development and delivery of comprehensive and fully documented support software for all System 4 Pi computers has resulted in refined innovations and techniques for assuring the quality and dependability of the software product. The support software incorporates the same spooling and paging techniques, the expanded macro assembly capability, and the documentation improvements that have been proven on numerous models of System/360 and System/370 and under multiple releases of operating system environments. The support software offers the integrity of design and maturity of concept demanded by large development efforts and is available for immediate use for software development.

The software provides complete facilities for modular operational program development and simulation using IBM System/360/370 equipment (minimum region size of 210K bytes).

The Assembler provides the programmer with symbolic language with which to generate a storage-efficient, high-throughput code. The Assembler language is similar in many respects to the IBM System/360 Assembler language, thus minimizing programmer training costs. Programming rules and techniques (such as modular programming, syntax, symbols and labels, source instructions, constants, comments, and subroutine linkages) are largely identical to the System/360 operations. The Linkage Editor allows a modular approach to program development, enabling the programmer to separately develop portions of each final load module prior to combining them to form the complete module. The Simulator, which provides for static and dynamic simulation of application programs, fully simulates the instruction set, interrupts (and associated effects on registers), main storage, program status words, and program timers. The interrelation of the support software is shown in the Data Flow diagram.

IBM is actively involved in developing Higher-Order Language (HOL)/Compiler facilities for System/4 Pi computers, and has developed a HOL Intermediate Language that encompasses FORTRAN, CMS-2, HAL, JOVIAL, SPL, and CLASP. A FORTRAN front-end compiler and a working code generator have been developed and demonstrated.



Support Software Data Flow

3.2 ASSEMBLER

Allows Programmer to Convert Symbolic Specifications to Machine Language

The Assembler designed to aid in efficient use of all CPU features, provides optimized program generation, documentation, and maintenance.

The Assembler translates symbolic instructions into machine language instructions, assigns storage locations, and performs auxiliary functions necessary to produce a relocatable object program suitable for input to the Linkage Editor. Inputs from magnetic tape, cards or direct access devices containing source code will be accepted through the operating system. Assembler-language programs may consist of up to three types of statements: machine-instruction, macro-instruction, and assembler-instruction statements. Source statements allow sequence and comment specification.

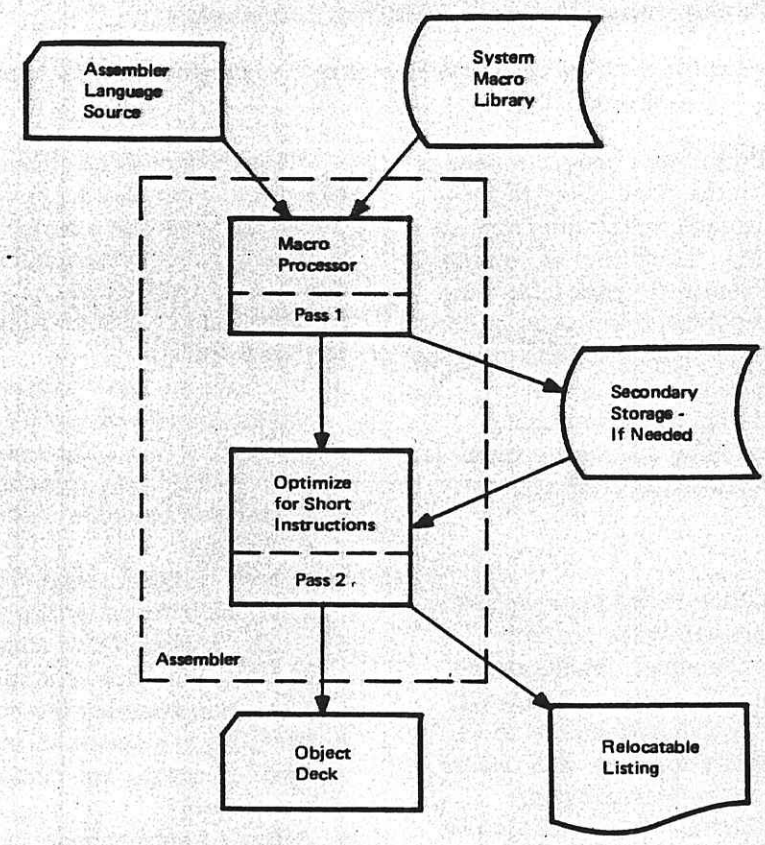
Machine-instruction statements are one-for-one symbolic representations of the machine instructions. The Assembler produces an equivalent machine instruction in the object program for each machine-instruction statement in the source program. Short format instructions are automatically selected by the assembler whenever possible.

Macroinstruction statements cause the Assembler to retrieve user-coded symbolic routines called macros, modify the routine according to the information in the macro instruction, and insert the modified routine into the source program for translation into machine language. Macros may be defined within the Assembler-language program or may be maintained on a system macro library. This powerful facility is comparable to the S-360 Assembler H macro facility.

The assembler program, by means of assembler-instruction statements, provides auxiliary functions to assist the programmer in checking, documentating and segmenting programs, defining data and symbols, controlling address assignments, generation of macro instructions and controlling the Assembler program itself.

The object programs produced by the Assembler are in a format suitable for relocation to any area of storage. In addition, they contain symbolic data used for program linkage and for reference within simulation.

The Assembler produces a thorough listing of the source program statements and the resulting object program statements for each source program assembled. Programmer control of the form, and content of the listing is provided. The Assembler also provides post-processor data following the assembled program listing. The post-processor data includes instruction and data symbols used in the assembled program, together with the core storage assignments and all references to those symbols. As a source program is assembled, it is analyzed for actual or potential errors in the use of the assembler language. Detected errors are indicated in the program listing.



Assembler Featurers:

1 — extensive symbolic assembler language
2 — user macro definition capability similar to the powerful S/360H level macro assembler
3 — facilities computer architecture changes
4 — allows for application dependent language expressions
5 — provides excellent base for compiler generated code
6 — generates truly relocatable code
7 — use of symbols, registers, labels, linkages, etc. largely similar to OS/360 counterparts
8 — provides auxiliary functions to assist in checking and documenting programs
9 — Supports decimal, binary, hexadecimal, and character data types
10 — extensive error diagnosis and reporting
11 — uses optimization techniques to generate efficient code
12 — generates thorough listings including:
- a — side-by-side source, generated machine instruction, and relative address
- b — assembler-generated statement number
- c — error messages
- d — symbol table with cross-reference information
- e — external symbol dictionary
- f — relocation dictionary

13 — affords independence in input/output device specifications.

Assembler Program Structure

3.3 LINKAGE EDITOR

Programmer Tool Permits Program Modules to be Written Independently

The Linkage Editor combines and relocates separately assembled program modules to produce an executable system.

The Linkage Editor is a program used in association with the Assembler to prepare machine-language programs from symbolic language. In addition to combining separately assembled programs, the Linkage Editor resolves references between these programs and prepares data for input to the Computer or the Simulator. It allows the user the option of assigning absolute target machine addresses when generating the loadable programs for execution.

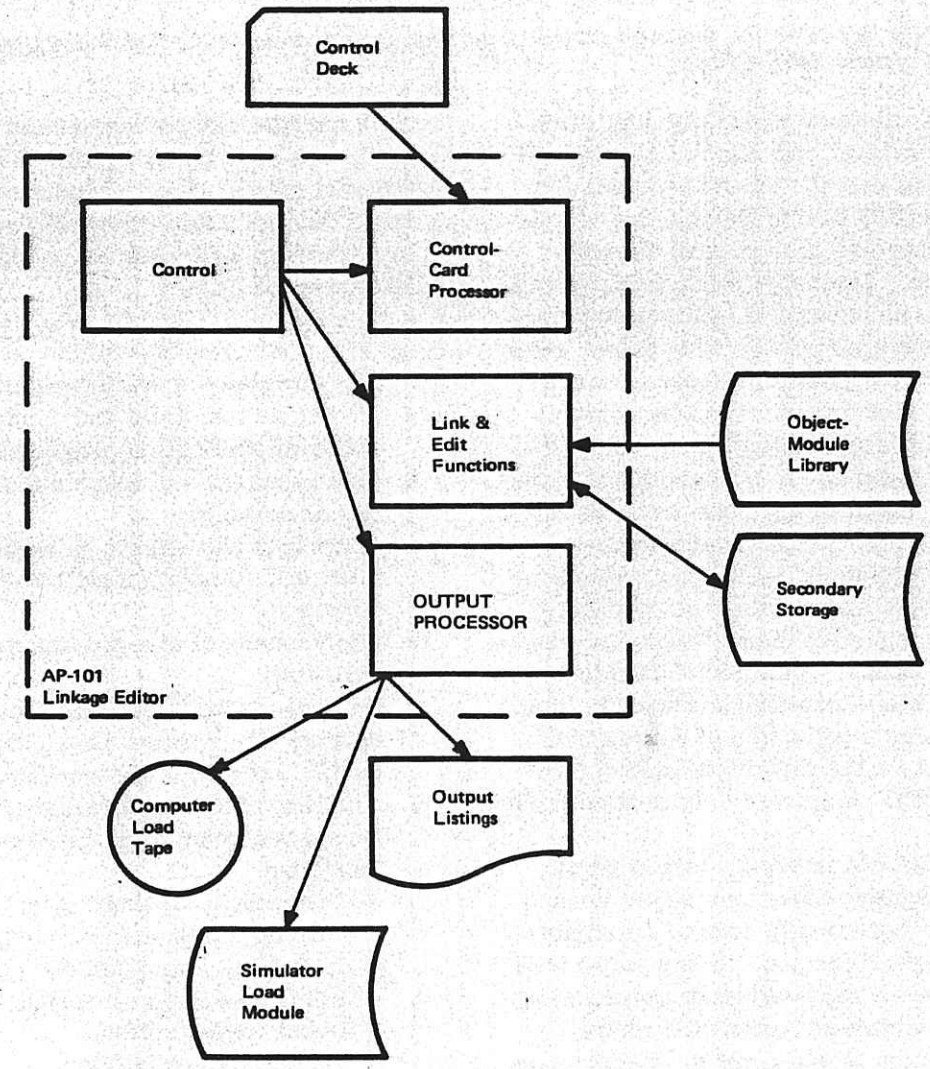
The Linkage Editor reads user-supplied cards (or card images) which control Linkage Editor processing. These cards define the load module content and layout, specify which optional functions are to be executed, and identify for inclusion one or more previously assembled program modules. Assembled modules can also be maintained in an object module library which is accessed by the Linkage Editor to improve processing speed.

The Linkage Editor provides, as output, a computer load module, a simulator load module including symbolic reference data, and a listing output. The standard listing output includes control card print, module map, entry points, and diagnostics. Optional printouts include an absolute listing and program statistics.

The Linkage Editor checks the input object modules and the specified load module layout for possible errors and prints diagnostics for any problems detected; e.g., unresolved references.

The Linkage Editor possesses the following features:

- Permits modular approach to program development and maintenance
- Fully resolves intermodule linkage among numerous relocatable modules
- Generates loadable and executable programs
- Allows user to assign absolute target machine addresses if desired
- Prints optionally an absolute assembly listing for entire load module
- Prints diagnostics for errors detected
- Permits replacement or rearrangement of component modules without reassembly
- Affords independence in input and output device specification
- Prints load module map information, including:
  - Absolute load address of each module
  - Module and program length
  - Identification of each module
  - Core-map layout of module entry points
  - Optionally an absolute machine code listing of the load module
- Provides program overlay capability.



Link Editor Program Structure

3.4 SIMULATOR

Provides an Alternate Environment for Program Verification and Debug

The Simulator is a powerful, thorough programming tool, used to evaluate the flight computer software on an IBM system 360/370 host.

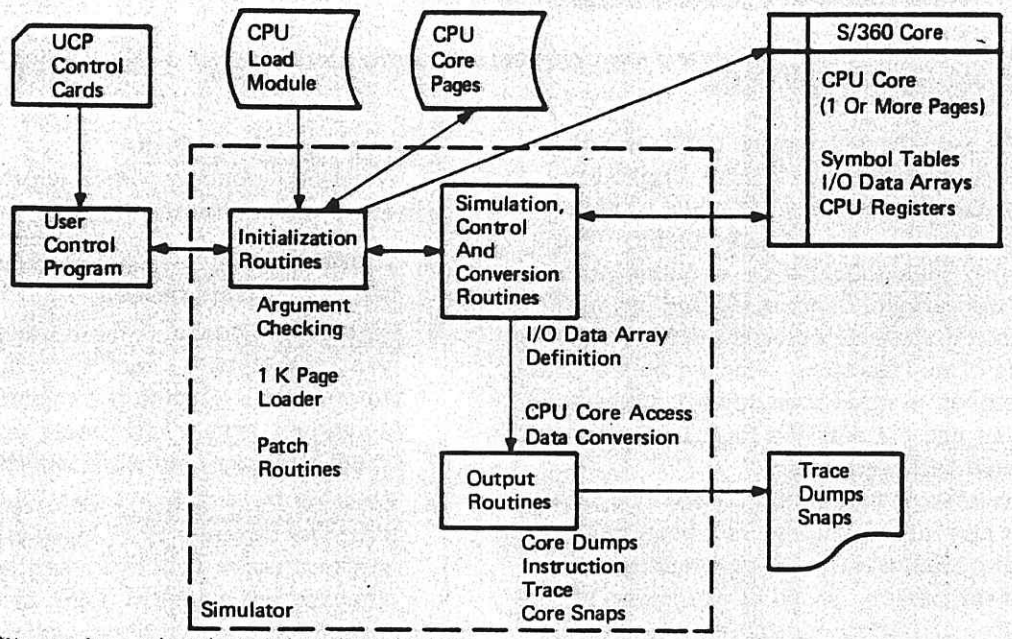
The Simulator functionally duplicates instruction-level operation of the Central Processing Unit (CPU) and provides for associated I/O device handling as well as allowing precise user control over the Simulation. Extensive debug facilities are provided, including full instruction trace, location stop, data snap, data patch, core dump, and diagnostic messages. Moreover, the instruction trace line contains detailed information that is suitable for processing and use in producing statistical reports. Included are such items as left- and right-hand symbols, mnemonics, specified registers, and updated condition code, carry, and overflow status bits. The user regulates the thoroughness, and hence the speed, of the Simulation to fit his needs. This is possible because the Simulator acts as a collection of subroutines available to a User-Written Control Program (UCP). A starter UCP is supplied by IBM.

The UCP can interpret user control cards, magnetic tape, or a direct access module to dynamically control the simulation throughout the run. If any error conditions are encountered during simulation, control is returned to the UCP where further action is determined. Returns are also generated for prespecified conditions, such as location stop or maximum simulation-time reached. Through the control cards, the user can dynamically instruct the UCP to continue simulation, respecify input conditions, or provide diagnostic printout. This level of control permits the user to tune the thoroughness of his simulation to match the state of his program.

The Simulator accepts the load module produced by the Linkage Editor as input. This module may be resident on either magnetic tape or a direct access device. Control and informational parameters are provided through subroutine calls or are passed in a common data area shared by the Simulator and the UCP.

The Simulator features include

- Provision for static and dynamic simulation of application programs
- Full simulation of instructions, both I/O and others
- Interrupts and effects on registers, memory, program status word, and timers
- Maintenance of elapsed simulated CPU time
- I/O simulation by means of user-defined data arrays, providing peripheral device independence
- Allowance for specification of all necessary simulation conditions, including
  - Program to be simulated
  - Starting locations for simulators
  - Initial and maximum CPU time
  - Error condition operation
  - Debug output options
  - Initial input condition
- Automatic paging of load module sections from disk to core reducing required S/360/370 region size.
- Expansion of instructions to an internal format for fast execution
- Provision for extensive debug facilities
- Symbolic, relative or absolute addressing
- Trace, dump, snap, and notification of all interrupts and error condition outputs.



*Note: Automatic paging technique allows simulation of large programs without increasing region requirements

Simulator Program Structure

3.5 OPERATIONAL READINESS SELF-TEST PROGRAM (STP)

The STP Provides Computer Operational Validation

The STP is an integrated software and microprogram monitoring function which provides computer operational readiness certification.

The Self-Test Program (STP) is co-resident with the Operational Flight Program (OFP), and can be invoked by the OFP executive on a periodic basis. The STP provides a Go/No-Go decision about CPU operational readiness, and saves information about detected failures. The status of malfunctions detected by the STP are stored in main memory for subsequent readout and for detailed fault isolation with external test equipment.

Division of functions between the program and microprogram has been made with careful consideration of core and microstore utilization, speed of execution, accessibility, and minimum interference to the OFP. The STP consists of the functional elements described in the following paragraphs.

Microprogrammed Tests

Functions uniquely tested via the microprogram include the following:

- Microinterrupts — Each microinterrupt latch is set and cleared.
- Real Time Clocks — the microprogram sets, increments, and reads the real-time clocks. Ordinary programming techniques cannot efficiently test the counters in a reasonable length of time.
- Floating Point Logic — Microprogramming the floating-point logic allows much quicker, more accurate, and more efficient testing since the logic is more accessible to the microprogram operational program.
- I/O Functions — The microprogram sets and clears interrupt latches.

3.6 FUNCTIONAL TEST PROGRAM (FTP)

The FTP Provides Computer Functional Checkout

The FTP is a comprehensive functional test program used for computer maintenance support and Acceptance testing.

The Functional Test Program (FTP), used in both computer checkout and Acceptance Test, tests and verifies all functions in the computer. In general, all data paths and functional circuits are exercised and their resultants are compared against expectant data.

The FTP, which utilizes the built-in test equipment (BITE) hardware, is designed to work with the Micro Programmed Test Set (MTS) and a test technician to detect and isolate failures. Detection coverage for the FTP is 99%. Isolation information is provided via an error log. With this information manual isolation is accomplished to the component level via a skilled technician using conventional troubleshooting techniques (scope, meters, etc.).

The five major modules of the FTP are:

- Supervisor — The FTP Supervisor five major tasks are:

Task	Function
1) Start Up	Provide initialization of FTP and CPU plus perform a basic instruction test.
2) Test Pass Control	Determine which test types (CPU, Memory, I/O, and/or Special) are to be executed on a per pass basis. This is an operator controlled function through AGE (MTS).
3) Error Logging	Builds an error log consisting of the first fifteen plus the last error that occurred.
4) Test Selection	Determine which test numbers, within a test type, are to be executed. Operator can modify selection via AGE.
5) Interrupt Handling	Provides interrupt identification of occurring interrupt and reports an error if interrupt is not allowed.

- CPU — The CPU test module, containing 79 separate tests, checks the proper operation of all instructions, along with register loading and data transferring capability. All addressing schemes are tested, including direct, indirect, indexing base, auto indexing, index modify, relative addressing, and extended addressing. Additional tests include register addressability and data transfer; condition code, carry, and overflow for all arithmetic and floating-point instructions; arithmetic interrupts and masks; logical bit handling; privilege instruction execution; privileged instructions in the problem state; PSW manipulation.
- Memory — The 15 memory tests check all functions related to main storage memory operations and the CPU memory interface. Tests are designed to assure that every data, parity, and storage protect bit (exclusive of three small areas; PSA, error log & test location) can be set to both the "1" and "0" state. The storage protect, instruction monitor, and parity interrupt functions are verified for correct operation.
- I/O — The 26 I/O tests check all functions related to I/O operations with the MTS operating as the device for I/O testing. The channel reset command capability is checked for proper operation. The program counters are checked for read and load capability. Direct I/O and DMA are tested with various data patterns and checked for proper results.
- Special — This module provides a location for tests which are not clearly CPU, Memory or I/O tests. The special test module contains three tests for testing the WAIT state, interrupt priority system, and operation of the microcode 'DETECT' instruction.

75-A97-001

Section 4

FUNCTIONAL DESCRIPTION

4.1 CPU FUNCTIONAL OPERATION

Basic Timing and Control Cycles Are Maintained by CPU and External Clock Control Logic

Functional operation and control of CPU subunits is maintained by microprogram control.

The Central Processor Unit Data Flow diagram depicts the functional areas within the CPU and the external interface. The following sections describe the functional operation of each area.

The Timing and control subunit generates all of the CPU and external clocks for basic timing and control functions. The normal CPU cycle is 200 ns and consists of two 100-ns segments. For Main Storage operations the CPU cycle is extended and synchronized with the Main Storage data return.

The Microprogram Control subunit contains the microprogram storage, the microprogram storage address logic, the microprogram storage output data register and the decode and control logic. The storage consists of up to 2,048 48-bit control words. During every CPU cycle a control word is read from microstorage to serve as the controlling word or microinstruction for the subsequent CPU cycle. Decoding of each individual microinstruction specifies the operation of the various functional subunits in the CPU. Included in the microinstruction is the microstorage address specification from which the next control word is to be obtained. This address may be directly specified in the microinstruction or it may be specified by a combination of status indicators.

The bus control subunits direct the routing of data from numerous internal and external sources onto the system input busses. This routing is principally accomplished by two multiplexers. The first or Master Bus Control Unit Multiplexer provides inputs to the CPU, Main Storage or I/O and selects the data from either the CPU, Main Storage, or I/O sources. The multiplexer is also capable of halfword reversal. The second or Intra CPU Control Unit Multiplexer provides inputs to the CPU Input Data Bus and selects the data from the Master Bus Control multiplexer or one of the Intra CPU Bus Control Registers.

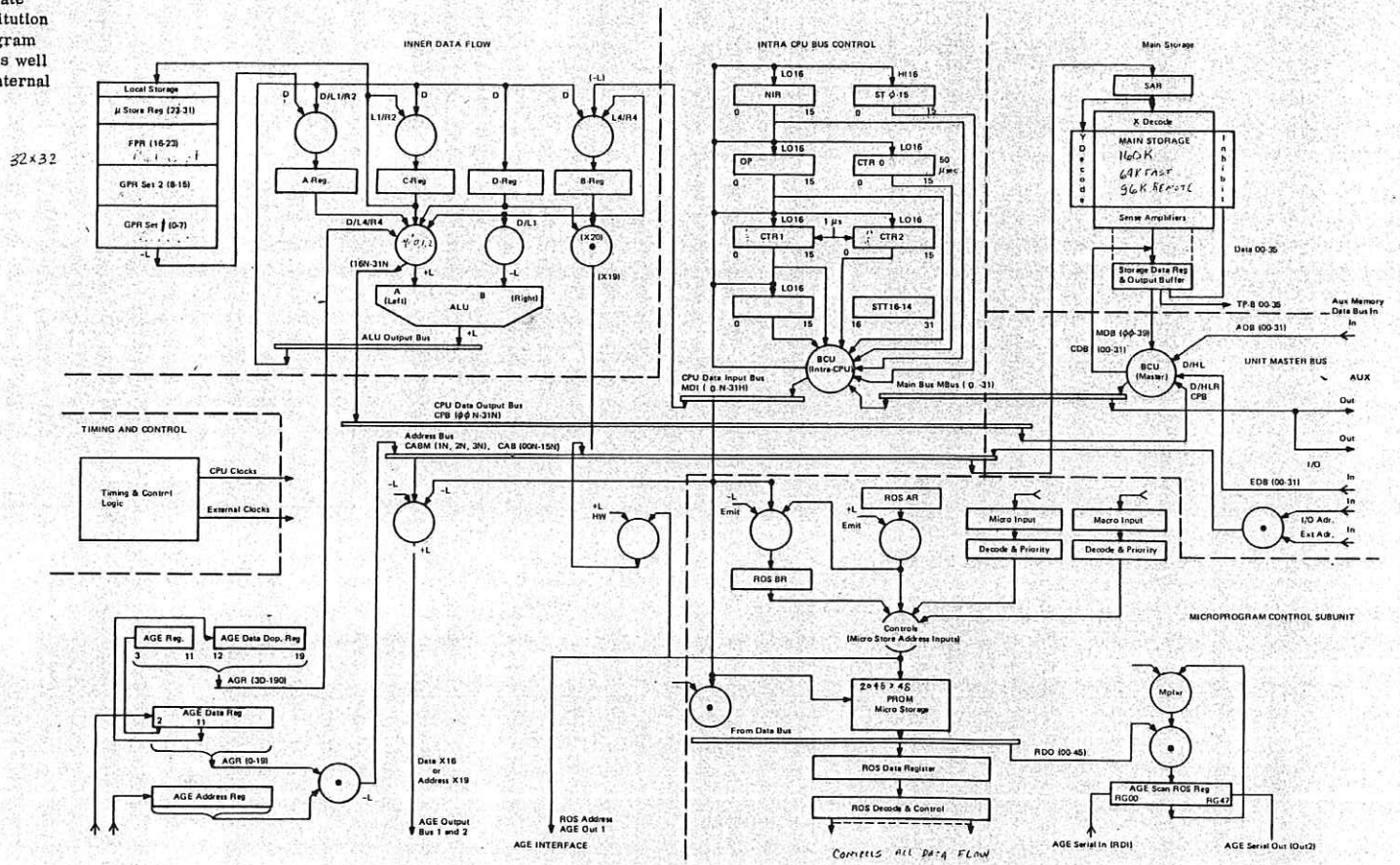
The elements of the inner data flow are the working registers, multiplexers and the function unit used by the microprogrammer to perform individual logic and arithmetic operations. Also included is the Local Store Array consisting of thirty-two fullword registers. The microprogram uses two sets of eight registers as fixed-point general purpose registers and 1 set of eight registers as floating point general purpose registers, while the other set of eight registers is reserved for microprogramming functions. The entire inner data flow is under control of the microprogrammer. Operation of the inner data flow may be explicitly specified (within constraints of the encoding) or conditionally specified allowing complex, data dependent procedures to be implemented by a single microinstruction.

The main storage interface provides control of an asynchronous main store of up to 128K fullwords, of which 32K is fast-access and the remaining 96K is remote. The interface also provides support for Read-Compute-Write operations and storage protection on individual halfword locations.

The Input/Output interface supports a bidirectional, thirty-two bit, parallel data interface. The I/O interface operating mode is either an externally initiated transfer or a program initiated transfer. The externally initiated transfer, designated Direct Memory Access (DMA), functions as a main storage cycle steal or fully independent main storage port. Thus, other than the interference in accessing main storage, the CPU processing functions are not affected by concurrent DMA operations. Program initiated transfers (PCIO) are macroinstructions utilizing CPU operations.

The AGE interface provides all the necessary control and data paths for loading and testing the CPU. In addition to

allowing direct access for main storage reading and writing, the AGE separate words interface allows direct substitution of microinstructions into microprogram control for execution by the CPU, as well as display of most registers CPU internal registers.



Central Processor Unit Data Flow

75-A97-001

4.2 MICROPROGRAM CONTROL

All CPU Functions Are Controlled By Microprogram Subunit

Microinstruction decode controls execution of the operations required during each program macroinstruction.

Microprogram Control Subunit

The Microprogram Control subunit contains the PROM storage, the ROS data register (ROSDR), the ROS address register (ROSAR), and the ROS backup register (ROSBR). A normal CPU cycle is 200 ns, and for each cycle a new 48-bit wide ROS data word is accessed from the PROM storage and loaded into the ROSDR. This 48-bit microword in the ROSDR is a microinstruction which is used to control all CPU functions. The PROM storage is capable of holding 2048 such microinstructions which make up all the macroinstructions and interrupt microsubroutines. While the current microinstruction in the ROSDR is executing a CPU function, this microinstruction is also generating the ROS address of the next microinstruction to be executed.

The PROM address bus is a 12-bit bus. The ROSAR is used to retain the high 6 ROS address bits until changed by microinstruction specifications. ROS address bits 6 through 9 can also be retained by the ROSAR when requested by the microinstruction. The last two bits are decode outputs from the microinstruction in the ROSDR. The ROSBR is used to save the next ROS address of an interrupt microsequence. When an interrupt occurs, Mux 1 selects the interrupt logic for generating the next ROS address. At the same time, the ROS address generated by the ROSAR and the ROS decode is loaded into the ROSBR via Mux 2. When the microsubroutine taken by the interrupt is completed, a microinstruction can specify the contents of the ROSBR as the next ROS address via Mux 1. This allows return to the microprogram sequencing which was occurring prior to the interrupt.

Microprogram Instruction Format

The microprogram instruction format consists of five main sections: 1) data flow controls, 2) local store control, 3) miscellaneous and emit control, 4) microprogram sequence control, and 5) parity and enable selected interrupt bits.

ROSDR bits 0 through 10 control the inner data flow function. The first three ROSDR bits (0 through 2) controls the left ALU input multiplexer. ROSDR bits 3 through 5 specify whether fullword, halfword byte, logical, or shifting operations are to be performed. The remaining ROSDR bits (6 through 10) specify inner data flow operations and destination of the ALU output.

ROSDR bits 11 through 16 specifies the local storage operation desired. Bit 11 defines a local storage read or write operation, while bits 12 through 16 specify indirectly the local storage address to be accessed. The contents of local store are further described in the Inner Data Flow section.

ROSDR bits 17 through 28 specifies how the Miscellaneous 1 field, Miscellaneous 2 field, and the Emit fields are to be used. Miscellaneous 1 field binary codes (ROSDR bits 17 through 20) from 0000 to 0111 use the 8-bit field ROSDR bits 21 through 28 as data. For example, Miscellaneous 1 binary code 0001 specifies that ROSDR bits 21 through 28 are to be gated directly to upper byte of the 'B' register. Miscellaneous 1 field binary codes (ROSDR bits 17 through 20) from 1000 to 1111 use ROSDR bits 25 through 28 to further define the operation while the remaining emit field bit (ROSDR bits 21 to 24) may be used as a Miscellaneous 2 field which specifies operations independent of the Miscellaneous 1 field. The Miscellaneous 1 field may specify the following operations:

1. Load the upper byte of the 'B' register via the 8-bit Emit field
2. Load set or clear status registers and interrupts

3. Specify the high 6 ROS address bits for the next microinstruction
4. Indicate the end of the instruction operation
5. Perform various I/O operations
6. Specify Floating Point operations
7. Specify memory read & write operations
8. Modify (via ROSDR bits 25 through 28) the lower 4 bits of the upper half of the ROS address (Address bits 6 through 9).

The Miscellaneous 2 field is used only when the last four operations (5 through 8) of the Miscellaneous 2 field are specified. The Miscellaneous 2 field with the above restrictions may specify special tasks such as disable and enable interrupts, request Master bus, etc.

ROSDR bits 29 through 43 of the Microprogram instruction specify the low-order 6 address bits of the next microinstruction to be executed. It is possible to perform a 64-way branch with this method. The microinstruction address field is 12 bits wide. The upper 6 address bits (0 through 5) can be altered by a Miscellaneous 1 field specification; otherwise these address bits remain the same when sequencing microinstructions. The Miscellaneous 1 field can modify all 6 address bits via the high and low emit fields in the microinstruction or address bits 2 through 5 via the low emit field of the microinstruction.

ROS address bits 6 through 9 are taken directly from microinstruction ROSDR bits 30 through 33, when bit 29 of the microinstruction is "0". When microinstruction bit 29 is "1" and bits 30 through 33 contain the code 0001, indirect branching occurs. For example, the contents of operation code register bits 4 through 7 are used as bits 6 through 9 of the following microinstruction address.

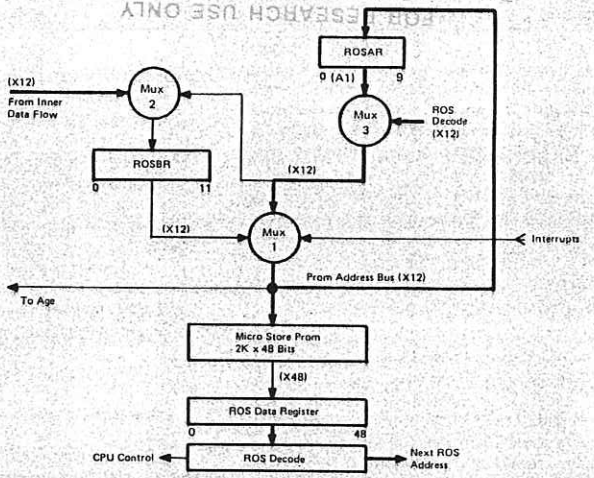
The A-branch field of the microinstruction ROSDR (bits 34 through 38) specifies ROS address bit 10 of the following microinstruction address. This address bit can be explicitly specified to be 1, 0, or set conditionally. Thirty tests can be specified using the A-branch field. For example, if the A-branch field contains the code 01101, then address bit 10 will be set to 1, if bits 12 through 15 of the operation code register are all zeros.

Similarly, the B-branch field of the microinstruction (bits 39 through 43) specifies ROS address bit 11 of the following microinstruction address. Again, this can unconditionally be set to 0, 1, or set conditionally, depending on the result of the additional 30 tests which can be specified. For example, code 01111 specifies that the contents of bit 15 of the operation code register are to be used as bit 11 of the microprogram address.

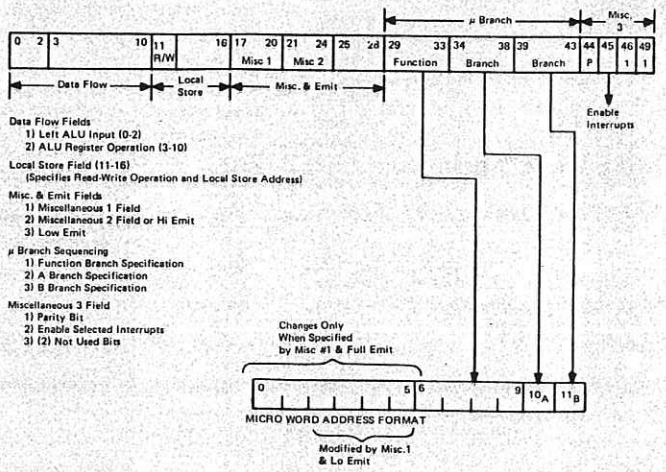
Microinstruction ROSDR bit 44 is used as a parity bit such that the 48 bits always have odd parity. Microinstruction ROSDR bit 45 is used to enable the following interrupts:

1. Main store parity error
2. I/O interface data parity error
3. Extended memory data parity error
4. Address specification
5. Store protection violations
6. Microinstruction parity error.

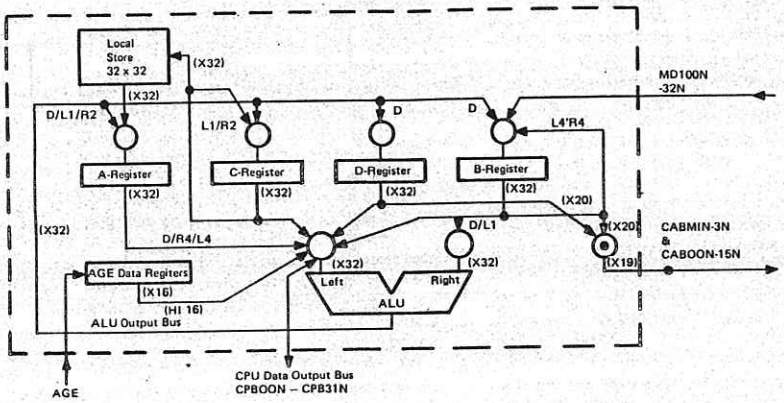
This interrupt control is necessary because there are periods during microinstruction operations that interrupts are not desired. An example of this would be during microinstructions that specify memory operations. The last two ROSDR bits 46 and 47 are spares and are set at an uplevel.



CPU Microprogram Control Subunit



PROM Data Register (μ-Word) Format



Inner Data Flow

4.3 INNER DATA FLOW

Focus Of All Processor Activity

Performs all basic computational and logic operations required for instruction processing.

The inner data flow performs all the logical and arithmetic operations called for by the microprogram. It is fully independent and can continue processing of complex or iterative functions without interference from or to the other portions of the data flow. Significant elements of the inner data flow are as follows:

- Arithmetic and Logic Unit (ALU)
- Working registers
- Miscellaneous multiplexers and gates
- Local Store.

The entire flow is 32 bits wide and supports divide and two-bit-at-a-time multiply. A 32-bit simple function, such as add, requires a single CPU cycle. Multiply of 32 bits by 32 bits is completed in 16 CPU cycles and gives a full 64-bit product. Divide of 64 bits by 32 bits yielding 32-bit quotient and 32-bit preliminary remainder uses 32 CPU cycles.

In addition to supporting the parallel operations used in the multiply and divide steps, other parallelism is built into the flow such as:

- Independent control of Local Store operation
- Independent control of the Left ALU Input Multiplexer
- Coupled shifts providing 64-bit shifting.

As the focal point of the data flow, the inner data flow is the source or destination of the three principal CPU busses.

- The CPU Data Input Bus (MDI 00N-31N) is used primarily to supply operands to the inner data flow.
- The Inner Data Flow Output Bus (CPB 00N-31N) is drawn directly from the Left ALU Input Multiplexer.
- The Address Bus (CABM7N-3N, CAB 00N-15N) is a dot-bus whose status may come from AGE, I/O, or the inner data flow.

Innner Data Flow - Working Registers

The inner data flow has four 32-bit working registers and a 16-bit AGE input register. The four working registers are used for temporary working storage during instruction execution and are accessible to the microprogrammer.

The A register is the Local Store output buffer and is loaded by each Local Store read. In addition, the A register may be loaded with the output of the ALU either directly, or shifted left one-bit position, or shifted right two-bit positions. The shifted loads occur only in response to shift, multiply, or divide commands. The A register is clocked whenever it is selected as a destination either explicitly or implicitly (shift, multiply, divide, local store reads). Normally, the entire 32 bits of the A register are clocked simultaneously. However, when the microinstruction specifies CMISC 0 (bits 17-20 = 1001; bits 25-28 = 0000), a clock is generated for the most significant eight bits of the A register. This clock will clear those bits of the A register provided the A register has not been otherwise selected as a destination.

The B register is the buffer for the Main Data Input Bus. The B register may also be loaded with the ALU output or with its own contents shifted left or right four bit positions. The shift occurs only when an ABL4 or ABR4 shift is specified (microinstruction bits 3-7 = 01000 or 01001 respectively). The B register, like the A register, is clocked whenever the B register is explicitly selected or implicitly selected as a destination. The most significant eight bits are also clocked whenever CM1SC 0 is specified.

The B register is an implicit destination:

- For ABL4 and ABR4 shifts
- For reads of the internal registers (microinstruction bits 17-20 = 1111, bit 28 = 1) and Program Store
- For emits* (bits 17-20 = 00001)
- For decodes (bits 17-20 = 1010, bits 25-28 = 10 xx when xx ≠ 11).

The C register is the Local Store input buffer. It may be loaded with the ALU output, and it may be shifted internally left one bit position or right two bit positions. During multiply and divide, the C register is coupled with the A register. In multiply, the C register initially contains the multiplier and receives the product bits as they spill out of the low end of the A register. In divide, the C register initially contains the least significant half of the dividend and feeds the A register as the partial remainders develop, while receiving the quotient bits.

The D register contains the instruction counter, condition code, and some system masks. It may be loaded with the ALU output. Certain portions (bits 16-19) are separately loadable at LSTEX** time with the appropriate condition codes, and are settable and clearable separately (microinstruction bits 17-20 = 010x, bits 21-24 = 111x). The D register is normally clocked when specified as an explicit destination. However, when RNOK is specified (microinstruction bits 17-20 = 1010, bits 25-28 = 1100) only the most significant 16 bits of the D register are clocked — this feature is normally used in executing a macroinstruction branch. The D register is also clocked as an implicit destination when microinstruction bits 3-10 = 111xx00xx.

The AGE input register is a serial-in, parallel-out register and is loadable only by the serial AGE interface.

Inner Data Flow - Multiplexers

In addition to the shift multiplexers on the inputs to the A, B, and C registers, there are three multiplexers for selecting left and right ALU inputs and input to the Address Bus.

The left ALU input multiplexer is directly controlled by bits 0-2 of the microinstruction. One of eight possible inputs is selected by the control decode function.

The AGE input register is also an input to the multiplexer when selected during AGE input operations or when W/LE-RBHC is specified (microinstruction bits 17-20 = 1111, bits 25-28 = 1101). The left ALU input multiplexer is the source of the Inner Data Flow Output bus.

The right ALU input multiplexer selects the B register contents directly or shifted left one bit position. In certain circumstances, the multiplexer is degated to provide an input of zero. The multiplexer is normally set direct. It will shift left one only in the case of certain arithmetic DWIMS*** (microinstruction bits 3-5 = 111 and bit 8 = 1) and in the case of certain multiply steps. The multiplexer will be degated only in the case of the decode DWIMS (microinstruction bits 3-5 = 111, bits 8-10 = 00x and then only if the processor opcode does not indicate an instruction with explicit immediate data.

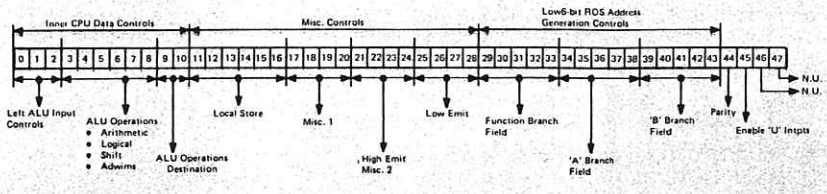
The Address Bus multiplexer provides gating of processor addresses from either the B register of the D register. The multiplexers will be on only if the CPU has priority assigned by the timing logic (GCPAD is up). Bits 1-15 of the Address Bus receive B register bits 1-15 or D register bits 1-15, depending on the microinstruction bit 27. Bits M1, M2, M3, and 0 will have D register 24-27 if microinstruction bit 27 is zero and D register bit 0 is one, or D register 28-31 if microinstruction bit 17 is one and B register bit 0 is zero.

Inner Data Flow - Arithmetic and Logic Unit

The arithmetic and logic unit performs binary addition and subtraction and logical operations as specified by the microinstructions. All logical operations are 32 bits and the repertoire of 16 operations uses all the capability available in the LSI arithmetic and logic modules.

Addition and subtraction are normally 32 bits with two additional high-order bits implemented for multiply. The carry into the low-order position of the ALU is generated to support two's complement subtraction (certain multiply and divide steps, arithmetic and fullword arithmetic with carry specified). The propagation of carries can be broken between bit position 7 and 8 and between 15 and 16 to provide halfword and byte add or subtract. These byte and halfword operations do not preserve unused portions but only guarantee that extraneous carries will not propagate into the computation or that required carry-ins appear. Byte operation may only be specified by microinstruction bits 3-5 = 110. During byte operations, normal carry propagation from bit position 8 is blocked and, if microinstruction bit 8 = 1, a carry is inserted into bit position 7. During halfword operations, normal carry propagation from bit position 16 is blocked and a carry is inserted into bit position 15 if the microinstruction specified halfword arithmetic and microinstruction bit 8 = 1 or if the microinstruction specifies a decode DWIM.* These functions result from halfword arithmetic operations with microinstruction bits 3-5 = 101 and from arithmetic DWIM decodes with microinstruction bits 3-5 = 111 and bits 8-10 = 000 or 100.

*DWIM - denotes a complex microprocedure especially designed to assist machine performance. They provide multiple or conditional operations not otherwise specifiable by a single microinstruction.



012

000 — A Register to Left ALU Input
001 — B Register to Left ALU Input
010 — C Register to Left ALU Input
011 — D Register to Left ALU Input
100 — A Register Left 4 to Left ALU Input (fills per shift opn)
101 — A Register Right 4 to Left ALU Input (fills per shift opn)
101 — Same as 'A' with B 0-7 inserted in positions 0-7 of ALU
111 — D Register to Left ALU Input with Low-order 1

3 4 5 6 7 8 9 10

0 0 x x x x x x — Logical Operation
0 1 x x x x x x — Shift Operation
1 0 0 x x x x x — Arithmetic Fullword Operation (i.e., — Hot one at bit 31)
1 0 1 x x x x x — Arithmetic Halfword Operation (i.e., — Hot 1 at bit 15; no prop. 16 to 15)
1 1 0 x x x x x — Arithmetic Byte Operation (i.e., — Hot 1 at bit 7; no prop. 8 to 7)
1 1 1 x x x x x — Miscellaneous Arithmetic Operation

Logical Operation

0 0 0 0 0 0 x x — B Reg complemented to ALU output
0 0 0 0 0 1 x x — Left ALU "and" B Reg complemented to ALU output
0 0 0 0 1 0 x x — Left ALU complemented "or" B Reg complemented to ALU output
0 0 0 0 1 1 x x — Left ALU "exclusive ored" with B Reg to ALU output
0 0 0 1 0 0 x x — Left ALU "or" B Register complemented to ALU output
0 0 0 1 0 1 x x — Left ALU to ALU output (used automatically when nothing is specified)
0 0 0 1 1 0 x x — Ones to ALU output
0 0 0 1 1 1 x x — Left ALU "or" B Reg to ALU output
0 0 1 0 0 0 x x — Left ALU complemented "and" B Reg complemented to ALU output
0 0 1 0 0 1 x x — Zeros to ALU output
0 0 1 0 1 0 x x — Left ALU complemented to ALU output
0 0 1 0 1 1 x x — Left ALU complemented "and" B Reg to ALU output
0 0 1 1 0 0 x x — Left ALU complemented "exclusive ored" with B Reg to ALU output
0 0 1 1 0 1 x x — Left ALU "and" B Reg to ALU output
0 0 1 1 1 0 x x — Left ALU complemented "or" B Reg to ALU output
0 0 1 1 1 1 x x — B Reg to ALU output

ALU Operation (3, 4, 5, = 100, 101, 110)

0 0 x x x — Left ALU is added to zero
0 1 x x x — ALU output = 1 less than 2's complement subtraction of Left ALU minus B Register
1 0 x x x — Left ALU plus B register
1 1 x x x — Left ALU minus 1

Hot 1 Operation

x x 0 x x — No carry into the adder
x x 1 x x — Carry into the adder as determined by fullword, halfword or byte operation

Destination

x x x 0 0 — ALU Output to A Register
x x x 0 1 — ALU Output to B Register
x x x 1 0 — ALU Output to C Register
x x x 1 1 — ALU Output to D Register

6 7 8 9 10

Misc Arithmetic Operations

1 1 1 x x 0 0 0 — If Op is explicit Immediate (Op Code 10110); B → C
If not Exp Imm and Long Instruction; D+ 1 → D unless FP overflow is on and not masked
Left ALU (Line 3) must be coded "D" for proper operation
(D + 1 → D is inhibited for ship op code — 11111)
1 1 1 x x 0 0 1 — 1) If Long Instruction; Bump IC unless FP overflow is on and not masked
Left ALU (Line 3) must be coded "D" for proper operation (360 operation)
1 1 1 x x 0 1 0 — Multiply Step [A±B(0, D, L1) $\xrightarrow{R2}$ A (Fill with ALU -1 & -2); C (R2) (Fill A30, A31)]
1 1 1 x x 0 1 1 — Divide Step [A±B $\xrightarrow{L1}$ A Fill Co; C $\xrightarrow{L1}$ C Fill Quotient] (Add = C31 B00)
(Precede divide step with A → A micro-order)
1 1 1 x x 1 0 0 — Left ALU plus B Reg L1 → B Reg [Halfword Add]
1 1 1 x x 1 0 1 — Left ALU plus B Reg L1 → A Reg
1 1 1 x x 1 1 0 — Left ALU minus B Reg L1 → A Reg

Shift Operation

0 1 0 0 0 x x x — ABL4 — A & B registers as a pair shifted Left 4 (See Note 1)
0 1 0 0 1 x x x — ABR4 — A & B registers as a pair shifted Right 4 (See Note 2)
0 1 0 1 0 x x x — AL4 — A Register shifted Left 4 (See Note 1)
0 1 0 1 1 x x x — AR4 — A Register shifted Right 4 (See Note 2)
0 1 1 0 0 x x x — ACL1 — A & C registers as a pair shifted Left 1 (See Note 3)
0 1 1 0 1 x x x — ACR2 — A & C registers as a pair shifted Right 2 (See Note 3)
0 1 1 1 0 x x x — AL1 — A Register shifted Left 1 (See Note 3)
0 1 1 1 1 x x x — AR2 — A Register shifted Right 2 (See Note 3)

Note 1 Left ALU Input opn must be AL4
Note 2 Left ALU Input opn must be AR4
Note 3 Left ALU Input opn must be A

Fill

0 1 x x x 0 0 x — Fill with zeros
0 1 x x x 0 1 x — Fill with sign (right shifts only)
0 1 x x x 1 0 x — Fill as Rotate (right shifts only)
0 1 x x x 1 1 x — Fill with Stat Bits (See spill/fill chart)

Spill

0 1 x x x x x 0 — Spill is lost
0 1 x x x x x 1 — Spill is Stat Reg Bits 12-15 (See spill/fill chart)

Microprogram Instruction Word Fields

Notes

*Emits — A micro function which furnishes 8 bits of immediate data directly from the microinstruction (bits 21-28).

**LSTEX — Instruction completion time represented by a LSTEX DWIM which causes automatic condition code update, interrupt enabling and a conditional request for an instruction fetch from main storage.

***DWIM — Denotes a complete microprocedure especially designed to assist machine performance. A DWIM provides multiple or conditional operations not otherwise specifiable by a single microinstruction.

Inner Data Flow - Local Store

The Local Store consists of thirty-two 32-bit register locations implemented in monolithic read/write devices. The Local Store contains all the general-purpose registers and floating-point registers visible to the macroprogrammer, as well as eight locations available only to the microprogrammer.

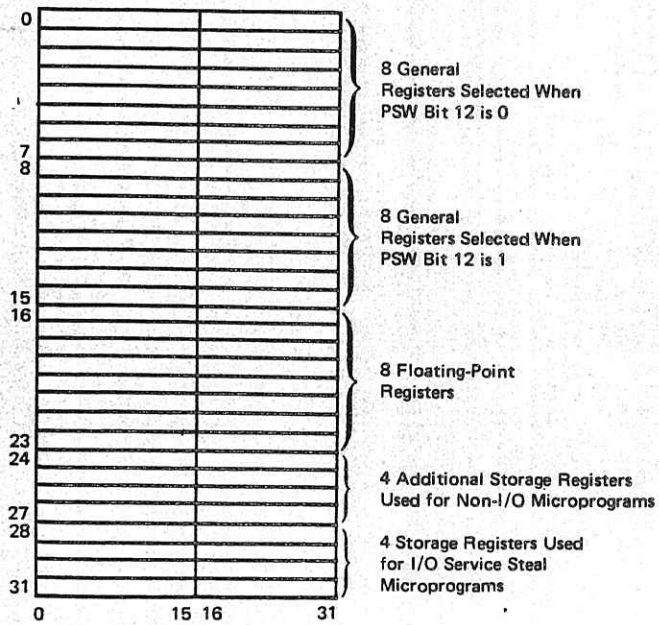
Within a CPU cycle, any location in the Local Store may be read from or written into. When read, the contents of the selected locations are loaded into the A register. When written, the local store location is loaded with the contents of the C register. If the A register is specified to receive data, other than Local Store, during a Local Store operation, the two pieces of data will be logically OR'ed in the A register.

Control of the Local Store consists of:

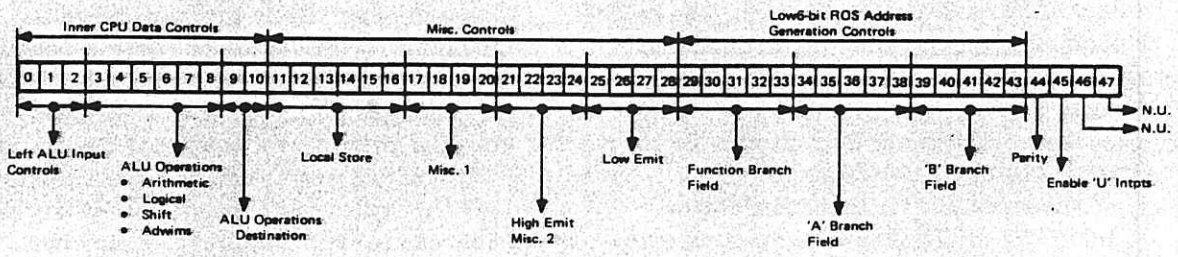
- Four address lines
- Two-chip selects
- A write enable.

The address lines are common to all the devices comprising Local Store. The two-chip selects each control 16 of the 32 register register locations in the Local Store. Chip selects are both inactive when no Local Store operation is specified, otherwise, one will be active depending on whether or not the register location lies in the general purpose registers. The write enable is common to all the Local Store devices and is active in the second half of the processor cycle on all writes to Local Store.

The control lines respond to bits 11 to 16 of the microinstruction. Bit 11 specifies whether a read or a write is to be done. Bits 12-15 indicate how the Local Store address is to be formed. A code of all zeros in bits 11-15 is the only valid no-op; all other codes may cause the ORing described earlier. The 31 possible address codes are configured to rapidly extract register addressing bits from the current macroinstruction. As implemented the codes support the processor architecture and furnish direct microcode addressing of the eight register locations reserved to the microprogrammer.



Local Store Assignments



11

0 xxxxx R -- Read

1 xxxxx W -- Write

Non-Zero Condition of Balance of Field Specifics LS Addressing

	12	13	14	15	16	
x	0	0	0	0	1	Register address = Stat 19, PSW 12, Op bits 5-7
x	0	0	0	1	0	Register address = Stat 19, PSW 12, Op bits 5-6, 1
x	0	0	0	1	1	Register address = Stat 19, PSW 12, Op bits 11-13
x	0	0	1	0	0	Register address = Stat 19, PSW 12, Stat bits 0-2
x	0	0	1	0	1	Register address = Stat 19, PSW 12, Op bits 13-15
x	0	0	1	1	0	Register address = Stat 19, PSW 12, 0, Op bits 14-15
x	0	0	1	1	1	Register address = Stat 19, PSW 12, 0, Op bits 14-15 (Read zero's to A Reg for 014, 014 = 11)
x	0	1	0	0	0	Register address = Stat 19, PSW 12, Op bits 13-14, 1
x	0	1	0	0	1	Use with DEC code (Misc B codes 8, 9, and 10)
x	0	1	0	1	0	
x	0	1	0	1	1	Register address = Stat 19, Stat bits 8-11
x	0	1	1	0	0	Register address = Stat 19, Op bits 8-11
x	0	1	1	0	1	Register address = Stat 19, Op bits 8-10, 1
x	0	1	1	1	0	Register address = Stat 19, Op bits 12-15
x	0	1	1	1	1	Register address = Stat 19, Stat bits 0-3 (Read zero's to A Reg for S0-3 = 0000)
x	1	0	0	0	0	Register address = Stat 19, 0, Op bits 5-7
x	1	0	0	0	1	Register address = Stat 19, 0, Op bits 5-6, 1
x	1	0	0	1	0	Register address = Stat 19, 0, Op bits 13-15
x	1	0	0	1	1	Register address = Stat 19, 0, Op bits 13-14, 1
x	1	0	1	0	0	Register address = Stat 19, 0, Op bits 9-11
x	1	0	1	0	1	Register address = Stat 19, 0, Op bits 9-10, 1
x	1	0	1	1	0	
x	1	0	1	1	1	Register address = Stat 19, Stat Bits 8-11
x	1	1	0	0	0	Register address = Stat 19, 1000
x	1	1	0	0	1	Register address = Stat 19, 1001
x	1	1	0	1	0	Register address = Stat 19, 1010
x	1	1	0	1	1	Register address = Stat 19, 1011 } Generally for CPU Use
x	1	1	1	0	0	Register address = Stat 19, 1100
x	1	1	1	1	0	Register address = Stat 19, 1110
x	1	1	1	1	1	Register address = Stat 19, 1111

4.4 DATA BUS CONTROL

Control Is Exercised Through Two Multiplexers

Master or main bus multiplexer controls main data flow to or from the CPU and the intra bus multiplexer controls the transfer into the CPU inner data flow.

Data flow control is exercised by two sets of 32 by 8-1 multiplexers. The multiplexers are controlled by an enable gate and three coded input controls. These coded controls are generated by a combination of hardware and microcode described in the following sections.

The main bus control multiplexer (BC MUX) multiplexes the main memory data output bus (MDB), the computer data bus (CDB), the auxiliary memory data output bus (ADB), and the external data bus (EDB) together to form the main bus (MBUS). Each of the four buses may be multiplexed direct (bits 0-15 to MB 0-15 and bits 16-31 to MB 16-31) or reverse (bits 0-15 to MB 16-31 and bits 16-31 to MB 0-15). The multiplexer outputs are always enabled. In the event of a default state, the control state is code 110, the computer data direct is selected. The control codes are listed in the multiplexer control code table.

MULTIPLEXER CONTROL CODES FOR MAIN BUS MULTIPLEXER

C_0	C_1	C_2	Function
0	0	0	Memory Data Direct
0	0	1	Memory Data Reverse
0	1	0	External Data Direct
0	1	1	External Data Reverse
1	0	0	Auxiliary Memory Data Direct
1	0	1	Auxiliary Memory Data Reverse
1	1	0	Computer Data Direct
1	1	1	Computer Data Reverse

Control of this multiplexer is either CPU microprogram or I/O direct memory access hardware. DMA data requests and CPU request bus micro-orders are prioritized within the CPU. Simultaneous requests are resolved in favor of the I/O but once priority is granted, a higher priority request will not cause a release of the lower priority. All I/O requests for bus priority and all CPU requests for bus priority not accompanied by a request I/O will initiate a memory cycle. External priority gates the direct memory access address to the computer address bus. CPU priority and memory operation gates either the D-register (Instruction Address) or B-register (Data Address) to the address bus in accordance with the microcode. The SAR clock for memory operations, or the latched request I/O for non-memory operations strobes the multiplexer control information into multiplexer control latches.

Once the memory control latches are strobed, they will remain latched until:

1) The external data request is deactivated for DMA operations.
2) The memory restore cycle is initiated for read-compute-write or store CPU originated memory operations.
3) The delayed memory data valid control deactivates for read CPU originated memory operation.
4) The next CPU cycle if a non-memory, non-latched bus request was issued. This is the normal function when CPU data must be reversed, or loaded into counters, Stat Register, P-Register, or AGE output.
5) The microcode commands a reset if a latched non-memory bus request was issued. This is the normal function for PCIO operations.

Data on the main bus is then available simultaneously to the I/O interface, the auxiliary memory interface, the main memory interface, and the inner bus control multiplexer (IBC MUX).

Data transfers into the CPU inner

data flow operations take place, is controlled by the 8-1 IBC MUX. The complement output of this multiplexer generates the memory data in bus (MDI) which feeds the B-register, AGE data out multiplexer, and the microprogram link register (ROSBR) multiplexer. The true output forms the data input to the various ancillary registers and counters.

Control of the multiplexer is via CPU microprogram and micro-assist hardware (DWIM) with an AGE override. Control of bits 0-15 is separate from control of bits 16-31 as illustrated in the control code tables for the intra bus multiplexer.

MULTIPLEXER CONTROL CODES FOR BITS 0–15 OF INTRA BUS MULTIPLEXER

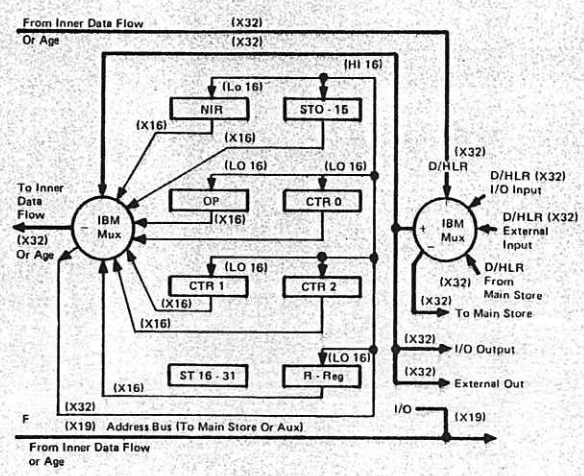
H_E	H_1	H_2	H_3	Function
0	X	X	X	Zeros to MDI 0-15.
1	0	0	0	Zeros to MDI 0-9, OP 8-13 to MDI 10-15.
1	0	0	1	Stats 0-15 to MDI 0-15.
1	0	1	0	Spare - Default ones to MDI 0-15.
1	0	1	1	Zeros to MDI 0-4, stats 5-15 to MDI 5-15.
1	1	0	0	MBUS 0-15 to MDI 0-15.
1	1	0	1	ROS Data Reg 21-28 to MDI 0-7, zeros to MDI 8-15.
1	1	1	0	MBUS 16 to MDI 0-15.
1	1	1	1	Spare - Default ones to MDI 0-15.

*The default state is 0XXX - Zeros to MDI 0-15.

MULTIPLEXER CONTROL CODES FOR BITS 16-31 OF INTRA BUS MULTIPLEXER

L_A	L_E	L_1	L_2	L_3	Function
0	0	X	X	X	Zeros to MDI 16-31.
0	1	0	0	0	Zeros to MDI 16-19, MBUS 20-31 to MDI 20-31.
0	1	0	0	1	Zeros to MDI 16-19, NIR 4-15 to MDI 20-31.
0	1	0	1	0	Zeros to MDI 16-19, STAT 4-15 to MDI 20-31.
0	1	0	1	1	Zeros to MDI 16-19, OPREG 4-15 to MDI 20-31.
0	1	1	0	0	Zeros to MDI 16-19, CNTRO 4-15 to MDI 20-31.
0	1	1	0	1	Zeros to MDI 16-19, CNTR1 4-15 to MDI 20-31.
0	1	1	1	0	Zeros to MDI 16-19, CNTR2 to MDI 20-31.
0	1	1	1	1	Zeros to MDI 16-19, PREG 4-15 to MDI 20-31.
1	1	0	0	0	MBUS 16-31 to MDI 16-31.
1	1	0	0	1	NIR 0-15 to MDI 16-31.
1	1	0	1	0	STAT 0-15 to MDI 16-31.
1	1	0	1	1	OPREG 0-15 to MDI 16-31.
1	1	1	0	0	CNTRO 0-15 to MDI 16-31.
1	1	1	0	1	CNTR1 0-15 to MDI 16-31.
1	1	1	1	0	CNTR2 0-15 to MDI 16-31.
1	1	1	1	1	PREQ 0-15 to MDI 16-31.

*The default state is 00XXX - Zeros to MDT 16-31.



Bus Control of Data Flow

4.5 TIMING AND CONTROL

20-MHz Crystal Oscillator

The timing source is divided to form a free running 10-MHz clock which in turn is used to establish the basic two clock system.

The basic CPU timing is developed from either an internal 20-MHz crystal oscillator or an external 20-MHz source. The timing source is then divided to form a 10-MHz free-running clock which becomes the primary clocking element in the unit.

The CPU timing consists of a two clock system denoted CP1 and CP2. These two clocks are developed from the 10-MHz clock and are nominally 100 ns each, although each clock may be extended in 100-ns increments in response to system activity.

A CPU cycle is defined as the period from the rise of CP1 to the rise of the next CP1. A CPU cycle consists of strobing the content of the addressed micro storage location into the ROS Data Register (RSDR) with the rise of CP1, performing the specified micro-operation, and strobing the result into the appropriate register with the rise of the next CP1 clock.

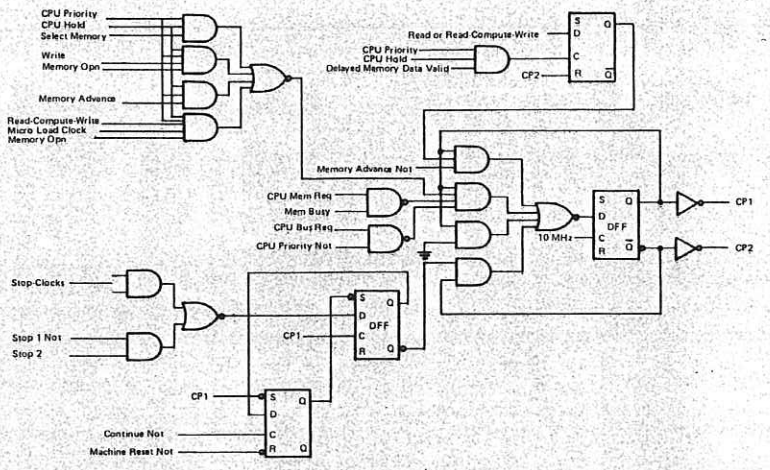
Interval timers are driven from a 1-MHz clock which is generated from division by five of a 5-MHz gated clock. The 5 MHz is normally free running, but when the CPU timing is stopped by support equipment command, the 5 MHz will stop as well. This stop preserves the real time relationship of the interval timers and instruction timing.

CPU Clocks

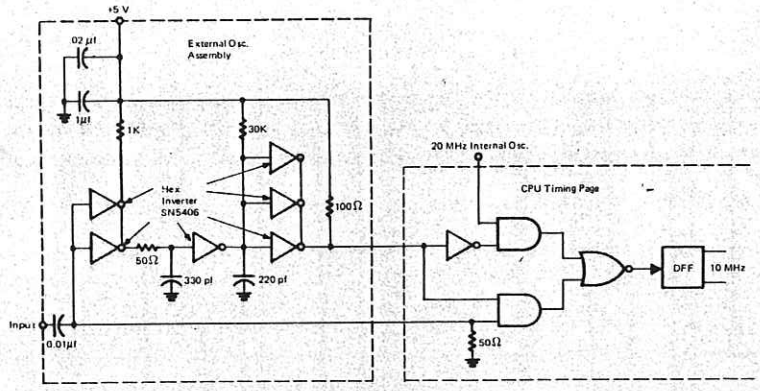
The CPU clocks CP1 and CP2 are generated from the Q and $\overline{Q}$ outputs of a D-Type Flip-Flop. With CP1 active, the CPU cycle may be extended indefinitely, in increments of 100 ns, through external control via the AGE control lines XSTP XSTP2N.

The external equipment command of XSTP1N=0, XSTP2N=1 commands an immediate stop and a command of XSTP1N=1, XSTP2N=0 commands a delayed stop upon the processor execution of a "Stop Clocks" micro order. The immediate stop is used for a single CPU cycle stop on micro address compare operation, whereas the delayed stop is used for a single instruction cycle stop on program address compare operation. When the cycle is stopped with CP1 active, initiating a pulse on the continue AGE control line (XCONT) restarts the timing.

With CP2 active, the CPU cycle is stretched, in increments of 100 ns, to synchronize the CPU to the asynchronous machine elements (main store and input/output). When the required asynchronous event occurs, the timing is restarted at the next 100 ns quantum. The table shows the start and restart conditions for CP2 active.



CPU Clock Generator



Input Requirements:

Frequency	19 to 20 MHz
Duty Cycle	45% Min., 60% Max.
Transition Times	(10% to 90%) 10 NS Max.
Amplitude (Peak-to-Peak)	3.5 V Min, 5.0 V Max.
Input Zo	0.01 µf in Series With 50Ω
Input Noise	0.4 V Max. Across 50Ω

CPU External Clock

CPU CYCLE STOP/RESTART CONDITIONS

CP2 Stop Conditions	CP2 Restart Conditions
CPU Main Bus Request Not CPU Priority	CPU Priority Granted
CPU Main Bus Request CPU Memory Request Memory Busy	Memory Not Busy
CPU Memory Select CPU Wait for Data Read or Read-Compute-Write	Memory Advance → Inactive
CPU Memory Select CPU Wait for Data Write	Data Load Clock → Inactive
Read-Compute-Write CPU Micro Load Clock	Data Load Clock → Inactive

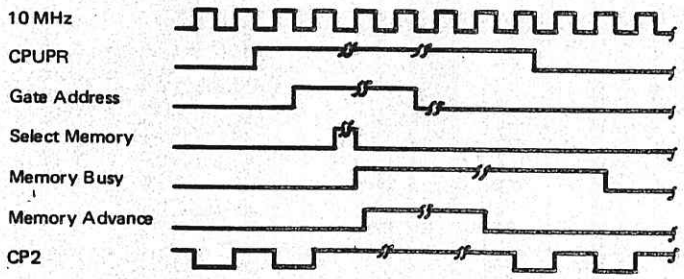
Memory Interface Timing

The interface with main storage is an asynchronous "handshaking" type of interface which can be activated by either a CPU micro order, an I/O Direct Memory Access, or a support equipment command. The CPU and Direct Memory Access sequences to main storage are essentially common. The operation is initiated by a Main Bus request, since the data is transferred over this bus. When priority is granted, the address is gated to the computer address bus (CAB), either internal or external SELECT MEMORY is generated and held until MEMORY BUSY is returned by the memory. When MEMORY ADVANCE is returned by the memory the SELECT MEMORY sequence is deactivated.

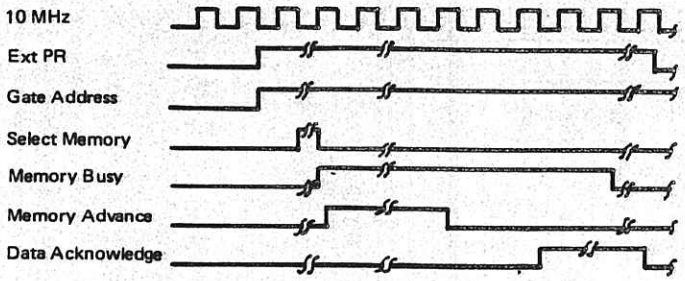
For CPU Memory Read operations, the CPU clock is restarted by the deactivation of MEMORY ADVANCE and for DMA Read Operations, the DATA ACKNOWLEDGE is generated by the deactivation of the MEMORY ADVANCE. Parity is checked 50 to 150 ns later.

For CPU Memory Store operations, the presence or absence of the store-protect bit is checked by the deactivation of MEMORY ADVANCE, the 100 ns DATA LOAD CLOCK (DLC) is generated 150 to 250 ns later and the CPU clock is re-started when DLC goes inactive. For DMA Store operations, the presence or absence of the store-protect bit is checked by the deactivation of MEMORY ADVANCE, the 100 ns DATA LOAD CLOCK is generated 150 to 250 ns later and DATA ACKNOWLEDGE is generated when DLC goes inactive.

The Read-Compute-Write operation is utilized only by the CPU with the sequence being similar to the CPU Read operation followed by a DATA LOAD CLOCK.



CPU Memory Read Interface and Timing Sequence



DMA Memory Write Interface and Timing Sequence

Main Bus Prioritization

The MAIN BUS provides the principal data path between the various elements of the CPU which consist of the main storage, auxiliary storage, I/O and, inner data flow. Data flow on the Main Bus is controlled by bus priority logic consisting of two flip-flops denoted as CPU Priority (CPUPR) and External Direct Memory Access Priority (EXTPR). Inputs to the flip-flops are strobed by the free running 10-MHz clock and establish the priority tree such that if a DMA request is pending it will always be granted unless CPUPR is active or the Disable DMA latch is set. Once either CPUPR or EXTPR is active, it will remain active until the data transfer is completed.

BUS PRIORITY CONDITIONS

EXTPR Active	EXTPR Inactive	CPUPR Active	CPUPR Inactive
DMA Priority (GP1) DMA Data Request Not Disable DMA Not CPUPR	DMA Data Request→Inactive Address Exception	CPU Bus Request Not EXTPR Active Not EXTPR Active Conditions Not GP1 Unless Disable DMA	Non-Memory Operation Micro Transfer Routine Complete Memory Read Operation Memory Advance→Inactive Memory Store Operation Memory Data Load Clock→Inactive Memory Read-Compute-Write Memory Data Load Clock→Inactive

4.6 MAIN STORAGE

Extended Performance/Modular Core Memory

Interchangeable 8K by 18 bit pluggable modules.

The Space Shuttle main storage consists of two, separate Extended Performance/ Modular Core Memories (EP/MCM). One of these memories, having a storage capacity of 40K words by 36 bits, is located in the CPU LRU. The other memory with a storage capacity of 24K words by 36 bits is located in the IOP LRU. Both memories communicate directly with the CPU. This section will summarize the characteristics of the CPU memory. A more detailed functional description of the EP/MCM can be found in the following sections. Those features unique to the IOP memory are covered in the IOP Functional Description.

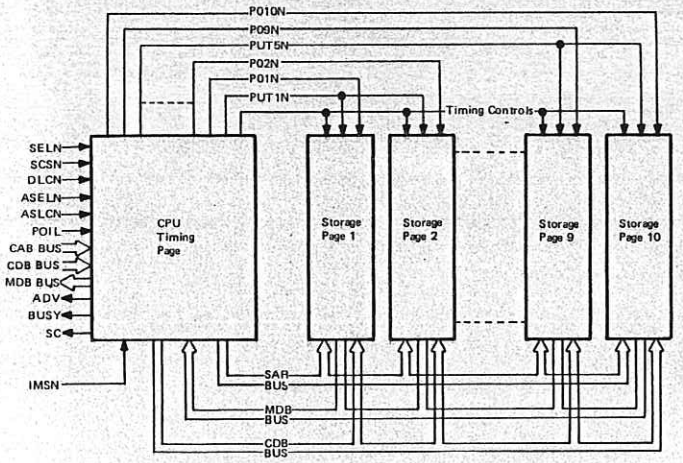
The EP/MCM is a 2-1/2 D organized ferrite core memory having a basic modularity of 8K words by 18 bits. This 8K by 18 bit pluggable module is called a storage page. The storage pages are constructed so as to permit unfolding, thereby, exposing the core array which is contained on the inside of the page. The two sides of the page are similar, with each providing 8K by 9 bits of storage capacity. The outside of the storage page contains buffer and control logic, address drivers, a diode decode matrix, temperature controlled voltage regulators, sense amplifiers, a storage data register (SDR) and the power switches.

Two of the 18-bit, 8192 word storage pages, in conjunction with a timing page, yield a memory capacity of 8K words by 36-bits. The CPU memory is expandable in 8K by 36-bit increments up to a maximum of 40K words within the CPU memory backpanel.

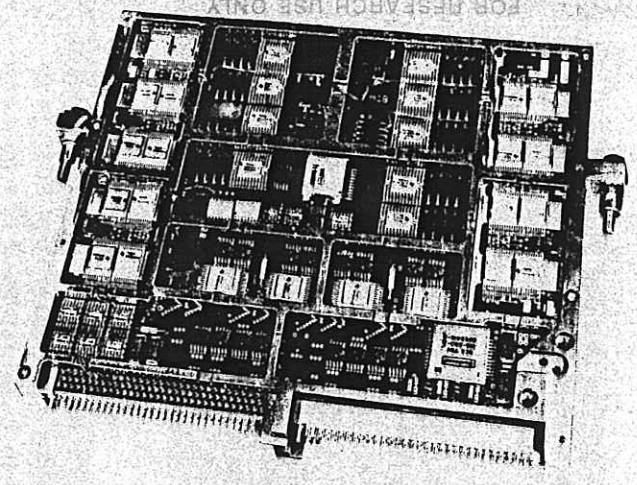
CPU MEMORY PERFORMANCE CHARACTERISTICS

Organization	2-1/2D
Storage Element	8/14 Wide-Temp. Ferrite Core
Electronics	Monolithic
Modularity	8,192 Words by 36 Bits
Addressability	Halfword/Fullword
Operational Modes	Read/Restore, Split Cycle Store
Temperature Range	-55° C to 95° C Rail
Required Voltages	+12V, +5V, -5V
Interface	TTL
Storage Capacity	40K - Fullwords
Cycle Time	800 ns Read/Restore
	1.0 μs Read/Modify/Write
Access Time	375 ns Max.
Power Up Time	470 ns
Power Down Time	20 μs
Power (1)	48.2W 2 Storage Pages Operational
	18.8W 2 Storage Pages Standby
	7.2W 6 Storage Pages Quiescent
	8.4W 1 Timing Page
	82.6W
Weight (2)	24.7 lb
Volume (2)	599 in^3

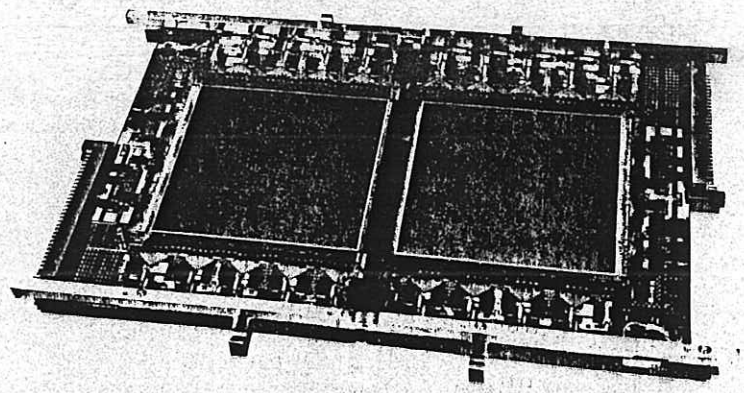
Notes: (1) 50% Fullword Duty Cycle, 50% I/O Data
(2) Includes Backpanel & Timing Page



CPU Memory Block Diagram 40K x 36 Bits



EP/MCM Storage Page (Folded View Showing Outer Surface)

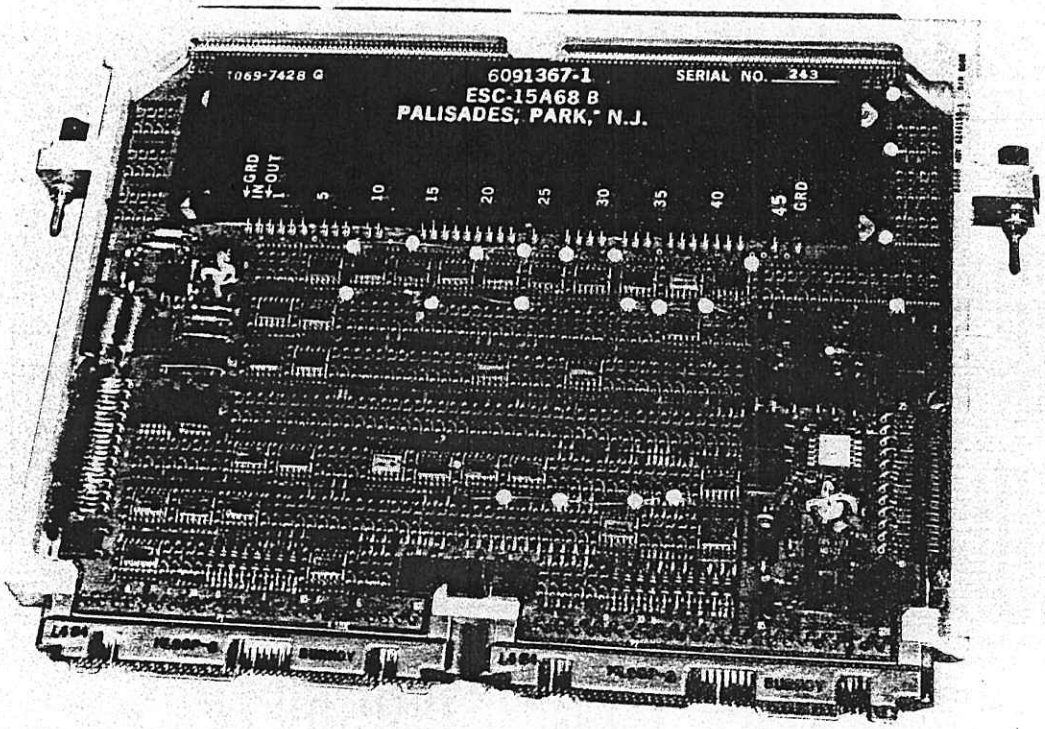


EP/MCM Storage Page (Unfolded View Showing Inner Surface with Core Array)

75-A97-001

The CPU timing page is similar to the standard 4 Pi logic pages, except that the height has been increased to allow space for mounting a delay line, also two connectors have been added to the top of the page for increased I/O capability. The timing page interfaces with the storage pages via the bottom two connectors on the page which plug into the memory backpanel. All CPU/memory interface signals are through the top two connectors of the timing page. This interface is standard TTL. A description of each interface signal follows:

- SELN - This signal is used to initiate a CPU memory cycle and to load the CPU Storage Address Register (SAR).
- SCSN - This signal is used to perform read/modify/write operations. When this signal is at a down level the restore portion of a memory cycle is inhibited. Return of this signal to an up level initiates the write portion of the cycle.
- DLCN - This signal is used to load data into the Storage Data Register (SDR). DLCN is issued only in conjunction with the SCSN signal.
- ASELN - This signal performs the same function as SELN with the exception of loading data into the SAR.
- ASLCN - This signal is used to load data into the SAR and must be issued prior to the ASELN signal.
- POIL - This signal is used to inform the CPU memory of an impending power transient condition. When POIL is at an up level the initiation of a memory cycle is inhibited.



CPU Memory Timing Page

- SC - This signal is issued by the CPU memory only after receipt of a POIL signal and only when a memory cycle is not in progress.
- IMSN - This signal, which originates in the CPU power supply, is used to clamp the memory address drivers in an off state during power transient conditions. IMSN may be issued only following a POIL-SC sequence. The IMSN signal also performs the memory reset function.
- ADV - This signal is issued by the CPU memory to signify that the SDR has been loaded with the addressed halfword or fullword.
- BUSY - This signal is issued by the CPU memory to signify that a memory cycle is in progress.
- CAB BUS - The CPU CAB BUS consists of 18 lines that determine memory addressability. The function of each bit is described in the section covering the memory timing page. The CAB lines must be stable prior to the issuance of SELN or ASLCN.
- CDB BUS - The CDB BUS consists of 36 lines that define halfword or fullword data to be stored in the CPU memory. The CDB lines must be stable for the duration of the DLCN signal.
- MDB BUS - The MDB BUS consists of 36 lines that contain halfword or fullword data just read from the CPU memory. Information on the MDB lines is valid only after the fall of ADV.

The CPU memory operates at a cycle time of 800 ns for a read operation and 1.0 us for a store operation. Maximum read access time is 375 ns. These times apply when the selected portion of memory is in a standby state. If the selected portion of memory is in a quiescent or low power state an additional 470 ns is required to power up to the standby state.

The EP/MCM timing page serves as a logical interface between the memory and the CPU as well as providing all the necessary timing and control signals for the storage pages. Functional elements contained on this page are: the timing generator, driver control logic, the storage address register (SAR), the power switch control logic and the voltage reference (V_{ref}) circuit.

The timing generator consists of a 470 ns delay line tapped at 10-ns increments, a delay line driver circuit, and high-speed (54S series) TTL gates. A memory cycle is initiated when a select command is received from the CPU. This starts a pulse propagating down the delay line. A cycle is divided into read and write portions. The write portion is accomplished by recycling a pulse through the delay line. This recycle operation is delayed in a read/modify/write mode. The total time required to complete the read and write portions of a cycle during the read/regenerate mode is 800 ns.

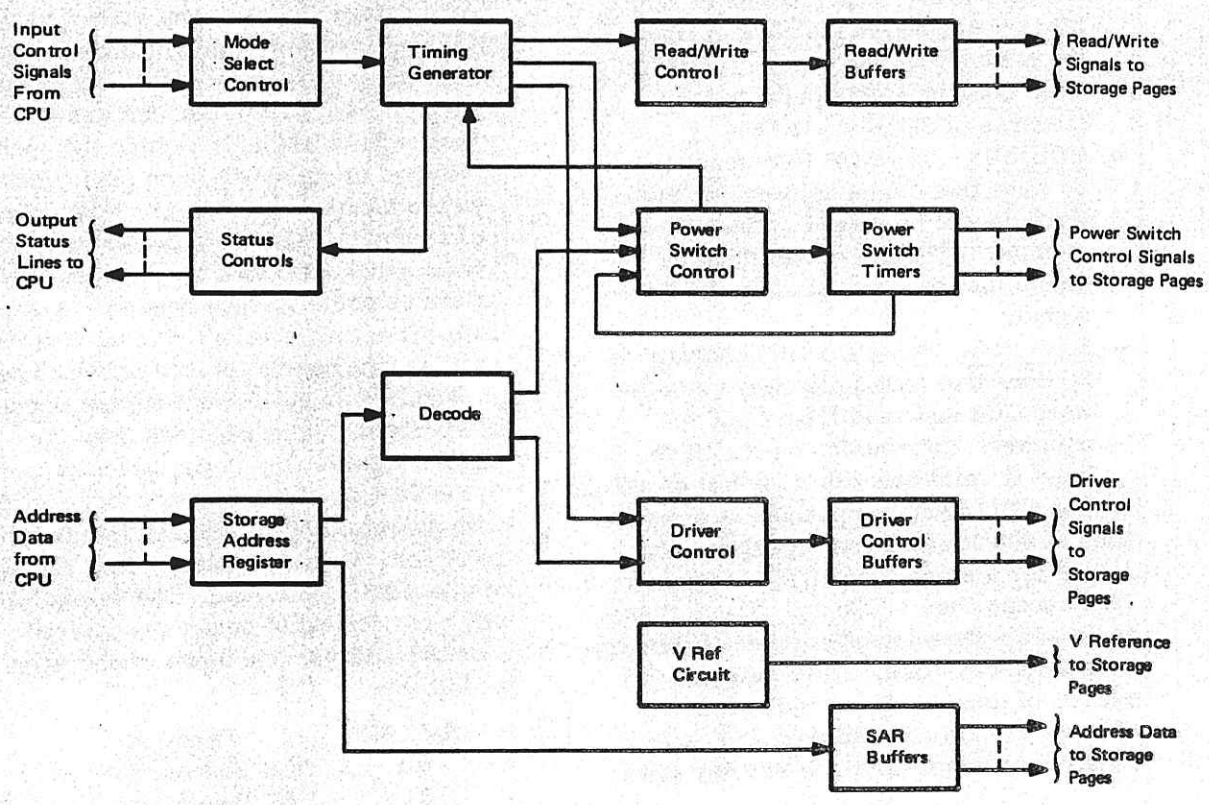
The driver control logic includes a decode stage and an output buffer stage which are made of a mix of Schottky and high-speed TTL gates. The decode stage receives inputs from the SAR and provides outputs which determine the operation (halfword or fullword) to be performed and the location in memory. The output buffer stage generates a power on (PO) signal which activates the address drivers on the selected storage page(s). A separate PO signal is supplied to each storage page so that control is on a halfword basis.

The SAR is an 18-bit register consisting of standard, high-speed and Schottky edge-triggered flip-flops. Schottky is used in the section of the register associated with driver control logic and the X-driver selection. Data is placed in this register via the SAR load clocks. The output SAR bits are decoded to select the desired memory address and mode of operation as follows:

SAR Bit	Function
M1	32K Boundary
0	16K Boundary
1	Full Word Selection
2-6	X Address Selection
7-14	Y Address Selection
15	Halfword Selection
C0	HW/FW Mode Selection

Power switch control logic consists of resettable single shots; one for each pair of storage pages, and a decode and timing control section. The single-shots determine the amount of time a given pair of pages (fullword) will remain powered up. This time is nominally set at 20 s. When a single-shot times out, the associated pair of storage pages revert to a low power or quiescent state. Powering up from a quiescent state to a standby state requires a 470-ns delay at the front end of a normal memory cycle, to allow decode and stabilization of voltages on the selected storage pages. This delay is accomplished by an additional pass through the delay line prior to the normal read and write portions of the cycle. During this first pass all timing and control signals to the storage pages are clamped. Following this 470-ns delay, an artificial select pulse (DSELN) is generated which initiates the normal read/write cycle. A comparison is made upon receipt of select (SELN) from the CPU to determine if the addressed pages are in a quiescent or a standby state and, hence, whether or not the 470-ns delay is required.

The V_{ref} circuit is a precision voltage source which supplies a -2.1-V reference to the sense amplifiers on the storage page. This voltage establishes the threshold detection level of the amplifier.



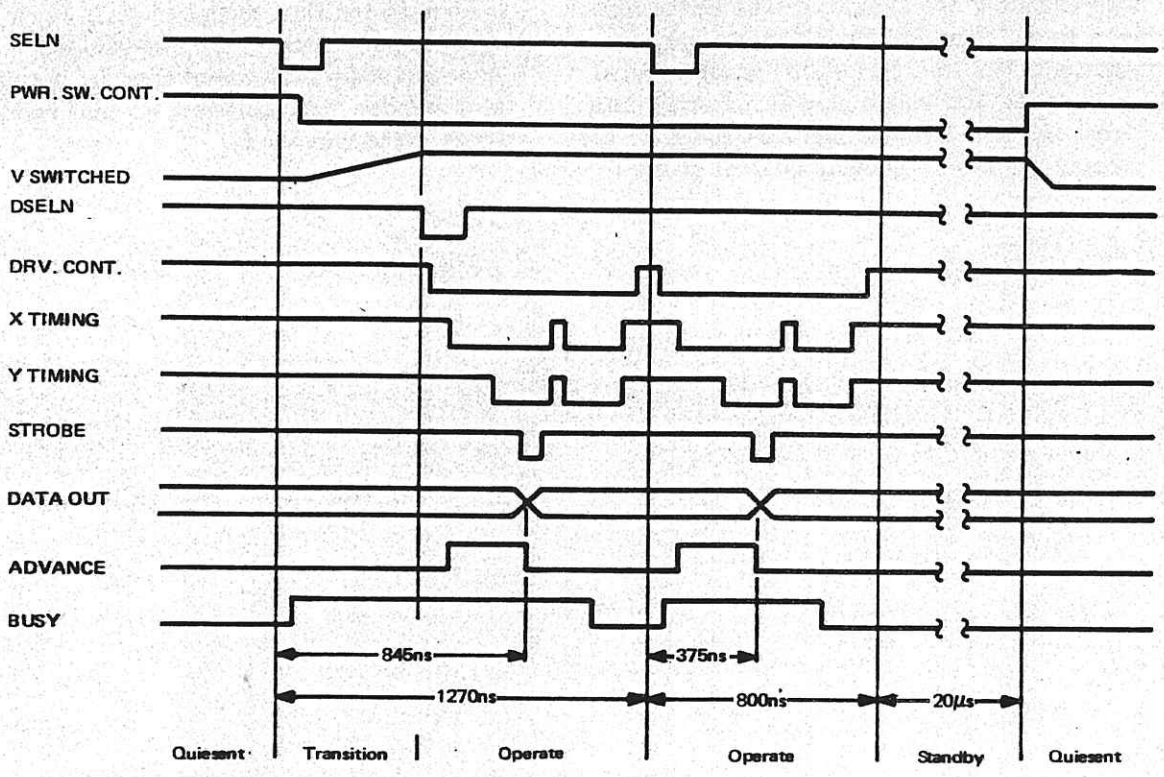
CPU Memory Timing Page Functional Block Diagram

The EP/MCM storage page is the basic building block of the CPU memory. Ten of these 8K word by 18-bit modules can be plugged into the memory backpanel to give a 40K word by 36-bit storage capacity. All storage pages interface exclusively with the memory timing page via the memory backpanel. This interface, with the exception of the V_{ref} voltage, is composed entirely of TTL compatible logic signals.

The core array on the storage page is a planar structure of 8-mil (I.D.), 14-mil (O.D.) cores on 15-mil centers arranged in two 272 by 288 matrices. These two core mats are attached to multilayer interconnection boards (MIBs) which are in turn bonded to separate support frames The two page sections fold together about flex tapes connecting the top parts of the MIBs. The Y address lines are continuous between the two core mats thus forming a core array having functional dimensions of 576 X lines by 256 Y lines. There are 16 additional Y lines which serve as spares.

The 2-1/2 D organization of this memory is implemented by segmenting the core array in the X dimension to form 18 8K word by 1 bit sections. Sense lines run parallel to the X lines and are connected such that each 8K bit section contains two 4K sense windings. Each bit has a unique set of address drivers and a sense amplifier. Since the Y lines are common to every bit, the selection of an 18 bit word is accomplished by addressing one Y line and 18 X lines (one in each bit section).

Address currents are supplied to the X and Y lines by monolithic driver modules. Each driver module has eight outputs capable of generating bipolar half-select currents. Also contained in these MSI modules are a 1 of 8 decoder, timing and polarity controls, and a power control section. Driver modules not required for active addressing are maintained in a low power (standby) state. The X dimension has two address drivers per bit. Addressing of only 16 of the 32 X lines in each bit section is made possible through the use of a phase reversal scheme. With phase



CPU Memory Timing Page Timing Profile

75-A97-001

reversal, two cores per bit receive both X and Y half select currents. In one core these currents are coincident while in the other core the currents are anti-coincident. Selection of either core is controlled by reversing the polarity of the X drive currents. In the Y dimension address currents are supplied through a diode matrix consisting of two diodes per line. This matrix is really two separate 8 by 16 matrices, each servicing 128 Y lines. Addressing of the 8 by 16 matrix is done with four driver modules. On the diode side of the matrix, one driver acts as a voltage source while a second driver acts as a current sink. On the common side of the matrix two drivers perform both source and sink functions.

Outputs from the two 4K sense segments associated with each bit are tied separately to the inputs of a dual channel monolithic sense amplifier. These inputs are time strobed to provide maximum signal to noise ratio. A precise reference voltage sets the discrimination threshold at a nominal value of 7 mV. The outputs of the two channels are ORed together at the input of a latch. This latch which serves as the Storage Data Register (SDR) is an integral part of the MSI sense module. Output data from the SDR is combined with the X driver timing signals to provide control of information to be stored in the core. When the output of the SDR represents a logical zero for a particular bit, the timing signal for that X driver is inhibited during the write portion of the cycle.

Each storage page contains three temperature-controlled voltage regulators (TCV). These hybrid circuits supply precision voltages which vary as a function of the page temperature. Address drive current levels are controlled by these voltages such that the core half select current is maintained at nominal over the full temperature operating temperature range.

The EP/MCM storage page contains a power switch feature which minimizes standby power. Voltages to all components on the page with the exception of the TCV circuits are switched. The power switch is a hybrid module having a response time of 400 ns. When the power switch is in an off condition, the storage page is in a quiescent power state. When the power switch is in an on condition, the storage page is in a standby power state. When the switch is on and the address drivers have been selected the storage page is in an operate power state. A storage page in a quiescent state must first be powered to a standby state before a normal read/write cycle can occur.

8K x 1-Bit Section of Storage Page

EP/MEM Functional Block Diagram

Storage Page Block Diagram

75-A97-001

4.7 INPUT/OUTPUT

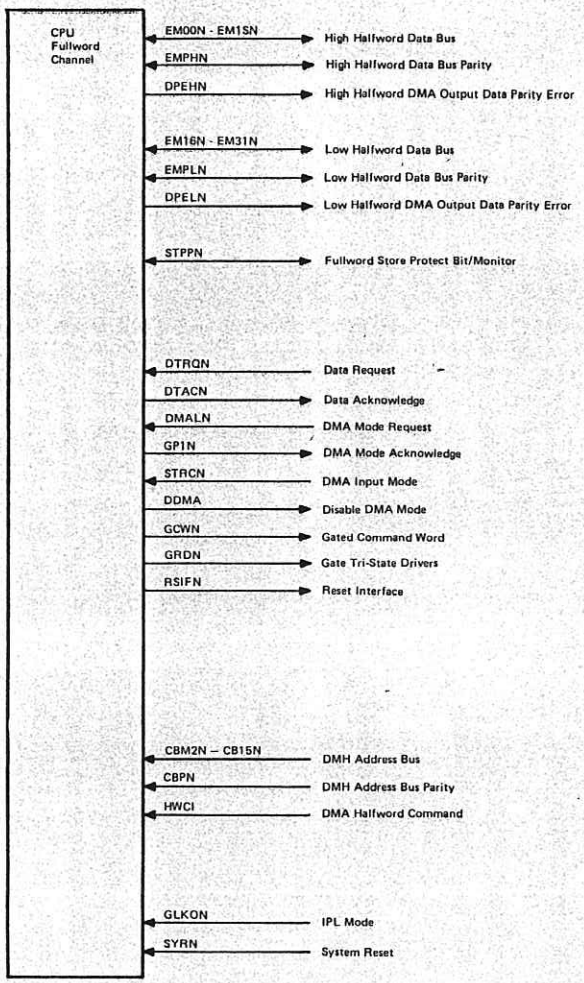
Full Word, Bi-directional, Multimode Channel

Program Controlled or Direct Memory Access modes for input output data transfer operations.

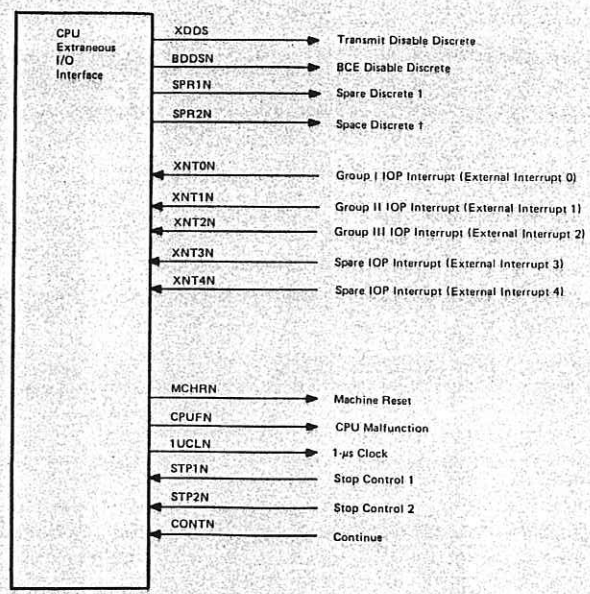
The input/output (I/O) consists of a bi-directional, multimode, 32-bit parallel data channel. The channel operates in either the program controlled input/output (PCIO) mode or the direct memory access (DMA) mode. The PCIO mode first transfers a 32-bit command word (CW) from the CPU to the Input/Output Processor (IOP) to establish initial interface command and control with a particular device. If a program control output function, the CPU then transfers a 32-bit data word to the IOP and if a program control input function the IOP then transfers 32-bit data word to the CPU.

The DMA mode is initiated by the IOP. In this mode, the IOP transfers a 19-bit memory address to the CPU via the DMA Address Bus for either input or output data transfer commands. For DMA input functions, the IOP provides a 32-bit data word input on the data bus concurrently with the address information to provide data for the memory store operation. For DMA output functions, the read portion of the memory cycle provides a 32-bit data word to the data bus for output to the IOP.

In addition to the I/O channel, extraneous I/O interface functions provide for four discrete outputs, five discrete externally activated interrupt inputs, and monitor/control signals.



Fullword Channel Interface



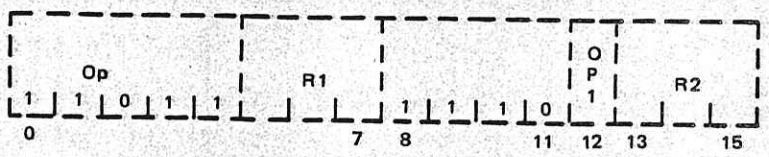
Discrete and Interrupt Interface

75-A97-001

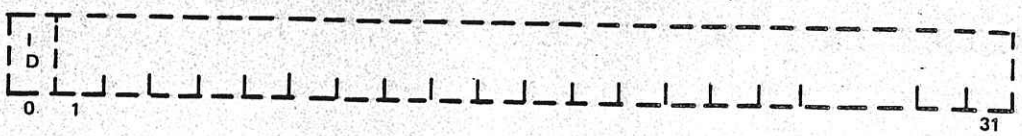
Program Controlled I/O (PCIO)

Program-controlled I/O operations transfer one fullword between a CPU general register and an I/O device. The operation is initiated by executing the privileged instruction "Input/Output." A control word (CW), in a second general register specified by the instruction, defines the specific I/O operation and the specific module or device associated with the operation.

The execution of a PCIO instruction transfers the content of the CPU general register specified by the R2 field to the IOP as a Control Word (CW) and then transfers to or from the general register specified by the R1 field. The CW specifies whether the data transfer is an input or output and further identifies the operation or command.



Program-Controlled I/O Instruction (PCIO)



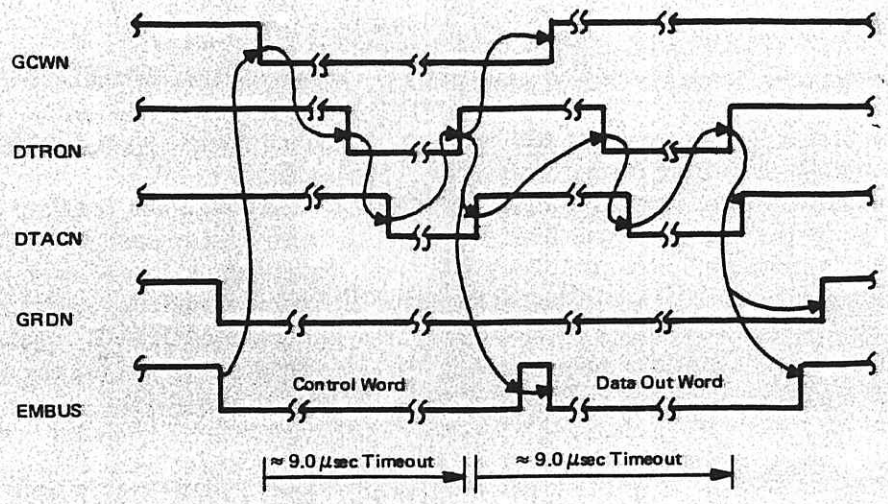
Control Word (CW)

The fields of the CW are defined as follows:

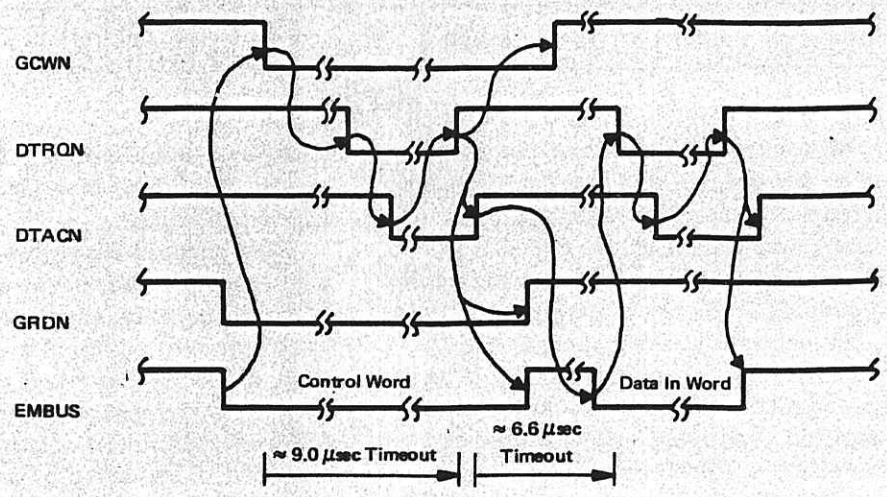
ID: For an input operation, bit 0 must be coded as 0. For an output operation, this bit must be coded as 1.

Command (M): Bits 1-31 specify the particular operation to be performed, and can be used to expand the basic input and output operations. For example, they can be coded to specify sense and control operations. In executing an input operation, the channel (1) transmits the 32-bit CW to the external device; and (2) subsequently loads 32 bits of information, transmitted from the addressed device, into general register R1. In executing an output operation, the channel (1) transmits the CW to the external device, and (2) subsequently transmits bits 0-31 of general register R1 to the addressed device. The specific definition of the command bits is application dependent. The only restriction placed on the system design is the definition of bit 0.

The interface sequences for both program controlled input and output operations are handshaking sequences which are asynchronous to CPU operations. However, if the I/O fails to respond within approximately 9 microseconds, the instruction will timeout and set a condition code of 01. If the instruction completes correctly, a condition code of 00 is set. Channel circuitry generates add parity for each high and low halfword of the data transfer.



PCO Interface Sequence

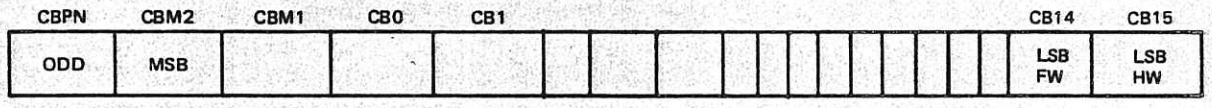


PCI Interface Sequence

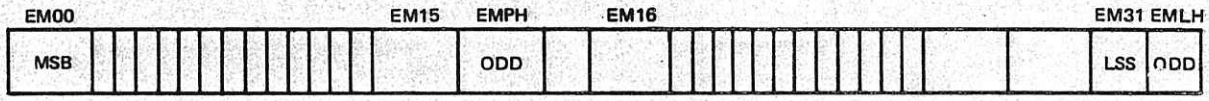
Direct Memory Access I/O Operation (DMA)

The Direct Memory Access I/O operation consists of reading from, or storing into, a halfword or fullword of memory as specified by the DMA Address Bus. The transfer is initiated by the IOP issuing a DMA mode request (DMAIN). The channel will grant DMA mode priority (GPIN) and inhibit CPU main data bus access unless a PCIO operation is pending. Once GPIN goes active, further CPU main bus access are inhibited until either the DMAIN line goes inactive or a disable DMA (DDMAN) occurs.

The data bus consists of two 16-bit halfword buses, each with an associated parity bit. The parity for each 17 bits must be odd. The address bus, containing 18 address bits plus one parity bit, must also have odd parity. Formats for the data and address words are shown.



Address Word



Data Word

For DMA outputs, mainstore parity is checked by the channel and if a parity error is detected for either halfword, the appropriate DMA output data parity error line (DPEHN or DPELN) is activated. Odd parity will be generated for the transfer. For DMA inputs, the parity bits are checked; and if parity is even, the store operation will be inhibited, and the addressed location will be unmodified. Likewise, if DMA address parity is even, a store operation is inhibited and the address location is unmodified, and for either a read or a store, DPEHN and DPELN are activated.

The DMA may be either burst mode or interleave mode. In burst mode, the I/O maintains control of the main data bus by holding the DMA mode request active and toggling data request for each word transferred. This provides maximum data transfer rate but prevents the CPU from executing instructions or servicing interrupts during the period of the burst transfer. In interleave mode, the I/O toggles both the DMA mode request and the data request for each word transferred. This permits the CPU to gain access to the main data bus and therefore to main store, thus allowing execution of CPU instructions and servicing of interrupts.

Four control discretes are associated with the DMA operation. DMA input mode (STRCN) must be active when data request goes active to perform a DMA input. If DMA halfword command (HWCI) is active when data request goes active, a halfword DMA operation will occur. High or low halfword is specified by address bit CB15. The reset interface (RDIFN) discrete is issued by the CPU to command the I/O to abort any I/O operations in progress, reset all Interface lines to the CPU, and prepare to accept a command. This discrete is used when abnormal I/O conditions exist which must be cleared immediately. The Disable DMA mode discrete (DDMAN) is issued by the CPU to inhibit DMA interference during critical CPU operations. A DMA operation in progress will be allowed to terminate normally but further DMA transfers are inhibited. The I/O should complete the data word transfer in progress and then deactivate the DMAIN 1in. The abnormal termination sequences are also shown.

Burst DMA Interface Sequence

Abnormal Terminate DMA Interface Sequences

Interleave DMA Interface Sequence

I/O Error Conditions

Specific error condition checks are made during I/O operations and any errors are reported as CPU interrupts. In addition, certain of these errors generate other error indications as shown in the following I/O Error Condition table.

I/O ERROR CONDITIONS

Error	Indication	Interrupt	Comments
PCIO Timeout	Condition Code 01	N/A	N/A
Even DMA Address Parity	Interrupt Code 05	I/O Group II	DMA inputs Do Not Modify Storage; Forces Output Parity Error Discrete
DMA Address Exception	Interrupt Code 03	I/O Group II	Disable DMA Mode Discrete
PCI Even Parity	Interrupt Code 01	I/O Group II	N/A
DMA Input Even Data Parity	Interrupt Code 02	I/O Group II	Does not Modify Storage
DMA Input to Store Protected Location	Interrupt Code 04	I/O Group II	Does not Modify Storage
DMA Output Storage Parity Error	Interrupt Code 02	Machine Check	Forces Output Parity Error Discrete, Sets Fault Latch

Initial Program Load Operation

Initial Program Load (IPL) operation using the DMA interface allows program loading without utilizing support ground equipments. Three additional interface signals IPL Mode (GLKON), System Reset (SYRN) and Store Protect Bit (STPPN) are required to support this function.

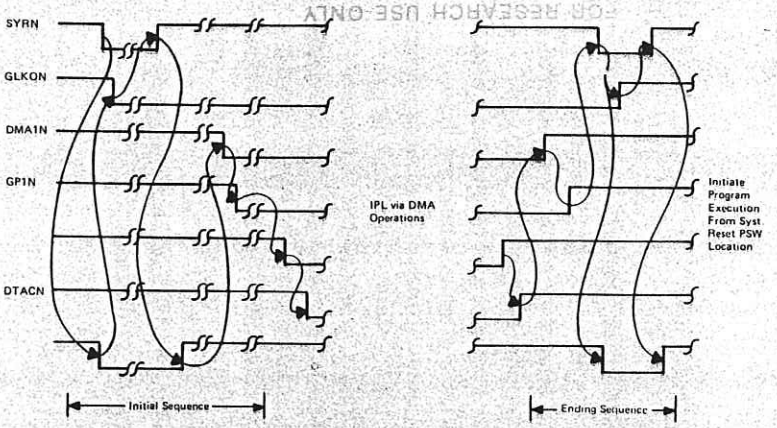
The IPL mode is entered by first activating System Reset then by activating GLKON and the deactivating SYRN. The CPU microprogram then executes a basic self test, enables the store protect logic and enters a microprogram loop testing GKLON and branching back. When the Disable DMA (DDMAN) line goes inactive the DMA mode may be initiated by the IOP for the program load operation.

The STPPN line must be utilized to load the store protect bit(s) for each word loaded. If STTPN is active on a fullword load, both halfwords willbe protected; if STPPN is inactive, both halfwords will be unprotected. For halfword operations, only the addressed halfword store protect bit is established.

After completing the IPL load function, a verification of the loaded locations may be accomplished by executing a DMA output. The STTPN Line will be active for fullword outputs if either halfword is protected; for halfword outputs it indicates that the addressed halfword is protected.

Following verification, SYRN is activated, GKLON is deactivated and SYRN then deactivated. The CPU is now ready to begin instruction execution from the System Reset PSW location.

During the IPL mode of operation, since the CPU microprogram is not operating in a normal mode, the normal DMA error indications do not apply. The IPL error conditions and the required actions by the IOP are listed in the IPL Error Conditions table.



Initial Program Load Beginning and Ending Sequences

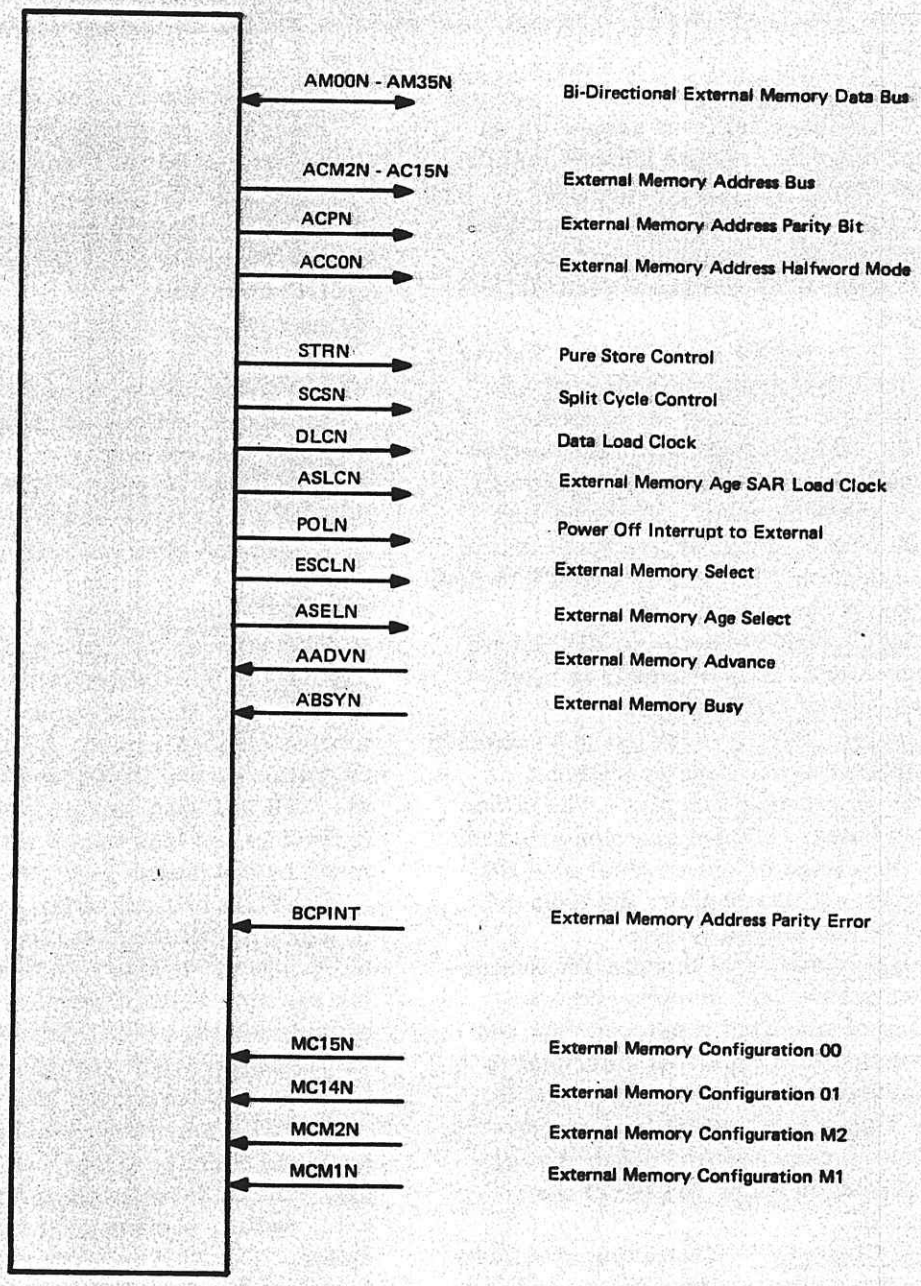
IPL Error Conditions

Error	Indication	CPU Action	Required IOP Action
Even DMA Address Parity	Output Parity Error Discrete	Inhibit Data Load clock, Recycle Interface	Re-try Transfer
DMA Address out-of-bounds	Disable DMA	Inhibit transfer, Inhibit DMA priority, Disable DMA	System Reset Re-try Transfer
Even DMA Input Data Parity	Output a Parity Error Discrete	Inhibit Data Load clock, Recycle Interface	Re-try Transer
Even DMA Output Storage Parity	Output a Parity Error Discrete	Set Fail Latch, Recycle Interface	Re-load Location

Within the space shuttle configuration, the mainstore designated "internal memory" is physically located in the CPU and consists of 40K fullwords. An additional mainstore area of 24K fullwords designated "external memory" is physically located in the IOP unit. A third External Memory Unit (EMU) may be utilized to add up to a total main store configuration of up to 256K fullwords.

The asynchronous memory interface used for internal memory is maintained for external memory with the exception that the data is transferred via a bi-directional tri-state bus and address parity is checked by the external memory. If even address parity is detected, the external memory generates the BCP INT interrupt and inhibits the Data Load Clock. The signal Pure Store Control (STRN) is utilized by the external memory to control enabling the tri-state Data Bus Drivers.

The external memory configuration bits are biased at the external memory drivers to specify the amount of external addressable mainstore in the system. The biasing on the CPU receivers is such that if the external memory is powered off, the memory configuration is limited to the 40K internal memory. For a 24K IOP memory, MCM2N is driven inactive and MCM1N, MC14N, and MC15N are driven active. These bits are used by the bus control unit to determine address exception violations.



CPU C/M External Memory Interface

4.8 FAULT DETECTION

Hardware and Software Detect Greater Than 95% of Faults

Built-in-test equipment (BITE) hardware and self-test program (STP), partially contained in microstore and main storage, combine to give a high degree of fault detection, thereby assuring high confidence in performance.

Built-In-Test Equipment

BITE hardware features are shown on the block diagram. These complement the Self-Test Program (STP) to provide greater than 95% detection of the hardware failures that affect computer operation. Details on the BITE hardware features are as follows:

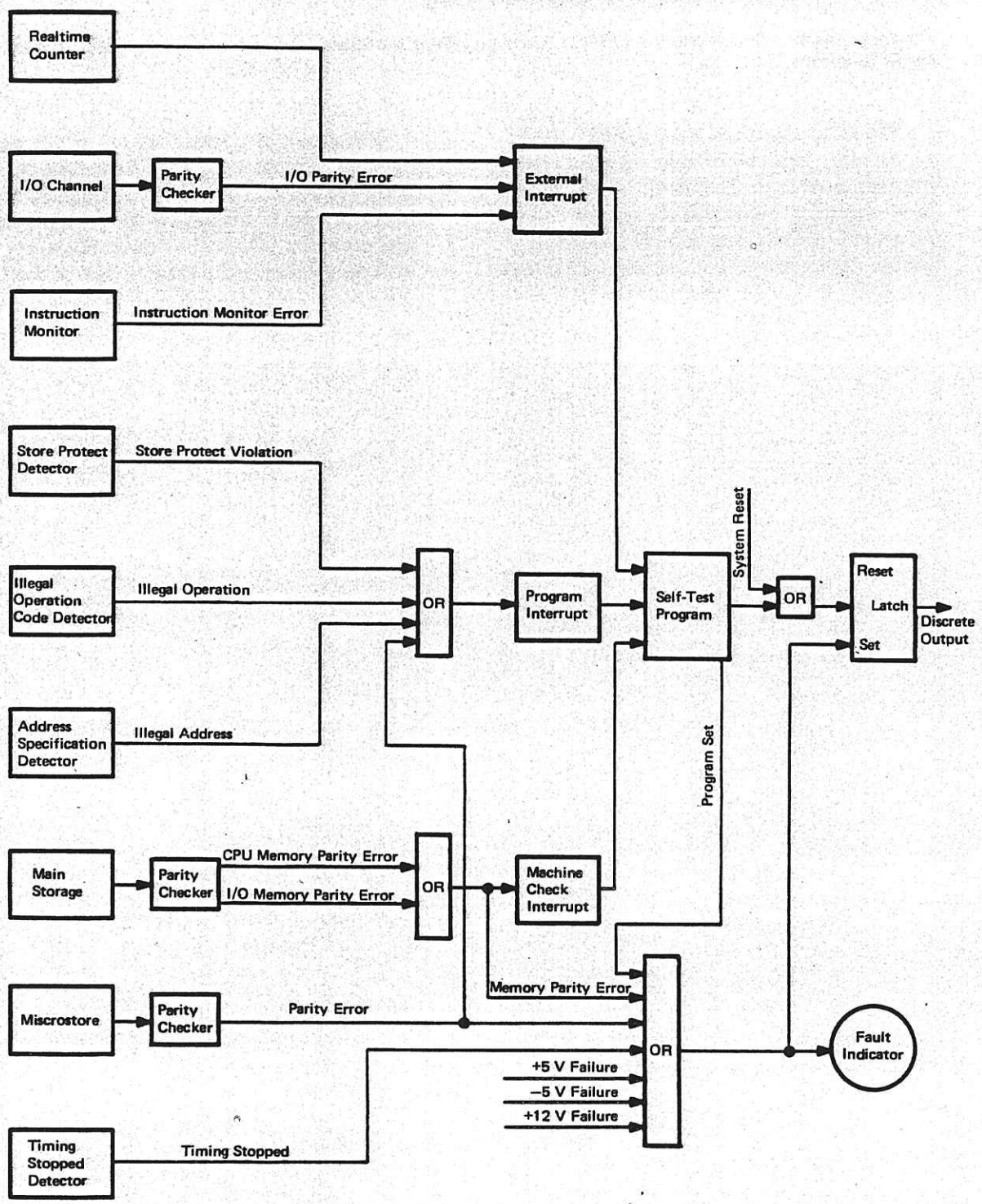
- Main Store Parity - All data and instruction transfers from main storage are checked for odd parity on a halfword basis. Odd parity enables detection of complete loss of information where all zeros are even parity. Selected data words with even parity are generated in PSA Location 2 for use by the STP to test the parity checking circuits.
- Timing Stopped Detect - An RC circuit, designed to remain charged, is used to detect loss of CPU timing.
- Microstore Parity - Odd parity is checked on all microstore read operations.
- Power Monitoring Circuits - The power supply output voltages are monitored for loss of voltage or out-of-tolerance DC voltages. If this occurs, the fault indicator is automatically set.
- Storage Protection - Storage Protection prevents hardware or software overwriting of specified prestored data and instruction locations. An interrupt is generated when an attempt is made to alter a protected location. Locations for protection are specified either during initial load or under control of the program.
- Illegal Operation - Instruction operation codes are tested automatically, as each is executed, to ensure that an invalid or unused operation resulting from a logic failure does not leave the CPU in an unknown state. Should this occur, a program interrupt is generated.
- Illegal Address - Any attempt to address main storage outside the configured number of words result in an address specification error.
- I/O Parity Assignment and Check - Odd parity is generated and checked on all I/O transfers.
- Instruction Monitor - The instruction monitor prevents the mistaken execution of variable data (non-storage-protected) as an instruction, and it generates a maskable interrupt.
- Real Time Counter - 32-bit counter with a 1 μs basic clock that can be loaded and read via software.

CPU Self-Test Program

STP utilizes microprogram test routines resident in microstore to reduce main storage and execution time requirements and to allow testing in a cyclic fashion. In the operation mode, the processor portion of the CPU is tested using fewer than 350 halfwords and less than 1 ms of time. The memory operations are tested using BITE.

Preflight or operational readiness test of memory utilizes less than 150 additional halfwords of storage and less than 300 ms, for 64K fullwords of memory. These tests provide testing of all drivers, amplifiers, addressing circuits and cores prior to flight or critical phases of a mission.

The STP may be co-resident with the operational flight program (OFP). The OFP may control the manner of STP execution; i.e., cyclic, background or pre-flight execution. The STP provides fail or pass decision about the CPU functions tested and saves information pertinent to a detected failure.



Fault Detection Functional Block Diagram

75-A97-001

4.9 AGE INTERFACE

Interface Is Compatible With Microprogrammed Test Set

Provides interface to exercise the CPU for software debug, hardware fault detection, and memory load/verify functions.

The CPU interface AGE signals listed in the table are compatible with the microprogrammed test set (MTS). These interface signals provide the necessary functional control and data transfers to enable external control and test of the CPU.

Execution of micro orders, which are entered into the CPU ROS Data Register from the AGE interface provides the capability to load and read all CPU registers and counters which are otherwise not available outside the boundaries of the CPU.

Pin	Signal Mnemonic	Signal Name	Signal Functional Description	Logic 0	Level 1	Input/ Output
13	XBOOON	Bus Out Position 0	20-Bit Parallel AGE Bus Out, All Scan Out Data Except Microprogram Instructions is Via This Bus	High	Low	Out
14	XBOO1N	Bus Out Position 1		High	Low	Out
21	XBOO2N	Bus Out Position 2		High	Low	Out
23	XBOO3N	Bus Out Position 3		High	Low	Out
32	XBOO4N	Bus Out Position 4		High	Low	Out
33	XBOO5N	Bus Out Position 5		High	Low	Out
43	XBOO6N	Bus Out Position 6		High	Low	Out
45	XBOO7N	Bus Out Position 7		High	Low	Out
56	XBOO8N	Bus Out Position 8		High	Low	Out
67	XBOO9N	Bus Out Position 9		High	Low	Out
79	XBO10N	Bus Out Position 10		High	Low	Out
80	XBO11N	Bus Out Position 11		High	Low	Out
82	XBO12N	Bus Out Position 12		High	Low	Out
92	XBO13N	Bus Out Position 13		High	Low	Out
102	XBO14N	Bus Out Position 14		High	Low	Out
104	XBO15N	Bus Out Position 15		High	Low	Out
113	XBO16N	Bus Out Position 16		High	Low	Out
114	XBO17N	Bus Out Position 17		High	Low	Out
127	XBO18N	Bus Out Position 18		High	Low	Out
128	XBO19N	Bus Out Position 19		High	Low	Out
51	XSTP1N	AGE Stop 1	Stop Controls Specify Unique Computer Run/Halt Conditions	High	Low	In
52	XSTP2N	AGE Stop 2		High	Low	In
62	XAION	Stop Internal Oscillator	Stop Internal Oscillator, Computer Timing Under Tester Control	High	Low	In
64	XSYSRN	System Reset	Initializes Computer to Known State	High	Low	In
57	XWAIT	Wait State	Wait State Indicator	Low	High	Out
58	XMSTL	Monitor Stop Time Level	Computer Stop Indicator, 1 or 2	Low	High	Out
69	DNUD	Don't Use Display	Control Signal to Inhibit Use of AGE Interface	Low	High	Out
09	XIADV	Advance	AGE Interface Simulates Computer Memory Advance	High	Low	In
15	XIBUSY	Busy	Simulates Computer Memory Busy	High	Low	In
16	XISEL	Inhibit Select	Inhibits all CPU Selection to Memory	High	Low	In
18	XIREQ	Inhibit Request	Inhibits All I/O Selection to Memory	High	Low	In
25	XLDCLK	Load Clock	Loads SAR or SDR per XBOC	High	Low	In
26	XECLKI	External Clock	Clocks Data (XEXDI) into AGE Data Register	Low	High	In
37	XEXDI	External Data	AGE Serial Data	High	Low	In
48	XEXSS	External Start	Starts Computer Memory Cycle	High	Low	In
49	IOPEN	IO Parity Error	IO Channel Parity Error Indicator	High	Low	Out
50	CPUPEN	CPU Parity Error	CPU Parity Error Indicator	High	Low	Out
59	GNGTN	Go/No-Go Time Out	N/A for Space Shuttle	High	Low	Out
53	XINTN	Inhibit Interrupt	Inhibits Generation of Macro Interrupts	High	Low	In
55	XBUMP	Bump	Clock that Enables the Transfer of Data according to the Microcode	High	Low	In
66	XRDIN	ROS Data In	Serial Data to the AGE ROS Scan Register	High	Low	In
65	XCONT	Continue	Continue in accordance with Stop Lines 1 and 2	High	Low	In
106	XMINTN	Microinterrupt	Microinterrupt pending Indicator	High	Low	Out
105	XMANTN	Macrointerrupt	Macrointerrupt pending Indicator	High	Low	Out

Pin	Signal Mnemonic	Signal Name	Signal Functional Description	Logic 0	Level 1	Input/ Output
38	XEDIS	Enable Display	Enable Selector Override Display	High	Low	In
40	XHDSC	Hard Scan	Enable Hard Scan	High	Low	In
24	XAEXSS	Allow External Start	Signal Inhibits Computer Generation of Memory Signals. Allows External Signals from Tester Control for Computer Memory Load/Verify	High	Low	In
34	XGOSDR	Gate Out SDR	High or Low Half SDR Gate to AGE Bus	High	Low	In
47	XGDI	Gate Data In	Gate into either SAR or SDR according to XBOC	High	Low	In
46	XBOC	Bus Control	Controls Gating of Data to either SAR or SDR	High	Low	In
27	XRDCLN	Register Clock	AGE ROS Scan Register Clock	High	Low	In
122	XP12V	+12VDC	+12VDC Power Supply Monitor Signal			
123	XP5V	+5VDC	+5VDC Power Supply Monitor Signal			
124	XN5V	-5VDC	-5VDC Power Supply Monitor Signal			
01	XEO	External Oscillator	External Timing Simulates Computer Oscillator	High	Low	In
03	XCP2	Clock Pulse 2	5 MHz CPU Clock 2 Monitor Signal	Low	High	Out
05	XCTCLK	Counter Clock	5 MHz Counter Clock Monitor Signal	High	Low	Out
110	XIFDI	Instruction Fetch Data Input	End of Operation Signal	Low	High	Out
119	XP3	Priority 3	Indicates that CPU is Using Memory.	Low	High	Out
75	XCADVN	Computer Advance	With this Signal and XP3, Memory Requests are Identified	High	Low	Out
88	XSATN	AGE Time Level	Start AGE data serial transfer for AGE in instruction	High	Low	Out
76	XRA00	ROS Address Bit 0	12 Bit Microprogram Instruction Address Register Scan Out	Low	High	Out
41	XRA01	ROS Address Bit 1		Low	High	Out
42	XRA02	ROS Address Bit 2		Low	High	Out
109	XRA03	ROS Address Bit 3		Low	High	Out
10	XRA04	ROS Address Bit 4		Low	High	Out
12	XRA05	ROS Address Bit 5		Low	High	Out
19	XRA06	ROS Address Bit 6		Low	High	Out
20	XRA07	ROS Address Bit 7		Low	High	Out
78	XRA08	ROS Address Bit 8		Low	High	Out
90	XRA09	ROS Address Bit 9		Low	High	Out
100	XRA10	ROS Address Bit 10		Low	High	Out
101	XRA11	ROS Address Bit 11		Low	High	Out
61	XGBAN	Gate ROS Bus Control A	Controls that Select the Function of the AGE ROS Scan Register to be Loaded or Read	High	Low	In
29	XGBBN	Gate ROS Bus Control B		High	Low	In
30	XGBCN	Gate ROS Bus Control C		High	Low	In
116	XRDOT	ROS Data Out	AGE ROS Scan Register Serial Data Output	High	Low	Out
31	XLDRDN	Load ROS Data Register	AGE Supplied ROS Data Register Load Clock	High	Low	In
115	XRPERN	ROS Parity Error	ROS Parity Error Indication	High	Low	Out
95	XSPERN	Store Protect Error	Store Protect Error Indication	High	Low	Out
87	XBUSYN	Busy	Memory Busy Indication	High	Low	Out
85	XADEXN	Address Exception	Address Exception Indication	High	Low	Out
97	XAOUTN	AGE Out	AGE data ready Indication for AGE out instruction	High	Low	Out
74	XAITPN	AGE Interrupt	Causes AGE Interrupt When Activated	High	Low	Out

75-A97-001

Input Formats

The input formats for the AGE Data register is defined below: The order of shifting in is bit 19 first.

See page 4-44 for a description of the AGE ROS Scan Register formats. The AGE ROS Scan Register is a 48 bit serially loaded register. The order of shifting in is bit 48 first.

Output Formats

This subsection defines the format of the AGE output bus. The control is set to select the desired register for output on the AGE bus. The control lines and XBO lines are defined in the table.

Input Formats

Input Destination	AGR																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
SAR	M3	M1	C0	M2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SDR High	MOP	ZERO	PH	0	1	2	3	4	5	6	7	STPH	8	9	10	11	12	13	14	15
SDR Low	MOP	ZERO	PL	16	17	18	19	20	21	22	23	STPL	24	25	26	27	28	29	30	31

*ZERO means logic zero

M3 = Most significant address bit
M2 = Next most significant address bit
M1 = Third most significant address bit
MOP = Memory operation bit; 0 = Read operation, 1 = Write operation
C0 = Control for fullword/halfword select; 0 = fullword select, 1 = halfword select
PH = Parity high; parity bit for the high data halfword
PL = Parity low; parity bit for the low data halfword
STPH = Store Protect High, store protect bit for the high data halfword
STPL = Store Protect Low, store protect bit for the low data halfword

AGE OUTPUT BUS

Output Displayed	XBO																			Control Signals			
	00N	01N	02N	03N	04N	05N	06N	07N	08N	09N	10N	11N	12N	13N	14N	15N	16N	17N	18N	19N	XBOC	XGOSDR	XGDI	SAR 15
CAB	M3	M2	CO	MI	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	I	Z	X	X
SDR	MPE	Z	PH	0	1	2	3	4	5	6	7	SPH	8	9	10	11	12	13	14	15	Z	Z	Z	Z
SDR	MPE	Z	PL	16	17	18	19	20	21	22	23	SPL	24	25	26	27	28	29	30	31	Z	Z	Z	I
SDR	MPE	Z	PH	0	1	2	3	4	5	6	7	SPH	8	9	10	11	12	13	14	15	Z	I	Z	Z
SDR	MPE	Z	PL	16	17	18	19	20	21	22	23	SPL	24	25	26	27	28	29	30	31	I	I	Z	I
*AGR	MPE	Z	GP	3	4	5	6	7	8	9	10	Z	12	13	14	15	16	17	18	19	I	Z	I	I
*AGR	MPE	Z	GP	3	4	5	6	7	8	9	10	Z	12	13	14	15	16	17	18	19	X	I	I	I

The information on the AGE output bus is defined as a logic one when the bus is at a low level and a logic zero when the bus is at a high level.

MPE = Memory Parity Error
X = Don't Care
Z = Logical Zero
I = Logical One
GP = Output of Parity Generator

Registers Definitions and Controls

The CPU contains a 20-bit serial input shift register denoted as the AGE Register (AGR). The loading of this register is controlled by two AGE lines denoted XEXDI and XECLKI. The data input line is XEXDI and XECLKI is the clocking line. The data is shifted serially into the AGR by setting the data line to the desired level and then cycle the clock line, see Figure 1.

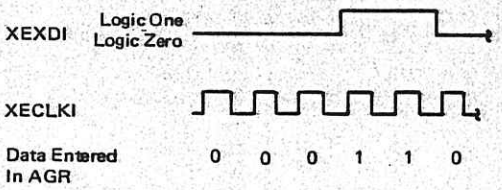


Figure 1. AGE Register Data Clocking

The data line is to be settled a minimum of 20 ns before the clock line is set to ONE. The clock pulse is to be stable for a minimum of 20 ns at ONE before it is set to ZERO. The maximum data rate transfer is one MHz. The minimum data rate transfer is determined by the MTS.

The AGR register positions are denoted as AGR00, AGR01...AGR19 where AGR19 is the first bit shifted in and AGR00 is the last bit shifted in.

The output of the AGE data register connects to the multiplexer which forms the computer bus (CPB) and via dot nands to the computer address bus (CAB). The AGR outputs are gated on whenever the signal XGDI = 1 or the current microorder contains a microorder miscellaneous Hex code of FXD (where X is any hex digit).

As shown in Figure 2, the XGDI signal gates the AGR outputs to both CPBHI (0-15) and CAB at the same time. The clock signals SLC and DLC enter data from the respective buses to the respective registers. Pulsing of the two clocks is controlled by XGDI, BOC, and XLDCLK signals as shown in Figure 3.

As indicated in Figure 3, the content of the AGR is transferred to the SAR when XGDI = XBOC = ONE and XLDCLK is pulsed. The B MPX controls are conditioned to load memory when XGDI = ONE, XBOC = ZERO, and XLDCLK is pulsed. The AGR data is actually transferred to the SDR during the memory cycle by an internally generated data load clock. The A MPX controls are conditioned during the memory cycle by maintaining XGDI = ONE until the end of the cycle.

Two signals are provided for external initiation of a memory cycle. The signals are defined as Allow External Start Store (XAEXSS) and External Start Store (XEXSS). When XAEXSS = ZERO, the XEXSS signal is inhibited. When XAEXSS = ONE, the rising edge of XEXSS issues a select to the memory (starts a memory cycle) (see Figure 4). AGE memory operations are halfword only.

Figure 4 indicates the selects are not issued to the memory when XAEXSS = ZERO. When XAEXSS = ONE selects are issued on the rising edge of XEXSS. To issue another memory SELECT, XEXSS must be lowered and again raised. The actual width of the down level of XEXSS is not important. The only important timing consideration is that the leading edge to trailing edge of the XEXSS must be 300 to 400 ns.

Figure 1 defines the gating of the AGR. The contents of the AGR are gated to the SDR and SAR, according to a previous paragraph as defined in the following paragraphs.

When loading the SAR, the AGR outputs 0, 3, 1 and 4 to 19 are gated in CAB positions M3, M2, M1 and 0 to 15, respectively. AGR position 2 is connected to the SARCO position. SARCO = ONE indicates the memory operation is halfword. This bit is used by the memory for controlling the power up/down circuits when a SELECT is issued to the memory.

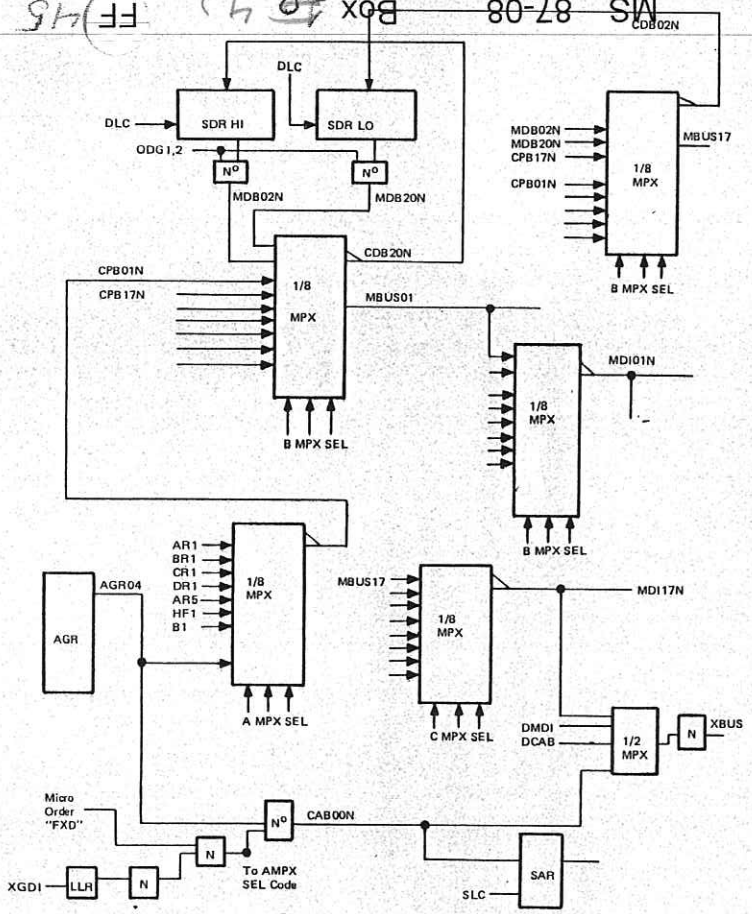


Figure 2. One Typical Bit of Bus

The MOP bit (ARGO) indicates either a read or store operation is to be executed when SELECT is issued to the memory. MOP = ONE causes a store operation and MOP = ZERO causes a read operation when SELECT is issued.

When loading SDR HIGH, the AGR outputs 2 to 19 are gated into SDRHP (High Parity), SDRO to 7 SDRHS (High Protect), and SDRO to 15 positions, respectively. When loading SDR LOW, the AGR outputs 2 to 19 are gated into SDRLP (Low Parity), SDR16 to 23 SDRLS (Low Protect) and SDR24 to 31 positions, respectively.

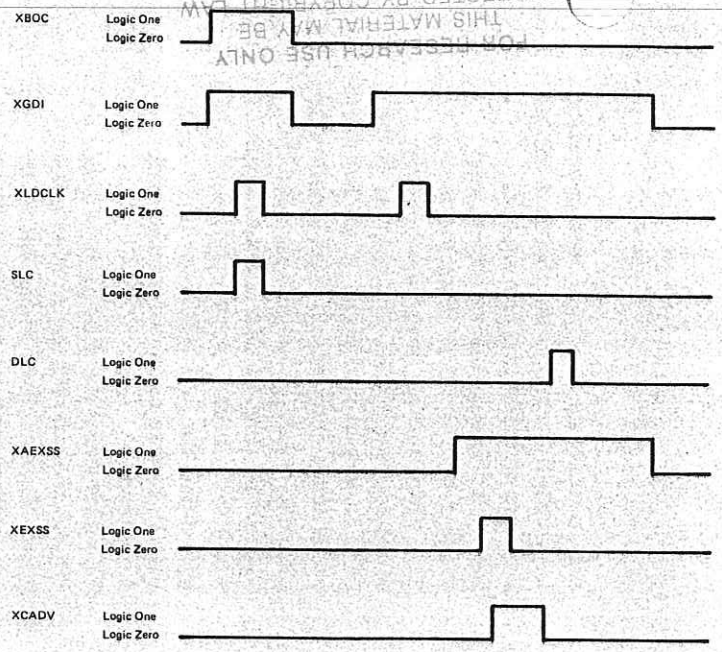


Figure 3. AGE Memory Load Sequence

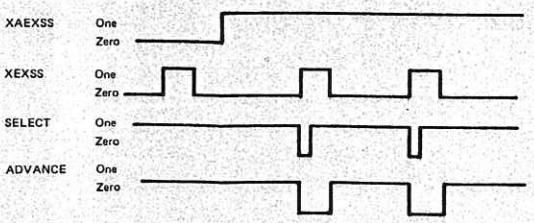


Figure 4. Memory Cycle Start

AGE ROS Scan Register (ARSR)

The AGE ROS Scan Register (ARSR) is a 48-bit parallel access shift register which is serially accessed by the MTS through the AGE connector. This register provides parallel access to the computer microprogram storage area, and through this access enables execution of the following functions:

1. Enter a micro order into the ROS Data Register for execution by the CPU. This function permits loading or reading of CPU registers.
2. Read the output of the ROS Data Regiser at a specified time.
3. Read the output of the PROM (microstore area) at a specified address.

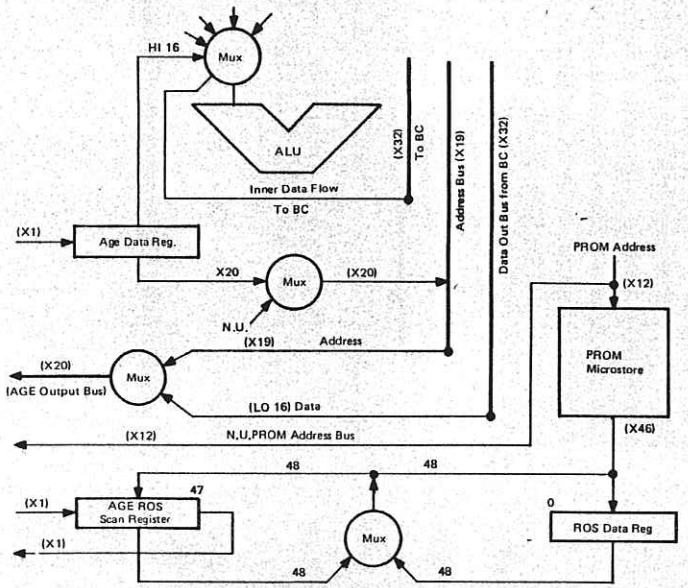
These three functions are available through AGE only when the CPU is halted.

4. A fourth function can be executed which enables observation of dynamically gleaned information. Each time a micro order is executed, the operation is finished by supplying the next PROM address to the microstore area, fetching the next micro order, and storing it in the ROS Data Register. This information is simultaneously stored in the ARSR. Therefore, if the CPU is halted at the end of a CPU cycle or is being operated in ROS single step, the next micro order can be read from the ARSR. The PROM address from which this micro order originated would have been the last address on the AGE PROM address bus before the halt command.

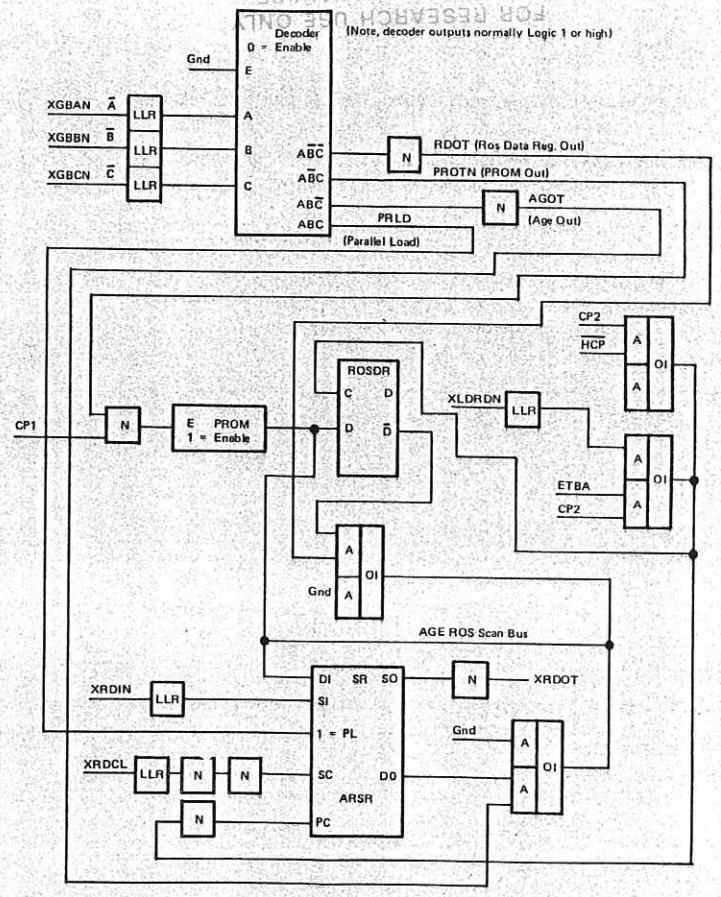
To understand the data flow of the ARSR and the associated AGE supplied signals. Mode control of the ARSR is accomplished through signals XGBAN, XGBBN and XGBCN as shown in the following table. Note XGBAN is used as a control bit to enable or disable ARSR.

XGBAN	XGBBN	XGBCN	MODE
1	X	X	ARSR Disabled
0	1	1	Place ROS Data Register Output on ARSR Parallel Input
0	1	0	Place PROM Data Output on ARSR Inputs
0	0	1	Place ARSR Parallel Data Output on ROS Data Register Input
0	0	0	Place ARSR in Serial Mode

Mode Control Truth Table



AGE Interface Data Flow



AGE ROS Scan Register (ARSR) Functional Diagram

75-A97-001

The ARSR is loaded by the MTS in the following manner. Mode control lines XGBAN, XGBBN and XGBCN are all set to logic zero, placing the ARSR in serial mode. Data is supplied on line XRDIN and clocked in by clock signal XRDCL.

The data line shall be set and stable a minimum of 50 ns before the clock line is set to one. The clock input is a symmetrical square wave with a maximum rate of 1 MHz and a minimum uptime of 500 ns.

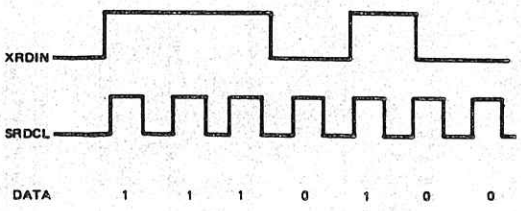
Once loaded, the contents of the ARSR are parallel loaded into the ROS Data Register by setting the mode control lines to logic 001. This places the ARSR outputs on the ROS Data Register inputs, and, by removing the prior 000 code, places the ARSR in parallel mode. The contents of the ARSR are loaded into the ROS Data Register by a single negative clock pulse on signal line XLDRN.

To read the output of the ROS Data Register, mode control lines XGBAN, XGBBN and XGBCN are set to logic 011. Note that this places a logic one on the RDOT Line and a logic one on the PRLD line. This places the ROS Data Register output on the AGE ROS Scan Bus, and places the ARSR in parallel mode. The contents of the ROS Data Register are loaded into the ARSR by a single negative clock pulse on signal line XLDRN.

The contents of the ARSR are now available to the MTS via the ARSR serial data output XRDOT. The ARSR is placed in serial mode by setting mode control lines XGBAN, XGBBN and XGBCN to logic 000. The register is shifted when the XRDCL line is pulsed to a logic one. The clock should be a symmetrical square wave with a maximum rate of 1 MHz and a minimum pulsewidth of 500 ns.

To read the output of the PROM at a specified address, two functional steps are necessary. First, a PROM address is supplied via the ROS Data Register, and second, the PROM output is scanned out via the ARSR. The PROM address is loaded in microorder form as previously stated. The table opposite lists information pertaining to establishing the microorder. Upon execution of this microorder, the next PROM address is derived and placed on the PROM address bus. PROM information is now placed on the AGE ROS Scan Bus by setting mode control lines XGBAN, XGBBN and XGBCN to logic 010. The information present on the bus is now stored in the ARSR by pulsing negative XLDRDN. Note that the PROM data is also stored in the ROS Data Register at this time. The PROM address is verified by reading lines XRA00 through XRA11.

The contents of the ARSR are now available to the MTS via the ARSR serial data output XRDOT. The ARSR is placed in serial mode by setting mode control lines XGBAN, XGBBN and XGBCN to logic 000. The register is shifted when the SRDCL line is pulsed to a logic one. The clock is a symmetrical square wave with a maximum rate of 1 MHz and a minimum pulsewidth of 500 ns.



ARSR Load Clock and Data Relationship

MICROORDER FORMAT FOR FORCING PROM ADDRESS

BIT NO:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
SET TO:	0	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0	R_{A_0}	R_{A_1}	R_{A_2}	R_{A_3}	R_{A_4}	R_{A_5}	0

BIT NO:	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
SET TO:	R_{A_6}	R_{A_7}	R_{A_8}	R_{A_9}	0	0	0	0	$R_{A_{10}}$	0	0	0	0	$R_{A_{11}}$	P	0

CPU Register Load and Display Functions

By issuing a single negative pulse of 200NS duration, minimum, on the XBUMP line while the CPU timing is stopped, the CPU is forced to execute the micro-order stored in the ROS Data Register. This pulse simulates the computer clock. By executing a series of AGE supplied micro-orders, CPU registers are loaded or displayed. The tables below illustrate specific micro-orders which are used during load and display functions.

ROS DATA REGISTER LOAD CODES

ROS Data Word	Function
3CE07FEB0000	AGR → BREG LO, BREG LO → BREG HI
228057F80000	BREG → AREG
22C057F80001	BREG → CREG
22E057F80000	BREG → DREG
22005F800001	BREG LO → ROS BACKUP REG
22005F880000	BREG LO → NIR
22C05F980001	BREG LO → OREG
22C05FA00000	BREG LO → CTR 0
22C05FA80001	BREG LO → CTR 1
22C05FB00000	BREG LO → CTR 2
22C05F900001	BREG BITS 24-27 → PREG BITS 8-11
22C05FB80001	BREG BITS 16-23, 28-31 → PREG BITS 0-7, 12-15
42C022A00001	SET STAT 19
42C02AA00000	CLEAR STAT 19
42D657F80000	CREG → LS PER STAT 19, OREG 8-11

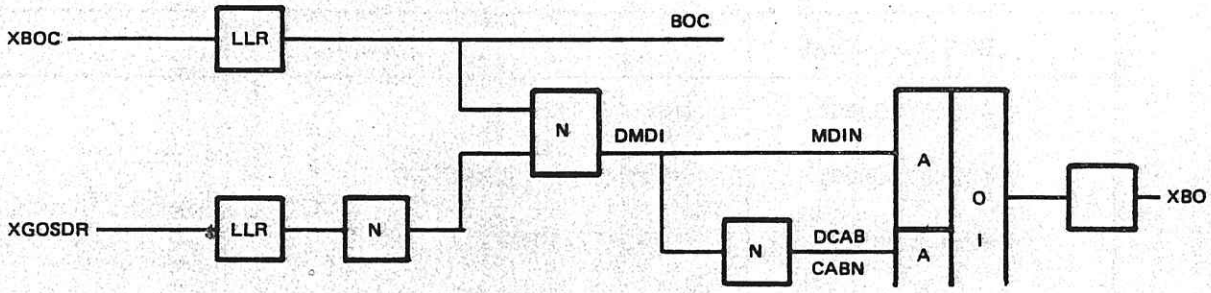
ROS DATA REGISTER DISPLAY CODES

ROS Data Word	Function
42C057C80001	NIR → BREG LO
42C057D00001	STAT 0-15 → BREG LO
42C057D80000	OPREG → BREG LO
42C057E00001	CTR0 → BREG LO
42C057E80000	CTR1 → BREG LO
42C057F00000	CTR2 → BREG LO
42C057F80001	PREG → BREG LO
02C07FA80000	AREG -R → BREG
02A057F80000	AREG → BREG
22C07FA80001	BREG -R → BREG
22A057F80001	BREG → BREG
42C07FA80001	CREG -R → BREG
42A057F80001	CREG → BREG
62C07FA80000	DREG -R → BREG
62A057F80000	DREG → BREG
42C657F80001	LS → AREG PER STAT 19, OREG 8-11

NOTE: -R → Indicates High to Low and Low to High

AGE Output Bus

The AGE output bus consists of 20 bits denoted XBO00N to XBO19N. The content of the bus is either the computer address bus (CAB) or the main data in bus (MDI) bits 16-31. Selection of which bus is gated is shown below.

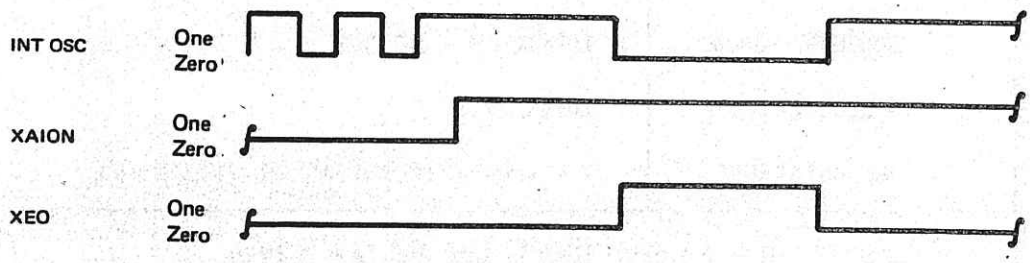


AGE Output Bus

Two signals determine if the MDI BUS or CAB BUS are to be gated onto the AGE output bus. When XGOSDR = ZERO and XBOC = ONE, the cab is gated to AGE Bus. When XGOSDR = ZERO and XBOC = ZERO, the MDI BUS is gated to the AGE BUS as shown in the figure above, and the CPU Registers (AR, BR, CR, DR) are gated onto the CPB. By proper selection of micro orders, the CPB is gated direct or reverse (CPB16-31, 0-16 to MBUS 0-31 respectively) to the MBUS and then to MDI 16-31. The Memory Bus (MDB) is gated direct or reverse based on the content of SAR 15. The MDB is gated to the MDI when XEDIS equals one. The table on page 4-41 shows the content of the Bus for memory verify functions.

Oscillator Simulation

Two signals are provided on the AGE interface for simulating the CPU internal oscillator. The XAION, when at ONE, inhibits the CPU internal oscillator to the timing generator. The signal XEO simulates the internal oscillator. XEO = ONE, the internal oscillator equals a ZERO and XEO = ZERO, the internal oscillator equals ONE as far as the timing generator is concerned. See Figure below.



Oscillator Simulation

Clock Control

Three signals are provided on the AGE interface for controlling the computer clocks. Two of the signals XSTP2N and XSTP1N determine how the clocks are to be generated and the XCONT enables the generation of the clocks. The logical states of XSTP2N and XSTP1N are defined in the table below. The stop line skew must be less than 100 ns.

CONTROL CLOCKS

XSTP1N	XSTP2N	Function
ZERO	ZERO	RUN
ZERO	ONE	Stop at End of CPU Cycle.
ONE	X	Stop at End of Instruction.

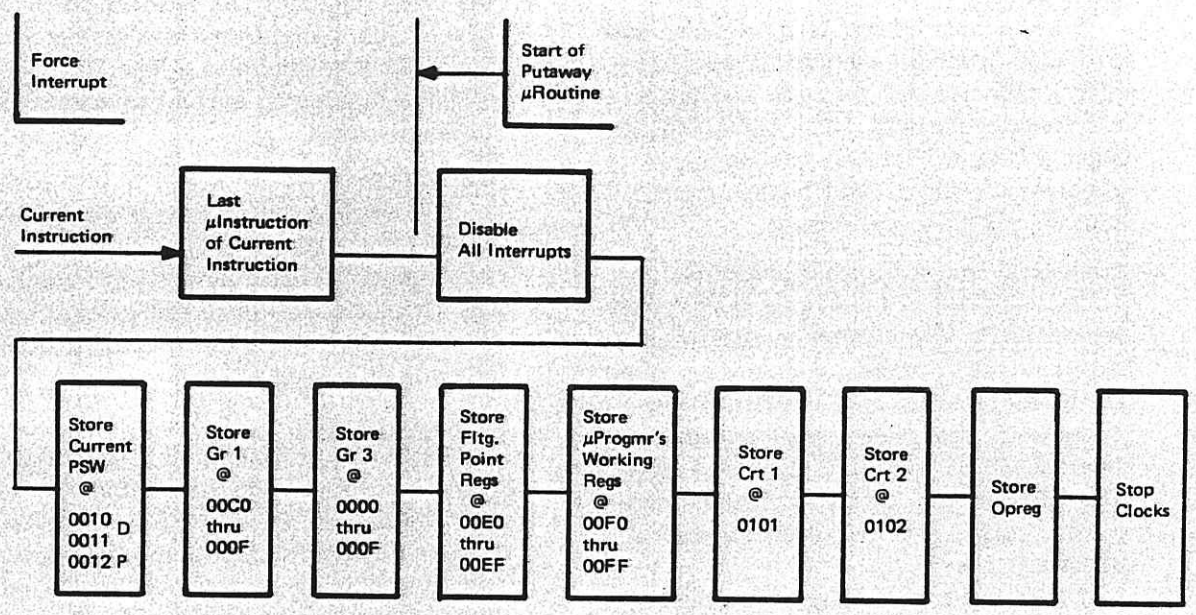
Single CPU Cycle

Suppose the computer is running (executing a program). To stop the computer at the end of a CPU cycle, XSTP2N is set to 1. The CPU is then halted and the I/O channels are reset at the end of the current CPU cycle. The CPU will remain halted until XCONT is set to 1 and then returned to 0. If XSTP1N = ZERO and XSTP2N = ONE, the CPU is advanced one CPU cycle and then halted. The CPU cycle is executed on the rising edge of XCONT. This is defined as single cycle. The minimum time for a CPU cycle is 0.20 us. The CPU is stopped with CPU clock 1 up. Note that during single cycle mode, Direct Memory Access (DMA) operations are possible only on the basis of one DMA per CPU cycle. Thus, for a 16-word DMA transfer, 16 CPU single cycles must be executed.

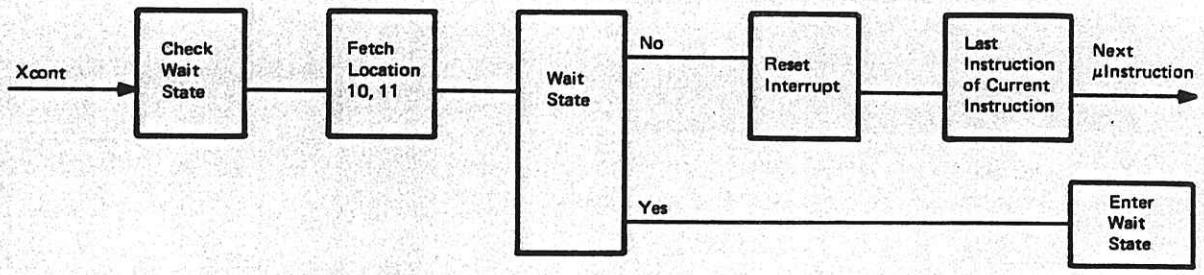
Instruction Stepping

Suppose the computer is running and the CPU is to be halted at the end of the current instruction. In this case, XSTP1N is set to 1 and at the end of the current instruction the CPU is halted. Note that the level of XSTP2N is a don't care in this case. Each time XCONT is raised to a 1 and returned to 0, a single instruction is executed. This is defined as instruction stepping. During instruction stepping the I/O channel will remain operating. If the CPU is halted, and XSTP1N and XSTP2N are then set to 0, the CPU will remain halted until XCONT is cycled to a 1 and returned to 0. The instruction is executed on the rising edge of XCONT.

To implement instruction stepping, an interrupt which causes the putaway routine to be entered is forced during the last micro instruction of the current instruction being processed. In the putaway routine (see figure) the current PSW, GPR set 1, 2, FPR set 1 and programmer's working register set, CTR #1, CTR #2 and OP Reg are stored and the clocks are stopped. To execute the next instruction, XCONT is pulsed which forces completion of the putaway routine. Here the condition of wait state is checked, then the current PSW is fetched. The instruction step interrupt is reset for one clock cycle, and a last microinstruction of the current instruction micro order is executed to begin execution of the next instruction, or the WAIT state is entered, depending on whether WAIT state was the condition prior to the interrupt.



Microcode Operations for Putaway Routine



Microcode Operations to Leave Putaway and Resume Operation

AGE Inhibits

Two signals on the AGE interface are provided to inhibit the CPU from recognizing I/O memory request, and inhibit all memory request. The signal XIREQ when a ONE inhibits all I/O memory request; the signal XISEL when a ONE inhibits all memory request.

Input Busy and Advance Simulation

Two signals are provided on the AGE interface for simulating the BUSY and Advance (ADV) signals which are outputs of the memory to the CPU. XIBUSY and XIADV equal ONE simulates a logic one in the memory BUSY and ADV signals respectively. The XIBUSY and XIADV at ZERO simulates a logic zero on the respective memory signals to the CPU.

These two signals along with the two memory inhibit signals allow the CPU and I/O to be checked out without the use of the memory unit.

System Reset

The XSYSRN signals allows the resetting of the CPU by external control. When XSYRN = 1, the CPU is reset and program execution is halted until XSYRSN is returned to 0.

Error Outputs

There are three error output signals denoted as XCPUPEN, XIOPEN and XGNGTN. For all signals an error has occurred when the signal is at low level and no error at a high level.

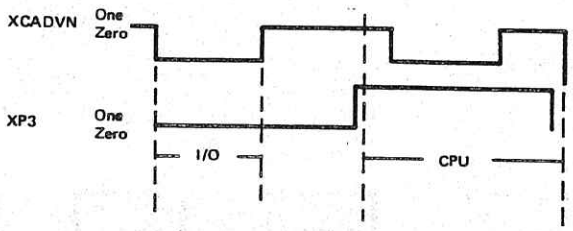
The XCPUPEN indicates a memory parity has occurred during a memory request. The XGNGTN signal indicates the Go-No counter has reached the count of 0. (This signal is not applicable on Space Shuttle machines.) The XIOPEN signal indicates an error has occurred in the channel.

Clock Outputs

Two signals denoted as XCP2, and XCTCLK are the CPU Clock Pulse 2, and Counter Clock respectively. Each output when at a high level indicates the clock is at a logic 1 and a low level indicates the clock is at a logic 0.

Memory Outputs

There are two memory related output signals denoted as XCADVN and XP3 on the AGE interface. The XCADVN signal when a logic 0 indicates the memory advance signal is being generated. The advance signal is generated as a result of a memory request by either the CPU or the I/O. The XP3 signal when at 1 indicates the CPU has issued the memory request. The XP3 signal is always raised before XCADVN is lowered. From these two signals, it can be determined if the CPU or the I/O memory request is being honored. See figure.



Memory Outputs

Time Level Outputs

Four CPU time levels denoted as XIFDI, XWAIT, XMSTL and XSATLN are on the AGE interface. the XIFDI when at ONE indicates the next CPU time level to be set is IFTCH. The IFTCH time level is the first CPU cycle executed at the beginning of every instruction. The XWAIT signal when at ONE indicates the CPU is in the WAIT state. The CPU will remain in the WAIT state until an interrupt is generated. The XMSTL when at 1 indicates the computer is in a stop state. This time level can only be issued when XSTPIN or XSTP2N = 1 and the CPU has halted. The XMSTL signal, when a 1, indicates the CPU has been halted.

The XSATLN signal when at ONE indicates a LAGE instruction is being executed. The LAGE instructions read a halfword from memory which is gated onto the AGE output bus. The memory halfword is used by the external equipment.

Channel Busy

The DNUD, when at ONE, indicates the channel is busy. When stopping the CPU at the end of an instruction, CPU or I/O registers cannot be displayed or loaded unless DNUD = 0 and XMSTL = 1. If an attempt is made to display or to load a register when the above two conditions are not true, the CPU or I/O operation being executed will interfere with the loading or display operation.

Load AGE and READ AGE Instruction Sequences

The LOAD AGE/READ AGE instruction provides computer software with the capability of entering AGE data into CPU registers via the AGE data register and/or outputting data from CPU registers to the AGE DATA BUS. The LOAD AGE/READ AGE instructions are RR format instructions in which an AGE control word is specified by Bits 16-31 of the R2 field (see Figure 1). Bits 0 - 15 of the R1 field are specified as the source of data in a LOAD AGE instruction. Bits 0 -15 of the R1 field are specified as the destination of data in a READ AGE instruction.

The sequence of events within the CPU during LOAD AGE/READ AGE is shown in Figure 2. Timing of these operations is further defined in Figure 3. Note that the microcode operations shown in Figure 2 are not on a one for one basis with actual microcodes, but are rather a block description of the microroutine which executes the given instruction.

An AGE interrupt is provided (XAITPN) to enable computer peripherals to gain CPU attention so that operator initiated LOAD/READ functions may be honored. This operation must be supported by software, which defines the particular operation through the AGE control word (see CPU Principles of Operation Manual).

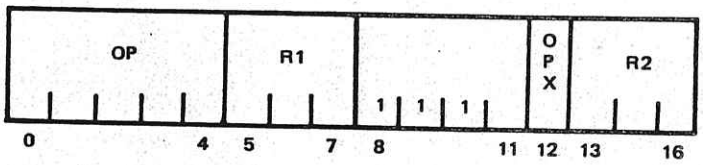


Figure 1. RR Format Used in Load AGE/READ AGE Instructions

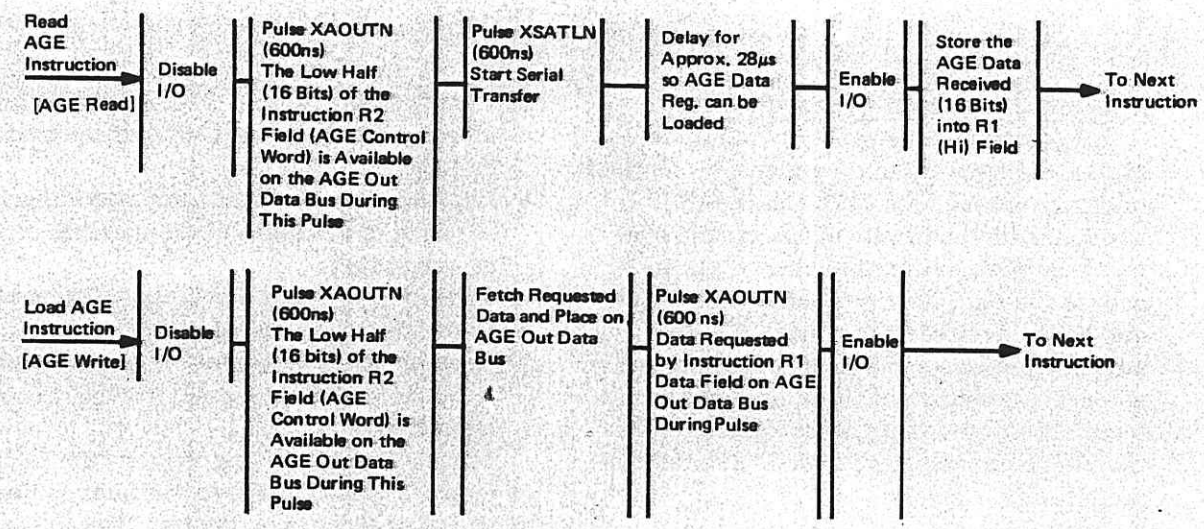


Figure 2. Microcode Operations for Load AGE/READ AGE Instructions

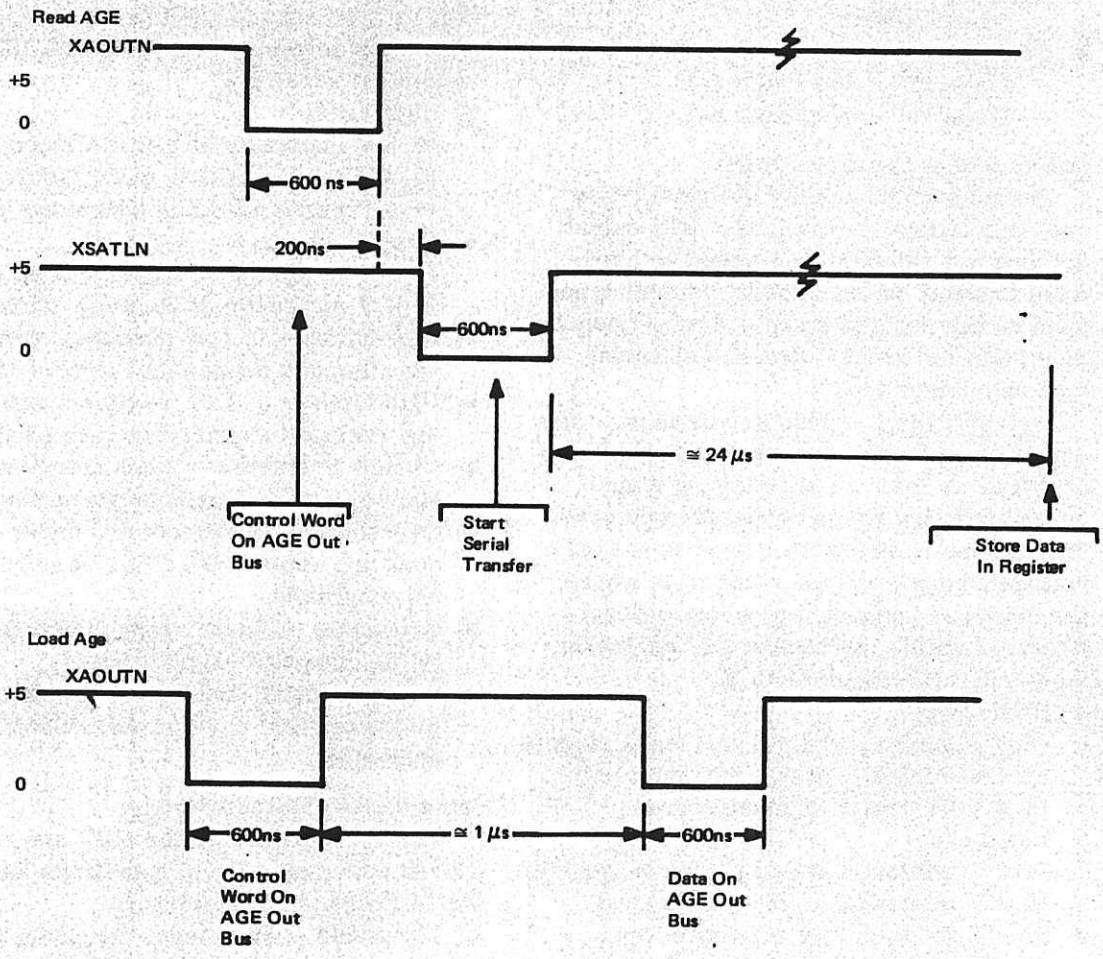


Figure 3. Load AGE/READ AGE Timing

75-A97-001

AGE Interface Signal Characteristics

All test signals between the CPU and the MST are single ended type signals. Ground integrity between the MST and the CPU is maintained by multiple signal ground connections through this interface. All inputs to the computer are received by special logic level receivers having high noise thresholds and are specified so that an open connector condition presents a logic zero to the internal hardware. At the interface, the logic signals are defined as follows:

- Inputs
  - Logic "1" = low voltage level
  - Logic "0" = high voltage level or "open" circuit
- Outputs
  - Logic "1" = See Table 1-1
  - Logic "0" = See Table 1-1

Output Signal Characteristics

Because of the variety of signal types and pulsewidths involved, the AGE output drivers are selected on a speed vs hardware tradeoff basis. The two driver types used on this interface are described below. (See table for cross reference of signal type and driver type.)

Driver Type I — This driver is used for signal types having minimum pulsewidths of 200 ns or less. The following signal characteristics are provided for two conditions; one in which the driver is the only voltage source, and the other case where the driver is connected to a Logic Level Receiver (LLR), or equivalent, having an input pull-up network to +5 V.

- High Level
  +5.5 V maximum at 0.0 mA (open circuit)
  +2.4 V minimum at 0.4 mA (sourced)
  +4.0 V minimum when connected to LLR.
- Low Level
  -1.0 V minimum at 0.0 mA (open circuit)
  +0.4 V maximum at 26 mA (sinking)
- Rise Time — 0.2 μs maximum when driving a lumped capacitance load of 250 pF.
- Fall Time — 0.2 μs maximum when driving a lumped capacitance of 250 pF.
- Circuit Protection — Each output withstands short to ground without permanent damage.
- Overshoot — Overshoot and undershoot does not exceed 20% of the nominal signal amplitude.
- Grounding — Each driver is referenced to the computer signal ground.
- Recommended Receiver — Logic Level Receiver (IBM part number 6088076) or equivalent.

Driver Type II — This driver type is used on output signals that have minimum pulse widths of greater than 200 ns. The following characteristics are provided for two conditions. One in which the driver is the only voltage source and the other case where the driver is connected to a LLR or equivalent receiver.

- High Level
  +5.5 V maximum at 0.0 mA (open circuit)
  +2.4 V minimum at 0.4 mA (sourced)
  +4.0 V minimum when connected to a LLR.
- Low Level
  -1.0 V minimum at 0.0 mA (open circuit)
  +0.4 V maximum at 20.0 mA (sinking)
- Rise Time — 0.2 μs maximum when driving a lumped capacitance load of 250 pF.
- Fall Time — 0.2 μs maximum when driving a lumped capacitance load of 250 pF.
- Circuit Protector — Each output withstands shorts to ground without permanent damage.
- Overshoot — Overshoot and undershoot does not exceed 20% of the nominal signal amplitude.
- Grounding — Each driver is referenced to the computer signal ground.
- Recommended Receiver — Logic Level Receiver (IBM part number 6088076) or equivalent.

Input Signal Characteristics

All test signals into the CPU are received with Logic Level Receivers having the following characteristics.

- Threshold — High level threshold is +3.0 V minimum
  — Low level threshold is +2.0 V minimum
- Common Mode Noise — ±1 V maximum

- Circuit Protection — All input lines withstand +15 V from signal line to ground without damage.
- Input Impedance — The typical input impedance is 5K ohms. Input currents are specified as -3.0 mA maximum at +0.4 V and +0.05 mA maximum at +15.0 V.
- Recommended Driving Source — Any TTL Series 5400 Nand gate.

SIGNAL AND DRIVER CROSS REFERENCE

Signal	Driver Type	Minimum Pulsewidth
XB000N - XB019N	I	100 ns Pulse
XMSTL	I	Level
XMINTN	I	200 ns Pulse
XMANTN	I	200 ns Pulse
XCP2	I	100 ns Clock
XCTCLK	I	100 ns Clock
XIFDI	I	200 ns Pulse
XP3	I	1 ms Pulse
XCADVN	I	450 ns Pulse
XRA00 - XRA11	I	100 ns Pulse
XRDOT	I	450 ns Pulse
XRPERN	I	200 ns Pulse
XSPERN	I	300 ns Pulse
XBUSYN	I	700 ns Pulse
XADEXN	I	100 ns Pulse
IOPEN	II	250 ns Pulse
CPUPEN	II	250 ns Pulse
GNGTN	II	250 ns Pulse
XSATLN	II	600 ns Pulse
X WAIT	II	Level
X AOUT	II	600 ns pulse

Cable Requirements

The cable type recommended for the AGE interface is a combination of twisted pair and coax cables with an overall outer gross shield. The twisted pair cable has a nominal impedance of 110 ohms line-to-line and the coax a nominal impedance of 90 ohms from center conductor to shield. The low side of each twisted pair and the shield of each coax is connected to signal ground at both the computer and tester ends. The outer gross shield is connected to the computer structure and tester frame via a low impedance RF connection. The table is a listing of all AGE signals and recommended wire type for each.

The AGE interface operates over a maximum cable distance between driving and receiving circuits of 15 ft.

AGE CONNECTOR PIN LISTING

Signal	AGE Pin	Recommended Wire Type
XWAIT	57	TP
XMSTL	58	TP Jumper Returns
DNUD	69	TP to Pin 70
RETURN	70	
XIADV	09	TP
XIBUSY	15	TP Jumper Returns
XISEL	16	TP to Pin 17
XIREQ	18	TP
RETURN	17	
XLDCLK	25	TP
XECLK1	26	TP Jumper Returns
XEXDI	37	TP to Pin 36
XEXSS	48	TP
RETURN	36	
IOPEN	49	TP
CPUPEN	50	TP Jumper Returns
GNGTN	59	TP to Pin 60
RETURN	60	
XSTPIN	51	TP
XSTP2N	52	TP Jumper Returns
XAION	62	TP to Pin 63
XSYSRN	64	TP
RETURN	63	
XINTN	53	TP
XBUMP	55	TP Jumper Returns
XRDIN	66	TP to Pin 54
XCONT	65	TP
RETURN	54	

AGE CONNECTOR PIN LISTING (cont)

Signal	AGE Pin	Recommended Wire Type
XCTCLK	05	Coax
RETURN	06	
XIFDI	110	Coax
RETURN	112	
XP3	119	Coax
RETURN	120	
XCADVN	75	Coax
RETURN	77	
XBION	83	Coax
XSATLN	88	Coax
RETURN	89	
XRA00	76	TP
XRA01	41	TP Jumper Returns
XRA02	42	TP to Pin 108
XRA03	109	TP
RETURN	108	
XRA04	10	TP
XRA05	25	TP Jumper Returns
XRA06	19	TP to Pin 81
XRA07	20	TP
RETURN	81	
XRA08	78	TP
XRA09	90	TP Jumper Returns
XRA10	100	TP to Pin 111
XRA11	101	TP
RETURN	111	
XGBAN	61	TP
XGBBN	29	TP Jumper Returns
XGBCN	30	TP to Pin 96
XRDOT	116	TP
RETURN	96	
XLDRDN	31	Coax
RETURN	11	
XRPERN	115	Coax
RETURN	107	
XSPERN	95	Coax
RETURN	94	
XBUSYN	87	Coax
RETURN	86	
XADEXN	85	Coax
RETURN	84	
XAOUTN	97	Coax
XAITPN	74	Coax

AGE CONNECTOR PIN LISTING (cont)

Signal	AGE Pin	Recommended Wire Type
XMINTN	106	TP
XMANTN	105	TP Jumper Returns
XEDIS	38	TP to Pin 39
XHSDC	40	TP
RETURN	39	
XAEXSS	24	TP
XGOSDR	34	TP Jumper Returns
XBOC	46	TP to Pin 35
XGDI	47	TP
RETURN	35	
XRDCLN	27	
RETURN	28	
XP12V	122	TP
XP5V	123	TP Jumper Returns
XN5V	124	TP to Pin 125
RETURN	125	
XB000N	13	TP
XB001N	14	TP Jumper Returns
XB002N	21	TP to Pin 22
XB003N	23	TP
RETURN	22	
XB004N	32	TP
XB005N	33	TP Jumper Returns
XB006N	43	TP to Pin 44
XB007N	45	TP
RETURN	44	
XB008N	56	TP
XB009N	67	TP Jumper Returns
XB010N	79	TP to Pin 68
XB011N	80	TP
RETURN	68	
XB012N	82	TP
XB013N	92	TP Jumper Returns
XB014N	102	TP to Pin 103
XB015N	104	TP
RETURN	103	
XB016N	113	TP
XB017N	114	TP Jumper Returns
XB018N	127	TP to Pin 121
XB019N	128	TP
RETURN	121	
XEO	01	Coax
RETURN	02	
XCP2	03	Coax
RETURN	04	

4.10 POWER SUPPLY

Modular Units - Functionally Oriented

The Modular Power Supply (MPS) provides modular functions with common design and hardware utilization.

The space shuttle general purpose computer uses two identical modular power supplies, one within the central processor unit (CPU) and one within the input output processor (IOP). The MPS converts 28 VDC input power to the required operating voltages.

Each MPS supplies operating voltages of +5 VDC, +12 VDC, -5 VDC for either CPU or IOP operation and a -12 VDC for IOP operation only. Additional logic circuits, used only in the CPU operation, monitors the power status for the IOP, EMU, and the CPU itself. At power up, at power down, or at loss of power, the MPS is sequenced on or off in an orderly fashion to assure that no stored data or program instruction is affected by input power transients.

Functional Operation

The power supply block diagram is shown on the opposite page. The Internal Voltage, Serial Dissipative Regulator (SDR) and the two Controller Assemblies are pluggable through a 49 pin connector into the power supply backpanel while the DC/DC converters consist of frame mounted high power dissipative components. The EMI filter is a separate module mounted in the CPU adjacent to the power supply module. The functional blocks in the power supply are as follows:

- DC/DC Converters (2)
- Internal Voltage Assembly
- Controller Assembly 1
- Controller Assembly 2
- Series Dissipative Regulator Assembly

DC/DC Converter

The CPU power supply contains 2 DC/DC converters, converter 1 provides the +5 V, 50 A output while converter 2 provides the +12 V, 19.3 A output and preregulated -8 V and -15 V (±8%) input voltages to the SDR Assembly. Each converter consists of an input ripple/energy storage filter, four power switching transistors and associated drive circuitry, a transformer/inductor assembly, output rectifiers and output filter capacitance. The conversion is accomplished in a push-pull (pulse width modulated) converter configuration operating at 25 kHz.

Converter 2 has -8 V and -15 V outputs. Each have individual rectifiers and output filters. The +12 V output, which is sensed by the controller, is regulated to +3, -2%. The other outputs are effectively preregulated to ±8% by the control loop. Further regulation is provided by the series dissipative regulators.

The controller assembly provides a clock logic signal, which contains the variable rate and pulse width control information, for converter operation. The converter driver logic converts the clock signal to alternate current drive to switch circuits for push-pull converter operation. Transformer coupled converter drive allows total ground isolation between input and output returns within the power supply.

Internal Voltage Operation

The Internal Voltage (IV) Assembly contains a self-starting low power single ended converter and the Power Off Intercept (POI) detector.

The IV converter voltages P40VI (+40 V P12VI (+12 V), P5VI (+5 V) and M6VI (-6 V) are used for internal power supply operations only.

The POI detector issues the POI signal only after the following two conditions have been met. 1) The input voltage is of sufficient amplitude to assure regulation of the DC/DC converter outputs and 2) the I. V. output voltages are within 95% of their nominal values which assures correct operation of the power system. The DC/DC converter operation is inhibited until POI is issued.

Controller 1 Operation

Controller 1 contains the control functions, Clock 1 and Start, for the +5 V DC/DC converter. The over current (OI) detector inhibits the clock 1 output when the +5 V output current exceeds an approximate 50 Amp limit. Other circuits monitor the power supply output voltages and generate a fault output signal when either an under voltage (-5%) or over voltage (+5%) condition occurs during normal operation.

The memory clamp signal (IMS/PORN) which inhibits memory operation during power up or power down transitions is also generated within controller 1.

Controller 2 Operation

Controller 2 performs the output voltage sense, Clock 2 and OI current sense functions for the +12 V DC/DC converter. Additional logic generates the Power Off Interrupt Delayed (POID), Power Down and Master Power On Reset (MPOR) required for the GPC system operation.

SDR Operation

The Series Dissipative Regulator (SDR) performs the ±2% regulating functions on the Pre-regulated DC/DC converter output voltages to provide the regulated CPU system voltage of -5 V. The control portion is powered by P12VI and M6VI internal voltage providing a low differential voltage from input to output minimizing power dissipation in the SDR. Foldback over current detection is also incorporated to attain low power dissipation during output overload conditions. An output overload condition only affects the associated SDR output.

Power Supply On-Off Characteristics

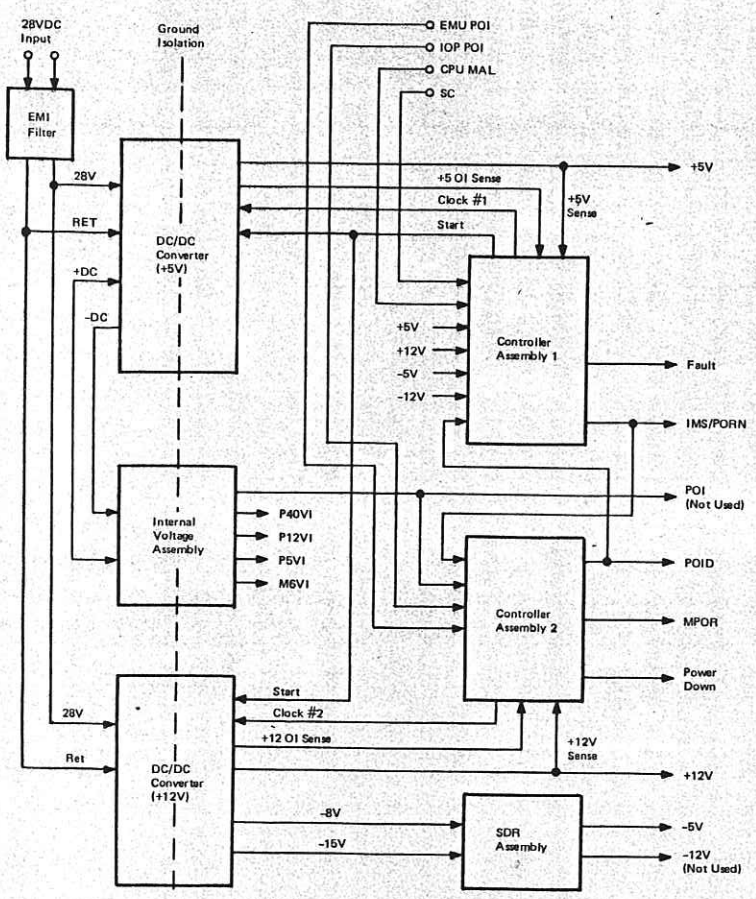
The power supply internal voltage circuit is the first section of the power supply that responds to the application or restoration of input power. When the filtered 28 V power reaches a voltage level of approximately 12 VDC, the internal voltage circuit starts. Regulated bias voltages are developed with a controlled rise time of 2 ±1 ms, during which time a continuously clamped POI signal prevents main power supply functions from beginning. A voltage detector that is set at 95% of nominal bias voltage levels releases the POI clamp when the filtered 28 V input has reached a minimum level of 21 VDC. The main converter-regulators are then enabled and regulated output voltages are generated with a controlled rise time of 4 ±1 ms. The Master Power-On Reset (MPOR) signal is at a down level condition until the regulated voltage monitor circuits indicate an in-tolerance condition. MPOR is available at the computer interface for external use and will be used in the power supply as an inhibit signal to the Inhibit Main Store (IMS) clamp to ensure that memory cannot be addressed when MPOR is at a down level. The IMS signal is at a down level condition to inhibit main store operations from sensed power application until 80 ±20 ms to ensure stabilization of the computer timing oscillator and the main store voltages. These events are illustrated in the timing diagram.

The power-down sequence is initiated by the power-off interrupt signal, whenever the filtered 28 V input voltage decreases to a level of 17.5 VDC. Upon receipt of the POI signal, the processor must complete any in-process instruction and store the general registers and program status data. A POI delayed (POID) signal is issued 100 μs later to the main store which, after completion of the current memory cycle, issues the signal Storage Complete (SC). The issuance of SC initiates the actual power-down sequence. The power supply control signal START inhibits the Master Power On Reset (MPOR) signal, which initiates the Inhibit Main Store Clamp (IMS) and sets all external logic signal interfaces to the logical one state. Once the power-down sequence has started, it continues to completion regardless of subsequent power line status.

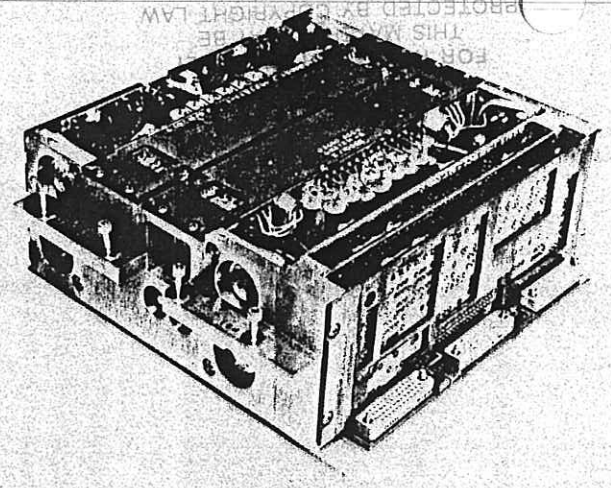
There is no danger of a power-down "race" condition, since the power supply output filters can sustain regulated voltage levels for a minimum of 25 μs, following the

inhibit application. The status of the general system clamp is assured during the power system output voltage decay interval.

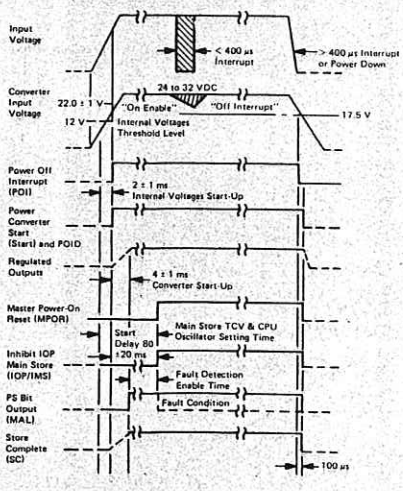
The CPU and IOP power supplies each have built-in energy storage capability in the form of input filter capacitors. The time interval over which the power supply can continue to function without applied power, starting at a minimum input voltage of 23.2 VDC, is 400 μs before issuance of POI.



CPU Power Supply Block Diagram



Modular Power Supply (MPS)



Power Supply On/Off Timing Diagram

75-A97-001

4.11 LOGIC TECHNOLOGY

Proven Circuit Families Fully Characterized

Three logic-circuit families, selectively applied for function and speed, are fully characterized for use in the CPU.

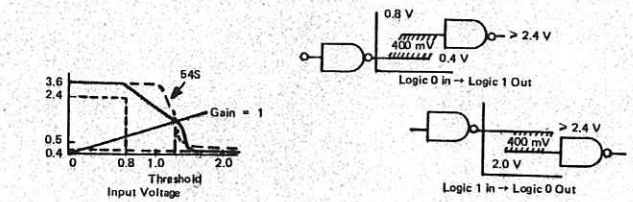
The CPU uses TTL (transistor/transistor logic) gates as the basic building block. The entire TTL family is designed so that each TTL version interfaces directly with each other in the same system, to provide the computer with the optimum speed/power product. The circuit diagrams for the three versions of TTL unit logic cricuits are shown.

Each version of TTL operates identically, and this operation can best be explained by examining the transfer curve. With the input low, the output will be two base-to-emitter voltages (V_{BE}) below V_{cc} or, typically, 3.6 V. When the input is allowed to rise to 0.8 V, the output voltage will be greater than 2.4 V, as guaranteed by the source control drawing (SCD). A further increase in input voltage will turn on the phase-splitter transistor and the output inverter, respectively, causing the output voltage to fall. The maximum down level is specified to be 0.4 V, maximum, at its rated load. Therefore, a 400-mV noise margin is guaranteed. Likewise, the SCD logical 1 input test condition is 2.0 V, which is 0.4 V below the guaranteed minimum logic 1 output of 2.4 V. In both states, guaranteed noise immunity is 400 mV. The above levels are guaranteed by the manufacturers over the full military temperature range of -55°C to +125°C. Note that the 54S family has a superior transfer characteristic, which raises the DC input level for maintaining the output high-level condition; this compensates for the slightly higher down-level offset voltage of 0.5 V to maintain noise margins.

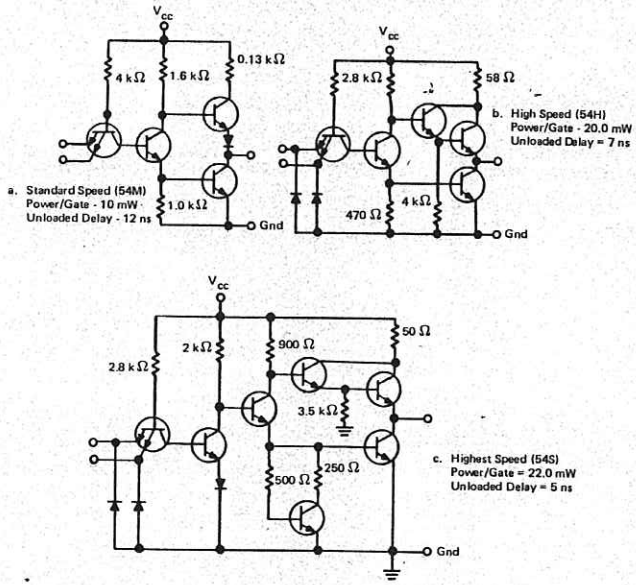
Detailed characterization of parts from each family consists of measuring all static input/output circuit parameters and dynamic performance under worst-case conditions, over the full military temperature range. The resulting vast file of TTL circuit performance data was used to develop the electrical acceptance test screens used to procure these circuits. These screens are more restrictive than the vendor's published screens and have proven very effective in maintaining control over TTL circuits purchased from three vendors. For instance, equipments developed from the original unit logic 54XX TTL circuits have had trouble-free production lives in excess of 5 years. These same screening methods are in use today, and the same characterization and screening philosophy is applied to new circuits as they are added to the list of circuits approved for use in Advanced System/4 Pi technology.

In addition to the electrical characterization, comprehensive examinations of the construction and processing of each circuit chip and package were performed. As part of this procedure, microsections of the chip and package are made to evaluate such parameters as diffusion depths, oxide thicknesses, metal thicknesses, intermetallic interfaces, and die attachments. Photo enlargements of the chips and package permit studying line spacings, diffusion widths, registration, bond placements, etc. Finally, scanning electron microscope micrographs reveal passivation, metal grain structure, and metallization integrity at oxide steps.

The logic design for the CPU takes advantage of maximum density integrated circuit functions. The complex functions and unique custom designs used are shown in the tables. All are bipolar TTL or TTL-compatible, with single-level metallization, and are packaged in 14- and 16-pin flatpacks.



TTL Transfer Curve *Guaranteed DC Noise Immunity*



Series 54 TTL Unit Logic Circuit Diagrams

UNIT LOGIC MODULES

Standard-Speed Series 54M	High-Speed Series 54H	Highest-Speed Series 54S
Quad-2 NAND	Dual, 4-Input NAND	Quad-2 NAND
Quad-2 Dot NAND	Single, Quad AOI	Triple-3 NAND
Quad-2 NAND	Dual Trigger	Dual 4 NAND
Triple-3 NAND		13-Input NAND
Dual-4 NAND		Dual Trigger
8-Way NAND		Hex Inverter
Hex Inverter		4-2-3 2 AOI
Dual 2-2 AOI		3-Way AND
2-2-2-2 AOI		Quad DFF
Quad Ex-OR		Quad Ex-OR
Dual Trigger		

COMPLEX FUNCTION AND MSI MODULES

Function	TTL Type
4-Bit Arithmetic Logic Unit	54S
Carry Look-Ahead Module	54S
8-to-1 Multiplexer	54S
8-Bit Parity Checker	54M
Dual Four-Bit Multiplex	54M
Dual Four-Bit Dot Multiplex	54M
Two-Bit Gated Register	54M
Four-Bit Shift Register	54M
Four-Bit Binary Counter	54M
Three-to-8 Decoder	54M
64-Bit Random-Access Memory	-
256 x 4 Programmable Read-Only Memory	-
9-Bit Parity Checker	54S
Dual Diff. RCV	-
Dual Diff. DRV	-
Dual Tri-State Diff. DRV	-

Section 5

CPU MICROCODE OPERATIONS

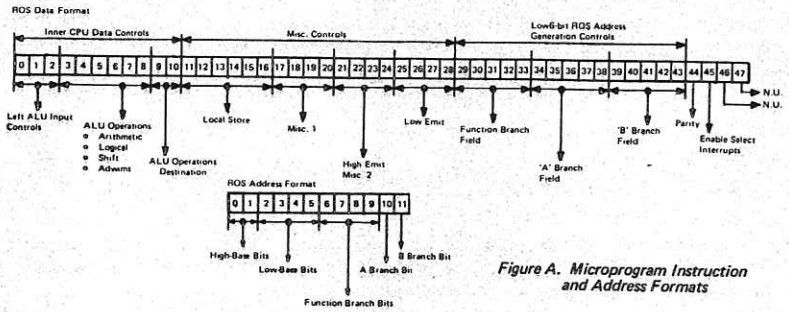


Figure A. Microprogram Instruction and Address Formats

Table 1
MISCELLANEOUS 1 CONTROL FIELD

ROS Data Bits 17 18 19 20	Function No.	Mnemonic	Function
0 0 0 0	0		No Operation
0 0 0 1	1	E → B	Full Emit to the B Register
0 0 1 0	2		Not Used
0 0 1 1	3		Not Used
0 1 0 0	4	LST/LE	Load or Set Per Low Emit | High Emit
0 1 0 1	5	LCL/LE	Load or Clear Per Low Emit | Specifies Function
0 1 1 0	6	E → ROS	Full Emit to High 6 Bits of ROS Address
0 1 1 1	7	LSTEX	Last Execution Micro-order in a macroinstruction
1 0 0 0	8	AMISC	'A' Miscellaneous Codes Specified by Low Emit
1 0 0 1	9	CMISC	'C' Miscellaneous Codes Specified by Low Emit
1 0 1 0	10	BMISC	'B' Miscellaneous Codes Specified by Low Emit
1 0 1 1	11	DMISC	'D' Miscellaneous Codes Specified by Low Emit
1 1 0 0	12	RD/LE	Read Operations Specified by Low Emit
1 1 0 1	13	RCW/LE	Read/Compute/Write Operations Specified by Low Emit
1 1 1 0	14	LE → RB	Low Emit to Low ROS Base Address Bits
1 1 1 1	15	W/LE	Write Operation Specified by Low Emit

Table 2
HIGH EMIT/MISC 2 CONTROL FIELD (USE WITH FUNCTIONS 4 & 5 OF MISC 1)

ROS Data Bits (High Emit) 21 22 23 24	Item No.	Mnemonic	Function
0	0	S2413	Load Status Registers 24, 1, 2, & 3
1	1	S47	Load Status Registers 4, 5, 6, & 7
2	2	S811	Load Status Registers 8, 9, 10, & 11
3	3	S1215	Load Status Registers 12, 13, 14 & 15
0 1 0 0	4		Not Used

5.1 CPU MICROCODE OPERATIONS

CPU Functions Are All Controlled by Execution of Microinstructions

Each 48-bit microinstruction controls a CPU cycle.

This section describes the Space Shuttle CPU microcode operations. The Shuttle CPU is a microprogrammable computer and all functions of the CPU are controlled by the execution of microinstructions. The CPU has a ROS capacity of 2048 microinstructions each of which is 48 bits wide. Figure A illustrates the format of the 48-bit microinstruction. The 48 bit wide microinstruction consists of three major fields — the Inner CPU Data Control field, the Miscellaneous Control Fields, and the Low 6-Bit ROS Address Generation Field. Unlike macroinstructions, the CPU microinstructions are not executed in sequence by incrementing the address. Each microinstruction specifies the ROS address of the next microinstruction. Figure A illustrates the 12-bit ROS address field format. Every microinstruction specifies the low 6 ROS address bits of the next microinstruction, while the high 6 ROS address bits remain unchanged unless altered specifically by the miscellaneous control fields.

Inner CPU Data Control Field

The Inner CPU Data Control Field (ROS data bits 0 through 16) manipulates the data for arithmetic operations, macroinstruction address generation operations, logical operations and shift operations. This field also controls the read and write operations into local store which contains the general and floating point registers. This field is described fully in the inner data flow section of this document and will not be described in this section.

- Instruction fetch and decode operations
- Main storage read and write operations.

This field will be described in more detail in the following paragraphs of this section.

Low 6 Bit ROS Address Generation Field

This field, whose format is shown in Figure A (ROS data bits 29 through 43) is divided into three subsections. The first subsection, represented by ROS data bits 29 through 33, is the Function Branch Field which specifies the ROS address bits 6 through 9 directly or indirectly by function. The Function branch field by itself provides a capability for a 16 way branch. The second subsection, represented by bits 34 through 38, is called the 'A' Branch field and specifies ROS address bit 10. The third subsection represented by ROS data bits 39 through 43 is called the 'B' Branch field and specifies ROS address bit 11. This address field is described in greater detail in subsequent paragraphs of this section.

Comments

ROS data bit 44 is the parity bit and is set to a value such that the 48 ROS data bits have odd parity. ROS data bit 45 when equal to a '1' enables the following interrupts:

- CPU Main Store Data parity error
- I/O interface address parity error
- I/O interface data parity error
- Extended memory storage data error
- Address specification

bit 17 = 0, both the high and low emit fields (ROS data bits 21 through 28) are used to further specify the function. When ROS data bit 17 = 1, only the low emit field (ROS data bits 25 through 28) is used to specify the Miscellaneous 1 field, while the high emit field (ROS data bits 21 through 24) is used independently as a Miscellaneous 2 field. The following paragraphs in this section will describe each function of the Miscellaneous 1 control field.

Function 0, ROS data bits 17, 18, 19 and 20 equal to '0000' is a no operation function.

Function 1, E →B, is used to load the high and low emit field directly into the high byte of the 'B' register, the remaining 24 bits are set to zero for this operation. With this function, constants can be generated in 8 bit chunks.

Functions 2 and 3 are presently not used.

Function 4, LST/LE, is used to 'load' or 'set' status registers, interrupt latches, and 'D' register bits. The latches or registers that are to be operated on are specified by the high emit field in groups of four and are listed in Table 2. The low emit field is used to either specify the data to be entered into the registers or to specify the register within the group that is to be set. An example of loading would be if ROS data register bits 17 through 28 equals 0100 0011 0101, respectively. This binary number combination selects Function 4 (LST/LE) in Table 1 which will load status registers 12, 13, 14 and 15 (Table 2, item 3) with the bits 0101, respectively. This operation is identical for register groups under items 0, 1, and 2 in Table 2. An example for setting registers would be if ROS data register bits 17 through 28 equals 0100 0111 0010, respectively. The first four bits of the binary number combination selects Function 4 (LST/LE) in Table 1. The next four bits select the Status Registers 28, 29, 30, and 31 (Table 2, item 7) to be operated on. The last four bits define which of the status registers specified in Table 2 item 7 is to be set. In this case with a binary code of 0010 status register 30 will be 'set' the others will not be altered. This operation is identical for the register groups specified by items 5 through 15. Table 3 lists the status registers and provides a description of each to give a better understanding of the Miscellaneous 1, Function 4 (LST/LE) operation. The microcode also branches on the status register bits which will be described in later paragraphs. The interrupt latches are set by the interrupt function, which is described under the "CPU Interrupt Structure" and by the Miscellaneous 1 Function 4 (LST/LE) ROS data register fields in the same manner as described for the status registers (Table 2 items 8, 9, 10, 12, 13). Miscellaneous 1, Function 4 (LST/LE) also can set the 'D' register bits 16, 17, 18 or 19 (Table 2, item 15) which are part of the current PSW. 'D' register bits 16 and 17 are the condition code indicators while 18 and 19 are the carry and overflow bits respectively.

Table 3
STATUS REGISTER ASSIGNMENTS

Status Reg	Function
00	'A' Reg Sign Bit
01	
02	
03	Exponent Counter: Used for handling the characteristic in floating point operations. Status Reg. 24 is the high bit of this counter and is used to detect characteristic overflows (O. F.) or underflows (U. F.).
04	
05	
06	
07	
08	
09	Iteration Counter: Used by micro-code to perform micro 'BRANCH ON COUNT' operations.
10	
11	
12	
13	Spill Status Bits: Used by shift micro operations to provide temporary storage for spilled bits.
14	
15	
16	Enables store protect logic when a 'one'
17	Store protect bit generator (only if enabled by STAT 16)
18	Performing 'DETECT' micro routine when a one
19	Always a zero
20	Interrupt in progress (Used by interrupt handling micro routines)
21	System Reset Register
22	Failure Indicator Register
23	Reserved for I/O operations (not used)
24	Exponent O. F. /U. F. Register (see description under STATS 1-7)
25	'B' Register Sign
26	ALU Output Sign
27	ALU Output 32-bit Zero Detect
28	Fixed Point Overflow Register
29	Fixed Point Carry Register
30	General usage for micro programmer
31	

Function 5, LCL/LE, is used to 'load or clear' the status registers, interrupt latches, and the 'D' register bits. The load function for items 0, 1, 2, 3 in Table 2 is identical to that described under Function 4. The 'clear' operations for items 5 through 15 perform in the same manner as the 'set' operation described under Function 4 (LST/LE) with the exception that ROS data register bits 25 through 28 specify which register or latches in the group of four is to be cleared.

Function 6, E→ROS, is used to change the high 6 bits of the ROS address of the next microinstruction to be executed. The high 6 ROS address bit illustrated in Figure A are labeled as the High-Base bits and the Low-Base bits. These 6 ROS address bits always remain the same until altered by the Function 6 (E→ROS) micro-operation.

Function 7, LSTEX, is used to identify the last micro operation of a macroinstruction. This function works in conjunction with the 8-bit emit field specified by ROS data register bits 21 through 28. The first two ROS data register bits 21 and 22 specify what 'D' register bits 16, 17, 18 and 19 will be (first 4 bits of the 2nd halfword of the current PSW). This function is illustrated in Figure B. An example of the Function 7 (LSTEX) operation is when ROS data register bits 21 and 22 are '1' and "0" respectively and the current microcode arithmetic result is greater than zero. Figure B shows that the 'D' register bits 16 and 17 (condition code) will be set to "0" and "1" respectively, and the 'D' register bit 18 will be set to a '1' or '0' depending on whether a carry occurs during the Function 7 (LSTEX) microcode operation. Also, if an overflow occurs 'D' register bit 19 will be set to a '1'. It should be noted once 'D' register bit 19 is set to a '1' it can only be reset to a '0' by a Function 5 (LCL/LE) operation (see Table 1). The remaining 6 ROS data register bits (23 through 28) are used directly as the high 6 bits of the ROS address for the next micro operation. These bits are shown in Figure A in the ROS Address Format as High-Base Bits and Low-Base Bits. Since Function 7 (LSTEX) is used to indicate the last micro-order executed per macroinstruction the CPU will decide (via hardware) whether a memory read operation is needed during the next micro-order to perform an instruction fetch. This decision is based on interrupt conditions, next macroinstruction address, and the condition of the Next Instruction Register latch.

Function 8, AMISC, is an 'A' miscellaneous field used to perform a variety of miscellaneous functions. Table 4 lists those functions which are specified by the ROS data bits 25, 26, 27, and 28 (low emit field). It should be noted at this time that for Functions 8 through 15 the high emit field (ROS bits 21, 22, 23, and 24) is free for use as an independent Miscellaneous 2 field, which will be described later. Item 0 (DISIO) and 1 (ENIO) in Table 4 are used to enable or disable DMAs. If a DMA is in process when a microinstruction is executed to disable DMAs, the DMA memory cycle in process will complete properly and then subsequent DMA memory cycles will be disabled by the CPU. Item 2 (DLC) is only used during Read Compute Write (RCW) Memory operations and is used to execute the write portion of the RCW operation. This will be described more fully in the third section which deals with memory microcode operations. Item 3 (STCL) stop clocks is used at the end of the putaway microroutine. This microcode function will stop the clocks, but still allow DMA operations to occur. Item 4 (RSINT) Reset Interface is used to reset the I/O - CPU interface logic. RSINT, when executed, activates the channel reset I/O interface line. Item 5 is not used. Item 6 (INVPAR), Invert Parity, when used sets a hardware latch. When this latch is set during a memory write operation, the parity bit stored is inverted causing this memory location to have bad parity. When the Invert Parity latch is set while reading memory, good

memory locations will look like they have parity errors. This function is used by the 'Detect' instruction. Items 7 and 8 are not used. Item 9 (ENMIN) is used to enable macrointerrupts. Items 10 and 11 are not used. Item 12 (REQIOA) is used in conjunction with a Miscellaneous 2 field item 4 (RQBS) (Table 5) Request Bus Operation. This function then sets up the bus for a non-memory operation involving a Program-Controlled I/O data transfer operation. The very next microoperation normally specifies a bus operation (a Function 12 or 15 operation, Table 1). When this bus operation is specified the hardware locks up the bus and it remains this way, allowing Program-Controlled I/O data transfers, until a Reset Acknowledge (RESACK) micro-operation is executed (shown in Table 4, item 15). Item 13 (REQIO) Request I/O operation must be used in conjunction with a Miscellaneous 2 field item 4 (RQBS) (Table 5) Request Bus Operation. This function like item 12 also sets up the bus for nonmemory bus operations. An example would be loading the hardware portion of the counters. This will be defined later. In this case unlike item 12 (REQIOA) the next bus request operation (generally Function 11, 12 and 15 in Table 1) will not lock up the bus. Item 13 (REQIO) is used for CPU bus operations such as loading counters, and reversing upper and lower data halfwords. This data will be more fully explained under the Miscellaneous 1, Functions 11, 12, and 15.

Function 9 (CMISC), in Table 1 is the 'C' Miscellaneous operations which uses the low emit field for operation identification. Table 5 lists the various items specified by the low emit field. Item 0 (LEC/17) is a specialized function which is used for floating point normalization operations. Item 1 (SHC→S) is used for shift microoperations. Items 2, 3, 5, 6, and 7 in Table 5 are used to load or set the various status registers. The microcode at a later period can then branch (via "A" & "B" branch fields) on these status registers which reflect the various ALU results described in Table 5. These status branches will be further described. Items 8 through 15 are used for incrementing or decrementing by "1" the status counters specified. Items 8 through 11 are used for microcode looping. A maximum of 16 iterations may occur if the status registers specified are loaded with zero's. Items 12 through 15 are used in the same manner but are designed for use in floating point operations specifically in the normalization routines.

ROS Data Reg. Bit 21	ROS Data Reg. Bit 22	Operation	Current Program Status Word (PSW) Condition Code Bits ('D' Register Bits 16 & 17) 00	01	10	11	Carry Bit 'D' Reg Bit 18	Overflow Bit D Reg Bit 19
0	0		No Change	No Change	No Change	No Change	No Change	No Change
0	1	Logical	Result = 0	Not Used	Not Used	Result ≠ 0	No Change	No Change
1	0	Arith 1	Result = 0	Result > 0	Not Used	Result < 0	Carry	*Overflow
1	1	Arith 2	Result = 0	Result > 0	Not Used	Result < 0	No Change	

*If overflow occurs bit 19 is set to a '1'. Once set it is never reset by this function.

Figure B. Current Program Status Word

Table 4
'A' MISCELLANEOUS CODES (USE WITH FUNCTION 8 OF MISC 1)

ROS Data Bits (Low Emit) 25 26 27 28	Item No.	Mnemonic	Function
0 0 0 0	0	DISIO	Disable Direct Memory Access (DMA) Operations
0 0 0 1	1	ENIO	Enable Direct Memory Access (DMA) Operations
0 0 1 0	2	DLC	Data Load Clock (Used only for Read/Compute/Write Oper.
0 0 1 1	3	STCL	Stop Clocks
0 1 0 0	4	RSINT	Reset Interface
0 1 0 1	5		Not Used
0 1 1 0	6	INVPAR	Invert Parity
0 1 1 1	7	TERRA	Not Used
1 0 0 0	8		Not Used
1 0 0 1	9	ENMIN	Enable Macro-interrupts
1 0 1 0	10		Not Used
1 0 1 1	11		Not Used
1 1 0 0	12	REQIOA	Request I/O and Acknowledge
1 1 0 1	13	REQIO	Request I/O
1 1 1 0	14		Not Used
1 1 1 1	15	RESACK	Reset Acknowledge

Table 5
'C' MISCELLANEOUS FIELD (USE WITH FUNCTION 9 OF MISC 1)

ROS Data Bits (Low Emit) 25 26 27 28	Item No.	Mnemonic	Function
0 0 0 0	0	LEC/17	Load Exp Cntr 1 thru 7 from ALU Left Input; Zero STATS 12 thru 15; Zero STAT 24; Clear A & B Register Bits 0 thru 7****
0 0 0 1	1	SHC → S	ALU Left Input 10 thru 13 (shift Count) to STATS 5 thru 11; ALU Output 14 and 15 to STATS 30, 31
0 0 1 0	2	LSC/47	Load Stat Cntr 8 thru 11 from Left Input 4 thru 7
0 0 1 1	3	IASS	Invert A sign Stat 0
0 1 0 0	4		
0 1 0 1	5	SSS	ALU Sign Bit (ALU 0) to Sign Stat 26
0 1 1 0	6	SZD	Set Zero Detect STAT/ALU Zero Output
0 1 1 1	7	SCO	Set Carry Stat 29 and Overflow Stat 25/ALU Carry/Overflow Conditions*****
1 0 0 0	8	+1 SC	Increment Stat Cntr (Stats 8 thru 11)
1 0 0 1	9	+1 SCB	Increment Stat Cntr (Stats 8 thru 11); Reuse func. Br until Cntr = 0*
1 0 1 0	10	-1 SC	Decrement Stat Cntr (Stats 8 thru 11)
1 0 1 1	11	-1 SCB	Decrement Stat Cntr (Stats 8 thru 11); Reuse func. Br until Cntr = 0*
1 1 0 0	12	+1 EXC	Increment Exponent Cntr (Stats 1 thru 7)
1 1 0 1	13	+1 EXCB	Increment Exponent Cntr (Stats 1 thru 7); Reuse func. Br until 'A' Reg Bit 0 ≠ Bit 1**
1 1 1 0	14	-1 EXC	Decrement Exponent Cntr (Stats 1 thru 7)
1 1 1 1	15	-1 EXCB	Decrement Exponent Cntr (Stats 1 thru 7); Reuse func. Br until 'A' Reg Bits 8-11 ≠ 0***

*Circuitry anticipates count of zero by recognizing prior count; therefore count of 0000 will allow 16 iterations to occur.
**Circuitry anticipates A0 ≠ A1 based on ALU 1 & 2; therefore the micro instruction expected is "A Reg LI" operation. The micro instruction preceding the repeating instruction must contain "A (B, C, or D) Reg → A (B, C, D)" for proper operation. Also, a minimum of one micro instruction is allowed by the repeating operation; therefore, a "normalized" condition must be separated by a prior branch.
***Circuitry anticipates 8 -11 = Zero based on ALU bits 12 - 15; therefore an "A Reg → A" operation must precede the repeating instruction. Also a minimum of one micro instruction is allowed by the repeating operation; therefore a "normalized" condition must be separated by a prior branch.
****A & B reg. HI bits will not clear if used as destinations by ALU operations (ROS Data Bits 9 & 10).
*****These STATS set by prior micro order results of ALU Output.

Table 6
'B' MISCELLANEOUS FIELD (USE WITH FUNCTION 10 OF MISC 1)

ROS Data Bits (Low Emit) 25	26	27	28	Item No.	Mnemonic	Function
0	0	0	0	0		Not Used
0	0	0	1	1		
0	0	1	0	2		
0	0	1	1	3		
0	1	0	0	4		
0	1	0	1	5		
0	1	1	0	6		
0	1	1	1	7		
1	0	0	0	8	DECB	Decode 'B' Set NIR OK Latch
1	0	0	1	9	DECE	Decode 'E' Set NIR OK Latch***
1	0	1	0	10	DECO	Decode 'O'
1	0	1	1	11	N → O	Next Instruction Register to Operational Register
1	1	0	0	12	RNOK	Reset NIR OK Latch (Inhibit "D" Reg Clock Bits 16 thru 31)
1	1	0	1	13		Not Used
1	1	1	0	14		
1	1	1	1	15		

RR — Read LS/OP 13-15
SRS — Op 8-13 → B_0 - 15; Read LS/Op 14 & 15
LRS — Stat 9-15 → B_0 - 15; Read LS/Op 14 & 15*
LRX — Stat 5-15 → B_5 - 15; Zero's to B_0 - 4; Read LS/Op 14 & 15*

*Read zero's LS when Op 14 & 15 = 11
**Read zero's LS when Stat 0-3 = 0000
***Status Register high to Next Instruction Register (NIR)

Table 7
'D' MISCELLANEOUS FIELD (USE WITH FUNCTION 11 OF MISC 1)

ROS Data Bits (Low Emit) 25	26	27	28	Item No.	Mnemonic	Function	
0	0	0	0	0	L → RBR	Load ROSBR from ALU Left Bits 20-31	* **
0	0	0	1	1	L → NIR	Load NIR from ALU Left Bits 16-31	
0	0	1	0	2	L → PD	Load 'P' Reg Bits 8 thru 11 from ALU Left Bits 24-27	
0	0	1	1	3	L → OP	Load OP from ALU Left Bits 16-31	
0	1	0	0	4	L → C0	Not Used (reserved)	
0	1	0	1	5	L → C1	Load Interval Timer 1 from ALU Left Bits 16-31	
0	1	1	0	6	L → C2	Load Interval Timer 2 from ALU Left Bits 16-31	
0	1	1	1	7	L → P	Load 'P' Reg from ALU Left Bits 16-31***	
1	0	0	0	8		(No Op)	
1	0	0	1	9	NIR → B	Load 'B' Reg Bits 16-31 from NIR	Zeroes → B_0 - 15
1	0	1	0	10	ST → B	Load 'B' Reg Bits 16-31 from Stat Bits 0-15****	
1	0	1	1	11	OP → B	Load 'B' Reg Bits 16-31 from OP	
1	1	0	0	12	C0 → B	Not Used (reserved)	
1	1	0	1	13	C1 → B	Load 'B' Reg Bits 16-31 from Interval Timer 1	
1	1	1	0	14	C2 → B	Load 'B' Reg Bits 16-31 from Interval Timer 2	
1	1	1	1	15	P → B	Load 'B' Reg Bits 16-31 from 'P' Reg	

*IO Bus Request (Misc 1 - Function No. 8, Item 13 and Misc 2 - Item No. 4) must be in prior micro order.
**If B Register is used as a destination during these micro orders, the Left ALU will be ORed into B Reg with the result.
***P Register Bits 8 thru 11 are not loaded by this item but are loaded via Item No. 2 only.
****B Register Bits 0 thru 15 are also loaded from Stat Bits 0 thru 15.
*****B Register Bits 24 through 27 always zeroed for this item operation.

Function 10, BMISC, in Table 1 is the 'B' miscellaneous operations which are basically used to perform decode operations on fetched macroinstructions. Table 6 lists these operations. Items 0 through 7 are not used. Items 8, 9, and 10 are decode functions and all three perform the hardware operations listed in Table 6. For Register to Register operations (RR) local store is read at the address specified by the Operation register bits 13 through 15 (R2). For SRS operations, Operation register bits 8 through 13 are transferred directly to the 'B' register bits 0 through 15. Also, the base is fetched from local store at the address specified by the operation register bits 14 and 15. For long instructions (RS) status register bits 0 through 15 (the second halfword of instruction loaded during a fullword fetch) is transferred to the 'B' register bits 0 through 15. Also, the base is fetched from local store at the address specified by the operation register bit 14 and 15. If these bits both equal 1, zero is used for the base. For long instructions that use the indexing format (RX), the status register bits 5 through 15 are transferred to the 'B' register bits 5 through 15 (displacement), and 'B' register bits 0 through 4 are zeroed. Also, the base is fetched from local store at the address specified by the operational register bits 14 and 15. It should be noted that DECB (Table 6, item 8) is used to decode instructions taken after a branch macroinstruction execution. The Next Instruction Register (NIR) latch is set during this operation indicating that the contents of the NIR are valid. DECE is used when the current instruction to be decoded is on an even halfword boundary in main storage. Again the NIR latch is set. Also during this micro-operation, the status register contents (second halfword from fullword memory instruction fetch) are transferred to the NIR. Table 6, item 11 (N→O) is a micro-order used in lieu of memory instruction fetch operations when the NIR contains a short instruction on the first halfword of a long instruction. The NIR contains the first 16 bits of the current instruction which is on the odd half word boundary in mainstore if the prior instruction was not a branch that was taken. Item 12 (RNOK) in Table 6 is used during branch operations (if taken) to indicate to the CPU that the NIR contents will not contain a valid instruction. Items 13 and 14 are not used and item 15 is used for no operation specifications.

Function 11, DMISC, of the Miscellaneous 1 field is used to load and read various 16-bit CPU registers. Items 0 through 7 in Table 7 indicate the load portion of this function. To prevent DMA interference on the master bus, the micro-order just prior to load operation must perform a nonmemory bus request. Both DMA and the DMISC load micro operations require use of the master bus. The nonmemory bus request is performed by a micro-order that uses Miscellaneous 1, Function 8, item 13, and Miscellaneous 2, item 4. The data transfers for register loads are from the low halfword of the 'B' register to the register specified in Table 8. Items 8 through 15 are used for register read operations. Read operations are data transfers from the registers specified in Table 7 to the B register bits 16 through 31.

Function 12, RD/LE, is the micro-operation used for main storage read. Table 8 lists all the read operations that can be performed by the CPU. Items 0 through 7 specify fullword (32 bits) reads from main store while items 8 through 11 are halfword operations. All read operations are memory operations where the data is loaded into the 'B' register. Table 8 also specifies whether the operation is direct, reverse or selected. For direct operation and fullword memory read, the data from the even halfword boundary of main storage is loaded into the upper halfword of the 'B' register and the odd halfword boundary from main storage is loaded into the lower halfword of the 'B' register. For reverse fullword operations, the two

halfwords from main storage are interchanged and loaded into the 'B' register. Selected operations (Table 8, items 8 and 9) are only used for halfword memory read operations. In these operations if the main storage least significant address bit is a '1', the odd boundary halfword from main storage is read and loaded into the high 16 bits of the 'B' register and the low 16 bits are zeroed. If the main storage least significant address bit is a '0', then the even boundary halfword from mainstorage is loaded into the upper halfword of the 'B' register while the lower 16 bits are again zeroed. The main storage address must already be established into the registers 'B' or 'D' prior to the execution of a read micro-operation. If item 4 in Table 8 is specified, the main storage address comes from the 'B' register. At the beginning of the read micro-order execution, the 'B' register will contain the main storage address. When the main store read micro-order is completed, the 'B' register will contain the data with the high and low halfwords reversed. Various read micro-sequences are shown in Figure C. To prevent DMA interference on the master bus, the micro-order prior to any memory read operation should be a Bus Request operation. This is a Miscellaneous 2 operation which will be further described in later paragraphs. In any case, this Bus Request operation which occurs prior to the memory read operation will test, via hardware, whether a DMA operation is in process. The CPU will be held in this Bus Request micro-order until the DMAs are complete and the bus is free for the memory read operation coming up next. A memory read micro-operation with wait (example Table 6, item 0) takes a total of 600 ns to complete. It is possible to perform an arithmetic operation if desired during a memory read cycle if a 'no wait' read is performed (example Table 9, item 1). The 'no wait' read will be 200 ns long and a 'wait' read will have to follow which is 400 ns long (example Table 9, item 0) to allow completion of the memory cycle. The data read from memory will be loaded into the 'B' register at the end of the 'wait' read micro-operation. Figure C illustrates the two memory read operations described.

Figure C also illustrates a nonmemory Bus microsequence used in Programmed Controlled I/O (PCI) input instructions. The first micro-order uses the Miscellaneous 2 field Request Bus operation which prevents DMA interference with the PCI on the master bus. Within this same micro-order, an 'A' miscellaneous Request I/O and Acknowledge operation (Table 4, item 12) is performed. This prevents a memory operation in the next micro-orders and locks up the bus as specified by the following micro-order. The next micro-order specifies a read operation as shown in conjunction with a Miscellaneous 2, I/O to main bus operation (IO→S). This sets up the bus and locks it to provide fullword and direct bus continuity between the I/O inputs and the 'B' register. At the end of this second micro-order, the 'B' register is clocked. If the data is good on the Bus, the 'B' register will contain the I/O data. To unlock the bus from the I/O operation, the final micro-order must be an 'A' miscellaneous Reset and Acknowledge operation (Table 4, item 15). Normally, more than one read micro-operation is performed after the Bus request micro-order and before the Reset and Acknowledge micro-order to allow time for I/O-CPU handshaking.

Table 8
READ OPERATIONS (USE WITH FUNCTIONS 12 OR 13 OF MISC. 1)

ROS Data Bits 25 26 27 28	Item No.	Mnemonic	Function
0 0 0 0	0	DBFW	Main Store Direct; B→Address; Full word; Wait
0 0 0 1	1	DBFN	Main Store Direct; B→Address; Full word; No Wait
0 0 1 0	2	DDFW	Main Store Direct; D→Address; Full word; Wait
0 0 1 1	3	DDFN	Main Store Direct; D→Address; Full word; No Wait
0 1 0 0	4	RBFW	Main Store Reverse; B→Address; Full word; Wait
0 1 0 1	5	RBFN	Main Store Reverse; B→Address; Full word; No Wait
0 1 1 0	6	RDFW	Main Store Reverse; D→Address; Full word; Wait
0 1 1 1	7	RDFN	Main Store Reverse; D→Address; Full word; No Wait
1 0 0 0	8	SBHW	Main Store Selected/Low Adr. Bit; B→Address; Halfword; Wait
1 0 0 1	9	SBHN	Main Store Selected/Low Adr. Bit; B→Address; Halfword; No Wait *
1 0 1 0	10	DDHW	Main Store Direct; D→Address; Halfword; Wait
1 0 1 1	11	DDHN	Main Store Direct; D→Address; Halfword; No Wait **
1 1 0 0	12		Not Used
1 1 0 1	13		Not Used
1 1 1 0	14		Not Used
1 1 1 1	15		Not Used

* Data→ Hi 'B' / Zero→ Lo 'B'

** If Low Addr Bit = 0: Data→ Hi 'B' / Zero→ Lo 'B'; If Low Addr Bit = 1: Data→ Low 'B' / Zero→ Hi 'B'

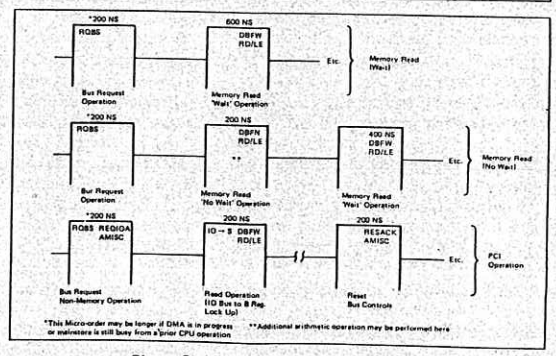


Figure C. Various 'Read' Micro-sequences

75-A97-001

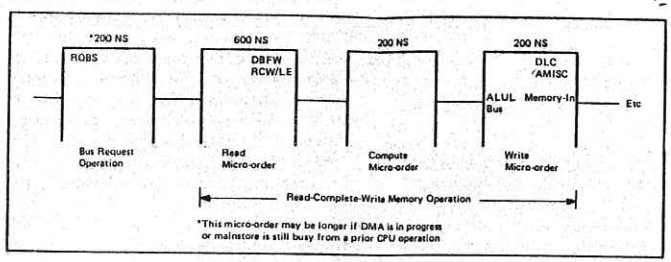


Figure D. Typical Read-Compute-Write Micro-Sequences

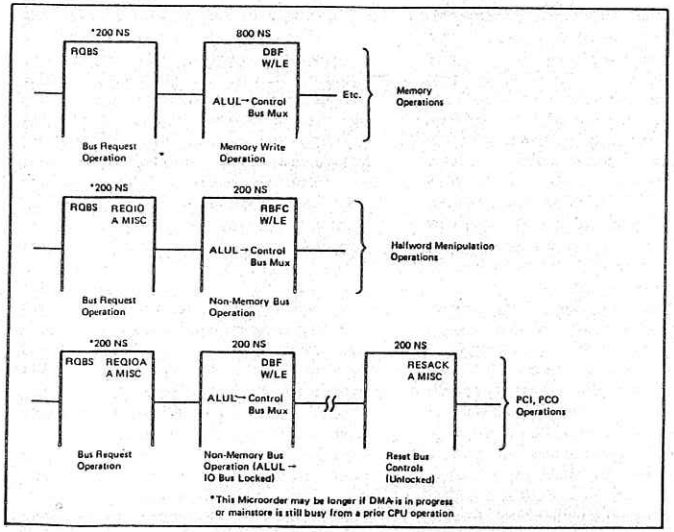


Figure E. Typical 'Write' Micro-Sequences

Function 13, RCW/LE, is used in conjunction with the items in Table 8 (also used for Function 12) to specify the read portion of a memory Read/Compute/Write (RCW) operation. Figure D illustrates a typical RCW operation. The first micro-order prior to the memory operation must be a Bus Request operation as described earlier to prevent I/O-CPU control bus interference. The second micro-order is a Read operation selected in Table 8, item 0. For this operation, the 'B' register initially has the mainstorage address. At the end of the read micro-order, the 'B' register contains the main store full word data (32-bit); and the address is saved in the mainstorage address buffers (SAR). While the memory is waiting for the write operation, the next micro-order shown is the compute cycle. Any arithmetic operation can be performed during this micro-order. The RCW memory operation is completed at the occurrence of the write micro-order. This micro-order issues a Data Load Clock (DLC) signal to the memory which gates the ALU left input to the main storage data buffers via the master bus. The ALU left source is selected by ROS Data registers bit 0, 1, and 2 which are more fully described under the Data Flow section.

Function 14, LE →RB, is used to change the 4 low base bits of the ROS address shown in Figure A as bits 2 through 5. These ROS address bits are changed to the values specified by the 4 low emit field of the ROS data register (ROS data bits 25 through 28).

Function 15, W/LE, is used to perform the write memory operations, nonmemory halfword manipulation, and to perform I/O data transfers to the I/O during PCI and PCO instruction execution. Table 9 lists all the write functions that the CPU is capable of performing. During a write micro-order, the ALU left input is multiplexed to the master bus. It must be remembered that the I/O, Mainstore, and the CPU take data directly from this bus. If Table 9, item 4 (RBF) is used, then ALU left input bits 16 through 31 and bits 0 through 15 are multiplexed to main bus out bits 0 through 31 respectively. In other words, this write operation transfers data, with the high and low halfwords reversed, to the master bus. If Table 9, item 5 (RBFC) is executed, the same operation occurs except that the 'B' register is clocked, thus, the data from the ALU left input via the Main Bus Out is clocked into the 'B' register with the high and low halfwords reversed. Figure E illustrates three typical micro-operations. The first micro-order is a memory write operation. A Bus Request is the first microcode necessary to prevent DMA interference and to initiate the memory sequence. The next micro-operation involves the execution of item 0 in Table 9. This item does a fullword and Direct data write, and the 'B' register specifies the mainstorage address. Since the 'B' register is not clocked, the address will remain intact through the memory write operation. The memory write micro is 800 ns in duration at the end of which the main storage data buffers have accepted the data on the master bus (multiplexed from the ALU left input). This micro-order must specify the ALU left data source by ROS data register bits 0 through 2 defined in the Inner data flow descriptions. Sometimes it is desirable to reverse the high and low halfwords of a 32-bit word in the data flow. This type of operation is illustrated in the second group of micro-operations shown in Figure E. The Bus Request micro-order is again needed to prevent Bus Interference but this time a nonmemory operation is desired. Therefore, in the first micro-order, a REQUIO operation is used (Table 4, item 13). This operation inhibits a memory cycle for

the next micro-order. The next micro-operation using RBFC (Table 9 , item 5) takes the ALU left data and loads it into the 'B' register with the high and low halfwords reversed. The third use for the write operations is to transfer data to the I/O using the PCI or PCO instructions. Again, a Bus Request is needed initially to prevent DMA bus interference. Also a REQIOA (Table 4, item 12) is used to prevent a memory operation and to lock up the main bus during the next micro-order. The next micro-order does does a DBF write operation (Table 9 , item 0). This multiplexes the ALU left fullword directly to the master bus making the data available to the I/O. The master bus multiplexer is now locked up, and as long as the following micro-orders thereafter specify the same ALU left data, the data on the master bus out will remain steady and available to the I/O. The master bus multiplexer is released with the execution of a micro-order containing a RESACK operation (Table 4, item 15).

Table 9
WRITE OPERATIONS (USE WITH FUNCTION 15 OF MISC 1)

ROS Data Bits (Low Emit) 25 26 27 28	Item No.	Mnemonic	Function
0 0 0 0	0	DBF	Fullword; Direct; B → Address; B Not Clocked (No Change)
0 0 0 1	1	DBFC	Fullword; Direct; B → Address; B Clocked (ALUL → B)
0 0 1 0	2	DDF	Fullword; Direct; D → Address; B Not Clocked
0 0 1 1	3	DDFCH	Fullword; Direct; D → Address; B Clocked ($ALUL_{0-15} \rightarrow B_{0-15}$; Zeroes → B_{16-31})
0 1 0 0	4	RBF	Fullword; Reverse; B → Address; B Not Clocked
0 1 0 1	5	RBFC	Fullword; Reverse; B → Address; B Clocked ($ALUL_{16-31,\ 0-15} \rightarrow B_{0-31}$)
0 1 1 0	6	RDF	Fullword; Reverse; D → Address; B Not Clocked
0 1 1 1	7	RDFCH	Fullword; Reverse; D → Address; B Clocked ($ALUL_{16-31} \rightarrow B_{0-15}$; Zeroes → B_{16-31})
1 0 0 0	8	SBH	Halfword; Selected/Low Adr Bit; B → Address; B Not Clocked (Data in High 16 Bits)
1 0 0 1	9		Not Used
1 0 1 0	10	SDH	Halfword; Direct; D → Address; B Not Clocked
1 0 1 1	11	SDHCL	Halfword; Direct; D → Address; B Clocked ($ALUL_{16-31} \rightarrow B_{16-31}$; Zeroes → B_{0-15})
1 1 0 0	12	RBH	Halfword; Reverse; B → Address; B Not Clocked
1 1 0 1	13	RBHC	Halfword; Reverse; B → Address; B Clocked ($ALUL_{16-31} \rightarrow B_{0-15}$; AGE → B_{16-31})
1 1 1 0	14	RDH	Halfword; Reverse; D → Address; B Not Clocked
1 1 1 1	15	RDHCL	Halfword; Reverse; D → Address; B Clocked ($ALUL_{0-15} \rightarrow B_{16-31}$; Zeroes → B_{0-15})
Computer Bus (CPB) → CPU Data Input Bus per reverse or direct function			

Table 10
MISCELLANEOUS 2 FIELD

ROS Data Bits (High Emit) 21 22 23 24	Item No.	Mnemonic	Function
0 0 0 0	0	IO → S	IO to Main Bus Out
0 0 0 1	1		Not Used
0 0 1 0	2	CW	Command Word
0 0 1 1	3		Not Used
0 1 0 0	4	RQBS	Request Bus
0 1 0 1	5	RB → A	ROS Backup Register to ROS Address Bus
0 1 1 0	6	S015O	Clock Status Reg 0 thru 15 and the Operational Register
0 1 1 1	7	S015N	Clock Status Reg 0 thru 15 and the Next Instruction Register
1 0 0 0	8	DISI	Disable Micro Interrupts (disabled until enabled by Misc 2 Item 9 or LSTEX)
1 0 0 1	9	ENI	Enable Micro Interrupts (enabled until disabled by Misc 2 Item 8)
1 0 1 0	10	AOUT	AGE Out
1 0 1 1	11	SAT	Start AGE serial data in operation
1 1 0 0	12	ABS → S	Load A & B sign with Stats (Stats 0 & 25)
1 1 0 1	13	SDSRB	Set Latch to use DSR from 'B' Reg
1 1 1 0	14	RDSRB	Reset Latch to use DSR from 'B' Reg (Reset automatically by LSTEX)
1 1 1 1	15		No-Op

*If Misc 1 code 14 (LE → RB) is used with this code (RB → R) the LE → RB will overload the base address bits while the function, A, and B branch bits will be loaded from ROS backup register. (Note – No-op in Misc 1 and low Emit is B Misc No-op with low Emit a No-op when Misc 2 code shown above is used without Misc 1.

Miscellaneous 2 Control Field

This field can only be used if desired in conjunction with the Miscellaneous 1 field items 8 through 15 (when ROS data register bit 17 = 1). Table 10 lists the functions specified by ROS data register bits 21 through 24 that can be executed for the Miscellaneous 2 operations.

Item 0, IO → S, gates the I/O Output Bus onto the Main Bus out for use in the I/O to CPU (PCI instructions) data transfers. This was described under Miscellaneous 1 Function 12.

Item 1 is not used.

Item 2, CW, is used to tell the I/O, via an interface hardware, that the Control Word is on the Main Bus out.

Item 3 is not used

Item 4, RQBS, is used to prevent Bus interference (such as during DMA occurrences) when the CPU microsequencing desires to use the main bus out. If during a 'RQBS' micro-order the main bus is in use (such by as DMA operations), the CPU will hold in this micro-order until the bus is free and the CPU has set up bus priority. After the CPU has its bus priority, the micro-sequencing will continue. Use of this function 'RQBS' was also described under the Miscellaneous 1 field, Functions 11, 12, 13, and 15.

Item 5, RB → A, utilizes the contents of the ROS back-up register, which is 12 bits wide, as the ROS address of the next micro-order to be executed. When a microinterrupt occurs during normal microsequencing, the ROS back-up register is loaded, via hardware with the address of the microinstruction that was interrupted. Therefore, if the microprogrammer desires he may return from the interrupt micro-routine if he executes the Miscellaneous 2, 'RB → A' function.

Item 6, S015O, is used during fullword memory fetchs for instructions when the instruction to be executed is on an even boundary in main store. The even boundary halfword instruction is loaded into the operational register while the odd boundary halfword portion is loaded into status register bits 0 through 15.

Item 7, S015N, is used during instruction fetch operations when preparing to execute a long instruction whose first halfword is on an odd boundary. The function S015N is used when a fullword memory read is performed to acquire the second halfword of the instruction. Thus with this operation status register 0 through 15 will be loaded with the second halfword and the Next Instruction Register (NIR) will be loaded with the next instruction. It should be noted that items 6 or 7 are specified during the memory read micro-orders. At the end of the memory read micro-order the fullword data loaded into the 'B' register will also be loaded into status registers 0 through 15 and operational register if S015O is specified or loaded into the status registers 0 through 15 and the next instruction register S015N is specified.

Item 8, DISI, microoperation is used to disable the micro-interrupts. When these interrupts are disabled, they remain pending.

Item 9, ENI, is used to enable the micro-interrupts.

Item 10, AOUT, is used during the execution of AGE read and write instructions to signal the AGE, via a hardwire on the AGE/CPU interface, that the CPU data is ready on the master bus.

Item 11, SAT, is also used during the execution of AGE read instructions. This micro-operation signals the AGE, via a hardwire on the AGE/CPU interface, to begin the serial transfer of data from the AGE to the CPU.

Item 12, ABS → S, execution causes status register position 0 and 25 to be loaded with the sign bits (higher order bit) of the 'A' and 'B' registers respectively.

Item 13, SDSRB, is used during the execution of non branch type instructions using the fullword Indirect Address Pointer. When this Miscellaneous 2 function is executed a latch is set. When this latch is set instead of using the DSR from the 'D' register, the CPU will use the DSR from the 'B' register. This is to allow temporary use of a new DSR field without altering the current DSR.

Item 14, RDSRB, resets the latch set by the SDSRB operation. Also Miscellaneous 1, Function 7 (LSTEX) resets this latch.

Item 15 in the Miscelleneous 2 field is a no operation field.

Low 6-Bit ROS Address Generation Logic

This section will describe the microcode operations required to generate the 6 low-order ROS address bits of the next microinstruction to be executed. Three 5-bit fields in the ROS data field (illustrated in Figure A) achieve the address generation.

- The function branch field (ROS data bits 29 through 33)
- 'A' branch field (ROS data bits 34 through 38)
- 'B' branch field (ROS data bit 39 through 43).

The Function Branch Field specifies the ROS address bits 6 through 9. When ROS data bit 29 equals '0', the ROS data bits 30 through 33 are used directly as the function branch bits (ROS address bits 6 through 9). When the ROS data bit 29 equals '1', the ROS data bits 30 through 33 must be decoded to determine what the ROS address bits will be. In either case, this field provides a 16-way branching capability. Table 11 lists the indirect functions for all the combinations of the ROS data bits in the Function Branch field. Most of the operations given in Table 11 are used for instruction decode operations with the exception of items 2 and 3 which are used for floating point operations and item 10 which is used for shift operations. An example of how these operations work will be given using item 7 (BRAP) as an example. Item 7 is specified when the ROS data register bits 29 through 33 equals 10111. ROS address bit 6 is specified by the logical AND of the Operation Register bits 8, 9, and 10 'ANDed' with the inverted Operation Register bit 11. This is shown in Table 11 as Op8 $\cdot$ Op9 $\cdot$ Op10 $\cdot$ $\overline{\text{Op11}}$. ROS address bit 7 is specified by the logical 'AND' of the Operational Register bits 8, 9, 10, and 11. ROS address bit 8 is specified by the logical 'AND' of Operational Register bit 0, 1, 2, and 3. ROS address bit 9 is specified by the 'AND' of Operation Register bits 0, 1, 2, 3, and 4.

The 'A' Branch field, ROS data bits 34 through 38 specify ROS address bit 10 Table 12 lists all the 'A' branch operations, Items 0 and 1 sets ROS address bit 10 to a '0' or '1' respectively. The remaining items specify the ROS address bit 10 indirectly and are described in Table 12 for each combination of the 'A' branch ROS data bits. The 'A' Branch items which depend on the ALU output (such as item 12) must not be executed in the micro-operation generating the desired ALU output but in the very next micro-order.

Table 11
FUNCTION BRANCH FIELD

ROS Data Bits 29 30 31 32 32	Item No.	Mnemonic	Function
1 0 0 0 0	0	O03	Op 0-3
1 0 0 0 1	1	O47	Op 4-7
1 0 0 1 0	2	FPEXBR	Set to (0), (AL 1-3 ZD $\cdot$ C1**), (ALU 1-3 = 111 $\cdot$ $\overline{\text{C1}}$**), (ALU 1-7 ZD)
1 0 0 1 1	3	FPRSBR	Set to (ALU0), (ALU 4-7 ZD), (ALU 8-11 ZD), (ALU 0-31 ZD)
1 0 1 0 0	4		Not Used
1 0 1 0 1	5		Not Used
1 0 1 1 0	6	OPAP	Set to (Op 8 $\cdot$ 9 $\cdot$ 10), (Op 8 $\cdot$ 9 $\cdot$ 10 $\cdot$ 11), (Op 0 $\cdot$ 1 $\cdot$ 2 $\cdot$ 3), (Op 8 $\cdot$ 9 $\cdot$ 10 $\cdot$ 11 $\cdot$ 13)
1 0 1 1 1	7	BRAP	Set to (Op 8 $\cdot$ 9 $\cdot$ 10 $\cdot$ $\overline{\text{11}}$), (Op 8 $\cdot$ 9 $\cdot$ 10 $\cdot$ 11), (Op 0 $\cdot$ 1 $\cdot$ 2 $\cdot$ 3), (Op 0 $\cdot$ $\overline{\text{OP1}}$ $\cdot$ Op2 $\cdot$ OP3 $\cdot$ $\overline{\text{OP4}}$)
1 1 0 0 0	8	IFAP	Set to (D15), (D15 $\cdot$ NOK), (1), (D15 $\cdot$ NOK $\cdot$ Long Instruction in NIR)
1 1 0 0 1	9		Not Used
1 1 0 1 0	10	SB	Set to: (Zero Det ALU 8-15), (Op 14 $\cdot$ Zero Det ALU 8-15), (OP15 $\cdot$ Zero Det ALU 8-15), (ZD ALUL 10-13)
1 1 0 1 1	11		Not Used
1 1 1 0 0	12	EXBR	Set to: (Op 8 $\cdot$ 9 $\cdot$ 10 $\cdot$ $\overline{\text{11}}$ $\cdot$ 12), (Op 8 $\cdot$ 9 $\cdot$ 10 $\cdot$ 11 $\cdot$ 12), (Op1), (Op2)
1 1 1 0 1	13	IBR	Set to: (0), (Op5), (Op6), (Op7)
1 1 1 1 0	14		Not Used
1 1 1 1 1	15		Not Used

*When ROS Data Bit 29 = 0, Use ROS Data Bits 30, 31, 32 and 33 directly to establish ROS Address Bits 6, 7, 8, and 9.
**C1 = Cout $\oplus$ Ovfl = cary out of bit 1.

Table 12
A BRANCH FIELD

ROS Data Bits 34 35 36 37 38	Item No.	Mnemonic	Function
0 0 0 0 0	0	0	Set to 0
0 0 0 0 1	1	1	Set to 1
0 0 0 1 0	2	S12	Set to Stat 12
0 0 0 1 1	3	S30	Set to Stat 30
0 0 1 0 0	4	DS	Set to Stat 18 (Detect)
0 0 1 0 1	5	360S	Set to Stat 19
0 0 1 1 0	6	OS	Set to Stat 28 (Overflow Stat)
0 0 1 1 1	7	ZDS02	Set to 1 if Zero Detect on Stats 0-2
0 1 0 0 0	8	P15	Set to P15 (Problem State/Current PSW bit 46)
0 1 0 0 1	9	O1	Set to Op bit 1
0 1 0 1 0	10	SSUL	Set to 1 if A & B Sign Stats are unequal (Stats 0 & 25)
0 1 0 1 1	11	CRY	Set to 1 if ALU output carry condition exists
0 1 1 0 0	12	ZD47	Set to 1 if Zero Detect on ALU output bits 4 thru 7
0 1 1 0 1	13	ZO1215	Set to 1 if Zero Detect on Op Reg bits 12 thru 15
0 1 1 1 0	14	O0:O1	Set to 1 if Op0 $\cdot$ Op1 = 1
0 1 1 1 1	15	O14	Set to Op14
1 0 0 0 0	16	S3	Set to Stat 3
1 0 0 0 1	17	O3	Set to Op bit 3
1 0 0 1 0	18	STBR	Set to 1 if $\overline{\text{Op1}}$ $\cdot$ OP2 $\cdot$ Op3 = 1
1 0 0 1 1	19	MOVF	Set to Multiply Overflow Condition (33 bit overflow)
1 0 1 0 0	20	IOLK	Set to Lockout I/O Condition (GLK Out)
1 0 1 0 1	21	S	Set to ALU output bit 0 (sign bit)
1 0 1 1 0	22	ZD015	Set to 1 if Zero Detect on ALU output bits 0 thru 15
1 0 1 1 1	23	O11	Set to Op Bit 11
1 1 0 0 0	24	OVF	Set to 1 if Output Bit 32 Overflow Condition exists
1 1 0 0 1	25	EXOUS	Set to Stat 24 (Exp Overflow - Underflow)
1 1 0 1 0	26		Not Used (Reserved)
1 1 0 1 1	27	BOC	Set to Branch Condition for BOC Instructions
1 1 1 0 0	28	BVC	Set to Branch Condition for BVC Instructions
1 1 1 0 1	29	BIAN	Set to Break-in Address Condition Not (BKADN)
1 1 1 1 0	30	SB14	Set to 1 if ALU Left Input Bit 14 $\cdot$ Zero Det ALUL Input Bits 10 thru 13 = 1
1 1 1 1 1	31	NCB	Set to 1 if ALU Output Bit 0 does not equal ALU Output Bit 1

The 'B' Branch Field, ROS data bits 39 through 43 specify ROS address bit 11. Table 13 lists all the 'B' Branch operations. This field works in the same manner as the 'A' Branch field.

Table 13
'B' BRANCH FIELD

ROS Data Bits	Item No.	Mnemonic	Function
0 0 0 0 0	0	0	Set to 0
0 0 0 0 1	1	1	Set to 1
0 0 0 1 0	2	S31	Set to Stat 31
0 0 0 1 1	3	S15	Set to Stat 15
0 0 1 0 0	4	SYSRTS	Set to Stat 21 (System Reset)
0 0 1 0 1	5	FS	Set to Stat 22 (Fail Safe)
0 0 1 1 0	6	RTS	Set to Stat 23
0 0 1 1 1	7	SBS	Set to Stat 25 (Sign B Stat)
0 1 0 0 0	8	SS	Set to Stat 26 (Sign Stat)
0 1 0 0 1	9	ZDS	Set to Stat 27 (ALU Zero Detect Stat)
0 1 0 1 0	10	IS	Set to Stat 20 (Interrupt in Process)
0 1 0 1 1	11	CS	Set to Stat 20 (Carry Stat)
0 1 1 0 0	12	SAS	Set to Stat 0 (Sign A Stat)
0 1 1 0 1	13	ZD	Set to 1 if Zero Detect on ALU Output Bits 0 thru 31
0 1 1 1 0	14	O0	Set to Op 0
0 1 1 1 1	15	O15	Set to Op 15
1 0 0 0 0	16	D22	Set to "D" Reg Bit 22 (FP Exponent underflow mask)
1 0 0 0 1	17	D23	Set to D Reg Bit 23 (FP significance mask)
1 0 0 1 0	18	BYTE	Set to 1 if B Register Bit 30 = 1
1 0 0 1 1	19	ALU1	Set to 1 if ALU Output Bit 1 = 1
1 0 1 0 0	20	C1	Set to Carry out of Bit 1 (Cout ⊕ OVFL)
1 0 1 0 1	21	D2	Set to 1 if Interval Timer 1 Interrupt Pending
1 0 1 1 0	22	D3	Set to 1 if Interval Timer 2 Interrupt Pending
1 0 1 1 1	23	O8101D	Set to 1 if Op Bits 8 thru 10 = 111
1 1 0 0 0	24	CPIO	Set to CPU/IO Failure Condition from Storage*
1 1 0 0 1	25	ZD811	Set to 1 if Zero Detect on ALU Output Bits 8 thru 11
1 1 0 1 0	26	C31F	Set to C Register Bit 31 (Fill Bit)
1 1 0 1 1	27	O4	Set to Op 4
1 1 1 0 0	28	O12	Set to Op 12
1 1 1 0 1	29	SB15	Set to 1 if ALU Left Input Bit 15 • Zero Det ALU Left Input 10 thru 13 = 1
1 1 1 1 0	30	S4	Set to Stat 4
1 1 1 1 1	31		Not Used (Reserved)

*=1 When I/O has responded, or when Memory Full occurred during a CPU transfer.

Section 6

MECHANICAL ASSEMBLY AND PACKAGING

6.1 MECHANICAL DESIGN

Evolved from a Family of Successful Airborne and Spaceborne Computers

Technologies and Applications data derived from previous programs were analyzed, improved upon and employed in the CPU design.

The CPU is packaged in a single LRU (Line Replaceable Unit) which conforms to the standard ATR case dimensions of 10.125 inches wide by 7.625 inches high by 19.562 inches long. Weight of the unit with 40K core memory is 57.9 pounds (prototype).

Top and bottom covers use captive, quick release fasteners to provide access to subassemblies. The top cover provides access to the processor/IO pages, the memory storage pages and the memory timing page. The bottom cover provides access to the processor/IO and memory storage back panels, power supply and RFI filter.

Four interface pages in the processor/IO section use a unique page/cable assembly construction. In addition to their use as a pluggable component carrier or module, the pages contain integral flexible polyimide, printed-circuit layers, which exit from the top edge of the laminated MIBs and terminate at connectors. This interface method provides repeatable electrical characteristics, reduces weight, increases packaging density, and reduces interface solder connections.

Five number 10-32 tapped holes in the front and rear panels are used for installation in accordance with RI Spec MC615-0001. A 3.576-inch diameter hole on the rear panel mates with a rack-mounted blind pressure fitting, for conditioned cooling air. The front panel provides five connectors for interface cables, a fault indicator, elapsed time indicator, handles, and the installation holes.

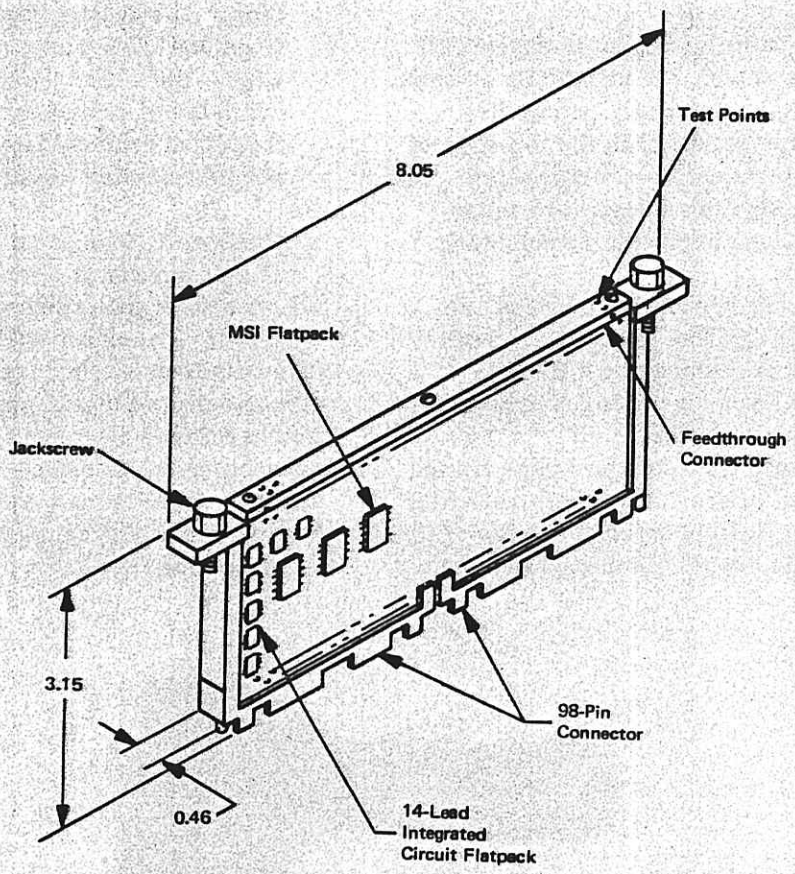
ESTIMATED WEIGHT BREAKDOWN BY MAJOR SUBASSEMBLY

Prototype

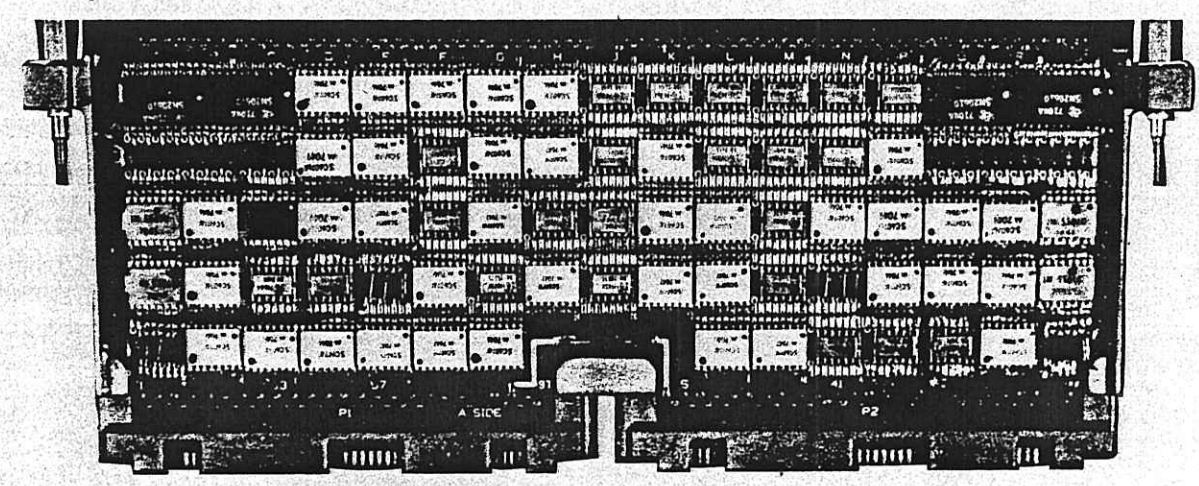
Subassembly	Weight (lb)
Structure	8.40
Covers	2.60
RFI Filter	1.61
Power Supply	9.10
CPU/IO Assembly	10.09
Storage Assembly	23.5
Miscellaneous (Includes Handles, ETI, BITE)	1.00
Total	57.9

Flight

Subassembly	Weight (lb)
Structure	8.40
Covers	1.60
RFI Filter	1.61
Power Supply	9.10
CPU/IO Assembly	10.09
Storage Assembly	23.5
Miscellaneous (Includes Handles, ETI, BITE)	1.00
Total	56.9



Basic Page Assembly



Logic Page

75-A97-001

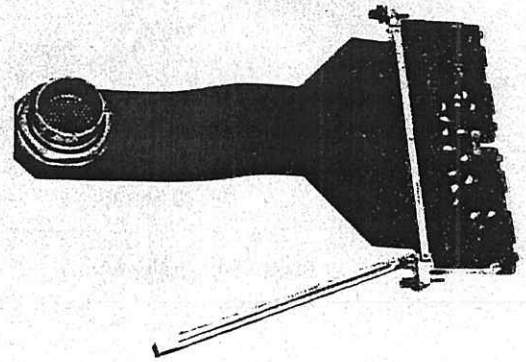
Page/Cable Assembly

The page/cable assembly is essentially a standard page with 3/5 high MIBs on each side. A flexible, integral, polyimide-layer flat cable exists from the top edge of the MIB and terminates at a connector. The basic page design is as described in the previous section along with the following modifications.

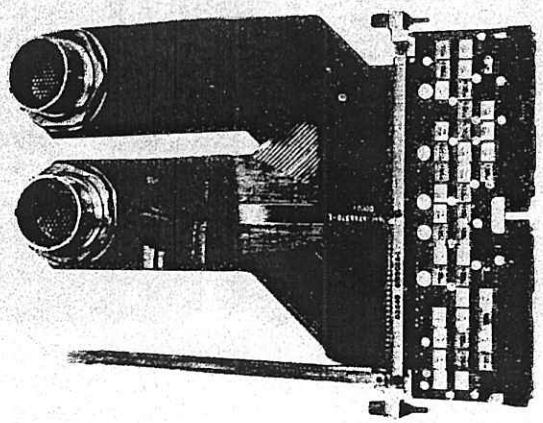
1) MIBs: In the fabrication process, an inner layer of glass epoxy in the laminated MIB is replaced by a flexible polyimide layer containing etched copper circuitry on each side.
2) Page Frame: The standard page frame is modified to provide a strain relief for the exiting flexible layer at the top of the page. This design eliminates the contact blocks, containing feed-through connections and test points, along the upper edge. However, each signal line is available at plated-through holes along the top of each cable assembly MIB to permit softwire rework.

Processor/IO Backpanel

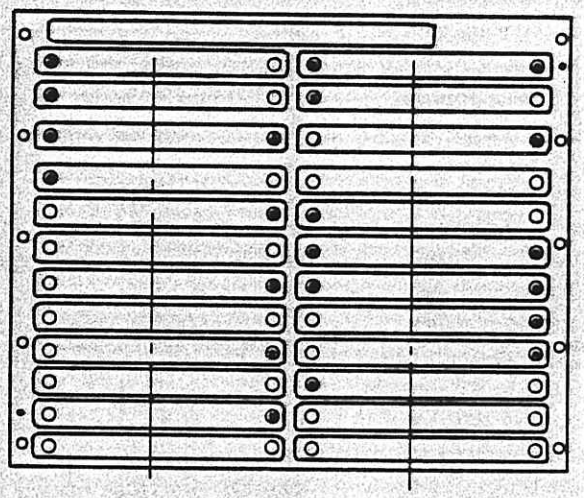
The backpanel is a single MIB in which all processor/IO page connectors are mounted. Two short wiring harnesses, using discrete wiring, are attached and strain-relieved at one end of the backpanel. The harnesses terminate with pluggable connectors for interface with the power supply. An aluminum support plate is bonded to the backpanel to provide rigidity and mounting facilities.



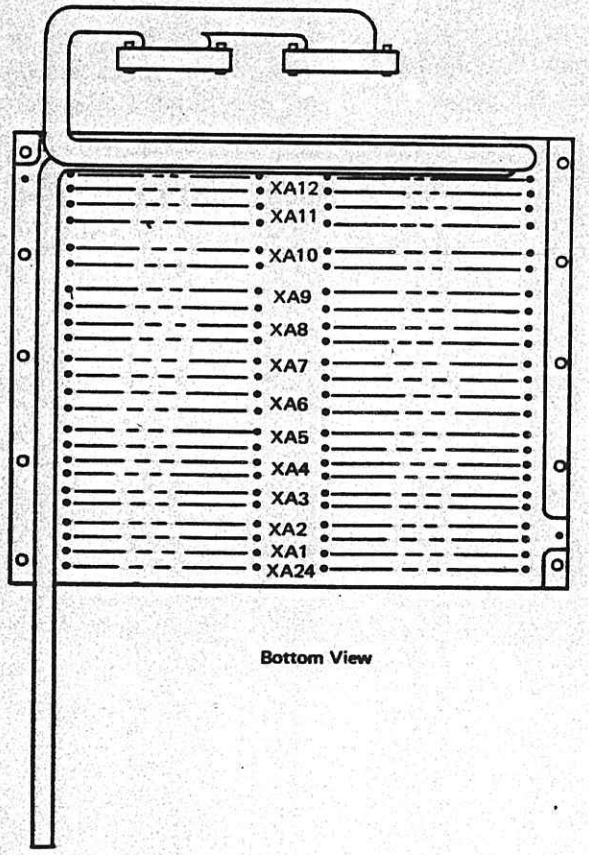
AGE Page/Cable Assembly



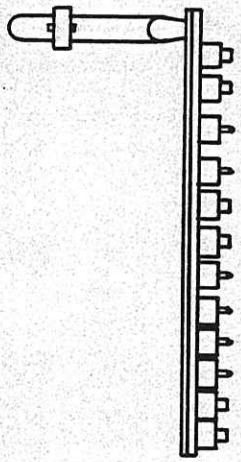
I/O Interface Page/Cable Assembly



Top View



Bottom View



Edge View

Processor/IO Backpanel (IBM Drawing 6246123)

75-A97-001

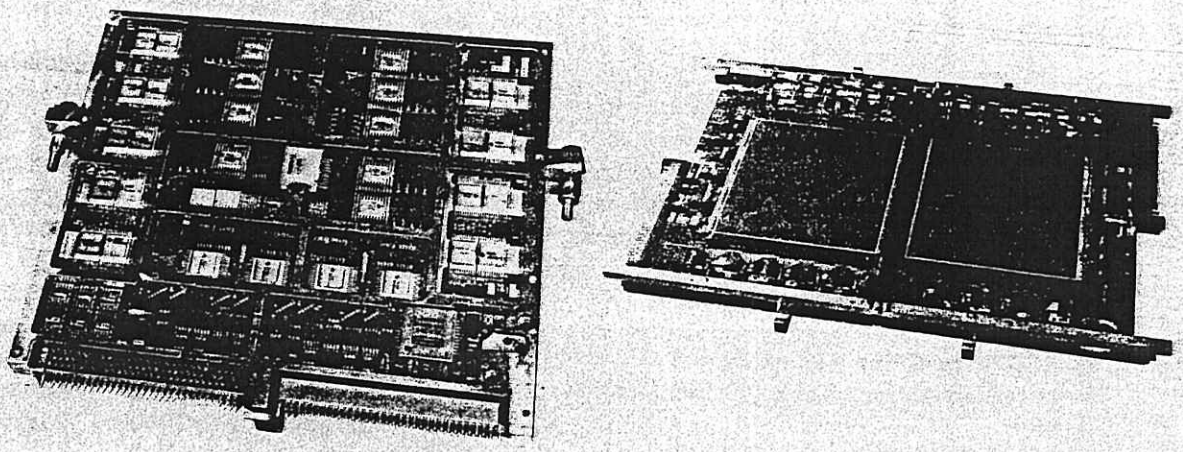
Core Storage Page

The core storage page consists of two multilayer circuit boards, each bonded to a separate heat-sink frame. The core mat is soldered to a pattern on the back of the two circuit boards, which are then folded, and fastened together. By positioning the core mat in between the two separable boards, the outside faces and portions of the inside faces of the circuit boards are left free for component mounting. This configuration allows sufficient mounting area for 8192, 18-bit halfwords per page. The core mat is an array of 0.008-inch inside diameter, 0.013-inch outside diameter cores on 0.015-inch centers, in a 256 by 576 matrix. The wires are continuous between the two circuit boards with no interference with the cores where the fold is made. The electronic circuits associated with the arrays are packaged mainly in MSI flat pack modules. A Burndy UPC 92-pin right-angle connector is soldered to each circuit board along the bottom edge, one connector on each end of the assembly. These connectors interconnect the pages through the backpanel multilayer board, which is bonded to a metal support plate.

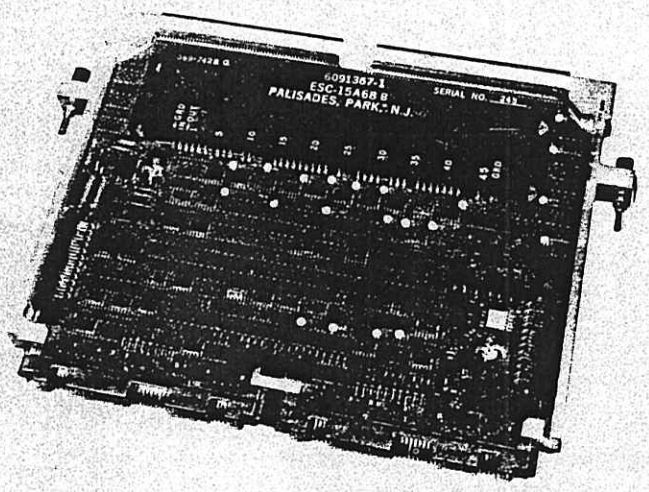
The page is fastened to the computer supporting structure, through mounting flanges extending from each side of page. Additional support is provided by the connectors and the backpanel support plate, which also is mounted to the computer structure. Stiffness, for maintaining low page deflections under vibration, is obtained with rigid aluminum support plates.

Timing/Logic Page

The timing and logic page is very similar in size (9.00 by 5.70 by 0.60 inches) to the core storage page; however, detail construction is more like the CPU pages. This assembly consists of two multilayer printed circuit boards bonded to each side of a metal plate frame. Insulators separate the boards from the frame. Components are attached to the outer surface of each board, and two 98-pin connectors are soldered along the bottom edge of the frame between the boards. The page is installed and removed by jackscrews in ears on each edge of the frame.



Main Memory Core Storage Page



Main Memory Timing and Logic Page

6.3 ASSEMBLY PACKAGING

Assembly Packaging is Functionally Oriented

Processor/IO, Main Storage and Power Supply are packaged within the CPU as removable functional assemblies.

Processor/IO Assembly

The processor/IO assembly consists of 11 pluggable pages interconnected by a multilayer circuit backpanel. The backpanel is mounted by socket head cap screws to rails brazed to the structure interior walls. The following pages are then plugged into the backpanel and secured to rails with captive mounting hardware:

1) AGE Flex-Cable Assembly*
2) Extended-Memory Page/Cable Assembly*
3) I/O Page/Cable Assembly*
4) ROS Control
5) Lo Data, Processor
6) Hi Data, Processor
7) Selector
8) Processor Control
9) PROM
10) Processor Timing
11) Memory Interface Page/Cable Assembly**

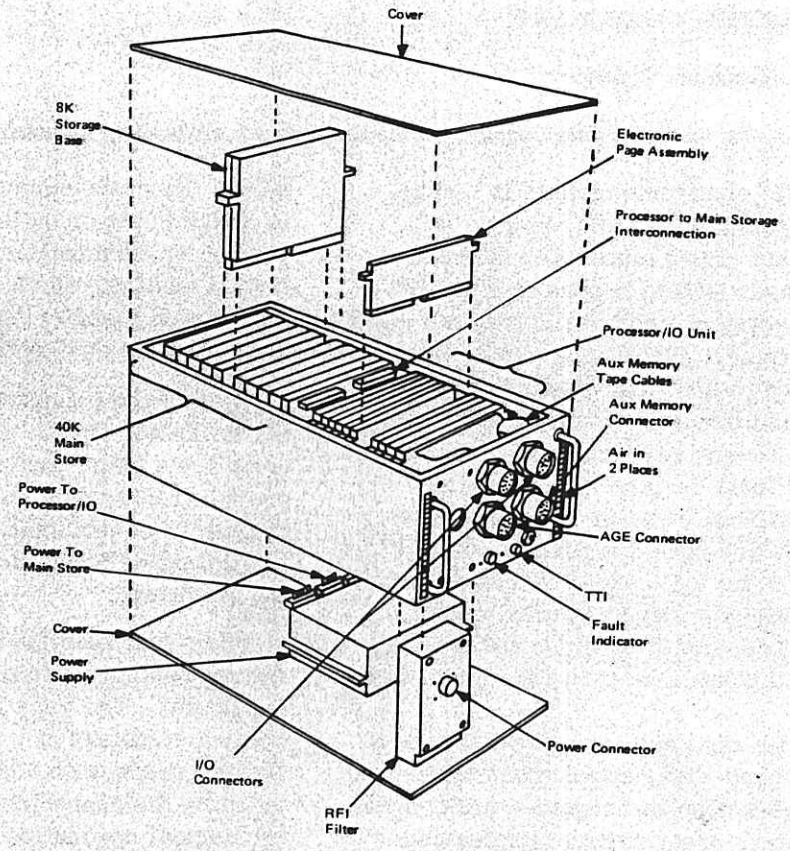
DRO Main Storage Assembly

The main store assembly consists of eleven pluggable pages interconnected by a multilayer circuit backpanel. Ten of the pages are core storage pages, and the eleventh is a timing and logic page. The dimensions of the storage pages are approximately 9.00 by 6.42 by 0.87 inches. All of the core storage pages are identical and interchangeable; therefore, only two page part numbers are required for the final assembly. This presents a significant advantage in maintainability and spares required.

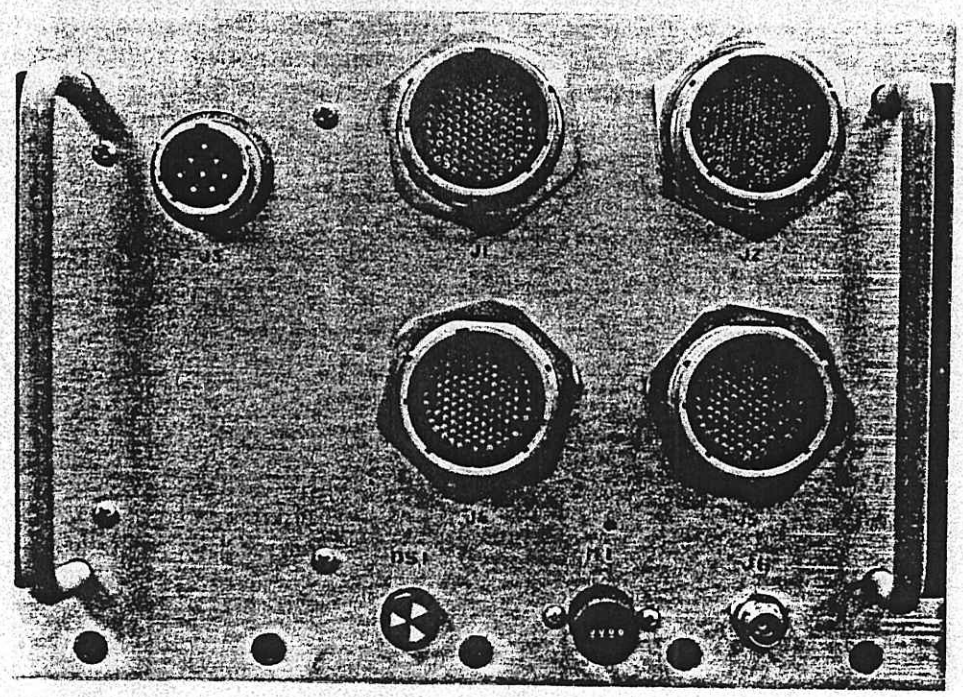
The main storage backpanel is mounted directly to the structure rails with socket head cap screws. The memory interface page/cable assembly is plugged into the backpanel. One memory timing and ten storage pages are then plugged into the backpanel and secured with captive mounting hardware.

*These page/cable assemblies also contain circular connectors which are mounted to the front panel.

**Opposite end of the Memory Interface page/cable assembly terminates at the storage backpanel providing a signal interface between the processor/IO and storage.



Space Shuttle CPU—Exploded View



Space Shuttle CPU (Prototype Shown)

75-A97-001

6.2 SUBASSEMBLY PACKAGING

Pluggable Subassembly Concept

All subassemblies, including pages, cables, and backpanels, are replaceable subassemblies.

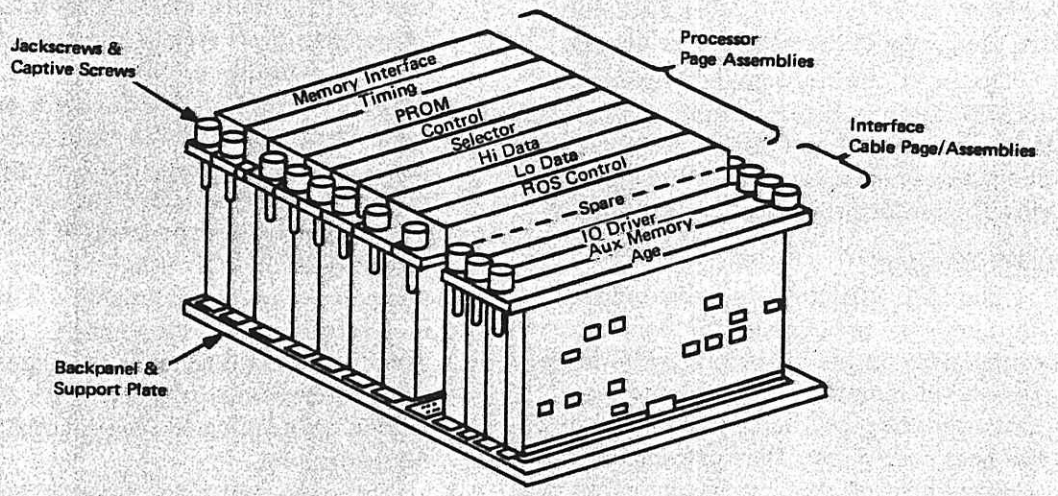
The basic electronic module is a pluggable subassembly called a "page". A page consists of two multilayer interconnection boards (MIBs) bonded to a metal frame. Insulators separate the boards from the frame. Electronic components are attached to the outer surface of each board. The circuits are mainly MSI and unit logic-integrated circuits in flatpack form and PROM devices in dual in-line packages (DIPs), although some discrete components are used. Utilization of MSI circuitry on the logic pages increases circuit packaging density by approximately 50%. Heat generated by the component is conducted through the MIB to the metal page frame.

The MIBs are made of several layers of etched, copper-clad, epoxy-glass laminates which are bonded together under heat and pressure. Connections between conductor layers are made through plated holes. Automated design and automatic testing, together with advanced process control, has resulted in extremely low failure rates for the plated-through holes and conductors.

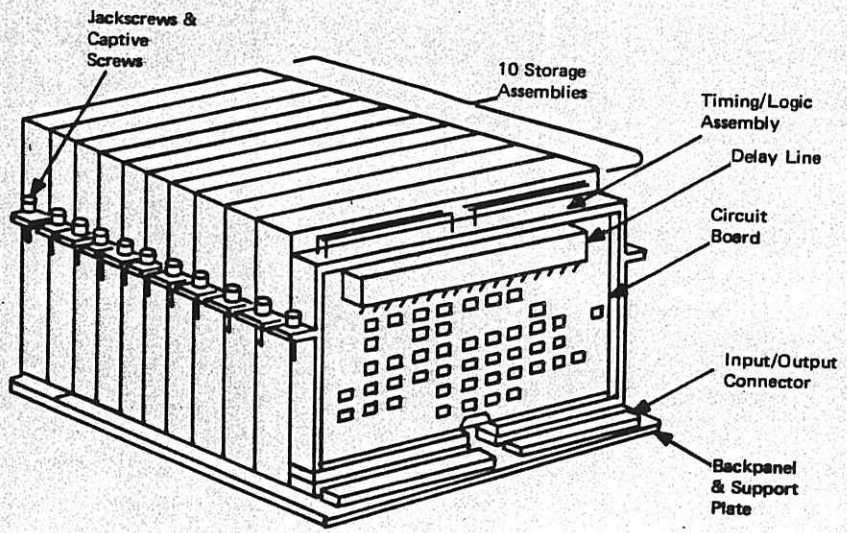
Integrated circuit flatpacks are soldered to the etched patterns on the surface of the MIBs. Discrete components are soldered in plated holes or on the board surface, depending on terminal configuration. A conformal coating, which can be easily removed with a soldering iron, during repair, is applied for environmental protection.

Two 98-pin connectors are fastened along the lower edge of the page frame. The contacts are a blade-and-fork configuration providing redundant current paths for each circuit. Gaskets seal the contact interface and the space between the receptacle and the backpanel.

Sixty-four feedthrough connections and 128 test points are along the upper edge of the frame. The page is fastened to a supporting structure at the two mounting flanges (ears) with additional support provided by the connectors. The page is cooled by thermal conduction through the frame and mounting flange thermal interface. Keyed guide pins project from the lower edge of the page to guide the page during insertion, to prevent mislocation and to insure that the correct page is installed in the proper location. Jackscrews on the mounting flanges permit installation and removal without special tools.



Processor/IO Assembly



Main Storage Assembly

75-A97-001

Power Supply Assembly

The power supply is mounted directly to the structure rails with captive fasteners. Regulated DC power distribution cables from the Processor/IO backpanel and memory interface page/cable assembly are then plugged into the power supply.

The input power connector attached to the RFI filter is inserted through an opening in the front panel and attached with screws. The filter is then secured to the structure mounting rails, with captive fasteners.

CPU Housing

The CPU structure is a dip-brazed, aluminum alloy housing with finned heat exchangers on the sidewalls is the main structural member. Forced cooling air is passed through, and contained within, the heat exchanger, thus preventing direct impingement of air on subassemblies and components. Aluminum rails, brazed to the heat exchanger, provide mounting surfaces and thermal conduction paths for all subassemblies.

Interconnections

The signal interconnection design eliminates the need for discrete wiring harnesses. All page-to-page interconnections in the processor/IO and storage assemblies are accomplished via MIB backpanels. Interconnections between the assemblies is via flex cable assembly.

Processor/IO interface with external systems is provided by the extended memory page/cable assembly, the I/O page/cable assembly and the AGE page/cable assembly.

The power interconnection for distributing regulated DC voltages is via three connectors on the power supply. Two of these connectors mate with wiring harness attached to the Processor/IO backpanel. The third connector mates with a discrete wiring harness on the main storage assembly backpanel.

External power interface is via a connector of the RFI filter which is mounted through the front panel. Discrete wiring connects the input power from the RFI filter to the power supply input.

(This page intentionally left blank)

6.4 THERMAL COOLING

Air-cooled Using Integral Heat Exchanger

Conduction cooling from components to subassembly frame and through mountings into integral heat exchangers.

The CPU is packaged in a dip-brazed aluminum structure with integral heat exchangers in two of the sidewalls. All major subassemblies are mounted to these heat exchangers using mounting flanges and rails. Conduction cooling is the principle mode of cooling for each component into its subassembly frame, and through the mounting ears into the brazed heat exchangers. Thermally, the CPU consists of three major areas, the power supply, the main storage assembly and the Processor/IO assembly.

The memory section, consisting of 10 modular core memory pages and a timing page dissipates 85 W. The Processor/IO section, consisting of interface page/cable assemblies, logic pages and a programmable read-only memory page dissipates 146 W. The power supply, including the RFI filter, dissipates 96 W. The total CPU power dissipation is 327 W.

Cooling air passes into two front panel inlet slots, through two heat exchangers into a common plenum and out a 3.576-inch exhaust outlet in the rear panel. The heat exchangers are 5.60 by 0.420 by 18.88 inches, with 70 fins per inch. All cooling air is indirect forced air with no impingement of cooling air on components. Thermal interfaces are through machined surfaces for metal-to-metal contacts and conformal coating under all components to ensure a good thermal path.

The CPU operates with a maximum inlet air temperature of 100°F. Temperature extremes down to 0°F will not have any detrimental effects on the unit and indirect air cooling, moisture, sand, dust, or salt spray cause no degrading problems on the cooling scheme. Minimum required normal cooling rates are shown in flow rate vs temperature chart. For normal operation the allowable inlet temperature range is 35°F to 100°F, however, under 8 psia emergency cabin pressure condition, the maximum allowable ambient temperature is 80°F with a maximum inlet temperature of 80°F at a maximum of 5.0 lbs/min/kW flow rate. Storage temperatures from -65°F to 150°F have no detrimental effect on the unit.

Typical pressure drop measurements for the CPU structure with various mass flows are shown in the pressure drop characteristics chart.

Cooling Air Passages and Flow

Minimum Flow Rate of Air as a Function of Temperature

Space Shuttle CPU Total and Static Pressure Drop Characteristics

75-A97-001

Section 7

MICROPROGRAMMED TEST SET

7.1 COMPUTER SUPPORT EQUIPMENT

Provides Control and Display of Computer Operation and Status

The computer support equipment (CSE) is used to load, control, and monitor the computer during hardware and software testing.

The basic CSE consists of a Microprogrammed Test Set (MTS), a magnetic tape transport, and a power and cooling unit. Under control of the MTS, the CPU is tested using the resident functional test program (FTP), which iterates continuously in a sequence that verifies the operational status of all portions of the CPU. The MTS, in conjunction with the CPU built-in-test equipment (BITE) and under the control of the FTP, can detect all processor, storage, and I/O problems.

The MTS provides the user with a CPU operator's console. The MTS interfaces with the CPU through a separate AGE connector for load, control, and monitor functions. This connection also serves as the primary interface for debugging software in the computer. The following functions can be performed through the MTS switches and displays:

- Enter information into data and control registers
- Start execution from preset address
- Stop execution at preset address
- System reset
- Memory load of a single word
- Stop execution
- Single step execution
- Single cycle execution
- Initiate manual memory read and write
- Perform memory load and verification from magnetic tape transport
- Perform memory dump from preset addresses
- Initiate interrupts
- Display registers
- Display microinstructions
- Display errors
- Address compare
- Stop on error
- History storage (ROS, Instruction, I/O address).

The magnetic tape transport uses a 7-inch reel (600 ft) of half-inch tape to record nine-track data at 800 b/in and 12.5 in/s. Physical characteristics are as follows:

- Weight: 60 lb
- Dimensions: 8.75H x 19W x 8.5D (inches)
- Power: 100-125 VAC ±10%, 1.0 A
  200-250 VAC ±10%, 0.5 A,
  48-62 Hz

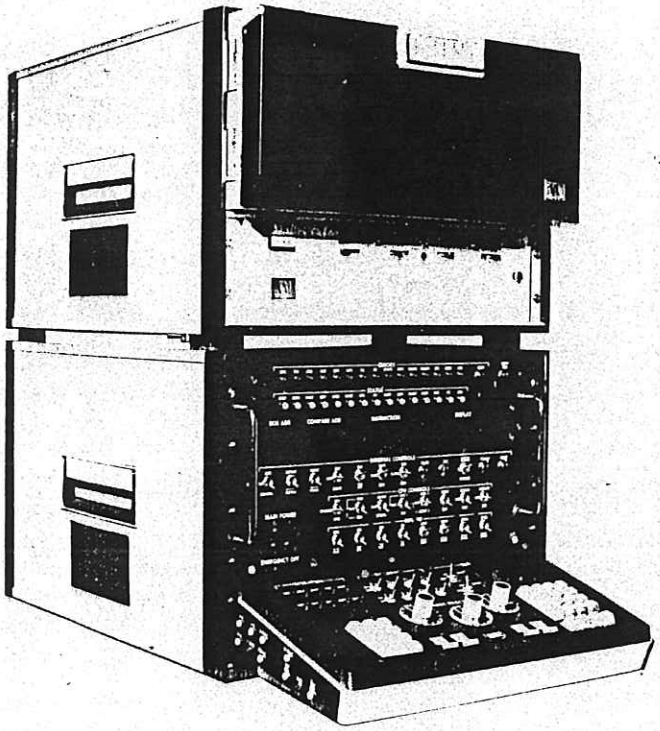
The power and cooling unit supplies 28-VDC input power and cooling air to the CPU. Test points and switches permit measuring supplied current and voltage.

To extend the versatility and man/CPU interface of the MTS for supporting hardware and software testing, programmable peripherals may be added to the basic equipment. These peripherals include a high-speed line printer, a CRT/Keyboard, and a second magnetic tape transport. The line printer may also be hard-wired, permitting hard copy memory dumps for program maintenance and debug aids. The line printer characteristics are shown below:

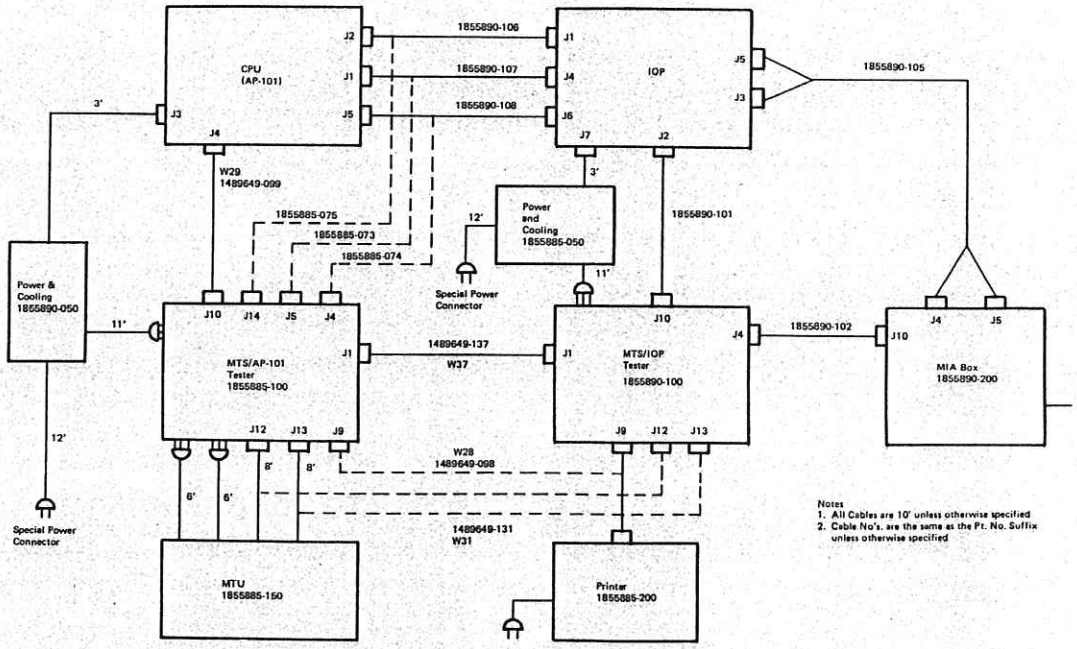
• Printing Rate:	165 characters per second, 60 lines of 132 characters per minute
• Character Structure:	5x7 dot matrix, 10-point equivalent
• Character Repertoire:	63 alphanumeric and special characters (USASCII)
• Printing Structure:	132 characters per line, 6 lines per inch
• Dimensions:	11-1/4H x 19-1/4D x 27-1/2W (inches)
• Weight:	155 lb

Character [illegible] of the CRT/keyboard are given belo

- Display: Up to 1000 alphanumeric characters on a 12-inch (diagonal) CRT; 25 lines at 40 characters per line
- Keyboard: Standard typewriter configuration, 64 alphanumeric characters, 16 edit control keys, 3 transmission keys, 2 mode control keys.



Typical Computer Support Equipment
(Microprogrammed Test Set and Magnetic Tape Transport)



Prototype CPU/IOP Cabling
Test Configuration

(Last page of document)

75-A97-001