Honeywell

LEVEL 6
HARDWARE

MODEL 43/53 CENTRAL PROCESSOR UNIT (VOL. I)

Model 43/53

CENTRAL PROCESSOR UNIT

(Volume I)

Document No. 71010300-200 Order No. FN28, Rev. 1

This manual has been revised to the -200 level. It supersedes all previous issues.

RECORD OF REVISIONS

REVISION	DATE	AUTHORITY	AFFECTED PAGES
-100	Jan. 1977	Original Issue	-
-200	March 1978	BLCD87346	Entire manual updated

Hardware Publications, M&TO, Billerica, MA 01821

Printed in the United States of America

CONTENTS

Section			Page
I	INTRODUCTION		1-1
	1.1	General Description	1-1
	1.2	Reference Documentation	1-5
	1.3	Software Visible Registers	1-5
		1.3.1 Base Registers	1-5
		1.3.2 Data Registers	1-5
		1.3.3 Indicator Register	1-6
		1.3.4 Mode Registers	1-6
		1.3.5 Program Counter	1-6
		1.3.6 Remote Descriptor Base Register	1-6
		1.3.7 Status/Security Register	1-6
		1.3.8 Stack Pointer	1-7
	1.4	Data Word Formats	1-7
		1.4.1 Memory Data	1-7
		1.4.2 Addresses	1-7
		1.4.3 Signed Integer Data Word	1-8
		1.4.4 Signed Integer Data Byte	1-8
		1.4.5 Sign Extended Integer Byte in Word	1-8
		1.4.6 Unsigned Integer Word	1-9
		1.4.7 Unsigned Integer Byte	1-9
		1.4.8 Unsigned Integer Byte in Word	1-9
		1.4.9 Signed Integer Double-Word	1-9
	1.5	Instruction Word Formats	1-9
		1.5.1 Double-Operand Instructions	1-10
		1.5.2 Single-Operand Instructions	1-10
		1.5.3 Input/Output Instructions	1-11
		1.5.3.1 Data and Command I/O Instructions	1-11
		1.5.3.2 Address and Range Output Instructions	1-12
		1.5.4 Short Value Immediate Instructions	1-13
		1.5.5 Branch on Register Instructions	1-14
		1.5.6 Branch on Indicator Instructions	1-14
		1.5.7 Shift Instructions	1-15
		1.5.7.1 Shift Short Format	1-15
		1.5.7.2 Shift Long Format	1-15
		1.5.8 Generic Instructions	1-16
	1.6	CPU/Memory Effective Addressing	1-16
		1.6.1 Displacement Address Type	1-17
		1.6.2 Address Syllable	1-17

CONTENTS

Section				Page
		1.6.2.1	Register Address Syllable	1-17
		1.6.2.2	Immediate Operand Address Syllable	1-20
		1.6.2.3	Memory Address Syllable	1-20
	1.6.3		Indexing	1-23
		1.6.3.1	Bit Addressing	1-24
		1.6.3.2	Byte Addressing	1-24
		1.6.3.3	Word Addressing	1-25
		1.6.3.4	Double-Word Addressing	1-25
		1.6.3.5	Quadruple-Word Addressing	1-26
	1.7		Memory Address Boundaries	1-26
II	THEORY OVERVIEW			2-1
	2.1		CPU Hardware Overview	2-1
		2.1.1	CPU Clock	2-1
		2.1.2	Control Store	2-3
		2.1.3	Next Address Generation (NAG) Logic	2-3
		2.1.3.1	Test Logic	2-3
		2.1.3.2	Branch Logic	2-3
		2.1.3.3	Link Register	2-4
		2.1.4	Microprocessor	2-4
		2.1.5	Microprocessor Addressing Logic	2-4
		2.1.6	Internal Bus	2-4
		2.1.7	CPU Registers	2-5
		2.1.7.1	Indicator Register	2-5
		2.1.7.2	LINK Register	2-5
		2.1.7.3	Counter Register	2-5
		2.1.7.4	Select Register	2-6
		2.1.7.5	Byte Indexing Register	2-6
		2.1.7.6	Instruction Register	2-6
		2.1.7.7	H Register	2-6
		2.1.7.8	Status/Security Register	2-6
		2.1.7.9	P Register	2-7
		2.1.7.10	Memory Address Register	2-7
		2.1.7.11	Megabus Procedure Buffers	2-7
		2.1.7.12	Megabus Data Buffer	2-8
		2.1.7.13	Megabus Interrupt Register	2-8
		2.1.7.14	Mode Registers	2-8
		2.1.7.15	M Collector Register	2-8
		2.1.8	CPU Control Flops and Control Signals	2-8
		2.1.9	Miscellaneous CPU Hardware	2-9
		2.1.9.1	Random Access Memory	2-9
		2.1.9.2	Bootload PROM	2-11
		2.1.9.3	Address Bus	2-12

CONTENTS

Section				Page
		2.1.9.4	Memory Management Unit	2-12
		2.1.9.5	Cache Memory	2-14
		2.1.9.6	Hexadecimal Decoder	2-15
		2.1.9.7	Subcommand Generator	2-15
		2.1.9.8	Constant Generators	2-15
		2.1.9.9	Control Panels	2-15
	2.2	Megabus Operations		2-28
		2.2.1	Master/Slave Relationship	2-29
		2.2.2	Megabus Usage	2-29
	2.3	Megabus Formats		2-31
		2.3.1	Channel Numbers	2-31
		2.3.2	Unit Addressing	2-32
		2.3.3	Memory Read	2-33
		2.3.4	Memory Write	2-33
	2.4	CPU Firmware Overview		2-33
		2.4.1	Initialize	2-34
		2.4.2	Instruction Fetch	2-34
		2.4.3	WDT/RTC Servicing	2-34
		2.4.4	Control Panel Service	2-34
		2.4.5	Address Formation	2-36
		2.4.6	CIP Address Formation	2-36
		2.4.7	Indirect/Indexing/Operand Fetch	2-36
		2.4.8	Instruction Execution	2-36
		2.4.9	Result Write	2-36
		2.4.10	Trap and Interrupt	2-36
			2.4.10.1 Traps	2-37
			2.4.10.2 Interrupts	2-37
		2.4.11	Quality Logic Test	2-37
III	FIRMWARE OPERATION			3-1
	3.1	Firmware Word		3-1
		3.1.1	Left Select (LS) Field	3-1
		3.1.2	Right Select (RS) Field	3-3
		3.1.3	Central Processor Board, Internal Bus (DI) Field	3-3
		3.1.4	RALU Destination (AD) Field	3-4
		3.1.5	RALU Function (AF) Field	3-4
		3.1.6	RALU Source (AS) Field	3-7
		3.1.7	Processor Clock (CK) Speed Control Field	3-8
		3.1.8	Internal Bus (BI) Selectro Control Field	3-8
		3.1.9	Select Modify (SM) Field	3-12
		3.1.10	Megabus (BS) Control Field	3-13
		3.1.11	General Purpose (GP) Micro-Op Field	3-19
		3.1.12	Test Condition (TC) Field	3-24
		3.1.13	Branch Type (BR) Field	3-28
		3.1.14	C Field	3-31
		3.1.15	Next Address (NA) Field	3-31

CONTENTS

Section			Page
	3.2	Traps	3-31
		3.2.1 Dedicated Memory Locations	3-31
		3.2.2 Trap Save Area	3-32
		3.2.2.1 Entry 1	3-32
		3.2.2.2 Entry 2	3-32
		3.2.2.3 Entry 3	3-32
		3.2.2.4 Entry 4	3-32
		3.2.2.5 Entry 5	3-32
		3.2.2.6 Entry 6	3-33
		3.2.2.7 Entry 7	3-34
		3.2.2.8 Entry 8	3-34
		3.2.3 Interrupt Save Area	3-34
		3.2.4 Trap Handler Procedures	3-34
		3.2.5 Trap Functionality	3-34
	3.3	Interrupts	3-35
		3.3.1 Interrupt Vectors	3-36
		3.3.2 Activity Flags	3-36
		3.3.3 LEV Instruction	3-38
IV	HARDWARE OPERATION		4-1
	4.1	Master Clock	4-1
		4.1.1 Clock Initialization Network	4-3
		4.1.2 Clock Stall Network	4-3
		4.1.2.1 Read Cycle Initiate Stalls	4-3
		4.1.2.2 Write Initiate Stalls	4-4
		4.1.2.3 Data Stalls	4-4
		4.1.3 Clock Cycle Generator	4-4
	4.2	Control Store	4-4
		4.2.1 Control Store Local Register	4-4
		4.2.2 Control Store Addressing	4-6
	4.3	Next Address Generation (NAG) Logic	4-6
		4.3.1 Test Logic	4-8
		4.3.2 Branch Logic	4-8
		4.3.3 LINK Register	4-8
		4.3.4 Address Generator	4-8
		4.3.4.1 X0 Branch	4-9
		4.3.4.2 XL Branch	4-9
		4.3.4.3 XA Branch	4-9
		4.3.4.4 XB Branch	4-9
		4.3.4.5 XR Branch	4-9
		4.3.4.6 XE Branch	4-9
		4.3.4.7 XW Branch	4-9
		4.3.4.8 XF Branch	4-9
	4.4	Microprocessor	4-9
		4.4.1 Microinstruction Decoder	4-10
		4.4.2 Register File	4-10
		4.4.3 Q Register	4-12
		4.4.4 Data Source Selector	4-12
		4.4.5 Arithmetic/Logic Unit	4-12

CONTENTS

Section			Page
		4.4.5.1 Look-Ahead Logic	4-14
		4.4.5.2 Overflow Logic	4-15
		4.4.5.3 Zero Detection Logic	4-15
	4.4.6	Data Output Selector	4-15
	4.4.7	Shift Logic	4-16
		4.4.7.1 Single Left Shift	4-17
		4.4.7.2 Double Left Shift	4-17
		4.4.7.3 Single Right Shift	4-18
		4.4.7.4 Double Right Shift	4-18
4.5	RALU Addressing		4-19
4.6	Internal Bus		4-20
4.7	CPU Registers		4-22
	4.7.1	Indicator Register	4-22
		4.7.1.1 Arithmetic Indicators	4-22
		4.7.1.2 Bit Test Indicator	4-22
		4.7.1.3 Input/Output Indicator	4-24
		4.7.1.4 Comparison Indicators	4-24
	4.7.2	LINK Register	4-24
	4.7.3	Counter Type Registers	4-24
		4.7.3.1 Counter Register	4-24
		4.7.3.2 Select Register	4-25
		4.7.3.3 Byte Indexing Register	4-25
	4.7.4	Instruction Register	4-25
	4.7.5	H Register	4-25
	4.7.6	Status/Security Register	4-25
	4.7.7	P Register	4-25
	4.7.8	Memory Address Register	4-26
	4.7.9	Megabus Register	4-26
	4.7.10	Interrupt Register	4-26
	4.7.11	M Register	4-26
4.8	CPU Control Flip-Flops		4-27
	4.8.1	SIGN Flip-Flop	4-27
	4.8.2	MISC Flip-Flop	4-27
	4.8.3	SHIN1 Flip-Flop	4-27
	4.8.4	SHIN2 Flip-Flop	4-27
	4.8.5	ZERO Flip-Flop	4-28
	4.8.6	WRAP Flip-Flop	4-28
	4.8.7	NEWXR Flip-Flop	4-28
	4.8.8	ACK Flip-Flop	4-28
	4.8.9	YELLOW and PARER Flip-Flops	4-28
	4.8.10	EXTRAP, INTBSY, and TICK Flip-Flops	4-29
	4.8.11	LOAD, TRAFFIC, and PANOK Flip-Flops	4-29
	4.8.12	EFFRING, NONPROC, NOCHECK, SEGERR, and PROV Controls	4-29
4.9	Miscellaneous CPU Hardware		4-30
	4.9.1	Random Access Memory	4-30
		4.9.1.1 RAM Location 0	4-31
		4.9.1.2 RAM Locations 1 through 3	4-32
		4.9.1.3 RAM Location 4	4-32

CONTENTS

Section			Page
		4.9.1.4 RAM Location 5	4-32
		4.9.1.5 RAM Location 6	4-32
		4.9.1.6 RAM Location 7	4-32
		4.9.1.7 RAM Location 8	4-32
		4.9.1.8 RAM Location 9	4-32
		4.9.1.9 RAM Location A	4-32
		4.9.1.10 RAM Location B	4-32
		4.9.1.11 RAM Location C	4-32
		4.9.1.12 RAM Location D	4-32
		4.9.1.13 RAM Location E	4-33
		4.9.1.14 RAM Location F	4-33
	4.9.2	Bootload PROM	4-33
	4.9.3	Address Bus	4-34
	4.9.4	Memory Management Unit	4-34
	4.9.5	Cache Memory	4-34
	4.9.6	Hexadecimal Decoder	4-35
	4.9.7	Subcommand Generator	4-35
	4.9.8	Constant Generators	4-37
	4.9.9	Control Panel	4-37
4.10	Megabus Network		4-37
	4.10.1	Interface Logic	4-37
		4.10.1.1 Timing Signal Lines	4-37
		4.10.1.2 Information Signal Lines	4-40
		4.10.1.3 Information Control Signal Lines	4-40
		4.10.1.4 Status/Error Signal Lines	4-41
		4.10.1.5 Tie-Breaking Control Signals	4-42
		4.10.1.6 Operational Control Signals	4-42
	4.10.2	Megabus Network Operation	4-43
	4.10.3	Megabus Tie-Breaking Function	4-44
	4.10.4	Memory Read Request	4-46
		4.10.4.1 Double-Word Fetch	4-46
		4.10.4.2 Single Word Fetch	4-46
	4.10.5	Response Summation	4-48
	4.10.6	Megabus Register	4-48
		4.10.6.1 P1 and P2 Buffers	4-48
		4.10.6.2 BD Buffer	4-49
4.11	Interrupt Control Logic		4-49
		4.11.1 Address Compare Logic	4-49
		4.11.2 Level Check Logic	4-50
		4.11.3 Interrupt Register	4-50
4.12	Full Control Panel		4-53
	4.12.1	Physical Characteristics	4-54
	4.12.2	Principles of Operation	4-54
		4.12.2.1 Control Panel Functionality	4-59
		4.12.2.2 Data Input Devices	4-60
		4.12.2.3 Register Displays	4-60

CONTENTS

Section		Page
4.12.2.4	Panel Controls/ Indicators	4-60
4.12.2.5	Panel Lock Request	4-67

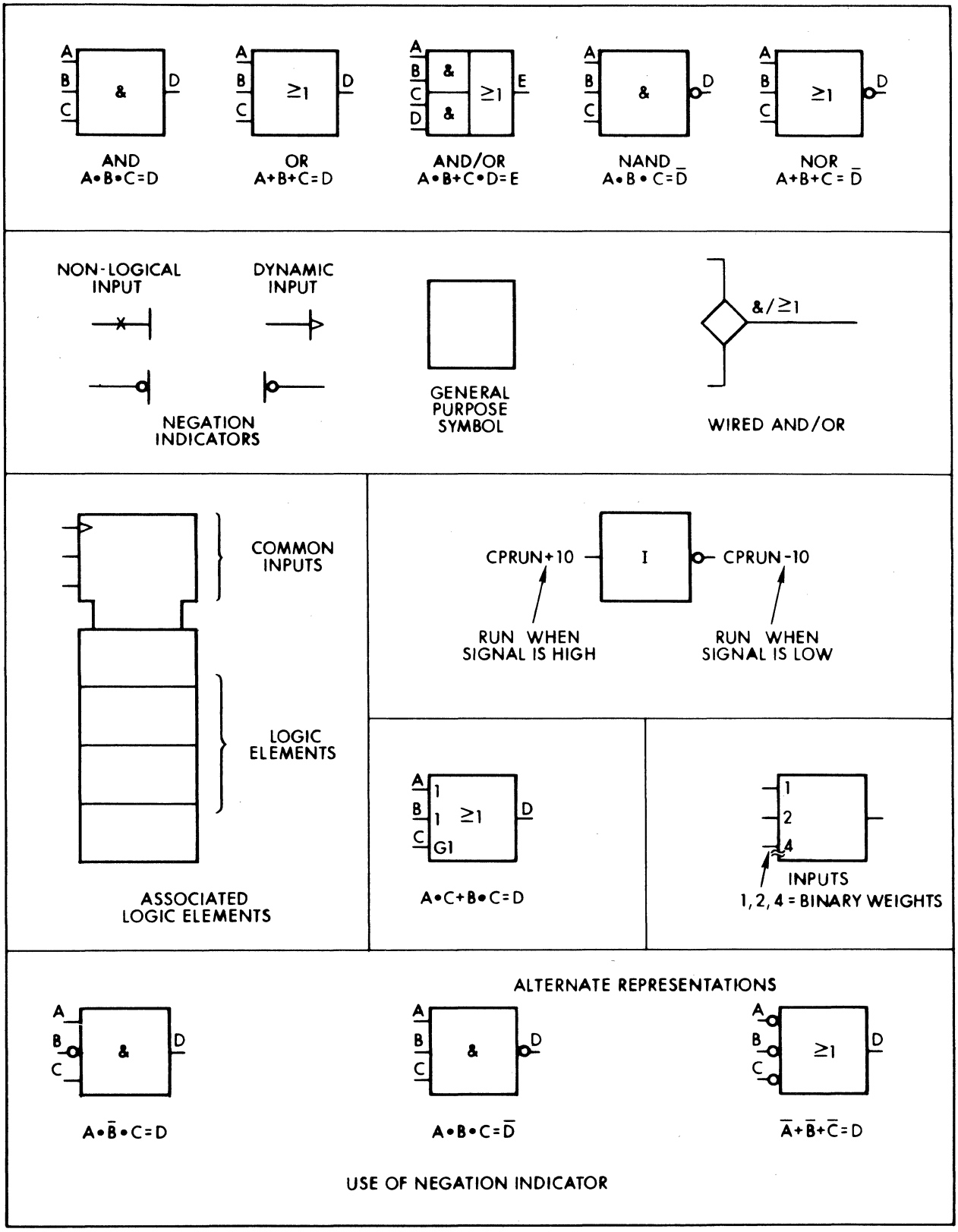
ILLLUSTRATIONS

Figure		Page
1-1	CPU Communications Block Diagram	1-3
1-2	CPU/Megabus Interface Signals	1-3
1-3	CPU Functional Block Diagram	1-4
1-4	Address Syllable	1-18
2-1	CPU Major Block Diagram	2-2
2-2	Full Control Panel	2-17
2-3	Basic Control Panel	2-26
2-4	Typical Level 6 System	2-29
2-5	Megabus Formats	2-31
2-6	CPU Firmware Overview Flow Chart	2-35
2-7	Dedicated Memory Locations in SAF	2-38
2-8	Dedicated Memory Locations in LAF	2-39
3-1	Firmware Word Format	3-2
4-1	Master Clock Logic	4-2
4-2	Control Store Local Register	4-5
4-3	Next Address Generation Logic	4-7
4-4	Microprocessor Functional Overview	4-11
4-5	Internal Bus Sources and Destinations	4-23
4-6	RAM Format	4-31
4-7	Subcommand Generator Logic	4-36
4-8	Bus Timing Diagram	4-44
4-9	Megabus Tie-Breaking Logic	4-45
4-10	Megabus Control and Memory Read Request Logic	4-47
4-11	Address Compare Logic	4-51
4-12	Level Check Logic	4-52
4-13	Interrupt Register	4-53
4-14	Interconnections and Rocker Arm Switch Assembly	4-55
4-15	Control Panel to CPU Signal Cable Connections	4-56
4-16	CPU Panel Control Logic	4-57
4-17	Key-Pad Array Logic	4-63

TABLES

Table		Page
1-1	Address Syllable Notation	1-19
2-1	Full Control Panel Switches and Indicators	2-18
2-2	Control Panel Register Selection	2-24
2-3	Basic Control Panel Switches and Indicators	2-26
2-4	Common Types of Megabus Operations	2-30
2-5	Trap Even, Trap Vectors, and Halt Locations	2-40
3-1	Left Select Codes	3-3
3-2	Right Select Codes	3-3
3-3	Central Processor Board Internal Bus Codes	3-4
3-4	RALU Destination Codes	3-5
3-5	RALU Function Codes	3-6
3-6	RALU Source Codes - AS Field	3-7
3-7	RALU Source Codes - AS/AF Field	3-8
3-8	Clock Speed Codes	3-8
3-9	Internal Bus Codes, Constants	3-9
3-10	Internal Bus Codes Selector	3-10
3-11	Internal Bus Control Indicators	3-11
3-12	Select Modify Codes	3-13
3-13	Register Selection	3-14
3-14	Megabus Control Codes	3-15
3-15	General Purpose Micro-Op Codes	3-20
3-16	Test Condition Codes	3-24
3-17	Test Condition 10 (Group A)	3-27
3-18	Test Condition 10 (Group B)	3-28
3-19	Test Condition 10 (Group C)	3-28
3-20	Test Condition 20 (Group M)	3-28
3-21	Branch Type Codes	3-29
3-22	Next Address Code	3-31
3-23	Level Assignments for Interrupt Events	3-35
3-24	LEV Operand Decode	3-39
4-1	RALU Logical and Arithmetic Operations	4-13
4-2	RALU Shift Operations	4-16
4-3	M Register Format	4-26
4-4	Megabus Interface Signals	4-38
4-5	Control Panel to CPU Internal Bus Bit Assignments	4-59

LOGIC SYMBOLOGY



1
INTRODUCTION

This manual describes the functionality and provides the theory of operation for the Level 6 Model 43/53 Central Processor Unit (CPU).

1.1 GENERAL DESCRIPTION

The CPU is a high-speed general purpose digital computer designed to process data from main memory and associated system devices. Its logic elements are fabricated on a 15- by 16-inch control board and a 12- by 14-inch four-layer CPU board. The CPU board is physically supported by the control board, hereafter referred to as the controller, in a piggyback manner. The controller houses the CPU control logic including some of the CPU registers, clock network, and Megabus* interface logic. The CPU board contains associated functional elements including the microprocessor, Random Access Memory (RAM), and bootstrap Programmable Read Only Memory (PROM). A Memory Management Unit (MMU) and cache memory are available as CPU options and, when included as part of the Model 43/53 system configuration, they are also housed on the CPU board. For details concerning the MMU and cache memory options, refer to the applicable documentation listed in subsection 1.2

CPU communication with main memory and other units of the Level 6 system is over a common bus, called the Megabus (see Figure 1-1). The interface signals between the CPU and the Mega-

*Trademark of Honeywell Information Systems, Inc.

bus are shown in Figure 1-2. The CPU can also operate in a multi-processor environment consisting of up to four central processing units, and each CPU can communicate directly with the following optional processor type controllers over the Megabus.

- Commercial Instruction Processor (CIP)
- Scientific Instruction Processor (SIP)

For details concerning the CIP and SIP options, refer to the applicable documentation listed in subsection 1.2.

The CPU uses hardware, with the aid of firmware, to communicate directly with the MMU or cache memory, to communicate with main memory over the system Megabus, and to decode all CPU instructions, performing the necessary arithmetic, logical, or shift operations (see Figure 1-3). The following list provides additional information relative to the general characteristics of the CPU:

1. 8-, 16-, or 32-bit data
2. Up to two megabytes of directly addressable main memory
3. Bit, byte, word, and double-word registers
4. Bit test, set, and mask capability
5. 26 program visible general registers, including multiple accumulators, multiple address, index, and control registers
6. Immediate, register-to-register, and register-to-memory operations
7. 64 vectored interrupt levels
8. Stack/Queue handling
9. Multiple vectored trap structure
10. Hardware supported context save and restore
11. Multiple addressing modes, including indexing, indirect, base plus displacement, program counter relative, auto increment/decrement, etc.
12. Real-time clock and watchdog timer
13. Power failure detection
14. Automatic restart
15. Bootstrap
16. Writable Control Store (WCS).

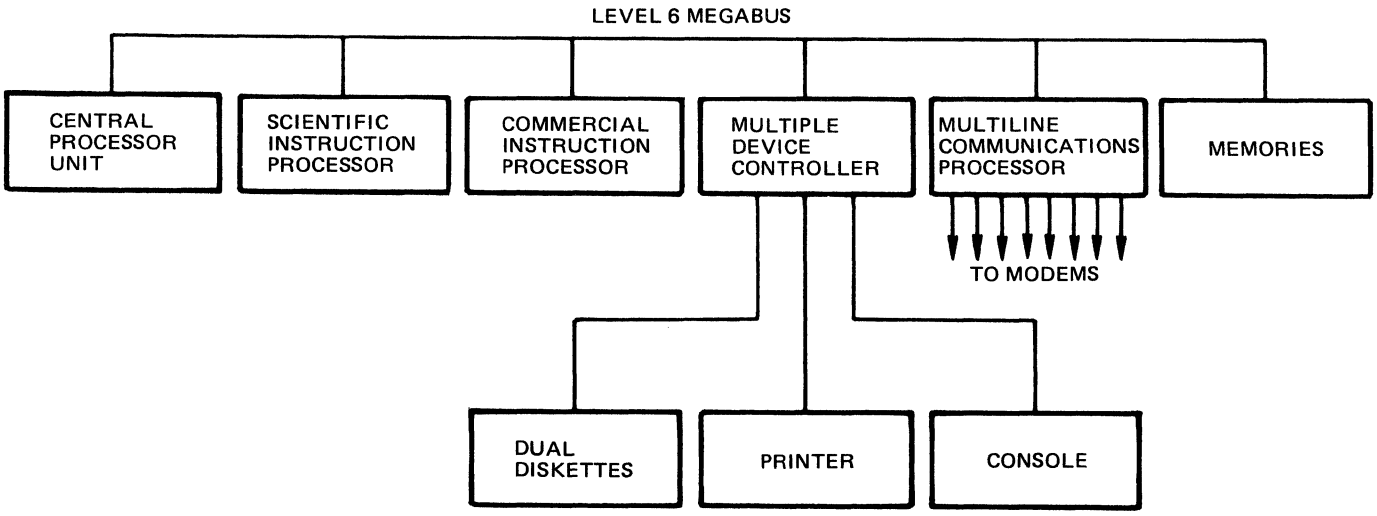


Figure 1-1 CPU Communications Block Diagram

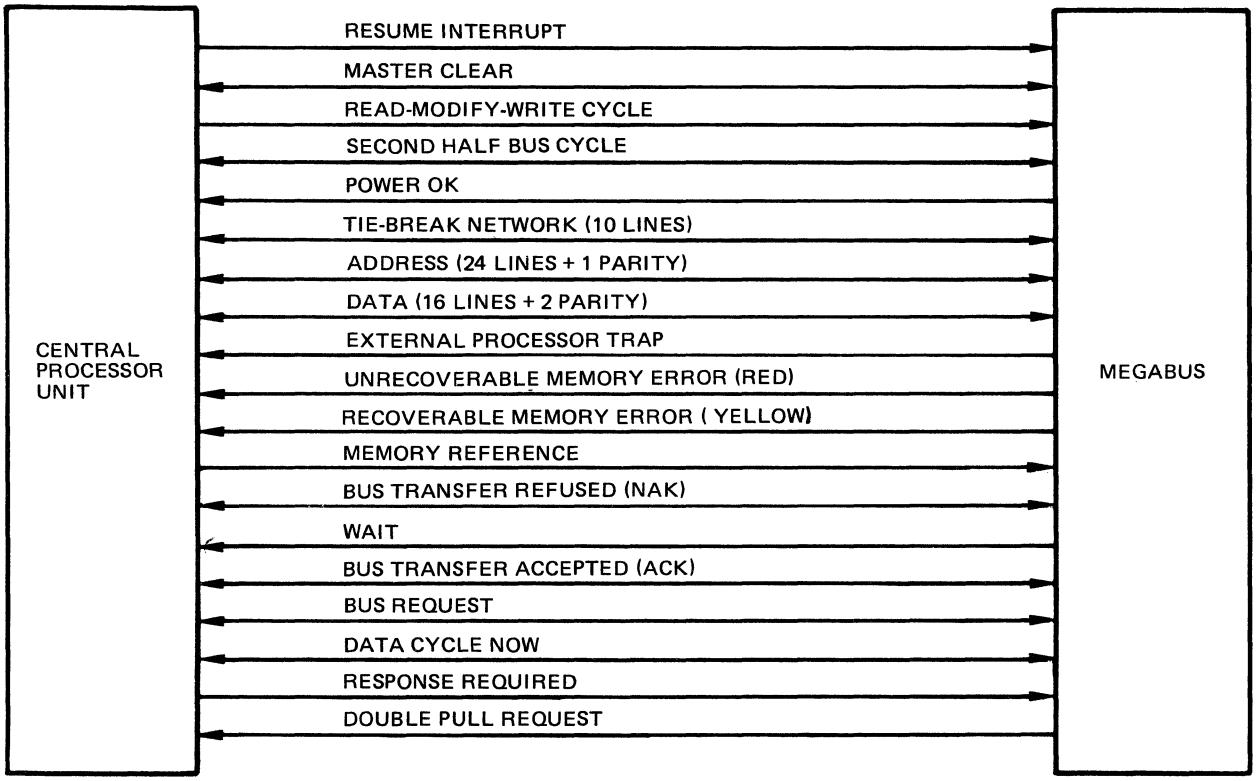


Figure 1-2 CPU/Megabus Interface Signals

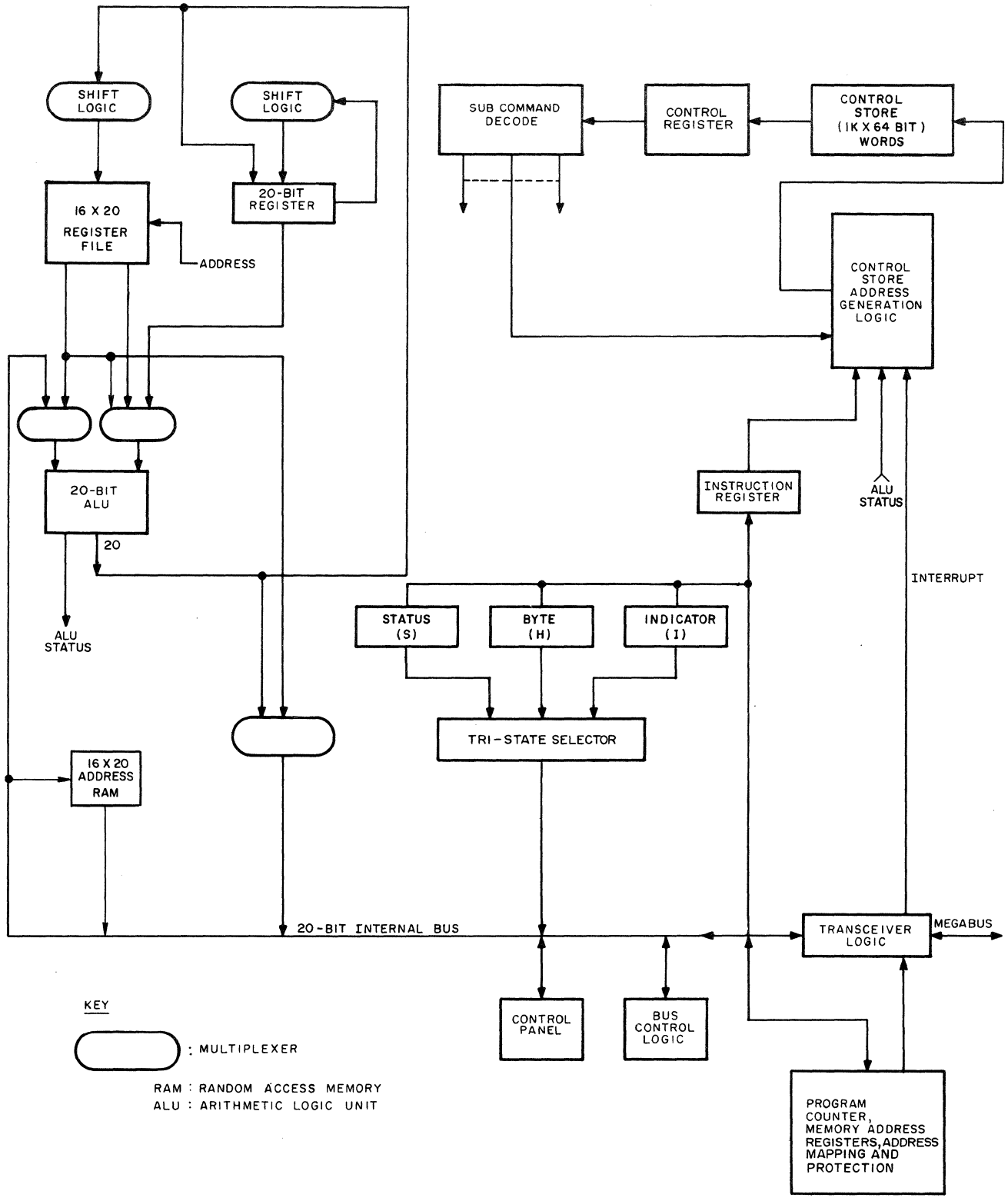


Figure 1-3 CPU Functional Block Diagram

1.2 REFERENCE DOCUMENTATION

The following support documents for the Model 43/53 CPU contain supplementary information for the data presented in this manual.

	DESCRIPTION	ORDER NUMBER
1.	Model 53 CPU Manual - Volume 2 (cache memory)	FQ30
2.	Model 53 CPU Manual - Volume 3 (MMU)	FW85
3.	Model 47 CPU Manual - Volume 2 (CIP)	FW82
4.	Type CPF9503 Scientific Instruction Processor Manual	FN30

1.3 SOFTWARE VISIBLE REGISTERS

Twenty six program visible registers, counters, and discrete signal lines are available for use by software to maintain software related data or instructions, and to display selected CPU status conditions for software interrogation. These elements include:

- Seven base registers
- Seven data registers
- Indicator register
- Seven mode registers
- Program counter
- Remote data descriptor base register
- Status/security register
- Stack pointer.

The software application for each of the above is described in the following subsections.

1.3.1 Base Registers (B1 Through B7)

Seven base registers (B1 through B7) are located in the register file portion of the microprocessor (refer to subsection 4.4.2), and maintain a 20-bit address of any procedure, data, array, or arbitrary location in memory.

1.3.2 Data Registers (D1 Through D7)

Seven data registers (D1 through D7) are located in the register file portion of the microprocessor (refer to subsection 4.4.2), and serve as 16-bit (10 through 1F) general purpose registers or accumulators. Bit 10 of each register is considered

the most significant bit. Each data register can also be used for post-indexing of addresses (i.e., as index registers).

NOTE

In software notation these registers are designated R1 through R7.

1.3.3 Indicator Register (I)

The indicator register (I) contains several single bit indicators that provide temporary storage for overflow and program status information.

1.3.4 Mode Registers (M1 Through M7)

The mode registers (M1 through M7) reside in locations 1 through 7 of the CPU Random Access Memory (RAM), refer to subsection 4.9.1. These registers retain mode information pertaining to the CPU and other processors (i.e., CIP and/or SIP), and can be modified by the MTM instruction. Currently, registers M2, M6, and M7 are reserved for future use, while the remaining four registers (M1 and M3 through M5) define the following system features:

- M1 - CPU trace and overflow trap masks
- M3 - CIP truncation and overflow trap masks
- M4 - SIP memory and accumulator length control and round/truncate mode control
- M5 - SIP exponent underflow, significance, and precision error trap masks.

1.3.5 Program Counter (P Register)

The program counter (or P register) normally contains the storage address of the next instruction to be executed by the CPU.

1.3.6 Remote Descriptor Base Register (RDBR)

The Remote Descriptor Base Register (RDBR) resides in location B of the CPU Random Access Memory (RAM), refer to subsection 4.9.1. The contents of this register point to a table of up to 4,096 data descriptors.

1.3.7 Status/Security Register (S)

The status/security register (S) contains the system status and security keys.

1.3.8 Stack Pointer (T Register)

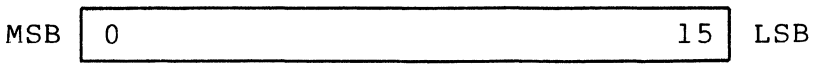
The stack pointer (or T register) resides in location A of the RAM. The contents of this register point to the base of the current stack structure in memory.

1.4 DATA WORD FORMATS

This subsection defines the various data word formats that are used by the CPU.

1.4.1 Memory Data

All data elements, such as a bit or byte, are based on 16-bit memory words. The format of each word is defined from left to right with the first bit numbered 0:

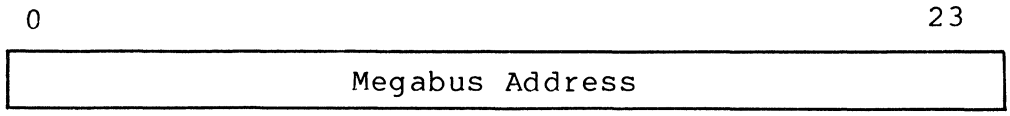


Memory data may be accessed by instructions to the bit, byte, digit, word, or multiword data item level. In all cases, the leftmost element is the most significant element of the word; e.g., bit 0 (above) is the first bit, bit 1 is the second bit, bits 0 through 7 are the first byte, bits 8 through 15 are the second byte, etc. Multiword items require successive word locations; the lowest address is defined as the leftmost or most significant part of the data item.

1.4.2 Addresses

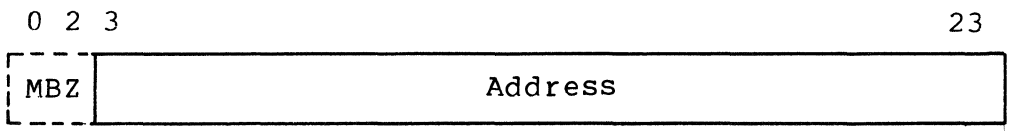
An address pointer is used to point to bit, byte, word, or multiword data items. This address indicates the leftmost and most significant element of the data item. Within an array, data items are numbered from left to right.

The CPU may operate in either Long or Short Address Form (LAF or SAF). LAF provides virtual addressability to 1M words, whereas SAF provides addressability to 64K words. Addresses are unsigned. Physical byte addresses must be presented to the Megabus and must contain 24 bits.



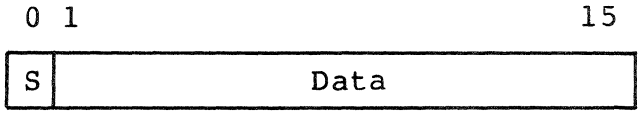
Bit positions of processor address registers are numbered to correspond to their positions on the Megabus with an appropriate number of leading zeros.

The CPU generates addresses which may contain 21 significant bits:



1.4.3 Signed Integer Data Word

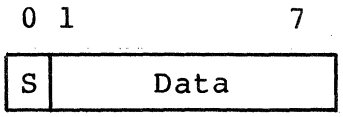
The data field is a 16-bit integer (in two's complement form) with the radix point to the right of bit 15, the least significant bit. Bit 0 indicates the Sign (S) of the data field. The format of the signed integer data word is:



S = Zero, sign is positive.
S = One; sign is negative.

1.4.4 Signed Integer Data Byte

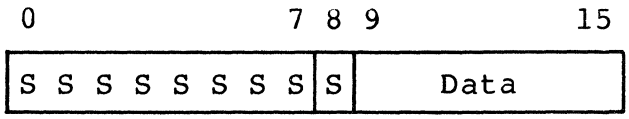
The data field is an 8-bit integer (in two's complement form) with the radix point to the right of bit 7, the least significant bit. Bit 0 indicates the Sign (S) of the data field. The format of the signed integer data byte is:



S = Zero; sign is positive.
S = One; sign is negative.

1.4.5 Sign Extended Integer Byte in Word

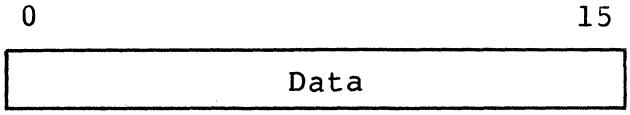
The data field is a 16-bit integer (in two's complement form) with the radix point to the right of bit 15, the least significant bit. The Sign bit (S) is extended from bit 8 through bit 0 of the data word. The format of the sign extended byte in a word is:



S = Zero; sign is positive.
S = One; sign is negative.

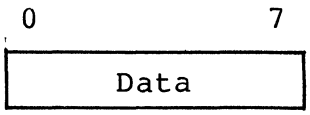
1.4.6 Unsigned Integer Word

The data field is a 16-bit integer. The format of the unsigned integer word is as follows:



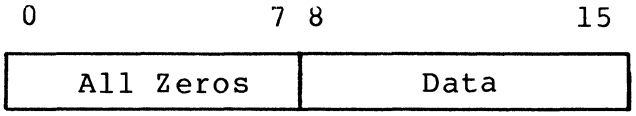
1.4.7 Unsigned Integer Byte

The data field is an 8-bit integer. The format of the unsigned integer byte is as follows:



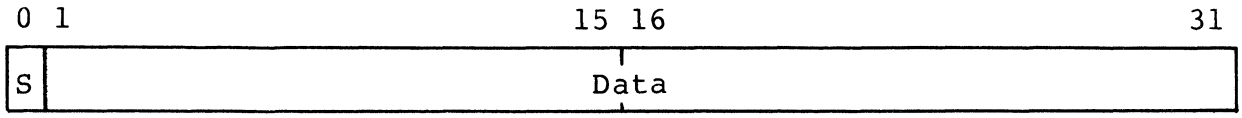
1.4.8 Unsigned Integer Byte in Word

The data field is an 8-bit integer and bits 0 through 7 are Zero. The format of the unsigned integer byte in a word is:



1.4.9 Signed Integer Double-Word

The data field is a 32-bit integer (in two's complement form) with the radix point to the right of bit 31, the least significant bit. Bit 0 indicates the Sign (S) of the data field. The format of the signed integer double-word is as follows:



S = Zero; sign is positive.
S = One; sign is negative.

1.5 INSTRUCTION WORD FORMATS

The CPU instruction set is divided into eight categories:

- Double-operand instructions
- Single-operand instructions
- Input/Output instructions
- Short value immediate instructions
- Branch on register instructions
- Branch on indicator instructions
- Shift instructions
- Generic instructions.

For a complete list of Level 6 instructions and their definitions, refer to the Honeywell Level 6 Minicomputer Handbook (order number AS22).

The double-operand and single-operand type instructions use an Address Syllable (AS) field to generate address references. A decode of the AS field usually results in the formulation of an Effective Address (EA), which points to an operand. The address syllable can take one of the following three formats:

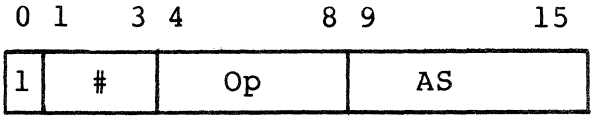
1. Register AS (RAS): The source or destination of the operand is a register (D or B).

2. Immediate Operand (IMO): The operand follows the instruction (special case of MAS, see item 3).

3. Memory AS (MAS): This form specifies a memory location that contains the operand.

1.5.1 Double-Operand Instructions

Double-operand instructions have the following format:



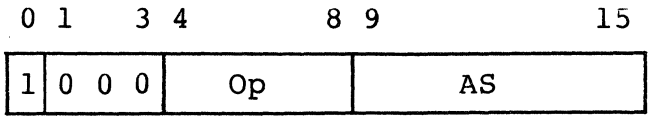
Op = op-code field
# = register number
AS = Address Syllable; refer to subsection 1.6 for the AS format.

Within this group, three types of instruction are available: (1) address register instructions, (2) data register instructions, and (3) mode register instructions. The type of register (D, B, or M) selected by the register number field is a function of the op-code. Depending on whether the address syllable specifies RAS, MAS, or IMO format, these instructions are defined as having the following formats, respectively:

- RR: Register to Register
- RM: Register to Memory
- RI: Register Immediate

1.5.2 Single-Operand Instructions

Single-operand instructions have the following format:



Op = op-code field
AS = Address Syllable; refer to subsection 1.6 for the AS format.

Within the group, three types of instructions are available: (1) control instructions, (2) bit instructions, and (3) modify operand instructions. Depending on whether the address syllable specified RAS, MAS, or IMO format, these instructions are defined as having the following formats, respectively:

- R: Register only
- M: Memory only
- I: Immediate only.

NOTE

Some instructions that modify operands in memory operate in the Read Modify Write (RMW) mode. In this mode, the selected memory cannot be accessed by any other processor in RMW mode until the location is modified by the current RMW. This feature is useful for synchronization in a multiprocessor environment.

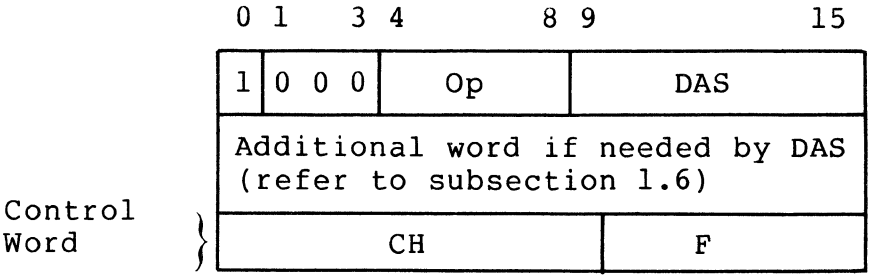
1.5.3 Input/Output Instructions

The I/O instructions are defined by their format as follows:

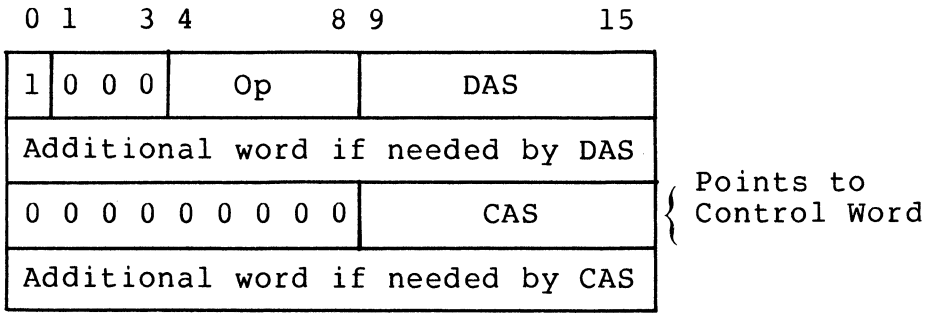
- Data and command I/O instructions
- Address and range output instruction.

1.5.3.1 Data and Command I/O Instructions

These instructions specify two quantities: (1) the data word that is identified by an address syllable identical to the one used for single-operand instructions, and (2) the control word that identifies the external channel (or device) and the function it has to perform. The control word may be imbedded in the procedure as follows:



or it may be nonprocedural, in which case the format is as follows:



Op = op-code field.

DAS = Data Address Syllable; specifies the location from/to which the data are transferred to/from the Megabus (refer to subsection 1.6 for DAS format).

CH = Channel number or the device address.

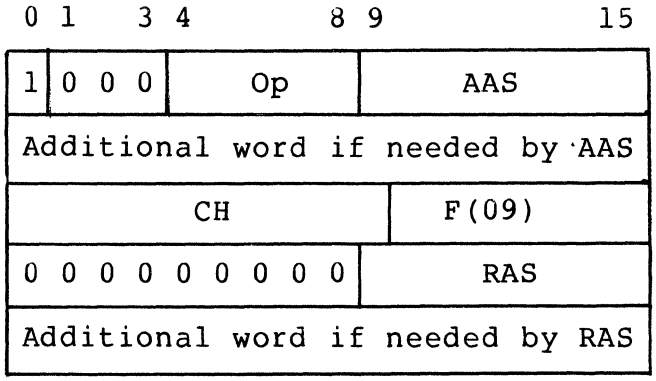
F = Function code, where:

- If F is even, data are transferred from the controller to the CPU.
- If F is odd, data are transferred from the CPU to the controller.

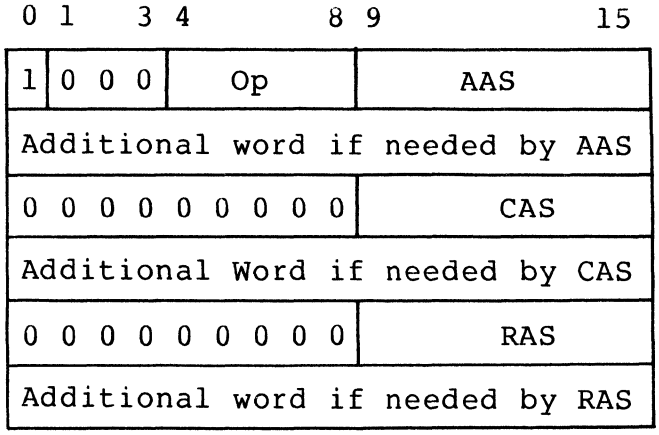
CAS = Controll Address Syllable; points to control word that contains CH and F (refer to subsection 1.6 for CAS format).

1.5.3.2 Address and Range Output Instructions

This instruction specifies three quantities: (1) the address, which is identified by an address syllable that is identical to the one used for single operand instructions, (2) the control word, which identifies the external channel (or device) and the function it has to perform, and (3) the range which is identified by an address syllable. The control word may be imbedded in the procedure as follows:



or it may be nonprocedural, in which case the format is as follows:



The definitions of the above words are the same as those specified in subsection 1.5.3.1 with the following additions:

AAS = Address Address Syllable; the byte effective address formulation from the AAS is transferred to the Megabus (refer to subsection 1.6 for AAS format).

RAS = Range Address Syllable; specifies the location from which the range is transferred to the Megabus (refer to subsection 1.6 for RAS format).

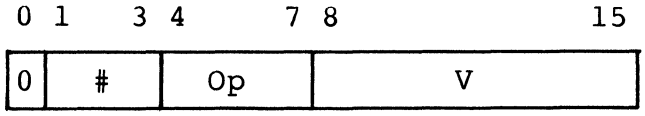
F = Function code; must specify the function code that is used to load the channel address register; otherwise, the operation is unspecified.

NOTE

All I/O instructions are privileged. If the privilege bit is zero, a trap will result (using trap vector 13) in lieu of execution (refer to Table 2-5).

1.5.4 Short Value Immediate Instructions

Short value immediate instructions have the following format:



Op = op-code filed.

= register number; selects one of seven word operand registers.

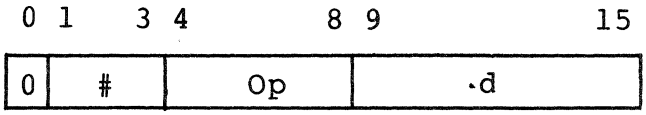
V = immediate operand value; the value is between -128 and +127, inclusive.

These instructions operate on the word operand (D) registers and perform the following operations:

- Load
- Compare
- Add
- Multiply.

1.5.5 Branch on Register Instructions

Branch on register instructions have the following format:



Op = op-code field.

= register number; selects one of seven word operand registers.

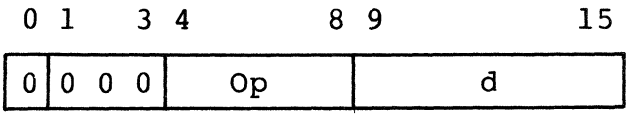
d = displacement; the relative address of the branch destination (refer to subsection 1.7.1).

These instructions enable branching on specified conditions of a selected word operand register, such as:

- Equal to zero
- Less than zero
- Increment and test
- Decrement and test.

1.5.6 Branch on Indicator Instructions

Branch on indicator instructions have the following format:



Op = op-code field.

d = displacement; the relative address of the branch destination (refer to subsection 1.7.1).

These instructions enable branching on various indicators, such as:

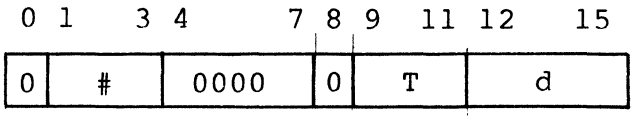
- Carry
- Equal
- Less than
- Greater than
- I/O bit.

1.5.7 Shift Instructions

The shift instructions have two formats: shift short and shift long.

1.5.7.1 Shift Short Format

The shift short instruction format is as follows:



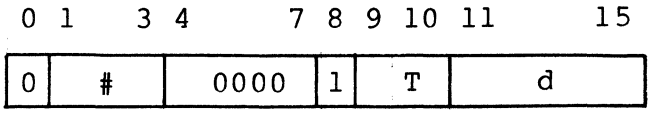
= register number; selects one of seven word operand registers.

T = type and direction of the shift.

d = distance (value between 1 and 15, inclusive); if d = 0, substitute the contents of D1 (bits 12 through 15) for distance.

1.5.7.2 Shift Long Format

The shift long instruction format is as follows:



= register number; selects one of three word operand register pairs.

T = type and direction of the shift.

d = distance (value between 1 and 31, inclusive); if d = 0, substitute the contents of D1 (bits 11 through 15) for distance.

Various types of shifts on single or double operands (two registers linked together) are possible; e.g., closed, open, arithmetic, left, right, etc. For double-operand shifts, the register number must equal 3, 5, or 7; otherwise, the operation is unspecified. The pairing of the registers is as follows:

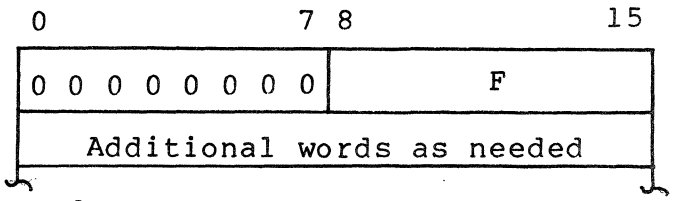
#'	#
2	3
4	5
6	7

#' = the number of the implied register that is linked together with the selected register (#).

Some shift operations modify the Carry (C) or Overflow (OV) indicators. The Carry indicator reflects the state of the last bit shifted out. If the Carry or Overflow indicator is to be modified and the actual shift distance is zero, the Carry or Overflow indicator is clear.

1.5.8 Generic Instructions

The generic instructions have the following format:



F = function code:

Within this group, the following types of instructions are available:

1. Acquire Stack Space
2. Activate Segment Descriptor
3. Breakpoint Trap
4. Control Real-Time Clock
5. Control Watchdog Timer
6. Dequeue On Address
7. Dequeue From Head
8. Halt
9. Load Remote Descriptor Base
10. Load Stack Pointer
11. Memory-To-Memory Move
12. Monitor Call
13. Queue On Head
14. Queue On Tail
15. Reconfigure External Processor
16. Relinquish Stack Space
17. Return From Trap
18. Store Stack Pointer
19. Store Remote Descriptor Base
20. Validate Address, Range, and Access Rights.

1.6 CPU/MEMORY EFFECTIVE ADDRESSING

Address generation depends on the following address types contained in the instruction.

- Displacement
- Address Syllable

In some cases, the contents of the Program Counter (P) are used to generate the effective address; P is assumed to be pointing to the word that contains the displacement in question.

1.6.1 Displacement Address Type

The first type of address definition field is the displacement type used with branch on register and branch on indicator instructions (refer to subsection 1.5.5 and 1.5.6, respectively).

The formulation of the effective address is:

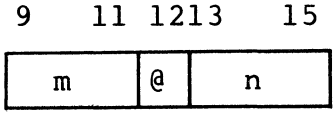
1. If d = 0, the effective address is obtained from the second (and third) word(s) of the instruction.

2. If d = 1, the effective address is obtained by adding to P the Displacement (DSP) contained in the second word of the instruction.

3. If d ≠ 0 or 1, the effective address is obtained by adding the displacement (a two's complement number with a value between -64 and +63, inclusive) to the address of the instruction (P).

1.6.2 Address Syllable

The single-operand and double-operand instructions generate address references through a field called the Address Syllable (AS). The format of the address syllable is:



m = address modifier.
@ = indirect addressing bit.
n = register number; values between 0 and 7, inclusive.

Figure 1-4 is a representation of the complete address syllable. Table 1-1 lists a set of definitions to facilitate the use of these AS descriptions.

1.6.2.1 Register Address Syllable

The Register Address Syllable (RAS) addresses a register that is the source or destination for the operand. The following subset of Figure 1-4 is defined as RAS:

m	N > 0
	@ = 0
5	B register or D register

n=0			n>0			
m	@=0	@=1	@=0	@=1		
0	IMA	*IMA	Bn	*Bn		
1	IMA+D1	*IMA+D1	Bn+D1	*Bn+D1		
2	IMA+D2	*IMA+D2	Bn+D2	*Bn+D2		
3	IMA+D3	*IMA+D3	Bn+D3	*Bn+D3		
4	P+DSP	*(P+DSP)	Bn+DSP	*(Bn+DSP)		
5	RFU	RFU	Bn register or Dn register	n=1,2, or 3 Bn + ↓D1	n=4 RFU	n=5,6, or 7 B(n-4) + D1↑
6	RFU	RFU	↓Bn	n=1,2, or 3 Bn + ↓D2	n=4 RFU	n=5,6, or 7 B(n-4) + D2↑
7	IMO	IV+DSP	Bn↑	n=1,2, or 3 Bn + ↓D3	n=4 RFU	n=5,6, or 7 B(n-4) + D3↑

NOTE

Entries in this figure are mnemonics for the various address forms available. These forms are described in subsections 1.6.1 through 1.6.3.

Figure 1-4 Address Syllable

Table 1-1 Address Syllable Notation

NOTATION	DESCRIPTION
DSP	DSP indicates a 16-bit signed displacement that follows the address syllable
*	Indirect operator (≠ @)
+D	Specifies indexing
↑	Auto-increment (B↑ or D↑ indicates postincrementation)
↓	Auto-decrement (↓B or ↓D indicates predecrementation)
IMA	Immediate Address
B	Base register
D	Operand register
P	Program counter; for the purpose of P relative addressing, P points to the word containing the displacement
()	Logical binding
[]	Contents of
+	Addition operator
-	Subtraction operator
x	Multiplication operator
←	Is replaced by
EA	Effective Address
IEA	Intermediate Address
IMO	Immediate Operand
IV	Interrupt Vector (points to Interrupt Save Area of Current Level)

The interpretation of RAS is determined by the op-code. For op-codes on Base (B) registers, RAS is defined as follows:

	n > 0
m	@ = 0
5	B register

Select word operand register n (n = value between 1 and 7, inclusive).

For all other single-operand and double-operand instructions, excluding LAB, LNJ, JMP, ENT, SAVE, and RSTR, RAS is defined as follows:

	n > 0
m	@ = 0
5	D register

Selects word operand register n (n = value between 1 and 7, inclusive).

1.6.2.2 Immediate Operand Address Syllable

The Immediate Operand Address Syllable (IMO) specifies an operand of appropriate size, which immediately follows the instruction. The following subset of Figure 1-4 is defined as IMO:

m	n = 0
	@ = 0
7	IMO

1.6.2.3 Memory Address Syllable

The Memory Address Syllable (MAS) specifies the effective address of a memory location. The MAS can have one of three formats: (1) P relative, (2) Immediate Address, or (3) B relative.

P Relative Format

The P Relative format is as follows:

	n = 0	
m	@ = 0	@ = 1
4	P + DSP	*(P + DSP)

P + DSP: The effective address is formed by adding DSP to the contents of P

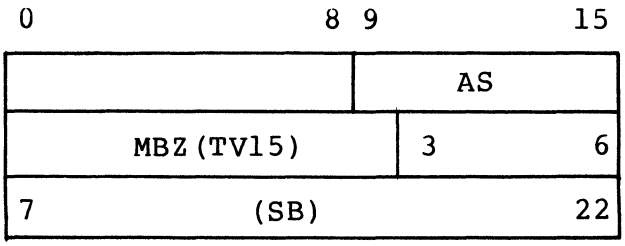
*(P + DSP): The effective address is pointed to by P + DSP.

Immediate Address Format

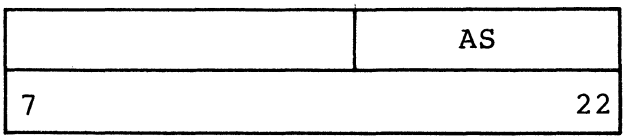
The Immediate Address (IMA) format is as follows:

	n = 0	
m	@ = 0	@ = 1
0	IMA	*IMA
1	IMA + D1	*IMA + D1
2	IMA + D2	*IMA + D2
3	IMA + D3	*IMA + D3

IMA: In LAF, use the two words that follow the address syllable as shown below:



IMA: In SAF, use the 16 bits in the word that follows the address syllable:



IMA: The effective address is contained in the location(s) pointed to by IMA (is the indirect operator).

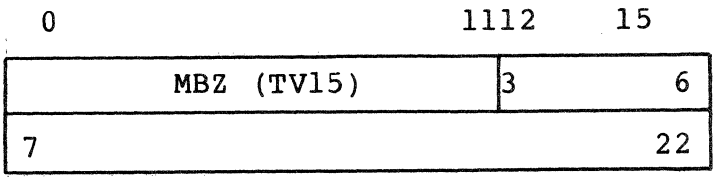
IMA + D (1, 2, or 3): The effective address is IMA indexed by the appropriate index register (see the following notes).

*IMA + D (1, 2, or 3): The effective address is obtained by adding the contents of the appropriate index register to the contents of the location(s) pointed to by IMA; indirect, post indexing (see the following notes).

NOTES

1. If the operand is larger or smaller than 16 bits, scale the index value before adding.

2. If in the LAF mode, indirection extracts a double-word (IMA and IMA + 1) as follows:



B Relative Format

The B Relative format is as follows:

m	n > 0		
	@ = 0	@ = 1	
0	Bn	*Bn	
1	Bn + D1	*Bn + D1	
2	Bn + D2	*Bn + D2	
3	Bn + D3	*Bn + D3	
4	Bn + DSP	*(Bn + DSP)	
5		n = 1,2,3 Bn + ↓D1	n = 5,6,7 B(n-4) + D1↑
6	Bn	n = 1,2,3 Bn + ↓D2	n = 5,6,7 B(n-4) + D2↑
7	Bn	n = 1,2,3 Bn + ↓D3	n = 5,6,7, B(n-4) + D3↑

Bn: The effective address is contained in base register n (n = value between 1 and 7, inclusive).

*Bn: The effective address is contained in the memory location pointed to by base register n (n = value between 1 and 7, inclusive).

Bn + D1, D2, or D3: The effective address is obtained by adding the contents of the appropriate index register to the contents of base register n (n = value between 1 and 7, inclusive). Refer to the following note.

*Bn + D1, D2, or D3: The effective address is obtained by adding the contents of the appropriate index register to the contents of the memory location(s) pointed to by base register n (n = value between 1 and 7, inclusive). Refer to the following note.

NOTE

If the operand is larger or smaller than 16 bits, scale the index value before adding.

Bn + DSP: The effective address is obtained by adding DSP to the contents of base register n (n = value between 1 and 7, inclusive).

*(Bn + DSP): The effective address is contained in the location(s) pointed to by Bn + DSP (n = value between 1 and 7, inclusive).

↓Bn: The effective address is contained in base register n after it is decremented by the operand size in words (n = value between 1 and 7, inclusive).

Bn↑: The effective address is contained in base register n (n = value between 1 and 7, inclusive). The base register is incremented by the operand size in words after effective address formation and prior to instruction execution.

Bn + Dx: The effective address is obtained by adding the contents of the appropriate (m - 4) index register (after it is decremented by 1) to the contents of the selected (n - 4) base register. If the operand is larger or smaller than 16 bits, the index value is scaled to operand size before addition.

B (n-4) + Dx: The effective address is obtained by adding the contents of the appropriate (m - 4) index register to the contents of the selected (n - 4) base register. After effective address formation and prior to execution of the op-code, the selected index register is incremented by 1. If the operand is larger or smaller than 16 bits, the index value is scaled to operand size before addition.

1.6.3 Indexing

Many address syllable forms specify indexing. During effective address generation, one of three index registers (D1, D2, or D3) is algebraically added as the last step, after any indirection, to the base address. The index value is treated as a signed integer data word.

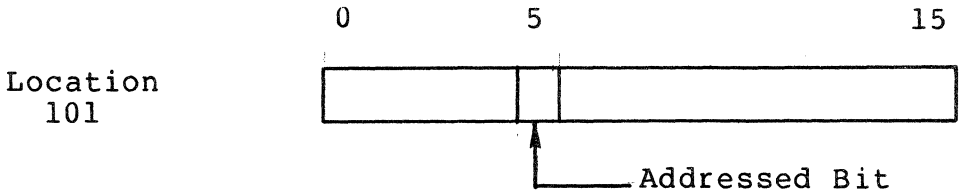
While indexing, the hardware automatically aligns the index value to correspond to the size of the item being referenced. The size of this item is determined by the op-code type (bit, byte, digit, word, double-word, or quadruple-word).

1.6.3.1 Bit Addressing

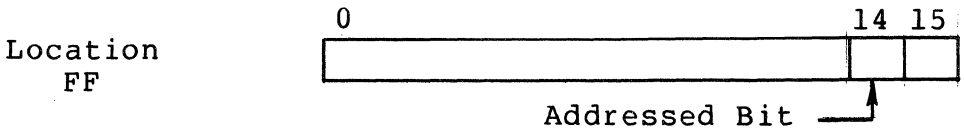
The index value for bit instructions (LB, LBF, LBT, LBC, and LBS), where the address syllable calls for indexing (IMA + Dx, Bn + Dx, *Bn + Dx, etc.), is aligned to denumerate bits in the array, with bit 0 being the leftmost bit in word 0 of the array. As an example, consider the following instruction.

```
LB, B1 + D1
```

If B1 = 100 and D1 = +21, then the addressed bit is bit 15 in location 101.



If B1 = 100 and D1 = -2, then the addressed bit is bit 14 in location FF.

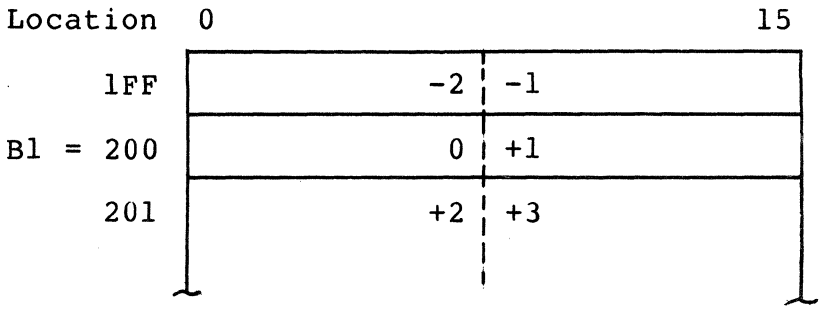


1.6.3.2 Byte Addressing

The index value for byte instructions (LDH, STH, CMH, etc.), where the address syllable calls for indexing, is aligned to denumerate bytes in the array, with byte 0 being the leftmost byte in word 0 of the array. As an example, considering the following instruction:

```
LDH, D4, B1 + D2
```

If B1 = 200 and D1 = ±n, then the addressed byte is as shown below:

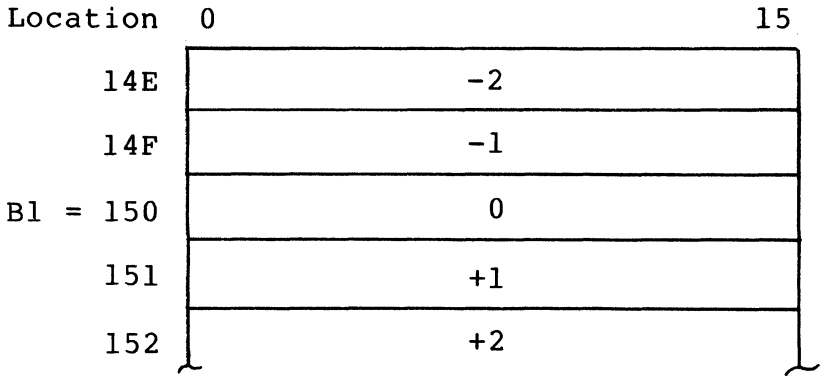


1.6.3.3 Word Addressing

The index value for word instructions (LDR, ADD, OR, etc.), where the address syllable calls for indexing, denumerates words in the array, with word 0 being the leftmost word in the array. As an example, consider the following instruction.

ADD, D3, B1 + D2

If B1 = 150 and D2 = ±n, then the addressed word is as follows:

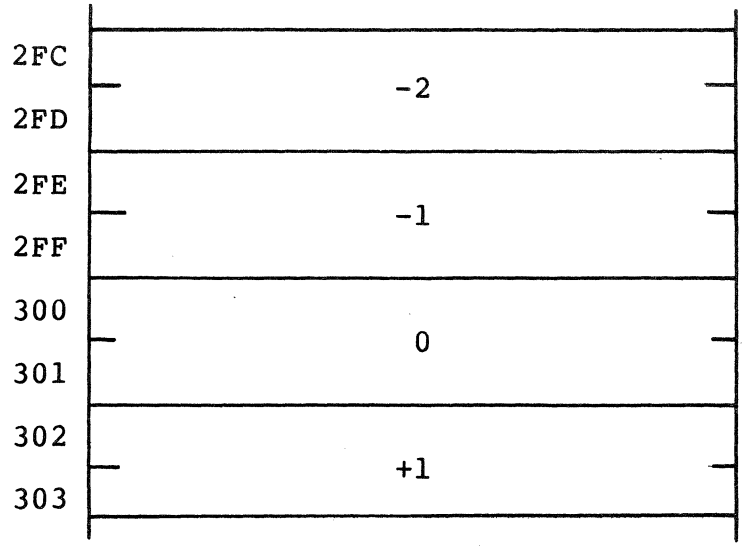


1.6.3.4 Double-Word Addressing

The index value for double-word operands, where the address syllable calls for indexing, denumerates double-words (32 bits) in the array, with the double-word zero (0) being the leftmost double-word. The processor assumes double-word operands for the following: (1) LDI, SDI, and AID instructions, and (2) scientific instructions (except branch instructions) if the memory operand length stored in M4 equals 2, and (3) all base register instrucitons (LDB, CMN, STB, CMB, and SWB) if the processor is in LAF mode. As an illustrative example, consider the following instruction.

LDI, (D6, D7), B4 + D3

If B4 = 300 and D3 = ±n, then the address operand is as follows:



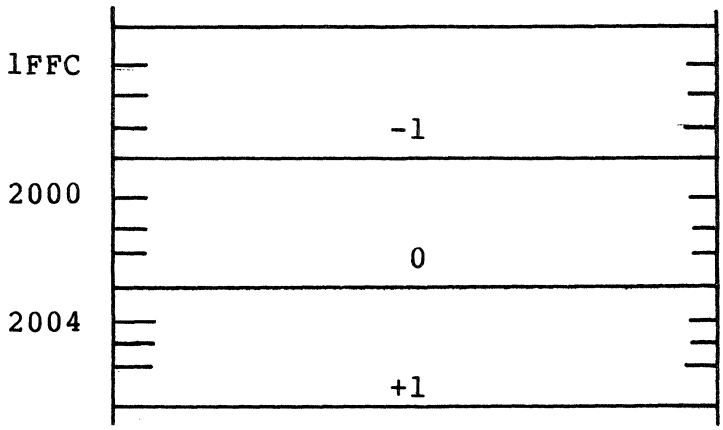
1.6.3.5 Quadruple-Word Addressing

The index value for scientific instructions (except branch instructions), where the memory operand length equals 4 (stored in M4) and the address syllable calls for indexing, assumes an array of quadruple words.

Consider the following example:

FAD,$S1,$B5.+$D1

If B5 = 2000 and D1 = ±n, then the addressed operand is as follows:



1.7 MEMORY ADDRESS BOUNDARIES

When the CPU initiates a memory cycle that addresses nonexistent memory. the processor either traps (using trap vector 15, see Table 2-5) or causes undefined results. These boundaries are of interest in defining nonexistent memory:

1. Addresses below zero

2. Addresses beyond the last module on-line at a given installation

3. Addresses greater than 1,024 K-words (i.e., main memory equals 1M-words).

If main memory is less than 1M-words, violation of any of the above three address boundaries causes a trap. If main memory is 1M-words, then trap conditions caused by incrementing the program counter, a hidden register, or a base register may not be detected and cause undefined results.

It should be noted that when the memory management unit (refer to subsection 1.2) is configured into the system, large virtual addresses may roam freely within the CPU and are not subjected to the scrutiny previously described until they are mapped and deposited onto the Megabus. The memory management unit may trap an address for other reasons, for example:

- Violation of protection (trap vector 14)
- Attempt to reference an invalid segment (trap vector 15).

II
THEORY OVERVIEW

The material in this section provides an overview description of the Central Processor Unit (CPU).

2.1 CPU HARDWARE OVERVIEW

Figure 2-1 is a major block diagram of the CPU, showing all of the major data transfer paths among the CPU major elements.

2.1.1 CPU Clock

The CPU clock generates the timing signals necessary for proper operation of the CPU. These clock signals are used to: (1) trigger all CPU registers and all test condition visible control flops, (2) stabilize registers after being loaded, and (3) signify that address, data, control, and parity are valid, allowing initiation of a Megabus cycle. The timing signals distributed throughout the CPU also provide four clock cycles that differ only in the duration of the cycle.

The clock is stalled early if needed Megabus data have not arrived and additional activity is to be performed in a cycle. It is stalled late to synchronize with a Megabus cycle if no activity is to be performed in the cycle.

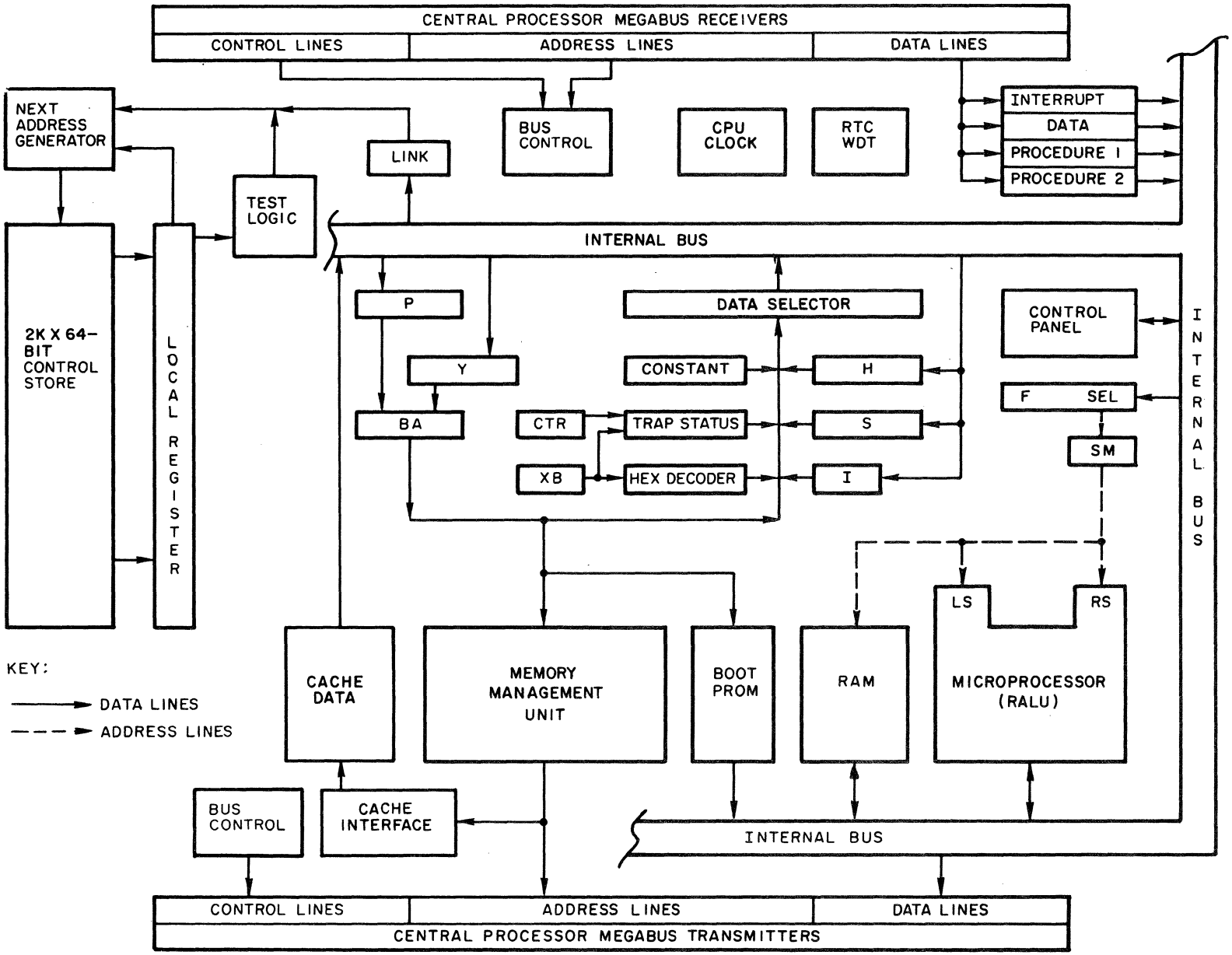


Figure 2-1 CPU Major Block Diagram

2.1.2 Control Store

Control store is comprised of a maximum of 2,048 by 64-bit locations (firmware words). These firmware words control almost all hardware operations within the CPU. Refer to subsection 3.1 for a detailed description of the control store (firmware) word.

All firmware words, as they are extracted from control store, are temporarily stored in a 72-bit register called the Control Store Local Register (CR).

2.1.3 Next Address Generation (NAG) Logic

The CPU uses one of the following three methods to generate the next firmware address:

- Method 1: Test and Branch
- Method 2: Major Branch
- Method 3: Subroutine Return.

Each of the above methods is a conditional branch based on a test condition selected by bits 42 through 47 of the control store word.

Method 1: This method is used when the next address(es) can be explicitly specified in bits 53 through 63 of the control store word.

Method 2: This method is used when branching to another firmware routine. The next address is conditionally obtained from the branch logic (refer to subsection 2.1.3.2) which generates numerous preassigned addresses.

Method 3: This method is used to conditionally return the firmware to the next control store location after execution of a firmware subroutine. The return address is obtained from the LINK register (refer to subsection 2.1.3.3), and must be stored in this register prior to entry to the specified subroutine.

2.1.3.1 Test Logic

The test logic selects 1 of 64 possible test conditions to participate in generating the next firmware address. Depending on whether the tested condition is true or false, the Test Condition True function is generated to control the NAG logic.

2.1.3.2 Branch Logic

The branch logic makes numerous preassigned addresses available for Major Branch operations. The address selected is determined by a decode of the instruction register contents, the control store outputs, and other control flops.

2.1.3.3 LINK Register (XL)

The XL register is an 8-bit register that is loaded from the internal bus and is available to the NAG logic.

2.1.4 Microprocessor

The microprocessor performs most of the arithmetic, logical, and shift operations required by the Level 6 system, including storage of operands for subsequent use by the firmware and over half the software visible registers. Included within this element are 17 storage registers, plus an arithmetic logic unit, that are used to implement the above functions. Of the 17 storage registers, 14 are software visible. The other three registers are visible only to the firmware, and are used as work registers for temporary storage of operands during firmware operations.

2.1.5 Microprocessor Addressing Logic

The Left Select (LS), Right Select (RS), and Selector Modifier (SM) logic areas comprise the microprocessor addressing logic. Although they are not physically part of the microprocessor, these logic areas perform the required register selection. The LS logic also provides addressing for the Random Access Memory (refer to subsection 2.1.9.1).

2.1.6 Internal Bus

The internal bus receives data from any one of several sources and makes this data available to destinations throughout the CPU.

Elements that function as internal bus sources include:

- Microprocessor output
- Sixteen 20-bit registers (RAM)
- Megabus buffer registers
- Other control registers (refer to subsection 2.1.7)
- Constant-generation facilities.

Elements that may serve as destinations for the internal bus include:

- Microprocessor input
- 16 RAM registers
- Memory address register and program counter
- Instruction register
- Other control registers (refer to subsection 2.1.7)
- Other control logic (e.g., Test logic).

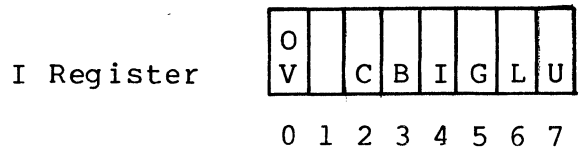
In general, the internal bus receives inputs from a single source and optionally delivers copies to one destination in each of the categories listed above. Internal bus data are also available to the Megabus.

2.1.7 CPU Registers

The CPU registers, except those contained within the microprocessor, are described in the following subsections. Refer to subsections 4.4.2 and 4.4.3 for a description of the microprocessor registers.

2.1.7.1 Indicator Register (I)

The I register is an 8-bit software-visible register that contains various single bit indicators. The register format is as follows:



The indicators contained in this register can be grouped as follows:

- Arithmetic indicators
  - OV (overflow indicator)
  - C (carry bit)
- Bit indicator
  - (bit test indicator)
- I/O indicator
  - I (input/output indicator)
- Comparison indicators
  - G (greater than indicator)
  - L (less than indicator)
  - U (unequal signs indicator)

2.1.7.2 LINK Register (XL)

Refer to subsection 2.1.3.3 for a description of the XL register.

2.1.7.3 Counter Register (CTR)

The CTR register is a 4-bit counter that indicates the number of procedure words consumed in the processing of the current instruction. Its value is reported in the trap status Z-word; otherwise, it is not software-visible.

2.1.7.4 Select Register (SEL)

The SEL register is a 4-bit register (not visible to software) that can be loaded from the internal bus. SEL can also be decremented and tested for zero by the firmware, and is thus useful, for example, to maintain the number of loops for a Multiply or Divide operation.

2.1.7.5 Byte Indexing Register (XB)

The XB register is a 4-bit shift register (not visible to software) that supplies trap context information regarding indexing of bit or byte operations. The output from this register is fed to both the internal bus and the hexadecimal decoder logic.

2.1.7.6 Instruction Register (F)

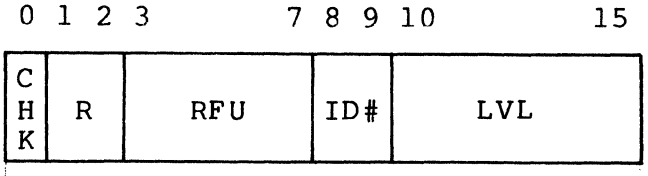
The F register is a 12-bit temporary storage register (not visible to software) that usually holds the most significant 12-bits of the instruction word as it is received from memory. Instruction words for the F register are received over the internal bus and loaded under firmware control.

2.1.7.7 H Register

The H register is a 16-bit register (not visible to software) that accepts data directly from the internal bus and makes these data available to the internal bus source selector for byte swap operations onto the internal bus (i.e., the least and most significant eight bits of the register are swapped as its contents are deposited onto the internal bus).

2.1.7.8 Status/Security Register (S)

The S register is a 16-bit software-visible register that contains the system status and CPU security keys. The format of this register is:



CHK - On if a system device did not pass its QLT

R - Ring Number

00 = Ring 3 } User
01 = Ring 2 }

10 = Ring 1 } Privileged
11 = Ring 0 }

RFU - Reserved for Future Use

ID# - CPU Identity Number (assigned during system configuration) provides the two least significant bits of the CPU channel number

LVL - Interrupt priority level 0 (highest) through 63 (lowest)

Instructions directed from a user ring at system resources (e.g., HLT, I/O, etc.) are not executed and cause a unique trap. Further, when the memory management unit is enabled, it scrutinizes each address to determine whether this user is permitted access to this location for the purpose intended (read, write, or execute).

All instructions directed from a privileged ring at system resources are permitted, subject to access rights checking.

2.1.7.9 P Register (Program Counter)

The P register is a 20-bit software-visible counter that is always incremented during instruction execution to point to the next procedure word.

2.1.7.10 Memory Address Register (Y)

The Y register is a 20-bit counter (not visible to software) that makes addresses available (via the address bus) to the Megabus, to the Memory Management Unit, and (via the internal bus) to the firmware.

2.1.7.11 Megabus Procedure Buffers (BP)

The Megabus procedure buffers, also known as the Procedure 1 (P1) and Procedure 2 (P2) registers, are 16-bit storage buffers

that receive procedure words requested from memory. Neither of these buffers is software visible.

2.1.7.12 Megabus Data Buffer (BD)

The Megabus data buffer is a 16-bit storage buffer (not visible to software) that receives non-procedural data from memory and/or I/O devices.

2.1.7.13 Megabus Interrupt Register (RUP)

The Megabus interrupt register is a 16-bit register (not visible to software) that receives the channel number and level number of an interrupting device. It stores this information from the time the CPU accepts the request until the CPU services the request.

2.1.7.14 Mode Registers

Refer to subsection 2.1.9.1 through 2.1.9.7 for a description of the mode registers.

2.1.7.15 M Collector Register

Refer to subsectgion 2.1.9.1.8 for a description of the M collector register.

2.1.8 CPU Control Flops and Control Signals

The CPU provides 16 control flops and five control signals. These elements serve as hardware controls that can be manipulated as directed by the firmware, and include:

CPU Control Flops

- SIGN
- MISC
- SHIN1
- SHIN2
- ZERO
- WRAP
- NEWXR
- ACK
- YELLOW
- PARER
- EXTRAP
- INTBSY
- TICK
- LOAD
- TRAFFIC
- PANOK

CPU Control Signals

- EFFRING
- NONPROC
- NOCHEK
- SEGERR
- PROV

2.1.9 Miscellaneous CPU Hardware

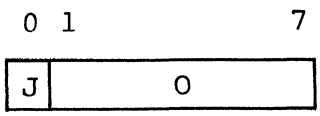
The following subsections describe those CPU elements not previously defined under one of the major CPU logic areas.

2.1.9.1 Random Access Memory (RAM)

The RAM consists of sixteen 20-bit locations. These locations contain data less frequently accessed than that stored in the microprocessor, including the seven software mode registers (M1 through M7). Addressing for the RAM is provided by the left select portion of the microprocessor addressing logic. Also included in this subsection is a description of the M collector register.

2.1.9.1.1 M1 Register

The M1 register is formatted to retain and provide the CPU trap masks as shown below:



J = Trace trap enable for jumps and branches.

- Zero = Trace Trap Disabled
- One = Trace Trap Enabled

01 through 07 = Overflow Trap - Enable controls for registers D1 through D7, respectively.

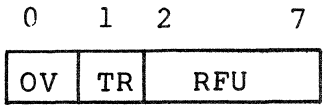
- Zero = Overflow Trap Disabled
- One = Overflow Trap Enabled

2.1.9.1.2 M2 Register

The M2 register is reserved for future use.

2.1.9.1.3 M3 Register

The M3 register is formatted as follows to retain the indicated CIP Trap Mask control (sent to the CIP for interpretaion).



OV = Overflow Trap Mask

- Zero = Trap Disabled
- One = Trap Enabled

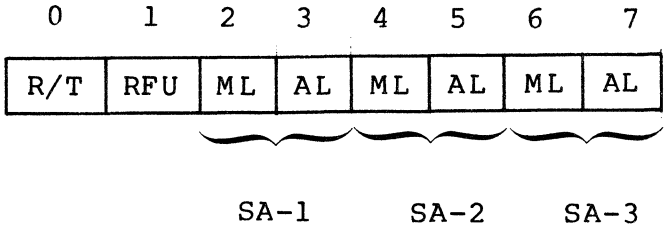
TR = Truncation Trap Mask

- Zero = Trap Disabled
- One = Trap Enabled

RFU = Reserved for Future Use (MBZ)

2.1.9.1.4 M4 Register

The M4 register is formatted as follows to retain the indicated SIP functions (sent to the SIP for interpretation).



R/T = Round/Truncate Mode

- Zero = Truncate
- One = Round

ML = Memory Operand Length in words

- Zero = 2 words (32 bits)
- One = 4 Words (64 bits)

AL = Accumulator Operand Length in Bits

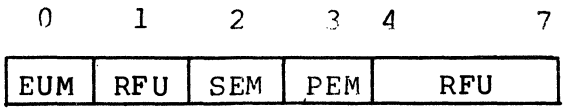
- Zero = 32-bit Operand
- One = 64-bit Operand

SA = Scientific Accumulator #1, #2 or #3.

RFU = Reserved for Future Use (MBZ).

2.1.9.1.5 M5 Register

The M5 register is formatted as follows to retain the indicated SIP functions (sent to the SIP for interpretation).



EUM = Exponent Underflow Trap Mask

- Zero = Trap Disabled
- One = Trap Enabled

SEM = Significance Error Trap Mask

- Zero = Trap Disabled
- One = Trap Enabled

PEM = Precision Error Trap Mask

- Zero = Trap Disable
- One = Trap Enabled

RFU = Reserved for Future Use (MBZ)

2.1.9.1.6 M6 Register

The M6 register is reserved for future use.

2.1.9.1.7 M7 Register

The M7 register is reserved for future use.

2.1.9.1.8 M Collector Register

Since the M register contents are stored in the RAM, they are not easily accessible to the test logic. The 8-bit M collector register is used to collect the pertinent bits that the CPU requies to make instantaneous decisions, and is formatted as follows:

```
M1(J) = Trace trap enable.
S1(D) = S1 memory operand length is quadruple-word.
S2(D) = S2 memory operand length is quadruple-word.
S3(D) = S3 memory operand length is quadruple-word.
      = Reserved for future use.
      = Reserved for future use.
      = The CIP is present and operational.
      = The SIP is present and operational.
```

2.1.9.2 Bootload PROM

The bootload PROM is comprised of 1,024, 16-bit locations that contain the standard bootload routines and internal test software for the CPU. It is automatically accessed in lieu of main memory whenever the Load mode flop on the control panel is On.

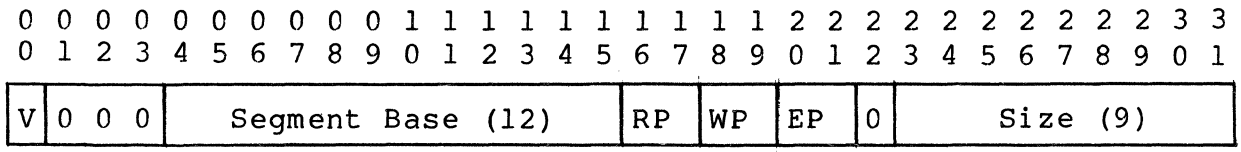
2.1.9.3 Address Bus

The address bus is a 20-bit wide bus that makes addresses for I/O and memory read or write cycles available to both the internal bus and the Megabus. It receives inputs from either the program counter or the memory address register; generally, the program counter is used for procedure references and the memory address register for all other references.

2.1.9.4 Memory Management Unit

When the Memory Management Unit (MMU) is installed, all addresses that reside in the internal processor registers (Y, B2, etc.) are reinterpreted before taking part in a memory reference (either through the Megabus or cache). Internal processor addresses are called virtual addresses; addresses after reinterpretation by the MMU are called physical addresses. Two steps are required to convert a virtual address into a physical address: (1) perform the virtual to physical mapping, and (2) determine, based on current processor states, whether this memory reference is permitted. The segmentation mechanism has been chosen to implement these requirements. Segments are sections of virtual memory space. Each segment is defined by a pattern of 32 bits stored in the MMU hardware.

2.1.9.4.1 Segment Descriptors



where:

V = Valid (this segment is currently valid)

Segment Base = Physical starting address of segment (in units of 256 words)

RP = Read Permisison (rings from which this segment is now readable)

WP = Write Permission (rings from which this segment is now writable)

EP = Execute Permission (rings from which this segment is now executable)

0 = Must be zero

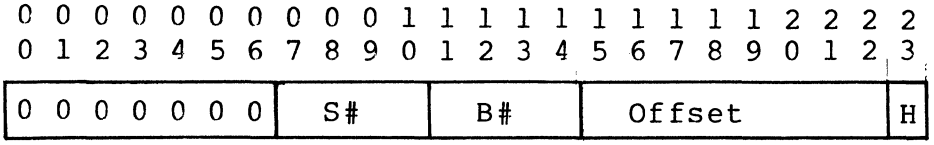
Size = Size of segment (in units of 256 words).

NOTE

A segment must begin on a 256-word boundary. The size field in the segment descriptor is in units of 256 words called a block.

2.1.9.4.2 Short Address Form (SAF) Virtual Address

When a virtual address is presented to the MMU, it is divided into three parts: a segment number, a block number, and an offset.



where:

S# = Segment number.

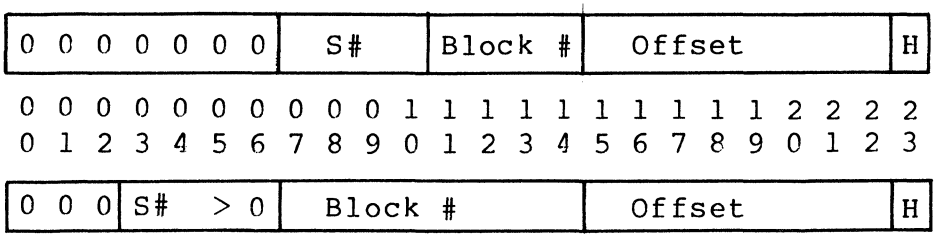
B# = Block number (maximum of 16 blocks of 256 words each).

Offset = These eight bits are never mapped.

H = Half-word address bit (never mapped).

2.1.9.4.3 Long Address Form (LAF) Virtual Addresses

LAF virtual addresses are also divided into three parts:



where:

S# = Segment number.

Block # = Maximum of 256 blocks of 256 words each (i.e., 64K).

Offset = These eight bits are never mapped.

H = Half-word address bit (never mapped).

2.1.9.4.4 Physical Address

The following steps take place when the MMU converts a virtual address into a physical address.

1. Use the segment number to fetch the segment descriptor.

2. Add the segment base (from the segment descriptor) to the block number (from the virtual address): the sum becomes bits 3 through 14 of the physical address.

3. Transmit the offset into bits 15 through 22 of the physical address.

4. Transmit the half-word bit 23 of the physical address.

2.1.9.4.5 Illegitimate Addresses

The following conditions cause a nonexistent system resource trap (TV15):

1. The validity bit in the segment descriptor is off.

2. The physical address is beyond the range of installed memory.

3. The block number in the virtual address exceeds the size field in the segment descriptor.

2.1.9.4.6 Access Rights

The MMU performs two types of checks each time it converts a virtual address into a physical address: (1) a comparison of the read/write/execute permission bits to the ring number in effect for this memory access, and (2) a determination as to whether this virtual address is legitimate (refer to subsection 2.1.9.4.5). Each of the three 2-bit fields (RP, WP, and EP) is coded to allow access to ring 0 only; 0 and 1; 0, 1, and 2; or all. The MMU (having knowledge of RP, WP, EP, the ring number, and the Megabus control lines that describe the intent of the processor regarding this memory cycle) may allow or disallow the memory reference. If the MMU disallows the memory reference, a protection violation results (TV14).

2.1.9.4.7 Activating the MMU

When Master Clear occurs, the MMU loads its segment descriptors so that no conversion takes place (trivial map). Thus, virtual and physical addresses are equal until the map is changed (by the ASD instruction or level change).

2.1.9.5 Cache Memory

The cache memory supplements main memory with a high-speed storage array for dedicated use by the CPU as a storage buffer for a copy of more frequently used CPU information. This effectively reduces access time, otherwise required by the CPU to fetch this information from main memory, and increases the overall speed and performance of the CPU. During communications

between the CPU and memory, the cache is effectively invisible to the system (i.e., if information requested by the CPU is not contained within the cache, a memory access will automatically be performed to obtain the required data).

2.1.9.6 Hexadecimal Decoder

The hexadecimal decoder consists of a 4-bit to 16-bit multiplexer that is used to generate a mask for bit and other operations. If the value of the 4-bit content of the XB register is $0 \leq N \leq 15$, then bit N of the decoder output is zero, and the other outputs are ones.

2.1.9.7 Subcommand Generator

The subcommand generator provides the control signals that permit firmware to manipulate the CPU hardware.

2.1.9.8 Constant Generators

The constant generators supply specific constants that are used by the CPU firmware.

2.1.9.9 Control Panels

Two types of control panels, full and basic, allow the operator to communicate with the CPU. The panels are available in one of the following hardware configurations.

- Bull Nose Configuration: The control panel is housed in a white molded housing, which projects from the CPU for panel visibility and operational ease.
- Industrial Configuration: The industrial configuration is mounted in a flat vertical dress panel, which is secured flush with the CPU outer exterior to allow the panel to be inside the cabinet.
- Remote Configuration: The remote configuration, using a repeater circuit board (refer to the Series 60 Level 6 Models 3X, 4X, and 5X Test and Verification Operators Guide, Order Number AW94), provides an extension of 10 cable feet between the CPU and the panel.
- Portable Configuration: The portable configuration is a self-contained full control panel, which is designed for exclusive use with the basic control panel.

2.1.9.9.1 Full Control Panel

The full control panel (see Figure 2-2) contains CPU controls, status indicators, and register displays, enabling the user to interrogate and analyze system performance. Table 2-1

describes the panel control and indicator functions. Table 2-2 lists the register selection codes. The logic design capabilities of the panel enable the user to arbitrarily:

- Stop the CPU
- Display and modify memory locations or CPU registers
- Single step through a program
- Load an operation or diagnostic program
- Manually store programs in memory
- Manually start programs.

The control panel also contains the facilities to accept diagnostic programs from an external tape cassette through the Honeywell Test and Verification Loader (TVL), which is designed specifically for this purpose. Panel coupling with this unit is accomplished by connecting the TVL signal cable to the ribbon connector at the lower right side of the control panel. It should be noted that the panel keypad array is automatically disabled while data is transferred from the cassette, to prevent erroneous alterations to its input data. For additional TVL technical and operational information and cassette specifications, refer to the Test and Verification Loader Manual (Order Number FL97).

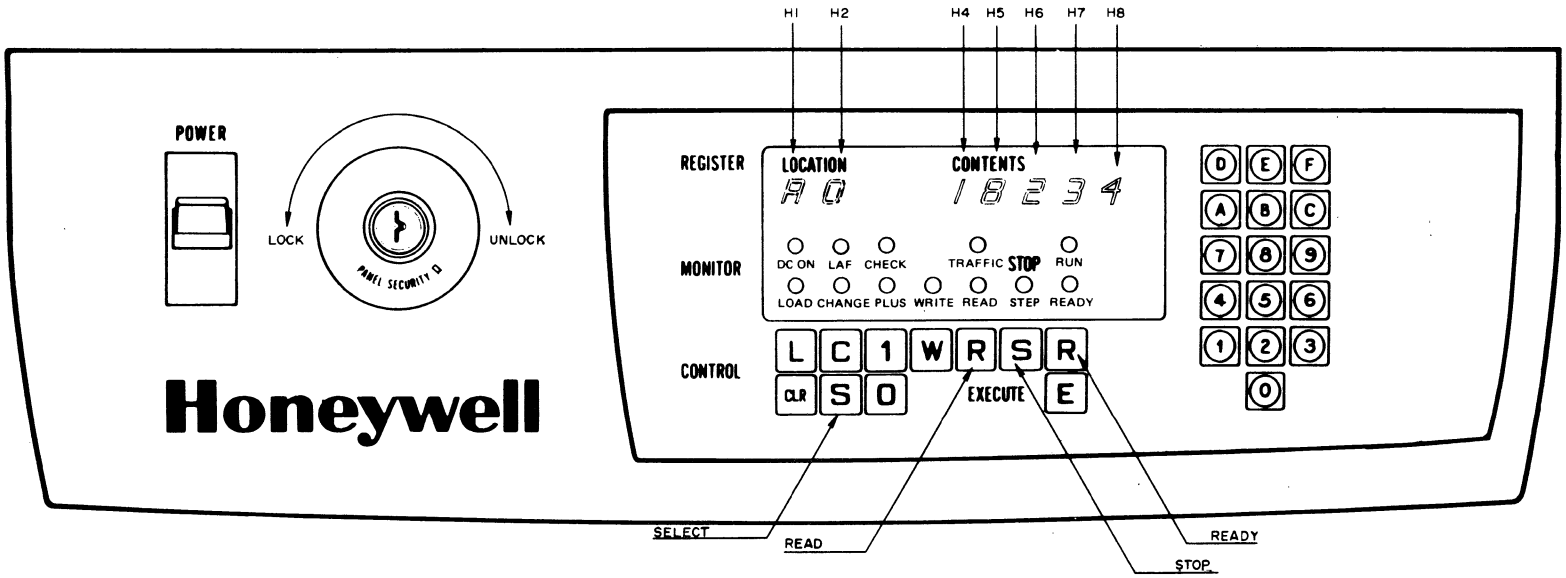


Figure 2-2 Full Control Panel

Table 2-1 Full Control Panel Switches and Indicators
(Sheet 1 of 6)

SWITCH/INDICATOR	FUNCTION
POWER Switch	Two-position switch used for engaging/disengaging the system power. Up position turns the power on; down position turns the power off. When the power is switched on, Master Clear is automatically activated and the DC ON indicator illuminates after DC power is attained.
PANEL SECURITY Switch	Two-position keylock switch used for enabling/disabling the control panel. Counterclockwise (locked) position disables all control panel switches/touch keys (except POWER) to prevent alteration of the memory or register contents; the register display is disabled (not illuminated). Clockwise (unlocked) position enables all control panel switches/touch keys; the register display illuminates, indicating that the control panel is operational.
0, 1, ... F Hexadecimal Pad Keys	Sets the appropriate digits into hexadecimal displays (H1, H2, H4 - H8).
CLR (Master Clear) Pushbutton	Activates Master Clear for the system, but is not effective while the CPU in the Run state. Depressing the CLR pushbutton: • Clears the Program Counter (E0), the H register, and the Instruction Register (D0). • Clears all pending interrupts and traps. • Stops the Real-Time Clock (RTC) and the Watchdog Timer (WDT). • Sets Status Register (S) bits 1 and 2 (ring number); clears bits 10 through 15 (interrupt priority level). • Creates trivial map in MMU, if present. • Starts the Quality Logic Test (QLT) in each controller. • Sets CPU address mode (LAF/SAF) from preselected switch setting.

Table 2-1 Full Control Panel Switches and Indicators
(Sheet 2 of 6)

SWITCH/INDICATOR	FUNCTION
CLR (Master Clear) Pushbutton (cont'd)	• Clears memory address (A0) and data (B0) registers.
Rocker Arm Switch Assembly	Located on the underside (component side) of the logic board and controls the following logical functions: • If switch 1 is enabled (On) and power fails, the CPU performs an auto boot operation (i.e., memory is volatile). If switch 1 is disabled and power fails, the CPU performs an auto restart operation (i.e., memory is nonvolatile). • Switch 3 is set to the enabled position when used with the full control panel (i.e., the full control panel drives its own LAF indicator). Switch 3 is set to the disabled position when used with the portable panel (i.e., the basic panel drives the portable panel LAF indicator). • If switch 4 is enabled, depressing the CLEAR pushbutton places the CPU in Long Address Form (LAF). If switch 4 is disabled, depressing the CLEAR pushbutton places the CPU in Short Address Form (SAF).
L (Bootload) Pushbutton	Causes the control panel to enter the Load mode. When the Execute (E) pushbutton is depressed, the QLT routine is entered. After successful completion of the QLT, the TRAFFIC indicator extinguishes; the P Register contains 00002. Depressing the Execute (E) pushbutton at this time initiates the Bootload operation. The default load device is automatically chosen by the firmware unless an alternate input device is selected (prior to the second depression of Execute (E) pushbutton) by inserting its channel number into Data Register 1 (D1). To select D1, refer to Table 2-2.
S (Select) Pushbutton	Causes the control panel to enter the Select mode, thereby enabling register selection via the 16 key hexadecimal entry pad:

Table 2-1 Full Control Panel Switches and Indicators
(Sheet 3 of 6)

SWITCH/INDICATOR	FUNCTION
S (Select) Pushbutton (Cont'd)	• H1 displays keys 8 through F • H2 displays keys 0 through 7 • H4 through H8 display contents of register selected by H1-H2. The Select mode may be initiated in any state. NOTE Selection of any D (0 through 7) register causes H4 to extinguish; thus, 16 bits are displayed. When register B0 is selected, H4 is illuminated, but its value may usually be ignored.
1 (Plus One) Pushbutton	Causes the control panel to enter Plus One mode. Enables incrementing of the address register before reading or writing successive memory locations from the control panel if the control panel is in either Read or Write mode. Each depression of the Execute (E) pushbutton while in these modes causes the memory address register to be incremented by 1 prior to its use.
0 (Plus Zero) Pushbutton	Causes the control panel to exit Plus One mode; consequently, the memory address register is not incremented during the memory read or write operations.
W (Write) Pushbutton	Clears Plus One mode and places the control panel in Write mode. Clears Load mode. When the Execute (E) pushbutton is depressed in Write mode, the contents of the selected register are written into the Memory Address Register (A0).
R (Read) Pushbutton	Clears Plus One mode and places the control panel in Read mode. When the Execute (E) pushbutton is depressed, the contents of the memory location addressed by the Memory Address Register (A0) are read into the selected register. If in Load mode when the Execute (E) pushbutton is depressed in Read mode, the contents of the Bootload location addressed by the Memory Address

Table 2-1 Full Control Panel Switches and Indicators
(Sheet 4 of 6)

SWITCH/INDICATOR	FUNCTION
R (Read) Pushbutton (Cont'd)	Register (A0) are read into the selected register.
S (Stop) Pushbutton	Causes the control panel to enter Stop mode and halts instruction execution. Each depression of the Execute (E) pushbutton causes the CPU to execute one instruction.
R (Ready) Pushbutton	Causes the control panel to enter Ready mode. If the Execute (E) pushbutton is subsequently depressed, the control panel enters Run mode (RUN and TRAFFIC indicators illuminate) and program execution begins.
C (Change) Pushbutton	Causes the control panel to enter Change mode, thereby enabling entry of one to five hexadecimal digits into a selected register. Any hexadecimal key that is depressed is shifted into display position H8 and the corresponding position of the selected register. At the same time, H8 is shifted into H7, H7 into H6, H6 into H5, and H5 into H4; must not be in Run mode.
E (Execute) Pushbutton	Initiates control panel operation appropriate to the current mode: • If in Ready mode, depressing the Execute (E) pushbutton places the control panel in Run mode, executing instructions that start with the one in the instruction register and continue at the location specified by the program counter. Execution continues until a Stop (S), Read (R), or Write (W) pushbutton is depressed. • If in Step mode, depressing the Execute (E) pushbutton causes execution of one instruction; the control panel remains in Step mode. • If in Read or Write mode, depressing the Execute (E) pushbutton displays or changes the contents of the memory location selected by A0 (see also Plus One).

Table 2-1 Full Control Panel Switches and Indicators
(Sheet 5 of 6)

SWITCH/INDICATOR	FUNCTION
E (Execute) Pushbutton (Cont'd)	• If in Load mode, and in Ready or Stop mode, depressing the Execute (E) pushbutton initiates the QLT operation. For further details concerning the Execute (E) pushbutton, refer to the L, R, S, R, and W pushbutton descriptions.
H1, H2 Hexadecimal Display	Displays selected register number: H1 displays register types 8-F. H2 displays register numbers 0-7. Table 2-2 shows the H1, H2 format.
H4, H5, H6, H7, H8 Hexadecimal Display	Displays the contents of the selected register. Display is four or five hexadecimal digits, depending on register type: H4 is the most significant digit. H8 is the least significant digit.
DC ON Indicator	Illuminates when operational DC power is available in the system.
CHECK Indicator	Illuminates when at least one bus element has not successfully completed its logic tests (QLT) or a bus element is not properly plugged into the bus.
TRAFFIC Indicator	Illuminates when the CPU is executing any instruction, excluding Halt.
RUN Indicator	Illuminates when the control panel is in Run mode (i.e., executing programs). If the TRAFFIC indicator extinguishes while the RUN indicator is illuminated, the CPU is executing a Halt instruction.
LOAD Indicator	Illuminates when the control panel is in Load mode; extinguishes when the Load operation is successfully completed.

Table 2-1 Full Control Panel Switches and Indicators
(Sheet 6 of 6)

SWITCH/INDICATOR	FUNCTION
CHANGE Indicator	Illuminates when the control panel is in Change mode. In this mode, the contents of the selected register can be modified by key-in data from the hexadecimal pad keys (except in Run mode).
PLUS Indicator	Illuminates when the Plus One (1) key has been depressed. When PLUS is illuminated, sequential memory locations can be read or written. The PLUS indicator extinguishes when the Plus Zero (0), Read (R), or Write (W) key is depressed.
WRITE Indicator	Illuminates when the control panel is in Write mode. When WRITE is illuminated, data can be written into memory from the control panel.
READ Indicator	Illuminates when the control panel is in Read mode. When READ is illuminated, data can be read from memory via the control panel.
STOP/STEP Indicator	Illuminates when the control panel is in Step mode. One instruction is executed with each depression of the Execute (E) pushbutton.
READY Indicator	Illuminated when the Ready (R) key is depressed, placing the control panel in Ready mode. If the Execute (E) pushbutton is depressed, the control panel enters Run mode.
LAF Indicator	Illuminates when the CPU is set to execute instructions in LAF (long address form). In this mode, pointers in memory are 32 bits in length. Extinguishes when the CPU is set to execute instructions in SAF (short address form). In this mode, pointers in memory are 16 bits in length. NOTE The CPU QLT routine always executes in LAF mode and with the LAF indicator ON, regardless of the LAF/SAF switch setting.

Table 2-2 Control Panel Register Selection
(Sheet 1 of 2)

LOCATION CODE		SELECTED REGISTER CONTENTS					
H1	H2	H3	H4	H5	H6	H7	H8
8	0**	R F U	0	0	0	CIP Indicators	
8	1**		0	0	0	SIP Indicators	
8	2**		0	0	0	CIP Indicators	
8	3**		0	0	0	SIP Indicators	
8	4**		0	SIP Accumulator 1, Word 0			
8	5**		0	SIP Accumulator 1, Word 1			
8	6**		0	SIP Accumulator 1, Word 2			
8	7**		0	SIP Accumulator 1, Word 3			
9	0**		0	SIP Accumulator 2, Word 0			
9	1**		0	SIP Accumulator 2, Word 1			
9	2**		0	SIP Accumulator 2, Word 2			
9	3**		0	SIP Accumulator 2, Word 3			
9	4**		0	SIP Accumulator 3, Word 0			
9	5**		0	SIP Accumulator 3, Word 1			
9	6**		0	SIP Accumulator 3, Word 2			
9	7**		0	SIP Accumulator 3, Word 3			
A	0		Memory Address (Y)				
A	1**		Physical Address equivalent to A0				
A	2**		Stack Pointer				
A	3**		Remote Descriptor Base				
A	4**		Latest CIP Instruction Address				
A	5**		Latest SIP Instruction Address				
A	6**		Work Location				
A	7**		Work Location				
B	0		Memory Data				
B	1		Base (B1)				
B	2		Base (B2)				
B	3		Base (B3)				
B	4		Base (B4)				
B	5		Base (B5)				
B	6		Base (B6)				
B	7		Base (B7)				
C	0**		0	Status (S)			
C	1**		0	Indicators (I)		Mode (M1)	
C	2**		0	(RFU)		Mode (M2)	
C	3**		0	Indicators (I)		Mode (M3)	

Table 2-2 Control Panel Register Selection
(Sheet 2 of 2)

LOCATION CODE		SELECTED REGISTER CONTENTS			
H1	H2	H3	H4	H5 H6	H7 H8
C	4**	R F U	0	Real Time Clock (RTC)*	Mode (M4)
C	5**		0	(RFU)	Mode (M5)
C	6**		0	Watch Dog Timer (WDT)*	Mode (M6)
C	7**		0	Multi NATSAP / RFU	Mode (M7)
D	0		***	Instruction (D0)	
D	1			Data (D1)	
D	2			Data (D2)	
D	3			Data (D3)	
D	4			Data (D4)	
D	5			Data (D5)	
D	6			Data (D6)	
D	7			Data (D7)	
E	0			Program Counter (P)	

*WDT/RTC value (displayed in HEX indicators 4 and 5) equal 00_{16} if disabled and equal FF_{16} if enabled.
**Read (Display) only.
***D register selection causes HEX indicator 4 to extinguish.

2.1.9.9.2 Basic Control Panel

The operation of the basic control panel and its system display capabilities are confined to two switchable control functions and five system status indicators (as shown in Figure 2-3). A brief functional description of each switch and indicator is provided in Table 2-3. The basic control panel is designed primarily to provide an operator with the minimum number of controls required to activate a system. However, a portable full panel assembly is available for expansion to a full panel status.

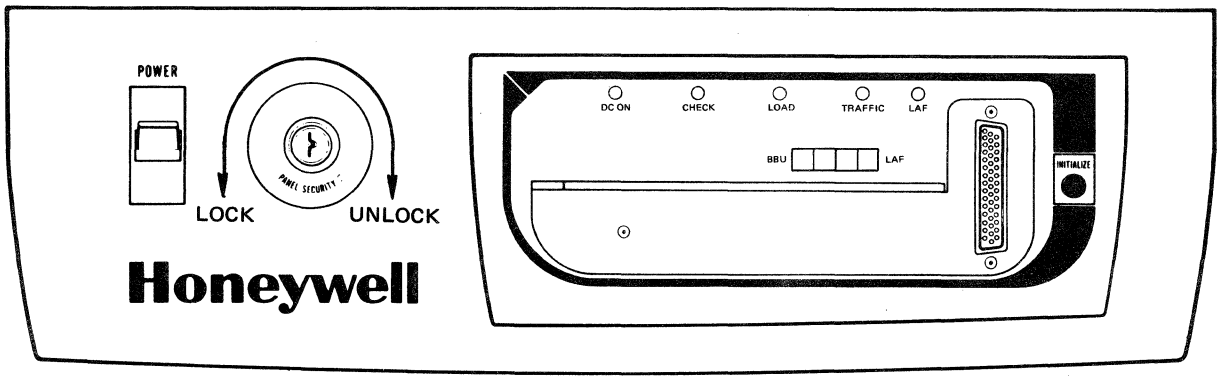


Figure 2-3 Basic Control Panel

Table 2-3 Basic Control Panel Switches and Indicators (Sheet 1 of 3)

SWITCH/INDICATOR	FUNCTION
POWER Switch	Two-position switch used for engaging/ disengaging the system power. Up position turns the power off. When the power is switched on, Master Clear is automatically activated and the DC ON indicator illuminates after DC power is attained.
PANEL SECURITY Switch	Two-position keylock switch used for enabling/disabling the control panel. Counterclockwise (locked) position disables the INITIALIZE switch and the interface to full control panel.
INITIALIZE Pushbutton	Depressing the INITIALIZE pushbutton: • Clears the Program Counter (E0), the M registers, and the Instruction Register (D0). • Clears all pending interrupts and traps. • Stops the Real-Time Clock (RTC) and the Watchdog Timer (WDT). • Sets Status Register (S) bits 1 and 2 (ring number); clears bits 10 through 15 (interrupt priority level). • Creates trivial map in MMU, if present. • Starts the Quality Logic Test (QLT) in each controller. • Sets CPU address mode (LAF/SAF) from preselected switch setting.

Table 2-3 Basic Control Panel Switches and Indicators
(Sheet 2 of 3)

SWITCH/INDICATOR	FUNCTION
INITIALIZE Pushbutton (Cont'd)	• Clears memory address (A0) and data (B0) registers. • If memory is volatile, runs the CPU QLT, followed by a memory test on the first 8K of memory (checks memory and zeros data), then executes an auto boot. • If memory is non-volatile, runs the CPU QLT, then starts program execution with the instruction beginning at main memory location 00000.
Rocker Arm Switch Assembly	Controls the following logical functions: • If switch 1 is enabled (On) when power is applied, the CPU performs an auto boot operation (i.e., memory is volatile). If switch 1 is disabled when power is applied, the CPU performs an auto restart operation (i.e., memory is non-volatile). • If switch 4 is enabled, depressing the INITIALIZE pushbutton places the CPU in Long Address Form (LAF). If switch 4 is disabled, depressing the INITIALIZE pushbutton places the CPU in Short Address Form (SAF).
DC ON Indicator	Illuminates when operational DC power is available in the system.
CHECK Indicator	Illuminates when at least one bus element has not successfully completed its logic test (QLT) or a bus element is not properly plugged into the bus.
TRAFFIC Indicator	Illuminates when the CPU is executing any instruction, excluding Halt.
LOAD Indicator	Illuminates when the CPU is in Load mode; extinguishes when the Load operation is successfully completed.

Table 2-3 Basic Control Panel Switches and Indicators (Sheet 3 of 3)

SWITCH/INDICATOR	FUNCTION
LAF Indicator	Illuminates when the CPU is set to execute instructions in LAF (Long Address Form). In this mode, pointers in memory are 32 bits in length. Extinguishes when the CPU is set to execute instructions in SAF (Short Address Form). In this mode, pointers in memory are 16 bits in length. NOTE The CPU QLT routine always executes in LAF mode and with the LAF indicator On, regardless of the LAF/SAF switch setting.

2.1.9.9.3 Portable Control Panel

The portable control panel, a self-contained full panel, is mounted on the basic panel and connects to the system through the 50-pin connector on the basic panel. Functionally, the portable panel is similar in operation to the full panel with two exceptions. Rocker arm switch assembly 1 is set to the auto restart position (memory is non-volatile), and rocker arm switch assembly 3 is set to the SAF position (Off), allowing the basic panel to drive the LAF indicator on the portable panel.

NOTE

With the basic panel PANEL SECURITY switch unlocked the portable panel functions exactly as if a full panel was configured. This includes execution of the CPU QLT for all available memory rather than just the first 8K which would be accomplished when only the basic panel is installed in a system.

2.2 MEGABUS OPERATIONS

The Megabus (see Figure 2-4) provides a common communication path (interface) among all units of the Level 6 system. The Megabus is asynchronous in design, permitting units of varying speeds to operate efficiently on the same system. Five types of communication are permitted over the Megabus: (1) memory read requests, (2) non-memory read requests, (3) read responses, (4) memory write requests, and (5) non-memory write requests, including interrupts.

The Megabus can accommodate a maximum of 23 units/controllers. The number of I/O devices supported by a single Megabus may be greater than 23 because several I/O devices may be connected through a single controller. For larger systems,

several Megabuses may be interconnected by using the Intersystem Links (ISLs).

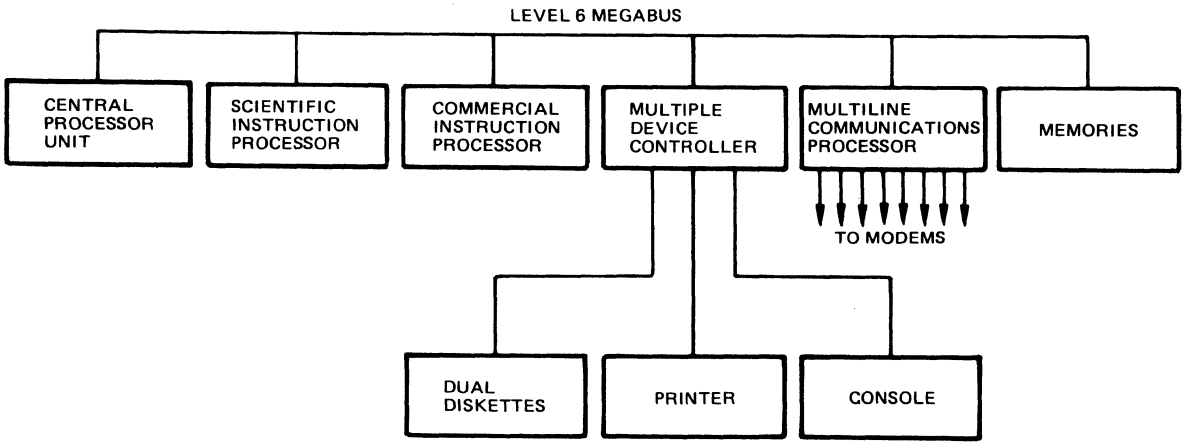


Figure 2-4 Typical Level 6 System

2.2.1 Master/Slave Relationship

The Megabus is bidirectional, thereby permitting any two units/controllers to communicate with each other at a given time. The transfer of information between units/controllers forms a temporary master/slave relationship (i.e., the unit/controller requesting and receiving access to the bus becomes the master unit while the unit/controller being addressed by the master unit becomes the slave unit). If the communication requires a reponse, the responding slave unit/controller assumes the role of master unit and the requesting unit/controller (previous master) becomes the slave unit.

All information transfers are from master unit to slave unit and each transfer is referred to as a bus cycle. This cycle is comprised of the following: (1) the requestor (master unit) asks for use of the bus, (2) if no other unit/controller of a higher priority is making a bus request, use of the bus is granted to the requestor (master unit), and (3) the master unit then transmits its information to the slave unit and the slave unit acknowledges or refuses the communication.

Communication between a master unit and slave unit requires a response from the slave unit when the master unit is requesting data (e.g., a memory read command). In this case, the request for information requires one bus cycle and the transmission of information back to the requestor requires an additional bus cycle to complete the task.

2.2.2 Megabus Usage

Common types of Megabus operations are listed in Table 2-4. These various operations require either one, two, or three bus cycles. Information transfers that are considered write operations require one bus cycle while transfers that are considered read operations require an additional bus cycle for the response.

NOTE

Once a bus cycle is granted, the types of operations performed between the master unit and the slave unit are a function of the specific functionality of the two units/controllers.

Other types of Megabus operations, such as controller-to-controller transfers, are not listed in Table 2-4; however, the bus architecture makes no restrictions in this regard.

Table 2-4 Common Types of Megabus Operations

TYPE OF OPERATION	ORIGINAL MASTER	ORIGINAL SLAVE	NUMBER OF BUS CYCLES
Instruction Fetch (one word)	CPU	Memory	2
Instruction Fetch (two words)	CPU	Memory	3
Operand Fetch (one word)	CPU	Memory	2
Operand Fetch (two words)	CPU	Memory	3
Operand Store (word)	CPU	Memory	1
Operand Store (byte)	CPU	Memory	1
DMA Read (word)	Controller	Memory	2
DMA Read (byte)	Controller	Memory	2
DMA Write (word)	Controller	Memory	1
DMA Write (byte)	Controller	Memory	1
I/O Output Command Word	CPU	Controller	1
I/O Input Command Word	CPU	Controller	2
Interrupt	Controller	CPU	1

2.3 MEGABUS FORMATS

Figure 2-5 depicts the information format for the data and address lines during various Megabus cycle operations. Each format shown reflects the occurrence of a single Megabus cycle. The following subsections provide a description of the formatting associated with the address and data lines.

OPERATION	NO. OF CYCLES	MASTER	SLAVE	ADDRESS LINES (BSAD)	DATA LINES (BSDT)
MEMORY READ REQUEST	1	CPU+CU	MEM	0–23: BYTE ADDRESS	0–9: MASTER CHANNEL NUMBER; 10–15: VARIABLE USAGE
I/O READ REQUEST	1	CPU	CU	0–7: (shaded); 8–17: SLAVE CHANNEL NUMBER; 18–23: FUNCTION CODE	0–9: MASTER CHANNEL NUMBER; 10–15: VARIABLE USAGE
READ RESPONSE	1	MEM+CU	CPU+CU	0–7: (shaded); 8–17: SLAVE CHANNEL NUMBER; 18–23: VARIABLE USAGE	0–15: DATA
MEMORY WRITE	1	CPU+CU	MEM	0–23: BYTE ADDRESS	0–7: DATA; 8–15: DATA
I/O DATA OUTPUT	1	CPU	CU	0–7: (shaded); 8–17: SLAVE CHANNEL NUMBER; 18–23: FUNCTION CODE	0–15: DATA
I/O ADDRESS OUTPUT	1	CPU	CU	0–7: (blank); 8–17: SLAVE CHANNEL NUMBER; 18–23: FUNCTION CODE	0–15: BYTE ADDRESS
INTERRUPT	1	CU	CPU	0–7: (shaded); 8–17: SLAVE CHANNEL NUMBER; 18–23: MBZ	0–9: MASTER CHANNEL NUMBER; 10–15: INTERRUPTING LEVEL

Figure 2-5 Megabus Formats

2.3.1 Channel Numbers

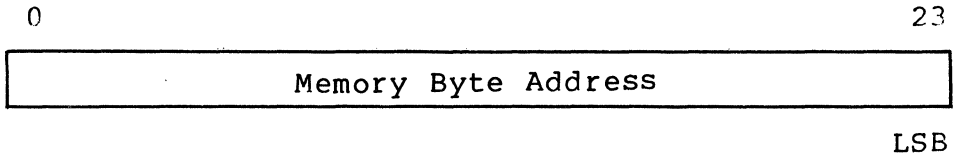
A channel number will exist for every end point in a particular system except for memory, which is identified only by a memory address.

The channel number of the slave unit will appear on the address bus for all non-memory transfers. Each unit will compare that number with its own internally stored number. The unit which achieves a comparison is, by definition, the slave and must respond to that cycle. No two end points on a single Megabus will be assigned the same channel number.

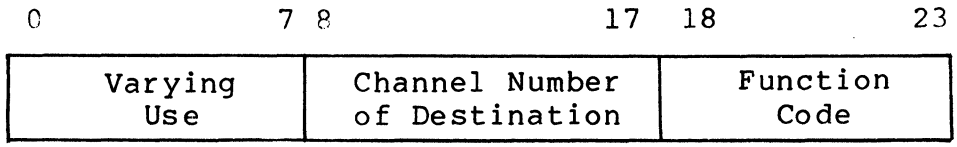
Processor channel numbers are restricted to the range of 0000 through $03C0_{16}$. The six upper bits of the channel number are fixed as Zeros by the processor logic and only the lower two bits are variable via a DIP switch. Processor channel numbers are not used by any other units.

2.3.2 Unit Addressing

A master unit may address any other unit on the bus as the slave unit. It does this by placing the slave address on the address lines. There are 24 address lines, which can have either of two interpretations, depending on the state of the Memory Reference signal. If Memory Reference is true, the following format applies to the address lines:

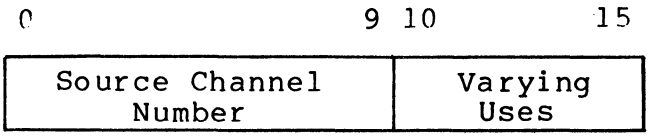


If Memory Reference is false, the following format applies to the address lines:



When units are passing control information, data, or interupts, they address each other by channel number. Along with the channel number, a 6-bit function code is passed that specifies which function this transfer implies.

When a master unit requires a response from the slave unit, it indicates this to the slave unit by forcing the Bus Write signal false. In addition, the master unit provides its own identity to the slave unit by means of a channel number. This is coded on the data lines of the bus as follows:



The response cycle is directed to the register (master unit) by a non-memory reference transfer. The Second Half Bus Cycle accompanies the transfer to indicate that this is the awaited cycle.

2.3.3 Memory Read (Word)

During a read operation, a 16-bit data word is requested from memory. The CPU accepts the complete data word into one of the bus input buffers, from where it is routed (under firmware control) to the internal bus for use, as appropriate.

2.3.4 Memory Write (Word or Byte)

If during a write operation the byte control line is false, a 16-bit data word is written into main memory. However, if the byte control signal is true, the memory unit is alerted to enable an 8-bit write mask for either the left or right byte, depending on the state of address bit 23. If bit 23 is true, the right byte is written into memory as new data; if bit 23 is false, the left byte is written into memory.

2.4 CPU FIRMWARE OVERVIEW

Figure 2-6 is an overview flow chart of the 15 major firmware subdivisions:

- Initialize
- Instruction Fetch (XF)
- WDT/RTC Servicing (TIX)
- Control Panel Service
- Address Formation (XA)
- Branch Instructions (XA)
- Generic Instructions (XA)
- Address Formation (XB)
- Indirect and/or Indexing (XR)
- Operand Read (XR)
- Instruction Execution (XE)

- Result Write (XW)
- Trap
- Interrupt
- Quality Logic Test (QLT).

It must be understood that the above subdivisions are only for convenience of discussion. Most firmware routines are parts of several subdivisions.

2.4.1 Initialize

The initialize fimrware is entered by Master Clear and performs the following:

- Clears P and D0
- Clears various control flops
- Clears RAM
- Loads a trivial map into the Memory Management Unit
- Determines whether a SIP and/or CIP is present.

If the control panel is locked, the Instruction Fetch firmware is entered; otherwise, the Control Panel Service firmware is entered.

2.4.2 Instruction Fetch (XF)

The XF firmware obtains the first word of the next instruction for execution. Once the instruction is received from memory, it is loaded into the F register. Copies of the instruction are written into D0 and into RAM0. During XF, checks are performed for the following conditions:

- Device interrupt
- External processor trap
- WDT/RTC/Control panel service.

2.4.3 WDT/RTC Servicing (TIX)

Every 8-1/3 milliseconds (independent of line frequency), the watchdog timer and real-time clock are updated, if enabled. Once these tasks are performed, the Control Panel Service firmware is entered.

2.4.4 Control Panel Service

Every 8-1/3 milliseconds the control panel interface is interrogated to determine whether the operator wishes a new display or wishes to stop program execution. The display is then updated.

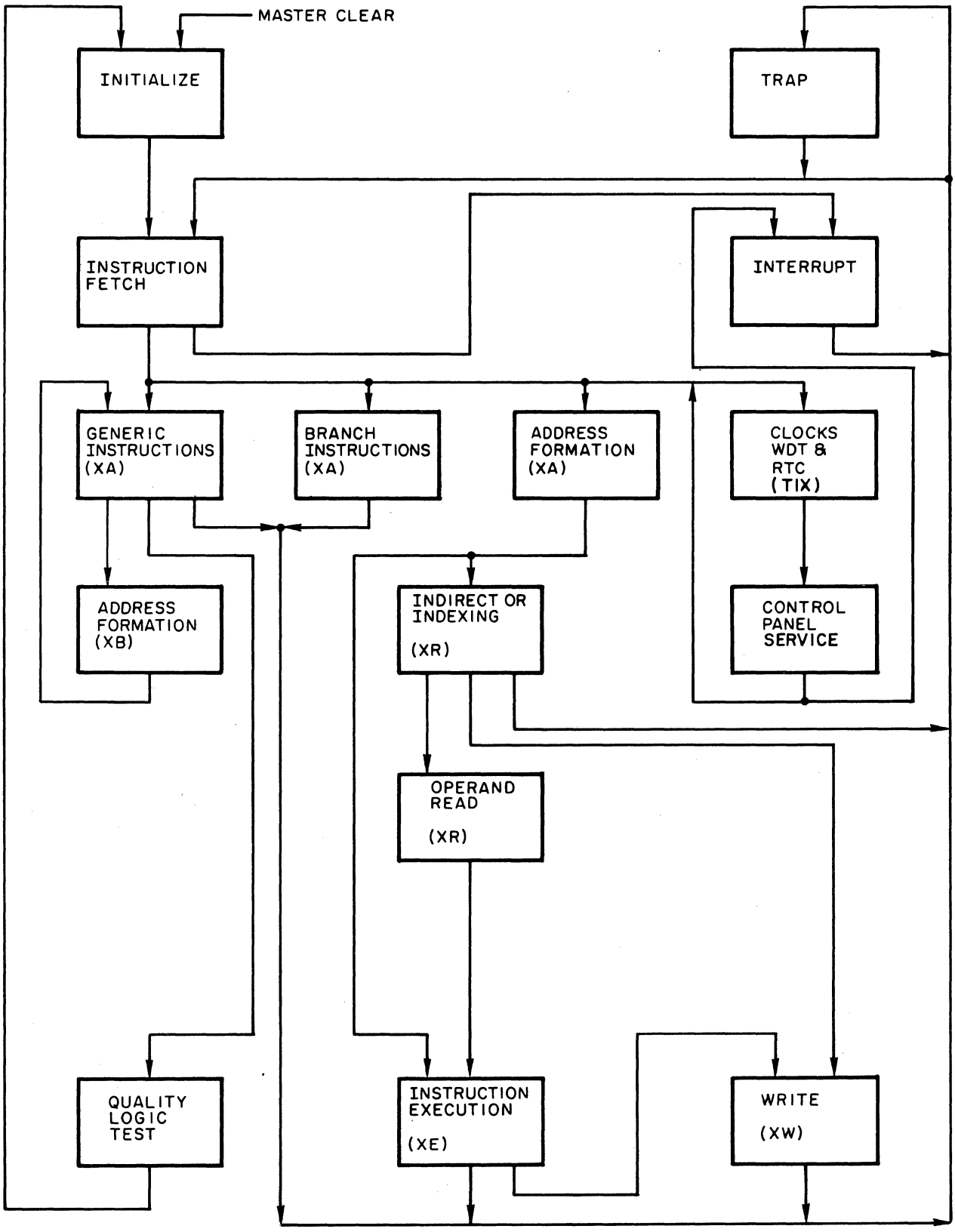


Figure 2-6 CPU Firmware Overview Flow Chart

2.4.5 Address Formation (XA)

The XA firmware examines the instruction, subdividing it into categories and subcategories:

- Category 1: Standard 7-bit address syllable (subcategories: P + D/B + D, B/@B, etc.).
- Category 2: Branches (subcategories: long displacement, short displacement, etc.).
- Category 3: Generics (subcategories: HLT, RTCN, MCL, MMM, etc.).
- Category 4: Shifts (subcategories: SOL, DCR, etc.).
- Category 5: Short-value Immediates (subcategories: +, -).

For instructions using the 7-bit address syllable, XA starts effective address generation. If the address syllable calls for a register or if the operand is immediate, the XA firmware exits to XE; otherwise, the XA firmware exits to XR.

2.4.6 CIP Address Formation (XB)

The XB firmware is similar to XA, but interprets a different address syllable format for commercial data descriptors.

2.4.7 Indirect/Indexing/Operand Fetch (XR)

The XR firmware performs indirection and indexing if needed. It then fetches an operand if the instruction requires it. Some instructions, such as jumps, are executed within XR (i.e., without further major branching). For others, XR exits to XE.

2.4.8 Instruction Execution (XE)

The XE firmware selects one of several op-code dependent entry points where instruction execution begins.

2.4.9 Result Write (XW)

The XW firmware stores the result, if necessary, after the instruction is executed. This firmware is entered only by instructions that must return their results to a palce specified by an address syllable.

2.4.10 Trap and Interrupt

Traps are distinguished from interrupts by being synchronous with and caused by the program currently being executed. On the other hand, interrupts are generally related to the program currently being executed, or at best are asychronous with it.

2.4.10.1 Traps

When processing traps, the processor generates addresses to dedicated memory locations (see Figures 2-7 and 2-8) that contain the Next Available Trap Save Area Pointer (NATSAP) and the trap vectors (i.e., pointers to the trap handler procedures). The various events that can cause a trap and their associated vector numbers are listed in Table 2-5. For a detailed description of the trap firmware refer to subsection 3.2.

The Trap firmware is enterd from any one of many locations within the firmware when a trap condition is detected.

2.4.10.2 Interrupts

The CPU interrupt hardware consists of two sections: (1) the interrupt busy flip-flop together with the logic for comparing the CPU level to the incoming interrupt level, and (2) the interrupt data register, which stores an acknowledged interrupt so that the processor can service it. Interrupts are classified by their source as follows:

1. Externally generated by the following events:

    a. Peripheral device completed an assigned activity
    b. Peripheral device changed state
    c. Power failure
    d. Dialog from other processors.

2. Internally generatged by the following events:

    a. Execution of a suspend, dispatch, or quick level change instruction
    b. Watchdog timer runout
    c. Real-time clock runout
    d. Trap save area pool exhausted.

For a detailed description of interrupts, see subsection 3.3.

The interrupt firmware is enterd from a number fo sources, depending on the operation currently being performed by the CPU. These interrupt sources are as follows:

- Use of last trap save area
- Programmed interrupt (LEV)
- External device task completion
- RTC runout
- WDT runout
- Incipient power failure.

2.4.11 Quality Logic Test (QLT)

The QLT provides a basic confidence test of the CPU logic and the memory. The QLT is entered via the MCL instruction when the control panel is in Load mode.

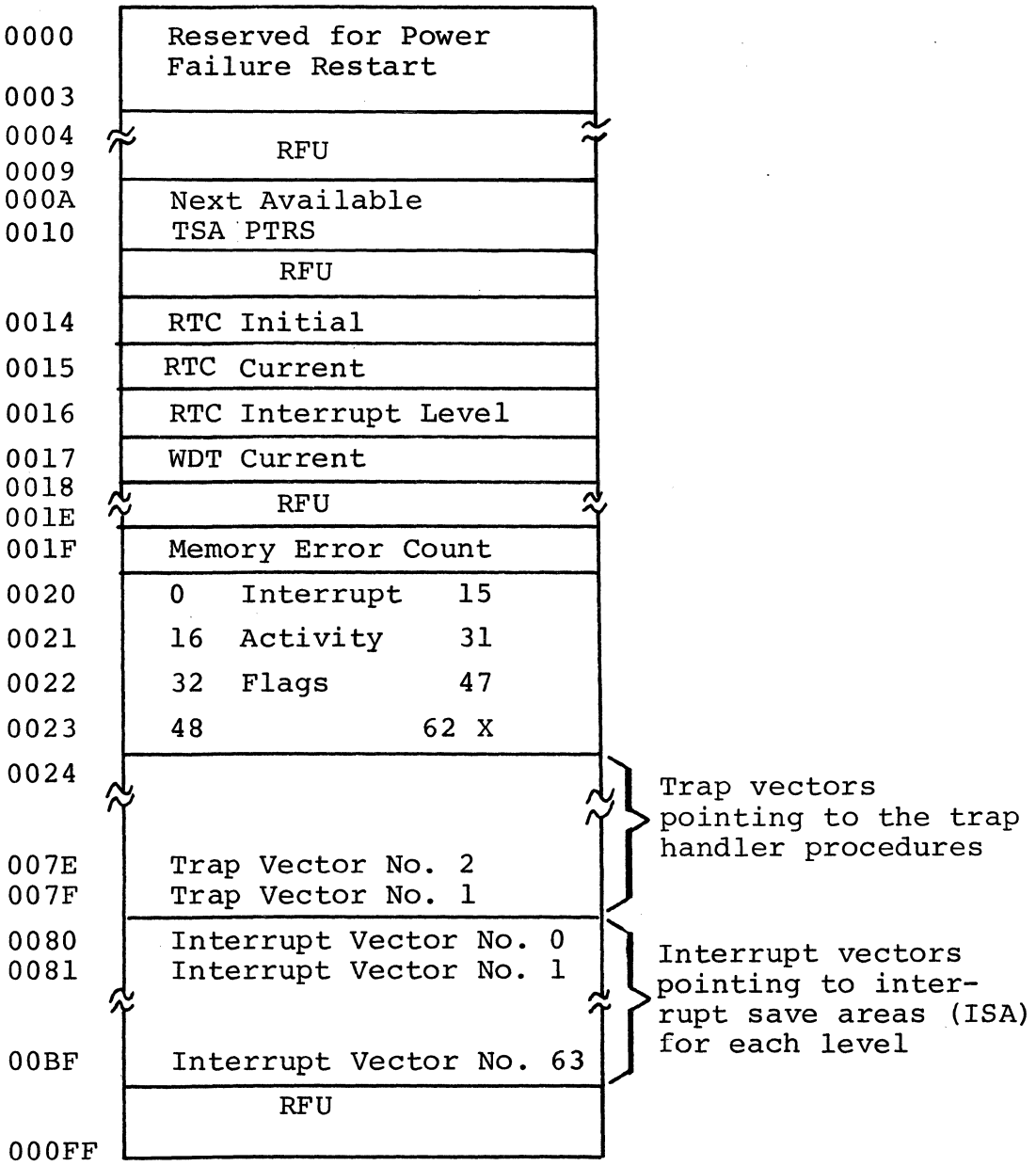


Figure 2-7 Dedicated Memory Locations in SAF

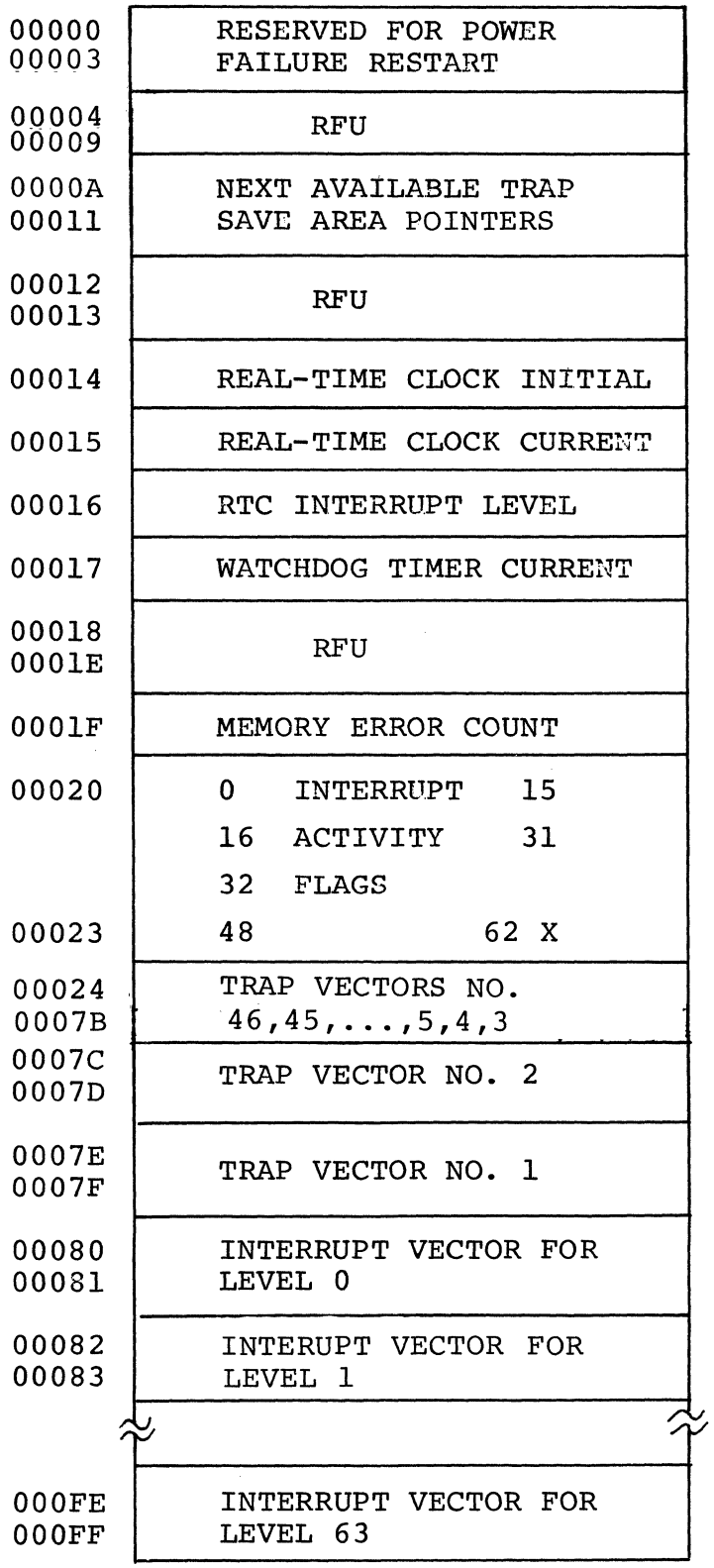


Figure 2-8 Dedicated Memory Locations in LAF

Table 2-5 Trap Event, Trap Vectors, and Halt Locations

TV NUMBER	TV		EVENT NAME	P*	
	SAF	LAF		SAF	LAF
1	7F	7E	Monitor Call	80	80
2	7E	7C	Trace/Breakpoint**	7F	7E
3	7D	7A	Unimplemented SIP Operation	7E	7C
4	7C	78	Reserved for Software Use	7D	7A
5	7B	76	Unimplemented Non-SIP Operation	7C	78
6	7A	74	Integer Register Overflow**	7B	76
7	79	72	Scientific Divide by Zero	7A	74
8	78	70	Scientific Exponent Overflow	79	72
9	77	6E	Stack Underflow	78	70
10	76	6C	Stack Overflow	77	6E
11	75	6A	-	76	6C
12	74	68	Illegal Remote Data Descriptor	75	6A
13	73	66	Privilege Violation	74	68
14	72	64	Protection Violation	73	66
15	71	62	Unavailable Resource	72	64
16	70	60	Program Error	71	62
17	6F	5E	Uncorrectable Memory Error	70	60
18	6E	5C	-	6F	5E
19	6D	5A	Scientific Exponent Underflow**	6E	5C
20	6C	58	Scientific Program Error	6D	5A
21	6B	56	Scientific Significance Error**	6C	58
22	6A	54	Scientific Precision Error**	6B	56
23	69	52	External Processor Unavailable Resource	6A	54
24	68	50	External Processor Uncorrectable Memory Error	69	52
25	67	4E	-	68	50
26	66	4C	-	67	4E
27	65	4A	-	66	4C
28	64	48	-	65	4A
29	63	46	-	64	48
30	62	44	CIP Failed QLT	63	46
31	61	42	SIP Failed QLT	62	44
32	60	40	-	61	40
33	5F	3E	-	60	40

*The CPU halts here if NATSAP is not null, but the TV is null.

**Mask controlled trap.

III

FIRMWARE OPERATION

This section provides a description of the Central Processor Unit (CPU) firmware, which consists of 2,048 64-bit words.

3.1 FIRMWARE WORD

The firmware word is illustrated in Figure 3-1. The firmware word is divided into 15 distinct fields: (1) LS, (2) RS, (3) DI, (4) AD, (5) AF, (6) AS, (7) CK, (8) BI, (9) SM, (10) BS, (11) GP, (12) TC, (13) BR, (14) C, and (15) NA. Each of these field controls a portion of the hardware (refer to subsections 3.1.1. through 3.1.15).

3.1.1 Left Select (LS) Field

The LS field consists of bits 1 through 3 of the firmware word. This field serves a dual purpose. Along with the Select Modify (SM) field, it provides a 4-bit address to both the RAM and the RALU left select inputs. The SM field is described in subsection 3.1.9. Since the LS field is only three bits long and since four bits are required to fully address either the RAM or the RALU, one bit must be created (see Table 3-1). The two-weight bit, which is not in control store, is created from the presence of either the four-weight or one-weight bits.

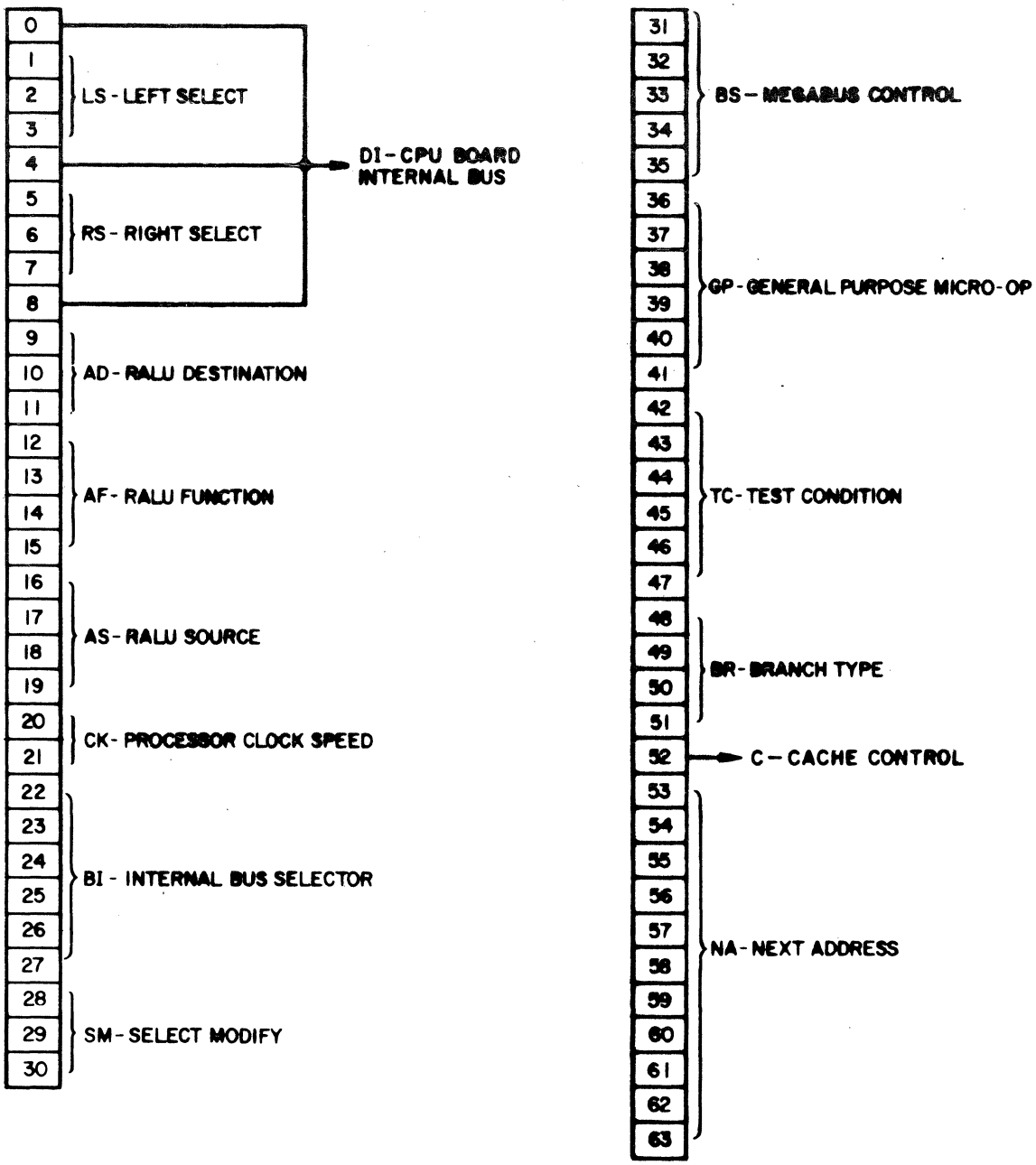


Figure 3-1 Firmware Word Format

Table 3-1 Left Select Codes

LS CODE	GENERATED ADDRESS 8	4	2	1	FIRMWARE NAME	REGISTER SELECTED
0	0	0	0	0	LSD0	D0
1	0	0	1	1	LSD3	D3
2	0	1	1	0	LSD6	D6
3	0	1	1	1	LSD7	D7
4	1	0	0	0	LSB0	B0
5	1	0	1	1	LSB3	B3
6	1	1	1	0	LSB6	B6
7	1	1	1	1	LSB7	B7

3.1.2 Right Select (RS) Field

The RS field consists of bits 5 through 7 of the firmware word. Only three bits reside in control store, thereby requiring that the two-weight bit again be created as in the LS field. The RS field provides a 4-bit address to the RALU right select inputs to transfer an operand to the right output of the reigster file. If data are to be written into the register file, RS selects the location into which the new data are to be loaded. As in the LS field, SM is utilized in determining the address delivered to the RALU right select inputs (see Table 3-2).

Table 3-2 Right Select Codes

RS CODE	GENERATED ADDRESS 8	4	2	1	FIRMWARE NAME	REGISTER SELECTED
0	0	0	0	0	RSD0	D0
1	0	0	1	1	RSD3	D3
2	0	1	1	0	RSD6	D6
3	0	1	1	1	RSD7	D7
4	1	0	0	0	RSB0	B0
5	1	0	1	1	RSB3	B3
6	1	1	1	0	RSB6	B6
7	1	1	1	1	RSB7	B7

3.1.3 Central Processor Board, Internal Bus (DI) Field

This 3-bit field (bits 0, 4, and 8) determines: (1) which portion (if any) of the RALU will be placed onto the internal bus, (2) whether the RAM will be transferred to the internal bus, and (3) whether the data currently on the internal bus will be written into the RAM (see Table 3-3).

Table 3-3 Central Processor Board Internal Bus Codes

DI CODE	FIRMWARE NAME	OPERATION PERFORMED
0	BIB*	Transfers RALU bits 0C through 0F to internal bus bits 0C through 0F.
1	DIA	Transfers RALU bits 0C through 1F to internal bus bits 0C through 1F.
2	DIC	Loads selected RAM location bits 0C through 1F with output of RALU bits 0C through 1F via the internal bus.
3	-	RFU
4	DIN	Null (neither RAM nor RALU is sent to the internal bus and the RAM is not written).
5	BIR*	Transfers RALU bits 18 through 1F to internal bus bits 18 through 1F.
6	DIW	Loads selected RAM location (bits 0C through 1F) with contents of internal bus (bits 0C through 1F).
7	DIR	Transfers RAM location contents (bits 0C through 1F) to internal bus (bits 0C through 1F).

*Refer to Table 3-10.

3.1.4 RALU Destination (AD) Field

This 3-bit field (bits 9 through 11) determines whether the RALU output will be shifted right, left, or not at all. The AD field also controls whether this value will be written into the Q register or the register file or neither, and whether the register file left output or the ALU output is made available to the internal bus (see Table 3-4).

3.1.5 RALU Function (AF) Field

This 4-bit field (bits 12 through 15) controls the type of operation that will be performed on the two operands, J and K. Fifteen different functions (see Table 3-5) may be performed (e.g., ADD, OR, AND, etc.). The most significant bit of this field controls the ALU input carry.

Table 3-4 RALU Destination Codes

AD CODE	FIRMWARE NAME	OPERATION PERFORMED
0	ADFQ	Sends the output of the ALU to the output multiplexer and writes the same value into the Q register.
1	ADFN	Sends the output of the ALU to the output multiplexer.
2	ADLR	Sends the contents of the left latch to the output multiplexer and writes the output of the ALU into the register file.
3	ADFR	Sends the output of the ALU to the output multiplexer and writes the same value into the register file.
4	ADDR	Sends the output of the ALU to the output multiplexer; performs a 32-bit shift right of the ALU output and the Q register contents, writing the shifted results to the register file and to the Q register.
5	ADSR	Sends the output of ALU to the output multiplexer; performs a 16-bit shift right of the ALU output, writing the shifted result to the register file.
6	ADDL	Sends the output of the ALU to the output multiplexer; performs a 32-bit shift left of the ALU output and the Q register contents, writing the shifted results to the register file and to the Q register.
7	ADSL	Sends the output of the ALU to the output multiplexer; performs a 16-bit shift left of the ALU output, writing the shifted result to the register file.

Table 3-5 RALU Function Codes

AF CODE	FIRMWARE NAME	OPERATION PERFORMED
0	AFADD	Adds input J and input K (J + K).
1	AFK-1	Performs a One's complement of J and adds it to K (K - J - 1).
2	AFJ-1	Performs a One's complement of K and adds it to J (J - K - 1).
3	AFIOR	ORs inputs J and K (J OR K).
4	-	Not used.
5	AFKNJ	Performs a One's complement of J and ANDs it with K (K AND $\overline{J}$).
6	AFXOR	Exclusive OR of J and K (J XOR K).
7	AFXNR	Performs a One's complement of exclusive OR of J and K.
8	AFINC	Adds input J and input K with an input carry (J + K + 1).
9	AFK-J	Subtracts J from K (K - J).
A	AFJ-K	Subtracts K from J (J - K).
B	AMIOR	ORs inputs J and K with input carry*.
C	AFAND	ANDs inputs J and K with input carry*.
D	AMKNJ	Performs a One's complement of J, then ANDs result to K with input carry*.
E	AMXOR	Exclusive OR of J and K with input carry*.
F	AMXNR	Performs a One's complement of exclusive OR of J and K with input carry*.

*An input carry during AND operations creates an output carry and overflow signal; during logic operations, in other non-arithmetic operations, it is used to control the Memory Management Unit (MMU).

3.1.6 RALU Source (AS) Field

This 4-bit field (bits 16 through 19) controls which pair of operands (internal bus, register file left output, register file right output, Q register, or Zero) will be designated as the J and K inputs to the ALU. The most significant bit of this field may alter one or both of the operands to become a 20-bit sign-extended value (see Tables 3-6 and 3-7).

Table 3-6 RALU Source Codes - AS Field

AS CODE	FIRMWARE NAME	OPERATION PERFORMED
0 8	AWLQ* ASLQ	Takes J from the register file left output and K from the Q register.
1 9	AWLR* ASLR	Takes J from the register file left output and K from the register file right output.
2 A	AWZQ* ASZQ	Takes J equal to Zero and K from the Q register.
3 B	AWZR* ASZR	Takes J equal to Zero and K from the register file right output.
4 C	AWZL* ASZL	Takes J equal to Zero and K from the register file left output.
5 D	AWIL* ASIL	Takes J from BI and K from the register file left oputput.
6 E	AWIQ* ASIQ	Takes J from BI and K from the Q register.
7 F	AWIZ* ASIZ	Takes J from BI and K equal to Zero.

*The ALU zero detector is extended to test 20 bits, and Carry Out is taken from bit 0C instead of bit 10.

Table 3-7 RALU Source Codes - AS/AF Field

AS/AF CODE	FIRMWARE NAME	OPERATION PERFORMED
1/6	AXXX*	Value in the ALU is equal to double the contents of the register file left output (sign-extended). Left Select (LS) must equal Right Select (RS).
2/0	AXLQ*	Value in the ALU is equal to the contents of the register left output (sign-extended) plus the contents of the Q register.
2/8	AXLP*	Value in the ALU is equal to the contents of the register file left output (sign-extended) plus the contents of the Q register plus 1.
3/0	AXIR*	Value in the ALU is equal to the contents of the register file left output (sign-extended) plus the contents of the register file right output.

*The ALU zero detector is extended to test 20 bits, and Carry Out is taken from bit 0C instead of bit 10.

3.1.7 Processor Clock (CK) Speed Control Field

The CK field (bits 20 and 21) permits the processor clock to operate at intervals of 160, 180, 200, or 340 nanoseconds. The duration of each firmware step is thereby determined by this field (see Table 3-8).

Table 3-8 Clock Speed Codes

CK CODE	FIRMWARE NAME	CLOCK SPEED, Nanoseconds
0	CKVL	340
1	CKHL	200
2	CKHF	180
3	CKVF	160

3.1.8 Internal Bus (BI) Selector Control Field

The BI codes are listed in Tables 3-9 through 3-11. This 6-bit field (bits 22 through 27) performs one of four functions:

1. It generates firmware constants (9-bit sign-extended).
2. It selects which register content is to be delivered to the internal bus.

3. It determines which signal(s) will be sampled by the indicator register.

4. It generates control words to communicate with external processors.

Table 3-9 Internal Bus Codes, Constants

BI CODE (BITS)						FIRMWARE NAME	BI ACTION LINES (HEX CHARACTER POSITIONS)				
22	23	24	25	26	27		0C-0F	10-13	14-17	18-1B	1C-1F
0	0	b	b	b	b	(K0yz#)	0	0	0	y	z
0	1	b	b	b	b	(KFyz#)	0	F	F	y	z
0	0	N	N	N	N	K0--	0	0	0	y	-
0	1	N	N	N	N	KF--	0	F	F	y	-
0	N	0	0	0	0	K--0	0	-	-	y	0
0	N	0	0	0	1	K--1	0	-	-	y	1
0	N	0	0	1	0	K--2	0	-	-	y	2
0	N	0	0	1	1	K--3	0	-	-	y	3
0	N	0	1	0	0	K--4	0	-	-	y	4
0	N	0	1	0	1	K--5	0	-	-	y	5
0	N	0	1	1	0	K--6	0	-	-	y	6
0	N	0	1	1	1	K--7	0	-	-	y	7
0	N	1	0	0	0	K--8	0	-	-	y	8
0	N	1	0	0	1	K--9	0	-	-	y	9
0	N	1	0	1	0	K--A	0	-	-	y	A
0	N	1	0	1	1	K--B	0	-	-	y	B
0	N	1	1	0	0	K--C	0	-	-	y	C
0	N	1	1	0	1	K--D	0	-	-	y	D
0	N	1	1	1	0	K--E	0	-	-	y	E
0	N	1	1	1	1	K--F	0	-	-	y	F

LEGEND:

bbbb = binary value of hexadecimal digit z.

y = hexadecimal digit extracted from the NA field (NA 4 through 7).

N = may be either One or Zero (don't care situation).

- = may be any hexadecimal digit (don't care situation).

Table 3-10 Internal Bus Codes Selector (Sheet 1 of 2)

BI CODE (HEX)	FIRMWARE NAME	BI RECEIVES:		
		BI(0C-0f)	BI(10-17)	BI(18-1F)
20	BIR*	0000	H(18) See Note 1	ALU (18-1F)
21	BITS	0000	Y(07), 0000000	1,CPID 0, NA(4-7)
22	BIL	0000	H(10) See Note 2	H(10-17)
23	BIN(ns)**	0000	Interrupt Buffer	
23	DSTx(ds)**	0000	Bus Data Buffer	
23	PSTx(ps)**	0000	Bus Procedure Buffer	
24	BIA	Address Bus (BA)		
25	BIP(ns)**	H(1C-1F)	Control Panel	
25	DSHx(ds)**	H(1C-1F)	Bus Data Buffer	
25	PSHx(ps)**	H(1C-1F)	Bus Procedure Buffer	
26	BIB*	AU(0C-0F)	00000000	0000, AU(0C-0F)
27	BIV	0000	00000000	10,S(LVL)
28	Not Used			
29	BITC	0000	Y(07), 0000001	0,CPID,0. NA(4-7)
2A	BIH	0000	H(18-1F)	H(10-17)
2B	BIZ	0000	Trap Status Z-Word	
2C	Not Used			
2D	BIX	0000	Hex Decode of XB(0-3)	

Table 3-10 Internal Bus Codes Selector (Sheet 2 of 2)

BI CODE (HEX)	FIRMWARE NAME	BI RECEIVERS:		
		BI(0C)-0F)	BI(10-17)	BI(10-17)
2E	BII	0000	Indicators	00000000
2F	BIS	0000	Processor Status Register (S)	

*See Table 3-3.

**LEGEND

ns = no stal.

ds = stall, awaiting Megabus data (see Table 3-12).

ps = stall, awaiting procedure from Memory (see Table 3-12).

NOTES

1. Repeat bit H(18) for all eight positions.
2. Repeat bit H(10) for all eight positions.

Table 3-11 Internal Bus Control Indicators (Sheet 1 of 2)

BI CODE	FIRMWARE NAME	OPERATION PERFORMED
30	-	None
31	RI1	Loads the indicator register from the BI(18 through 1F).
32	RI2	Sets the overflow indictor if BI(10) and BI (11) are not equal; otherwise, no action occurs.
33	RI3	Copies the ALU overflow into the overflow indicator.
34	-	Not used.
35	RI5	Copies the Megabus ACK flip-flop into the I/O indicator.
36	RI6	If the RALU output is not Zero, sets the bit indicator; otherwise, clears the bit indicator.

Table 3-11 Internal Bus Control Indicators (Sheet 2 of 2)

BI CODE	FIRMWARE NAME	OPERATION PERFORMED
37	RI7	Copies BI(10) into the bit indicator, copies the ALU overflow into the overflow indicator, and copies the ALU carry into the carry indicator.
38	RI8	Copies ALU(10) into the less than indicator. Clear the greater than indicator if either BI(10) is one or the ALU output is zero; otherwise, sets the greater than indicator.
39	RI9	Copies AU(0C) into the less than indicator. Clears the greater than indicator if either AU(0C) is one or the ALU output is zero; otherwise, sets the greater than indicator.
3A	RIA	Copies the $\overline{\text{SIGN}}$ flip-flop to the greater than indicator; copies the SIGN flip-flop to the less than indicator and copies BI(10) to the unlike signs indicator.
3B	RIB	Copies the ALU overflow into the overflow indicator and copies the ALU carry into the carry indicator.
3C	RIC	Copies the shift out from Q(1F) to the carry indicator.
3D	RID	Copies BI(1F) into the carry indicator.
3E	RIE	Copies BI(10) into the carry indicator.
3F	RIF	Copies the ALU carry into the carry indicator.

3.1.9 Select Modify (SM) Field

The SM field (bits 28 through 30) affects the LS and RS fields equally. When the SM code is Zero, the LS and RS codes directly address the RALU left and right select inputs. SM codes 1, 2, 6, and 7 cause one of four bit-groups in the F (RF) register and SEL (RSEL) register to be ANDed with the RS and LS addresses, directing the result to the left and right register file inputs. SM codes 3 and 7 create constants that are ANDed with the RS and LS addresses (see Table 3-12 and 3-13).

Table 3-12 Select Modify Codes

SM CODE	FIRMWARE NAME	OPERATION PERFORMED
0	-	The LS and RS addreses are left unmodified.
1	SMN	The LS and RS addresses are ANDed with 1, RF (01 through 03), the register number field of the instruction word.
2	SMX	The LS and RS addresses are ANDed with 1, RF (09 through 0B), the field of the address syllable.
3	SME	The LS and RS addresses are ANDed with 1, 1, 1, 0.
4	-	Not used.
5	SMD	The LS and RS addresses are ANDed with 1, 1, 0, 1.
6	SMR	The LS and RS addresses are ANDed with 1, RSEL (1 through 3), the field of the address syllable.
7	SMS	The LS and RS addresses are ANDed with RSEL (0 through 3).

3.1.10 Megabus (BS) Control Field

Refer to Table 3-14 for Megabus control codes. This 5-bit field (bits 31 through 35) performs three functions:

1. Initiates Megabus cycles.

2. Controls the loading or incrementing of either the memory address register (Y) or the program counter register (P).

3. When data is requested from memory or an I/O controller and the data has not yet arrived, the BS field stalls the clock (inhibiting advancement beyond the current firmware step) until the data request is satisfied.

Table 3-13 Register Selection

SM PATTERN	LEFT/RIGHT ADDRESS SELECTION								SM CODE
	0	3	6	7	8	B	E	F	
1,1,1,1	D0 M0	D3 M3	D6 M6	D7 M7	B0 A0	B3 A3	B6 A6	B7 A7	0
1,F(1-3)	D0 M0	- -	D#' M#'	D# M#	B0 A0	- -	- -	B#	1(SMN)
1,F(9-B)	D0 M0	DX	- -	DX7	B0 A0	- -	- -	BX	2(SMX)
1,1,1,0	DO M0	D2 M2	D6 M6	D6 M6	B0 A0	B2 A2	B6 A6	B6 A6	3(SME)
1,1,0,1	D0 M0	D1 M1	D4 M4	D5 M5	B0 A0	B1 A1	B4 A4	B5 A5	5(SMD)
1,SEL(1-3)	D0 M0	- -	DB' MB'	DB MB	B0 A0	Bb	- -	BB	6(SMR)
SEL(0-3)	D0 M0	- -	- -	- -	- -	- -	- -	RSEL RAMSEL	7(SMS)
	0	1	2	3	4	5	6	7	
SM PATTERN	LEFT/RIGHT SELECT CODES								SM CODE

DB', MB', D#', OR M#' = The most significant half of an addressed pair.

DB, MB, D#, OR M# = An addressed register.

BB or B# = Selected Base Register (1 of 7).

Bb = Selected Base Register (B1, B2, or B3).

BX = Selected Base Register (B0 through B7).

RSEL = Selected Register (1 of 16).

DX7 = Used with CIP.

DX = Selected Index Register (D0, D1, D2, or D3).

Table 3-14 Megabus Control Codes (Sheet 1 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
00	PMUS	Loads the Address Bus (BA) from the contents of the Program Counter (P) register.
01	PURG	Loads the BA from the contents of the P register; discards Procedures 1 and 2 (P1/P2); sets the MT flip-flop.
02	YMUX	Loads the BA from the contents of the Y register.
03	YINC	Loads the BA from the contents of the Y register; increments the Y register by 1.
04	YLOD	Loads the BA from the contents of the P register; loads the Y register from the internal bus.
05	PINC (See Note 1)	Loads the BA from the contents of the P register; discards the P1 and P2 procedures; increments the P and CTR registers by 1; sets the MT flip-flop.
06	PAGJ	Loads the BA from the output of the memory management unit (bits 03 through 14) and the Y register (bits 15 through 22); loads the Y register from the internal bus.
07	YGJW	Loads the Memory Management Unit (MMU) from the contents of the Y register through the address bus (request check for write permission from the MMU).
08	YBAY	Loads the BA from the contents of the Y register; loads the Y register from the internal bus.
09	PLOD	Loads the BA from the contents of the P register; discards the P2 and P2 procedures; loads the P register from the internal bus; sets the MT flip-flop.
0A	YREL	Loads the BA from the contents of the Y register; loads the Y register from the internal bus ORed with Central Processor Channel (CPID) number.
0B	YOLD	Loads the BA from the contents of the Y register; loads the Y register (bits 07 through 22) from the internal bus.

Table 3-14 Megabus Control Codes (Sheet 2 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
0C	DSTY (See Note 2)	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the Data Buffer (BD); loads the Y register from the internal bus.
	DSHY	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD; loads the Y register with the contents of the internal bus.
0D	DSTU (See Note 2)	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the BD; increments the Y register by 1.
	DSHU	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD; increments the Y register by 1.
0E	DSHP	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD; loads the P register from the internal bus; sets the MT flip-flop and discards the contents of procedure buffers P1 and P2.
0F	DSTL (See Note 2)	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the BD.
	DSHL	Loads the BA from the contents of the Y register (clock is stalled, if necessary, until the data word is received from memory); transfers the data word to the internal bus from the H register (bits 1C through 1F) and the BD.

Table 3-14 Megabus Control Codes (Sheet 3 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
10	DRCL	Loads the BA from the contents of the Y register; initiates a memory read cycle; locks or unlocks memory if executing a Read-Modify-Write (RMW) instruction.
11	DRCB	Loads the BA from the contents of the Y register; initiates a memory read cycle bypassing cache.
	DRCI	Loads the BA from the contents of the Y register; initiating a memory read cycle.
12	IORC	Loads the BA from the contents of the Y register; initiates an I/O read cycle.
14	DWHW	Loads the BA from the contents of the Y register; increments the Y register by 1; writes one byte from the internal bus to memory.
	DWWU	Loads the BA from the contents of the Y register; increments the Y register by 1; writes one word from the internal bus to memory.
15	DWHL	Loads the BA from the contents of the Y register; locks or unlocks memory if executing a Read-Modify-Write (RMW) instruction, writes one word from the internal bus to memory.
	DWWL	Loads the BA from the contents of the Y register; locks or unlocks memory if executing a Read-Modify-Write (RMW) instruction; writes one word from the internal bus to memory.
16	IOWU	Loads the BA from the contents of the Y register; writes one word to an I/O Device; increments the Y register by 1.
17	IOWH	Loads the BA from the contents of the Y register; writes one byte to an I/O Device.
	IOWW	Loads the BA from the contents of the Y register; writes one word to an I/O Device.
19	PRCI (See Note 4)	Loads the BA from the contents of the P register; initiates a memory read cycle for two words of procedure.

Table 3-14 Megabus Control Codes (Sheet 4 of 5)

BS CODE	FIRMWARE NAME	OPERATION PERFORMED
1A	PRCP	Loads the BA from the contents of the Y register; initiates a memory read cycle for two words of procedure; loads the P register from the internal bus.
1B		Not used.
1C	PSTY (See Notes 1, 2, and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP buffer is full); transfers the next procedure word to the internal bus from the Procedure Buffer (BP); loads the Y register from the internal bus; increments the P and CTR registers by 1.
	PSHY (See Notes 1 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necesssary, until the BP is full); transfers the next procedure word to the internal bus from the H register (bits 1C through 1F) and the BP; loads the Y register from the internal bus; increments the P and CTR registers by 1.
1D	PSTL (See Notes 1, 2 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP is full); transfers the next procedure word to the internal bus from the BP; increments the P and CTR registers by 1.
	PSTI (See Notes 1 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP is full); transfers the next procedure word to the I register via the internal bus from the BP; increments the P and CTR registers by 1.
	PSHL (See Notes 1 and 4)	Loads the BA from the contents of the P register (clock is stalled, if necessary, until the BP is full); transfers the next procedure word to the internal bus from the H register (bits 1C through 1F) and the BP; increments the P and CTR registers by 1.

Table 3-14 Megabus Control Codes (Sheet 5 of 5)

NOTES

1. If the XB register is clocked, the control store word (bit 54) is transferred to the CTR register rather than incrementing the CTR register by 1.

2. Internal bus bits 0C through 0F are Zero.

3. Loads only internal bus (bits 10 through 1F); BI field available to control loading of indicator register.

4. The read request is ignored unless procedure buffers P1 and P2 are empty.

3.1.11 General Purpose (GP) Micro-Op Field

Refer to Table 3-15 for GP micro-op decodes. This 6-bit field (bits 36 through 41) generates a total of 64 micro-ops and is divided into five groups:

1. 16 micro-ops (GP40 through GP7C) affect the SIGN, ZERO, and SHIN control flip-flops, as well as the XB register.

2. 16 micro-ops (GP80 through GPBC) affect the F, SEL, and H registers, as well as the MISC control flip-flop.

3. 16 micro-ops (GP00 through GP3C) combine the actions of groups 1 and 2.

4. 8 micro-ops (GPC0 through GPDC) affect the following:

   a. BOOT and WRAP control flip-flops.

   b. S, M, and LINK registers.

   c. Megabus RINT function and Memory Management Unit (MMU) control functions.

5. 8 micro-ops (GPE0 through GPFC) are used for control panel functions.

Table 3-15 General Purpose Micro-Op Codes (Sheet 1 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
00	GP00	None.
01	GP04	Loads the H register from the internal bus; loads the SIGN flip-flop from internal bus bit 10.
02	GP08	Loads the SIGN flip-flop from internal bus bit 10; loads the ZERO flip-flop from the AUZERO function; loads the MISC flip-flop from internal bus bit 1f.
03	GP0C	Loads F (bits 8 through B) and SEL registers from the internal bus (bits 18 through 1F); sets the SIGN flip-flop and the FIRST flip-flop.
04	GP10	Clears the SHIN1 and SHIN2 flip-flops; shifts the XB register right by one position (ALU bit 1F to XB register bit 0).
05	GP14	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); set the ZERO flip-flop.
06	GP18	Loads the F and SEL registers from the internal bus; clears the XB register; sets the FIRST flip-flop.
07	GP1C	Sets the SHIN1 and SHIN2 flip-flops; loads the SEL register from internal bus bits 1C through 1F; sets the FIRST flip-flop.
08	GP20	Loads the ZERO flip-flop from the AUZERO function; loads the F and SEL registers from internal bus; sets the FIRST flip-flop; clears the MISC, SHIN1, and SHIN2 flip-flops; clears the XB register.
09	GP24	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); clears the SHIN2 and MISC flip-flops; sets the SHIN1 flip-flop.
0A	GP28	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); clears the ZERO flip-flop.

Table 3-15 General Purpose Micro-Op Codes (Sheet 2 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
0B	GP2C	Loads the F (bits 8 through B) and SEL registers from the internal bus (bits 18 through 1F); loads the SIGN flip-flop with internal bus bit 1F; if the internal bus (bits 10 through 15) equal/unequal 0, sets/clears the MISC flip-flop; sets the FIRST flip-flop.
0C	GP30	Loads the SIGN flip-flop from internal bus bit 10; sets the MISC flip-flop.
0D	GP34	Loads the SIGN flip-flop from internal bus bit 1F; sets the MISC flip-flop.
0E	GP38	Loads the SIGN flip-flop from internal bus bit 1F; loads the ZERO flip-flop with the state of the panel QLT flip-flop.
0F	GP3C	RFU.
10	GP40	Loads the SHIN2 flip-flop from the $\overline{\text{SIGN}}$ flip-flop.
11	GP44	Loads the SHIN2 flip-flop from the $\overline{\text{SIGN}}$ flip-flop; loads the SIGN flip-flop with internal bus bit 10.
12	GP48	Loads the SHIN1 flip-flop from the BIT indicator.
13	GP4C	Sets the SIGN flip-flop.
14	GP50	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0).
15	GP54	Shifts the XB register right by one position (ALU bit 1F to XB register bit 0); loads the SIGN flip-flop with internal bus bit 10.
16	GP58	Clears the ZERO flip-flop.
17	GP5C	Sets the ZERO flip-flop.
18	GP60	Loads the ZERO flip-flop from the AUZERO function.
19	GP64	Loads the SIGN flip-flop from internal bus bit 10.

Table 3-15 General Purpose Micro-Op Codes (Sheet 3 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
1A	GP68	Loads the SIGN flip-flop from internal bus bit 0C.
1B	GP6C	Loads the SIGN flip-flop from internal bus bit 1F.
1C	GP70	Clears the SHIN1 and SHIN2 flip-flops.
1D	GP74	Clears the SHIN2 flip-flop and sets the SHIN1 flip-flop.
1E	GP78	Sets the SHIN2 flip-flop and clears the SHIN1 flip-flop.
1F	GP7C	Sets the SHIN2 and SHIN1 flip-flops.
20	GP80	Loads the F and SEL registers from internal bus bits 10 through 1F; sets the FIRST flip-flop.
21	GP84	RFU.
22	GP88	Loads F register bits 8 through B and SEL register from internal bus bits 18 through 1F, respectively; sets the FIRST flip-flop.
23	GP8C	Clears the MISC flip-flop.
24	GP90	Sets the MISC flip-flop.
25	GP94	Loads the MISC flip-flop from the carry function.
26	GP98	Loads the MISC flip-flop from the Megabus ACK flip-flop.
27	GP9C	Loads the MISC flip-flop from the Protection Violation function.
28	GPA0	RFU.
29	GPA4	RFU.
2A	GPA8	Loads the SEL register from internal bus bits 1C through 1F; sets the FIRST flip-flop.
2B	GPAC	Loads the H register from the internal bus; loads the SEL register with internal bus bits 1C through 1F; sets the FIRST flip-flop.

Table 3-15 General Purpose Micro-Op Codes (Sheet 4 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
2C	GPB0	Loads the H register from the internal bus.
2D	GPB4	RFU.
2E	GPB8	RFU.
2F	GPBC	RFU.
30	GPC0	Broadcasts RINT (Retry Interrupts).
31	GPC4	Clears BOOT flip-flop.
32	GPC8	Loads the WRAP flip-flop from the result of an exclusive OR operation between the CARRY function and the output of the $\overline{\text{SIGN}}$ flip-flop; clears the FIRST flip-flop.
33	GPCC	Loads the RING number in the S register from internal bus bits 11 and 12.
34	GPD0	Loads the interrupt priority level in the S register from internal bus bits 1A through 1F; clears INTBSY if F register bit 5 is zero.
35	GPD4	Loads the LINK register from internal bus bits 17 through 1E.
36	GPD8	Loads M collector register from Y register bit 15, F register bit A, SEL register bits 0 and 2, $\overline{\text{ZERO}}$ flip-flop state, and the $\overline{\text{AUZERO}}$ function.
37	GPDC	With the RALU function code bits (see Table 3-1), AF0 and AF3 both Zero, loads the Memory Management Unit (MMU) with the validity bit and base address field of a segment descriptor. With AF0 Zero and AF3 One, loads the MMU with the access rights and segment size field of a segment descriptor. With AF0 One and AF3 Zero, sets the MMU NO-CHECK flip-flop. With AF0 and AF3 both One, makes the MMU check the validity of range, address, and access rights.

Table 3-15 General Purpose Micro-Op Codes (Sheet 5 of 5)

GP CODE	FIRMWARE NAME	OPERATION PERFORMED
38	GPE0	Loads panel display H4 from internal bus bits 1C through 1F.
39	GPE4	Loads panel displays H5, H6, H7, and H8 from from internal bus bits 10 through 1F.
3A	GPE8	Clears QLT indicator if the panel is Load mode.
3B	GPEC	Sets MPLOAD mode; sets panel QLT indicator if panel is locked, memory is volatile, and the ZERO flip-flop is Off.
3C	GPF0	BI receives control word from panel (see Table 3-9, BIP code).
3D	GPF4	Loads the panel TRAFFIC indicator from the ZERO function; sets Run mode if panel is in Ready mode.
3E	GPF8	Clears panel Load mode; sets TRAFFIC indicator and Run mode if panel is in Ready mode.
3F	GPFC	Clears panel CHANGE control flip-flop.

3.1.12 Test Condition (TC) Field

This 6-bit field (bits 42 through 47) samples one of 64 signals upon which the firmware may branch (see Tables 3-16 through 3-20).

Table 3-16 Test Condition Codes (Sheet 1 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
0	TC00	False.
1	TC01	Scientific Store Operation.
2	TC02	SIP presence.
3	TC03	CIP presence.
4	TC04	Operand size is 8 bits.
5	TC05	Operand size is 16 bits.
6	TC06	Operand size is greater than 16 bits.

Table 3-16 Test Condition Codes (Sheet 2 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
7	TC07	Operand size is 64 bits.
8	TC10*	Branch operation condition; if F register (bits 0 through 7) is equal to 0, POWER OK is tested.
9	TC11	Control panel $\overline{\text{EXECUTE}}$ mode; copy control panel LOAD mode into MYBOOT flip-flop.
A	TC12	LAF.
B	TC13	Control panel $\overline{\text{LOAD}}$ mode.
C	TC14	Inclusive OR of MISC or ZERO flip-flop.
D	TC15	SHIN1 flip-flop.
E	TC16	SHIN2 flip-flop.
F	TC17	In previous cycle, the address bus was loaded with the contents of the Y register.
10	TC20**	If # (F register bits 1 through 3) is not equal to Zero, test the complement of the specified bit of the internal bus (19 through 1F); if # is equal to Zero, test complement of the I/O indicator.
11	TC21	Carry function.
12	TC22	WCS absence.
13	TC23	Internal bus bit 10.
14	TC24	SHIN function.
15	TC25	Internal bus bit 10 is not equal to internal bus bit 11.
16	TC26	Inclusive OR of SHIN and AUZERO functions.
17	TC27	AUZERO function.
18	TC30	Cache absence.
19	TC31	Read-Modify-Write Flip-flop (RMWF) (initialize MT logic).
1A	TC32	Control panel LOCK function.

Table 3-16 Test Condition Codes (Sheet 3 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
1B	TC33	M register bit 0 (J).
1C	TC34	Indicator (I) register bit 0 (carry).
1D	TC35	Megabus Acknowledge flip-flop.
1E	TC36	Ring number code greater than 1.
1F	TC37	Parity error indicator.
20	TC40	SEL register value not equal to Zero.
21	TC41	RFU.
22	TC42	F register # field equals 7.
23	TC43	XB register (bit 0).
24	TC44	F register (bit 4).
25	TC45	F register (bit 5).
26	TC46	F register (bit 6).
27	TC47	F register (bit 7).
28	TC50	F register (bit 8).
29	TC51	SEL register register = 0 (decrements the SEL register by 1).
2A	TC52	F register (bit 9).
2B	TC53	F register (bit B).
2C	TC54	SEL register (bit 0).
2D	TC55	SEL register (bit 1).
2E	TC56	SEL register (bits 1, 2, 3, = 7).
2F	TC57	SEL register (bit 3).
30	TC60	WDT/RTC request (SNAPIT).
31	TC61	YELLOW flag (clear SNAPER and RBYELF flip-flops).
32	TC62	Register address syllable.

Table 3-16 Test Condition Codes (Sheet 4 of 4)

TC CODE	FIRMWARE NAME	TEST CONDITION
33	TC63	ZERO flip-flop.
34	TC64	SIGN flip-flop.
35	TC65	MISC flip-flop.
36	TC66	SEL register bit 2.
37	TC67	Inclusive OR of GJBARF function and FFWRAP flip-flop.
38	TC70	Interrupt not busy.
39	TC71	Interrupt busy or External Trap; clears SIGN flip-flop.
3A	TC72	Internal bus bit 18.
3B	TC73	Overflow function; if AS field equals AWXX, test AU(0C).
3C	TC74	RFU.
3D	TC75	RFU.
3E	TC76	Q register shift right; if not shift right, test Data Descriptor Length flip-flop.
3F	TC77	Internal bus bit 1F.

*Three additional multiplexers are used to create the input for TC10 (see Tables 3-17, 3-18, and 3-19).

**An additional multiplexer is used to create the input for TC20 (see Table 3-20).

Table 3-17 Test Condition 10 (Group A)

FUNCTION DECODE			
NUMZ	F04	F05	FUNCTION TESTED
0	0	0	Refer to Group B decode
0	0	1	Refer to Group B decode
0	1	0	Refer to Group B decode
0	1	1	Not used
1	0	0	Refer to Group C decode
1	0	1	Refer to Group B decode
1	1	0	Refer to Group C decode
1	1	1	Logical One

Table 3-18 Test Condition 10 (Group B)

FUNCTION DECODE			FUNCTION TESTED
NUMZ	F06	F07	
0	0	0	FFSIGN (SIGN flip-flop)
0	0	1	FFZERO (ZERO flip-flop)
0	1	0	$\overline{\text{FFSIGN}}$ AND $\overline{\text{FFZERO}}$
0	1	1	FFMISC (MISC flip-flop)
1	0	0	RIOVFF (Overflow Indicator)
1	0	1	RIBITF (Bit Indicator)
1	1	0	RICRYF (Carry Indicator)
1	1	1	RIACKF (I/O Indicator)

Table 3-19 Test Condition 10 (Group C)

FUNCTION DECODE			FUNCTION TESTED
F04	F06	F07	
0	0	0	MYPROK (Power OK)
0	0	1	Not used
0	1	0	RILESF (Less Than Indicator)
0	1	1	RIGTRF (Greater Than Indicator
1	0	0	RILESF and RISNEF are different
1	0	1	RILESF and RIGTRF are both off
1	1	0	RIGTRF and RISNEF are different
1	1	1	RISNEF (Signs Unlike Indicator)

Table 3-20 Test Condition 20 (Group M)

FUNCTION DECODE			FUNCTION TESTED
F01	F02	F03	
0	0	0	Acknowledge Indicator ($\overline{\text{RIACKF}}$)
0	0	1	Internal Bus ($\overline{\text{Bit 19}}$)
0	1	0	Internal Bus ($\overline{\text{Bit 1A}}$)
0	1	1	Internal Bus ($\overline{\text{Bit 1B}}$)
1	0	0	Internal Bus ($\overline{\text{Bit 1C}}$)
1	0	1	Internal Bus ($\overline{\text{Bit 1D}}$)
1	1	0	Internal Bus ($\overline{\text{Bit 1E}}$)
1	1	1	Internal Bus ($\overline{\text{Bit 1F}}$)

3.1.13 Branch Type (BR) Field

This 4-bit field (bits 48 through 51) selects the mechanism that is to produce the address of the next firmware step (see Table 3-21).

Table 3-21 Branch Type Codes (Sheet 1 of 2)

BR CODE	FIRMWARE NAME	OPERATION PERFORMED
0	X0T	If the test condition is false, use the NA field. If the test condition is true, OR the NA field with 3.
1	XLT	If the test condition is false, use the NA field. If the test condition is true, use the LINK register.
2	XAT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation ouptuts for the XA branch.
3	XBT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation outputs for the XB branch.
4	XRT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation ouputs for the XR branch.
5	XWT	If the test condition is false, use the NA FIELD. If the test condition is true, use the next address generation outputs for the XW branch.
6	XET	If the test condition is false, use the NA field. If the test condition is true, use the next address generation outputs for the XE branch.
7	XFT	If the test condition is false, use the NA field. If the test condition is true, use the next address generation outputs for the XF branch.

Table 3-21 Branch Type Codes (Sheet 2 of 2)

BR CODE	FIRMWARE NAME	OPERATION PERFORMED
8	X0F	If the test condition is true, use the NA field. If the test condition is false, OR the NA field with 3.
	X-F	Use the NA field (test should never be false).
9	XLF	If the test condition is true, use the NA field. If the test condition is false, use the LINK register.
A	XAF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XA branch.
B	XBF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XB branch.
C	XRF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XR branch.
D	XWF	If the test conditon is true, use the NA field. If the test condition is false, use the next address generation outputs for the XW branch.
E	XEF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XE branch.
F	XFF	If the test condition is true, use the NA field. If the test condition is false, use the next address generation outputs for the XF branch.

3.1.14 C Field

This 1-bit field (bit 52) controls the (optional) cache memory usage.

3.1.15 Next Address (NA) Field

This 11-bit field (bits 53 through 63) defines the address of the next firmware step except as altered by the TC and BR fields (see Table 3-22).

Table 3-22 Next Address Code

NA CODE (HEX)	FIRMWARE NAME	INTERPRETATION
000 through 3FF	N	Specifies the address of the next firmware step unless altered by the Branching (BR) field. The middle digit of the NA field (bits 57 through 60 of the firmware word) becomes the 16 weight digit of 9-bit sign-extended firmware generated constants. Bits 57 through 60 are also used to create function codes for communication with external processors (refer to Table 3-10, internal bus codes 21 and 29).

3.2 TRAPS

The trap firmware is supported by four groups of main memory locations, which are identified as follows:

- Dedicated memory locations
- Trap save area
- Interrupt save area
- Trap handler procedure.

3.2.1 Dedicated Memory Locations

The first dedicated memory locations associcated with traps are those which contain Next Available Trap Save Area Pointers (NATSAP) to the next available trap save area. The second group of locations are hexadecimal 0024 through 007F. These locations contain pointers (trap vectors) to the trap handler procedures. The third group of locations, hexadecimal 0080 through 00FF, contain pointers (interrupt vectors) to the interrupt save areas. The dedicated memory locations are illustrated in Figures 2-6 and 2-7.

3.2.2 Trap Save Area

Each trap save area consists of eight sequential entries, which are used to store the context associated with a particular trap. The layout of each trap save area is shown below.

Entry 1	TSAL*
Entry 2	I Register
Entry 3	Data Register 3
Entry 4	Instruction
Entry 5	Z Field
Entry 6	A Field*
Entry 7	P Field*
Entry 8	Base Register 3*

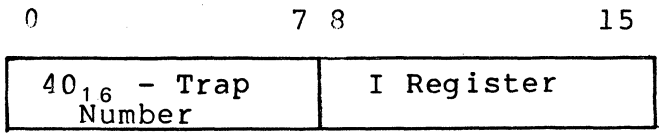
*Occupies two words in LAF

3.2.2.1 Entry 1

Entry 1 contains the Trap Save Area Link (TSAL). TSAL is an address pointer to the first location in the next trap save area. The first location (TSAL) in this trap save area points to the first location in the next one and so on, thereby forming a linked list of trap save areas. The eldest trap save area in the linked list is indicated by a null (Zero pointer) in entry 1.

3.2.2.2 Entry 2

Entry 2 stores the Trap Vector number and the current contents of the indicator register when a trap occurs. The format of entry 2 is:



3.2.2.3 Entry 3

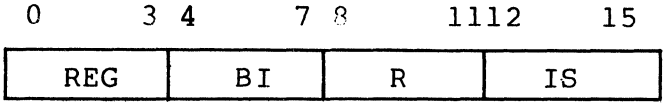
Entry 3 stores the current contents of Data Register 3 (D3) when a trap occurs.

3.2.2.4 Entry 4

Entry 4 stores the first word of the instruction that was being processed when the trap occurred.

3.2.2.5 Entry 5

Entry 5 is defined as the Z field and is used to store miscellaneous information when a trap occurs. The format of the Z field is:



3.2.2.5.1 REG FIELD

The REG field indicates whether or not the A field content is valid. If this digit equals Zero, the A field contains valid information; otherwise, the A field is unspecified.

3.2.2.5.2 BI Field

Two conditions must exist before this field is valid:

1. The REG digit must be zero.

2. The trapped instruction must be a bit, byte, or I/O instruction.

   a. Bit Instruction: The BI field contains the four low-order bits of the selected index register, or 0000 if no indexing was used.

   b. Byte Instructions: The BI field contains X000, where X equals the low-order bit of the selected index register, or 0000 if no indexing was used.

   c. I/O Instructions: If the A field (entry 6) contains the channel number, the most significant bit (4) of the BI field contains the low-order bit of the function code.

3.2.2.5.3 R Field

This field indicates the processor ring number when a trap occurs.

3.2.2.5.4 IS Field

The IS field indicates the size (in words) of the trapped instruction. However, if the trap occurs before the entire instruction was fetched, the IS field indicates the number of words that were fetched from the beginning of the Instruction Fetch firmware. For example, this condition may ocur when an I/O instruction is trapped, or if a nonrecoverable memory error occurs while fetching part of a multiword instruction.

3.2.2.6 Entry 6

Entry 6 is defined as the A field. This field is valid only when bit 0 of the Z field equals Zero. The A field contains the effective address generated by the trapped instruction address syllable, the address of the trapped instruction, or the address of the following instruction, depending on the particular trap activated.

3.2.2.7 Entry 7

Entry 7 is defined as the P field. This field contains the return address from the trap handler procedure. In most cases, this return address is the address of the location or instruction following the trapped instruction. However, when a multiword instruction is trapped, the return address may be within the multiword instruction. Consequently, modification of the P field may be required before returning from the trap (RTT). Subtracting the IS field of Entry 5 from Entry 7 always produces a pointer to the start of the current instruction.

3.2.2.8 Entry 8

Entry 8 stores the contents of Base Register 3 (B3) when a trap occurs. After thus saving its conents, and before entry into the Trap Handler procedure, B3 is loaded with a pointer to the A field of the trap save area.

3.2.3 Interrupt Save Area

There is a linked list of zero or more trap save areas attached to each interrupt save area. One interrupt save area is assigned to each of the 64 possible interrupt levels. Entry (-1) of each interrupt save area is a pointer to the string of trap save areas that are currently in use by traps in this level. If there are no trap save areas assigned to the interrupt save area, the interrupt save area contains a null (Zero address).

3.2.4 Trap Handler Procedures

The trap handler procedures are software programs which process traps. Return from the trap handler procedure is accomplished using the Return from Trap (RTT) generic instruction.

3.2.5 Trap Functionality

A trap can occur from many places in the firmware as a result of any number of causes. For example, a trap will be entered on reference to an unavailable resource (e.g., during an instruction or data fetch), or when an error condition (e.g., program error, protection violation, etc.), occurs, or when an unusual result (e.g., arithmetic overflow) is encountered.

When a trap does occur, the Next Available Trap Save Area Pointer (NATSAP) will be read from memory to obtain a new Trap Save Area (TSA). A context save will take place to store the trap vector number, indicators, data register 3, the instruction, the Z field, the A field, the P field, and base register 3 in sequential memory locations in the new TSA. The interrupt vector links the Interupt Save Area (ISA) of the current process to the new TSA. The trap vector pointer is used and the resultant trap handler procedure is entered to process the trap.

A return from the trap routine will take place when the generic instruction Return from Trap (RTT) is invoked. The TSA is unlinked from the current process and is returned to the pool of available trap save areas.

3.3 INTERRUPTS

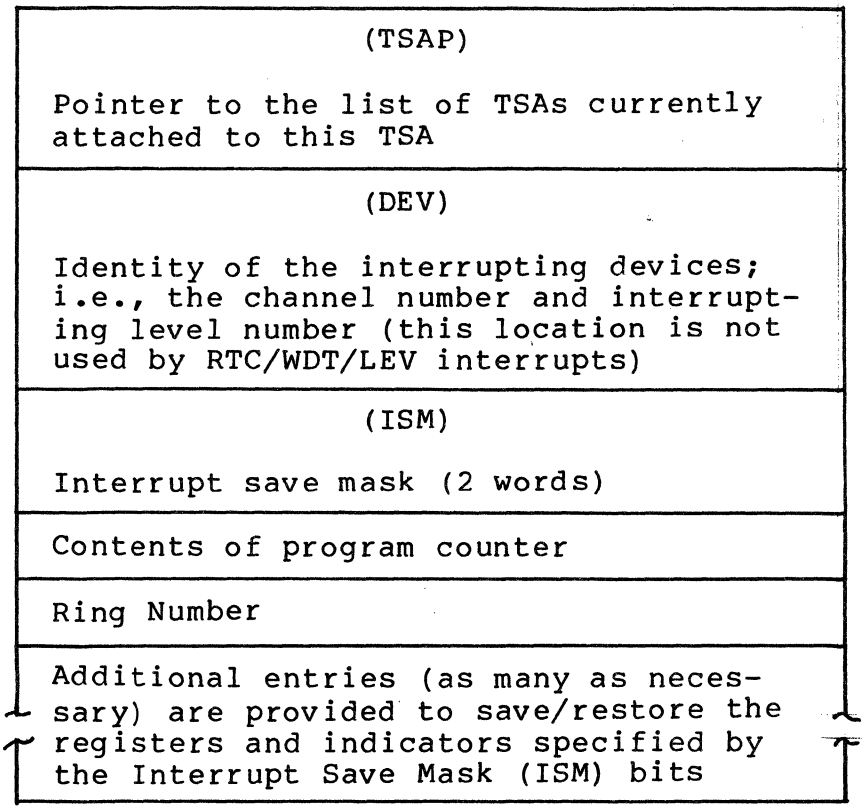
Every program in the CPU executes at a priority level, which is defined by the LVL field of the S register. This interrupt priority level has a range of 0 to 63, inclusive. Level 0 is defined as having the highest priority; level 63 has the lowest priority.

The program currently being executed can be interrupted by an event having a higher priority. The interrupt is serviced at the end of the instruction currently being executed. The level assignments for various events are shown in Table 2-23.

Table 3-23 Level Assignements for Interrupt Events

EVENT CAUSING INTERRUPT	LEVEL ASSIGNMENT	COMMENTS
Incipient power failure	0	Highest priority
Watchdog timer runout	1	-
Use of last TSA	2	-
Real-time clock	0 through 62	Level is contained in main memory location 0016 (Hex)
Device requiring service	1 through 62	Level is assigned to device by software
LEV instruction	0 through 63	Level specified by instruction

Each level has a corresponding interrupt vector, which points to an Interrupt Save Area (ISA). When the program currently being executed is interrupted, its context is stored in the assigned interrupt save area. The context of the interrupting process is then retrieved from the ISA that is assigned to its level and the process begins executing at its assigned level. The layout of each ISA is:



3.3.1 Interrupt Vectors

The interupt vectors (see Figures 2-6 and 2-7) consist of 64 address pointers. These address pointers are contained in dedicated memory locations 0080 through 00FF hexadecimal. The interrupt vector for level 0 is contained in the first entry; the interrupt vector for level 1 is in the second entry and so on up to the interrupt vector for level 63. Each interrupt vector points to one of the 64 interrupt save areas described above.

3.3.2 Activity Flags

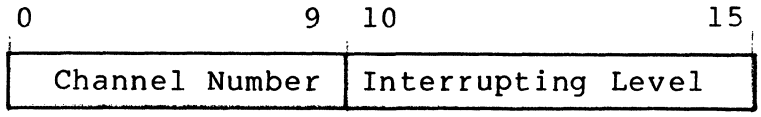
The activity flags (see Figure 2-6 or 2-7) associated with interrupts consist of 64 bits, which are stored in four dedicated memory locations: 0020, 021, 0022, and 0023 hexadecimal. The activity flag for level 0 is contained in bit 0 of location 20; the activity flag for level 15 is in bit 15 of location 20; the activity flag for level 16 is in bit 0 of location 21 and so on up to level 63, which is in bit 15 of location 23.

These activity flags indicate which processor levels are currently active (i.e., ready for execution or being executed). If a bit (activity flag) is On, an activity for the corresponding level will begin executing as soon as it is found to be the highest priority level active. If a bit is Off, the corresponding level is inactive. Activity flags are set by external interrupt requests and set/cleared by the LEV instruction.

NOTE

The activity flag for level 63 is always considered to be On.

When an external interrupt occurs, the identity of the interrupting channel is stored in the DEV entry of its interrupt save area. The format of location DEV is:



NOTES

1. If the interrupt vector for the interrupting level is null, the interrupt is acknowledged but not scheduled. However, the activity flags are scanned as described in subsection 3.3.3.

2. If the interrupting level equals the current level (or if different, and both levels have the same interrupt save area), the context save/restore process, described above, is bypassed.

3. Note 2 applies for all level changes, not just External Interrupts.

The Interrupt Save Mask (ISM) controls storing and loading of the processor registers during the save/restore process. The ISM format is:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M1	D1	D2	D3	D4	D5	D6	D7	I	B1	B2	B3	B4	B5	B6	B7
MMU	NATSAP			RFU		SIP	CIP	RFU						T	M

M = M registers 2 through 7

T = stack address register, T

CIP = the indicator register associated with the commercial instruction processor

SIP = the three scientific accumulators and the indicator register associated with the scientific instruction processor

NATSAP = the location where NATSAP will be obtained as shown in the chart below:

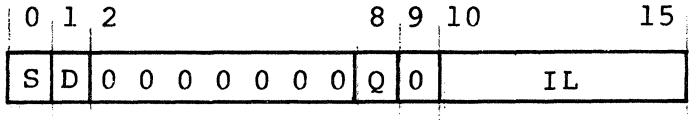
ISM BITS			MEMORY
17	18	19	LOCATION
0	0	0	0010
0	0	1	000E
0	1	0	000C
0	1	1	000A
1	0	0	0008
1	0	1	0006
1	1	0	0004
1	1	1	0002

MMU = the 16/31 segment descriptors of the memory management unit

The mask bits are scanned from right to left. If the mask bit is On, the corresponding processor register is either loaded from or stored in the ISA. The registers selected by the mask bits are loaded from/saved in consecutive memory locations, starting at the location immediately after that used to save/restore the S register. The number of memory locations utilized for the storage of processor registers is a function of the number of mask bits set.

3.3.3 LEV Instruction

The LEV instruction is one of the central processor single operand instructions and is used to set/clear activity flags. The LEV operand contains nine significant bits as shown below:



S = Suspend current level
D = Defer interrupt
Q = Quick level change (inhibit)
IL = Interrupt level number

The operations performed by the S, D, Q, and IL bits are listed in Table 3-24.

Table 3-24 LEV Operand Decode

OPERAND BITS S	D	Q	OPERATION PERFORMED
0	0	0	1. Sets the activity flag for level IL. 2. Scans activity flags to determine the highest priority active. 3. Saves the context of the current level*. 4. Loads the context for the highest priority active level*.
0	1	0	1. Sets the activity flag for level IL.
1	X	0	1. Sets the activity flag. 2. Clears the activity flag for the current level. 3. Scans the activity flags to determine the highest priority active interrupt level. 4. Saves the context of the current level*. 5. Loads the context for the highest priority active level*.
0	X	1	1. Sets the activity flag for level IL. 2. Sets the current level equal to IL without a context change. 3. Sets the IV of the new level equal to that of of the old level.
1	X	1	1. SEts the activity flag for level IL. 2. Clears the activity flag for the current level. 3. Sets the current level equal to IL without a context change. 4. Sets the IV of the new level equal to that of the old level.

*If interrupt save areas are the same, this operation is bypassed.

NOTE

An IL field equal to 63 is effectively a No-Op.

IV
HARDWARE OPERATION

The material contained in this section describes the hardware for the Central Processor Unit (CPU).

4.1 MASTER CLOCK

The CPU clock (see Figure 4-1) is a two-phase delay line clock that provides one variable and two fixed clock pulses. The variable clock pulse is called Master Clock (MCLOCK); the fixed clock pulses are called Data Cycle Now Permit (MCDCNP) and Local Register Valid (MLRVLD). All three clock pulses are generated by delay lines which are hereafter referred to as the clock cycle generator.

A Master Clock pulse is composed of an 80-nanosecond positive period and a variable length negative period. The negative intervals range from 80- to 240-nanoseconds in duration and are usually selected by bits 20 and 21 of the control store word (CK Field). Combining the selected negative period with the 80-nanosecond positive period produces one complete Master Clock cycle with an overall clock speed as indicated below.

CLOCK SPEED (nanoseconds)	SECOND HALF CYCLE (nanoseconds)	CK FIELD BITS	
		20	21
160	80	1	1
180	100	1	0
200	120	0	1
320	240	0	0

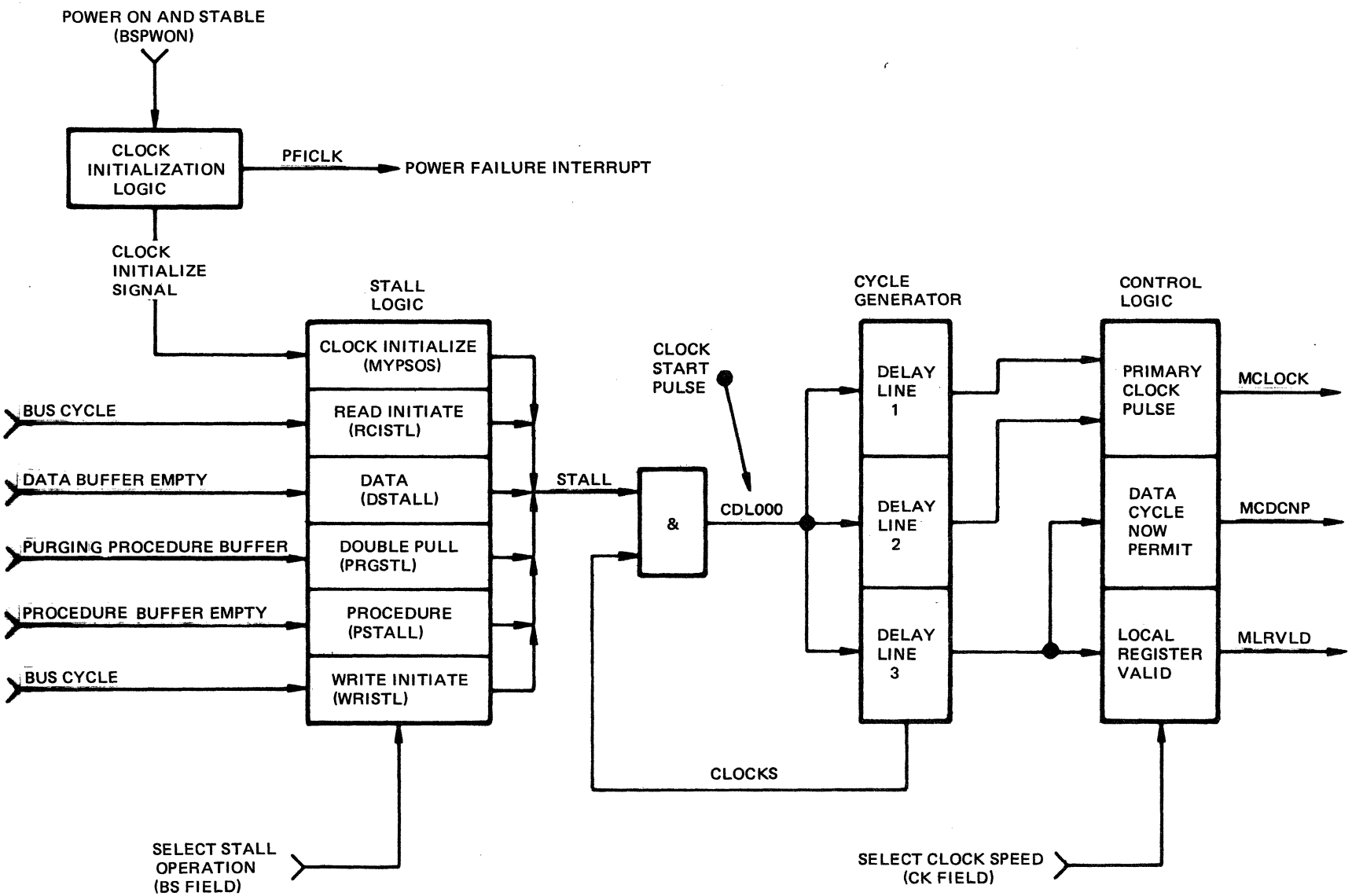


Figure 4-1 Master Clock Logic

A Data Cycle Now Permit pulse indicates that the Megabus address lines are valid; a Local Register Valid pulse indicates that the contents of the control store data register are valid.

The master clock logic can be divided into the following three areas:

- Clock initialization network
- Clock stall network
- Clock cycle generator.

4.1.1 Clock Initialization Network

The clock initialization network monitors the Megabus for a power-up sequence to initialize the CPU and the master clock, and to initiate the necessary CPU housekeeping operations when a power supply malfunction occurs. It performs these functions in conjunction with the clock stall network.

4.1.2 Clock Stall Network

The clock stall network momentarily stalls the clock when additional time is needed to complete one of the following CPU or firmware generated functions:

- A clock initialization that occurs during the CPU power-up sequence.
- A firmware initiated Megabus cycle to which no responding bus activity is expected (e.g., a memory write operation). The clock remains stalled until the selected unit acknowledges or rejects the request.
- A firmware cycle immediately following one in which a Megabus cycle was initiated and to which a response is expected. The clock remains stalled until the selected unit acknowledges or rejects the request.
- A firmware step that calls for data from a bus buffer that has not yet been filled. The clock remains stalled until the applicable buffer is full.

The clock is stalled by inhibiting the clock cycle generator (specifically MCLOCK pulses) when the CPU is initially powered-up, during a read or write cycle initiate, or for single-word or double-word data transfers from an external device (i.e., memory or I/O). The specific type of stall operation is determined by bits 31 through 35 of the control store word (BS Field).

4.1.2.1 Read Cycle Initiate Stalls

A Read Cycle Initiate is defined as a firmware step that requests data from a source external to the CPU (e.g., memory), and is intended to allow firmware to use the data in subsequent

firmware steps. The clock cycle generator is inhibited at the next MCLOCK pulse, preventing any further clock activity until the request is accepted or rejected over the Megabus.

4.1.2.2 Write Initiate Stalls

During a write operation to an external device, the clock is stalled until the device responds with a positive or negative acknowledgment.

4.1.2.3 Data Stalls

A Data Stall operation inhibits MCLOCK pulses to provide any additional time needed to fill the Data (BD) buffer or the Procedure buffer with pertinent data from memory. Two basic signals are used for this purpose. The first, ALOUTD, denotes that the data buffer is empty; the second, ALOUTP, denotes that the procedure buffers are empty. A third signal, PURGEF, inhibits additional double-fetch operations until all previous double-fetch requests have been answered and discarded.

4.1.3 Clock Cycle Generator

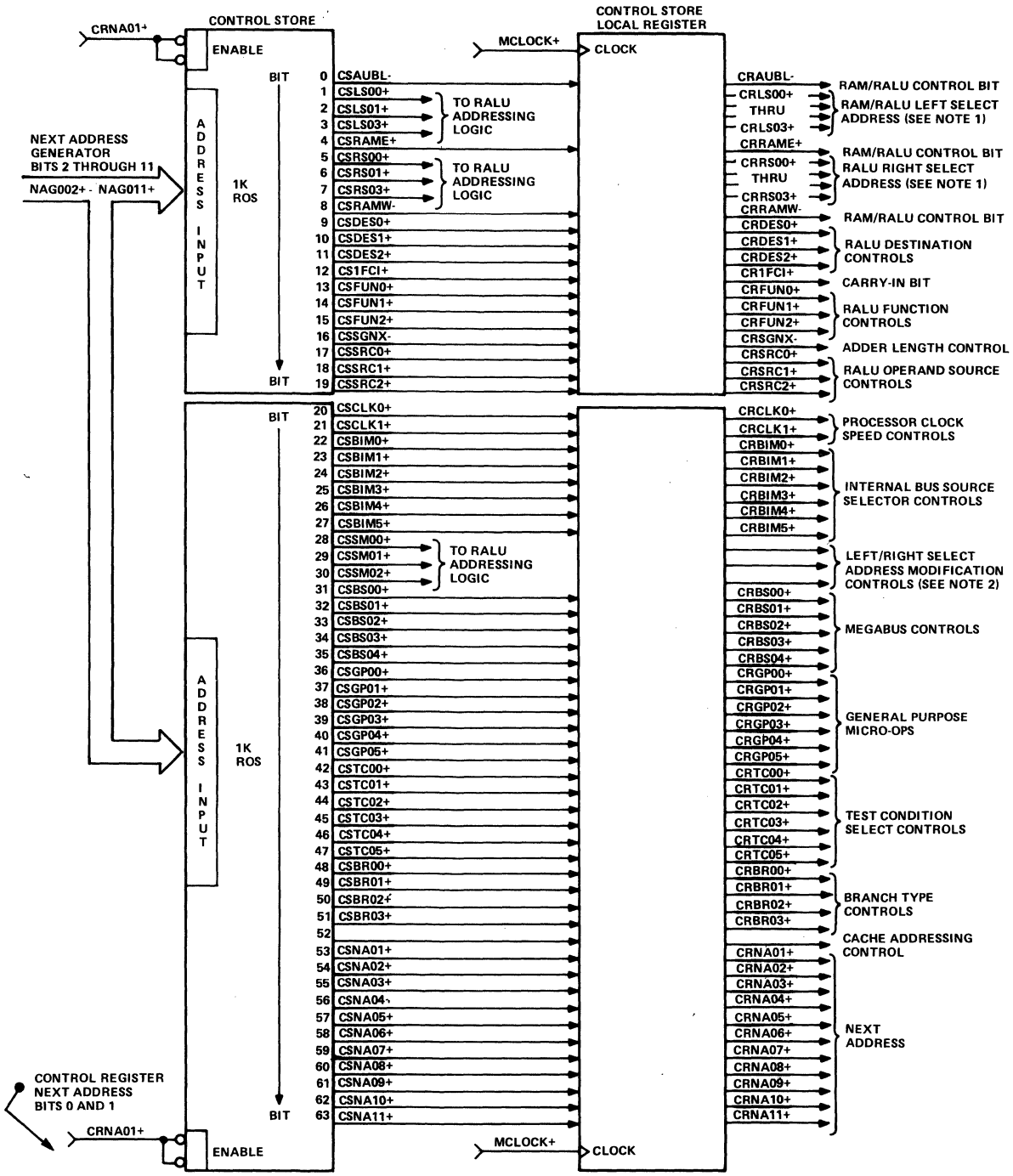
The clock cycle generator provides the selectable clock pulses used throughout the CPU, and consists of three delay lines plus associated control logic. Delay lines 1 and 2 generate the primary Master Clock (MCLOCK) pulse; delay line 3 generates the two previously described secondary clock pulses (MLRVLD and MCDCNP). All delay lines are tapped at selected intervals to generate their respective clock cycles.

4.2 CONTROL STORE

Control within the CPU is provided by the generation of specifically formatted 64-bit control words. Each word is selectively obtained from a 2,048 location Read Only Storage (ROS) memory, which is called the control store. The 64-bit output from the control store forms the input to the Control Store Local Register (CR). Figure 4-2 illustrates the control store logic and includes the control store local register.

4.2.1 Control Store Local Register (CR)

The CR register retains firmware control words that emanate from control store for one complete firmware cycle (i.e., the control store output, or firmware word, is strobed into the local register at the positive transition of the clock input to the register, and it retains this data until new data is available at the positive transition of the next primary clock pulse). Outputs from the CR register are available for distribution throughout the CPU.



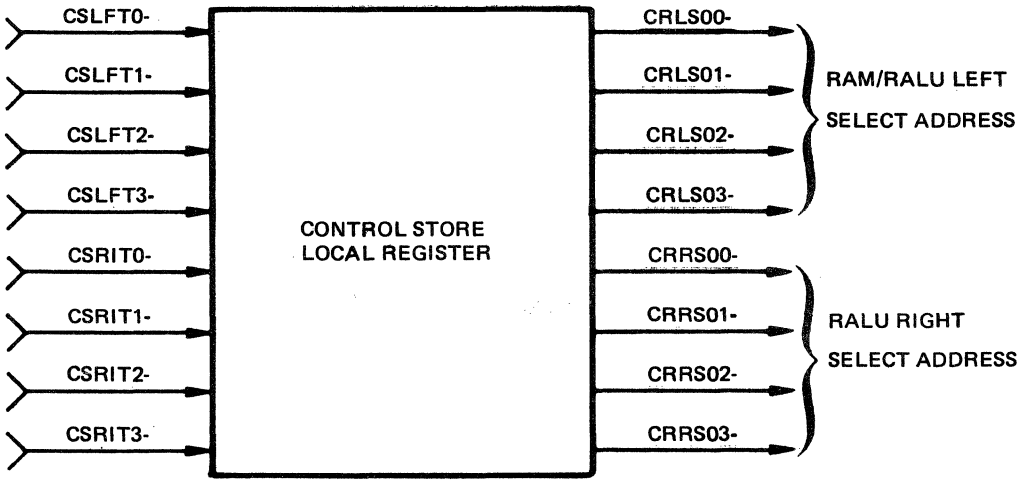
NOTES

1. INPUTS TO THESE SIGNALS ARE OBTAINED FROM THE OUTPUT OF THE RALU ADDRESSING LOGIC (REFER TO SUBSECTION 4.5)

2. THESE SIGNALS ARE NOT LOADED FROM THE OUTPUT OF THE CONTROL STORE. INSTEAD, THE OUTPUTS FROM CONTROL STORE (BITS 28 THROUGH 30) ARE FED TO THE RALU ADDRESSING LOGIC TO PARTICIPATE IN GENERATING RAM/RALU ADDRESSES.

Figure 4-2 Control Store Local Register

The Left and Right Select address bits (i.e., control store bits 1 through 3 and 5 through 7) are not stored directly into the local register. Instead, they are fed to the microprocessor (RALU) addressing logic where they are used in conjunction with the Select Modify bits (i.e., control store bits 28 through 30) to generate the 4-bit left and right select address inputs for the local register. These input signals to the local register and their corresponding output signals are:



4.2.2 Control Store Addressing

Control store addressing is primarily controlled by the next address generation logic (refer to subsection 4.3).

4.3 NEXT ADDRESS GENERATION (NAG) LOGIC

The NAG logic (see Figure 4-3) generates the next firmware address for the control store using one of three methods. All methods use bits 53 through 63 of the firmware word to form a tentative next address. These bits comprise the 11-bit NA field that can directly address any one of the 2,048 control store locations.

Method 1: This method uses as the alternate next address, bits 53 through 61 of the firmware word, in conjunction with logical Ones replacing bits 62 and 63, to form the 11-bit address.

Method 2: This method obtains the alternate next address from the CPU branch logic which generates numerous predefined addresses. The address generated is determined from a decode of the instruction register contents and other control logic.

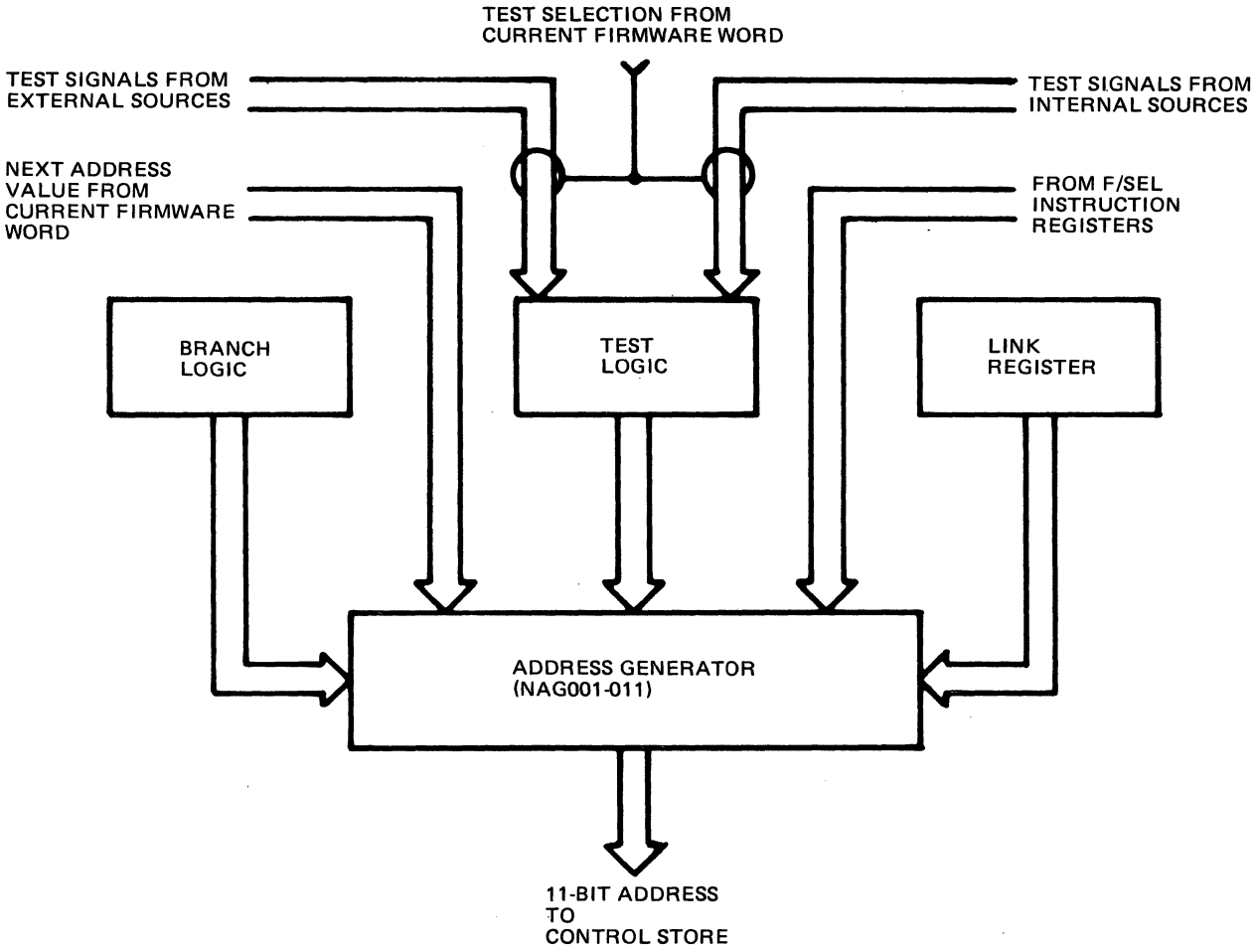


Figure 4-3 Next Address Generation Logic

Method 3: This method uses the 8-bit contents of the LINK register with three constant bits (as shown below) to form the alternate 11-bit address.

The next address generation logic can be divided into four areas: (1) test logic, (2) branch logic, (3) LINK register, and (4) address generator. These logic areas and selected fields of the firmware word make possible the above methods of generating the next firmware address.

4.3.1 Test Logic

The test logic receives inputs from sources both internal and external to the CPU, providing 69 hardware signals that can be used as test conditions. One of the 69 test signals is selected by the TC Field of the current firmware word to participate in generating the next firmware address. These test conditions are too numerous to catalog here, but are defined in Table 3-16.

The output from the test logic (hereafter referred to as the test signal) indicates whether or not the test condition is satisfied, and is fed directly to the address generator. The test signal is used by the address generator to determine whether to substitute the alternate next firmware address for the tentative value (CRNA01-11).

4.3.2 Branch Logic

The branch logic contains several branch PROMs that provide the next firmware address for major branch operations.

4.3.3 LINK Register (XL)

The XL register is an 8-bit firmware address register (not visible to software) that supplies eight bits of the next firmware address when a Link Branch (or subroutine return) type operation is selected; the most significant two bits of the address are forced to 01 and the least significant bit is forced to Zero.

4.3.4 Address Generator

The address generator provides the next firmware address for the control store. The precise manner used to generate the next address is determined by the BR Field of the current firmware word. This field specifies the type of branch operation being performed as a result of a specific test condition. The eight branch types that can be specified include two binary branches (X0 and XL) and six major branches (XA, XB, XR, XE, XW, and XF).

4.3.4.1 X0 Branch

The X0 branch type consists of both unconditional and conditional branches. The alternate next address is derived by performing a logical OR operation between the NA field and a value of 3 (hexadecimal).

4.3.4.2 XL Branch

The XL branch type returns control to the normal firmware sequence after execution of a firmware subroutine. The alternate next address is determined from the contents of the LINK register.

4.3.4.3 XA Branch

The XA branch type is used as the first step in analyzing each instruction. Formation of the alternate next address for XA branches depends on the instruction form.

4.3.4.4 XB Branch

The XB branch type is used to analyze the address syllable portion of the data descriptor for commercial type instructions.

4.3.4.5 XR Branch

The XR branch type is used to fetch an indirect address, perform indexing, read operand(s) from memory, or execute jump type or I/O type instructions.

4.3.4.6 XE Branch

The XE branch type completes the op-code decoding necessary to begin execution of the single- and double-operand instructions included in the CPU instruction repertoire.

4.3.4.7 XW Branch

The XW branch is used to store a result.

4.3.4.8 XF Branch

The XF branch type is used to exit the instruction currently being executed and return to the Instruction Fetch firmware.

4.4 MICROPROCESSOR

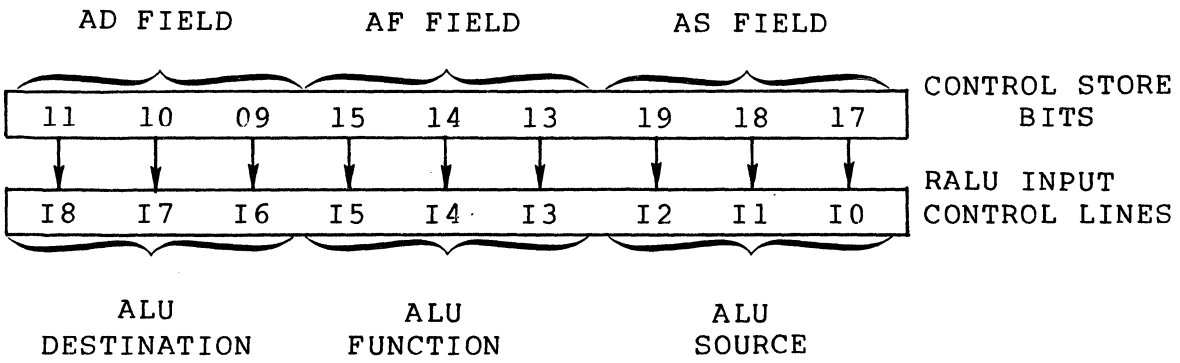
The microprocessor (see Figure 4-4), hereafter referred to as the Register File and Arithmetic Logic Unit (RALU), performs arithmetic, logical, and shift operations as directed by the current firmware word.

The major logic areas of the RALU are:

- Microinstruction decoder
- Register file
- Q register
- Data source selector
- Arithmetic/Logic Unit (ALU)
- Data output selector
- Shift logic.

4.4.1 Microinstruction Decoder

Control store generates a microprocessor instruction by encoding the RALU input control lines (I0 through I8) with the microinstruction code necessary to process data through the RALU. This is accomplished using the AD, AF, and AS fields of the current firmware word as shown below:



4.4.2 Register File

The Register File (RF) is the operand storage facility of the RALU, consisting of 16 registers; two working registers, seven data registers, and seven base registers. Each register is 20 bits wide.

The two working registers (D0 and B0) are not software-visible, and provide a temporary storage facility when manipulating data during firmware operations. These registers are located in RF locations 0 and 8, respectively.

The seven data registers (D1 through D7) are software-visible, representing software registers R1 through R7. These registers are located in RF locations 1 through 7.

The seven base registers (B1 through B7) are software-visible, and are located in RF locations 9 through F.

Of the 16 register file locations, any two can be simultaneously accessed by the firmware, providing dual operands in a single firmware step. The contents of the selected locations (or location, if the addresses are the same) appear as the left and right outputs from the register file. The left output may be routed directly (via the data output selector) to the internal bus source selector, while both the left and right outputs are available as sources for the J and K ports.

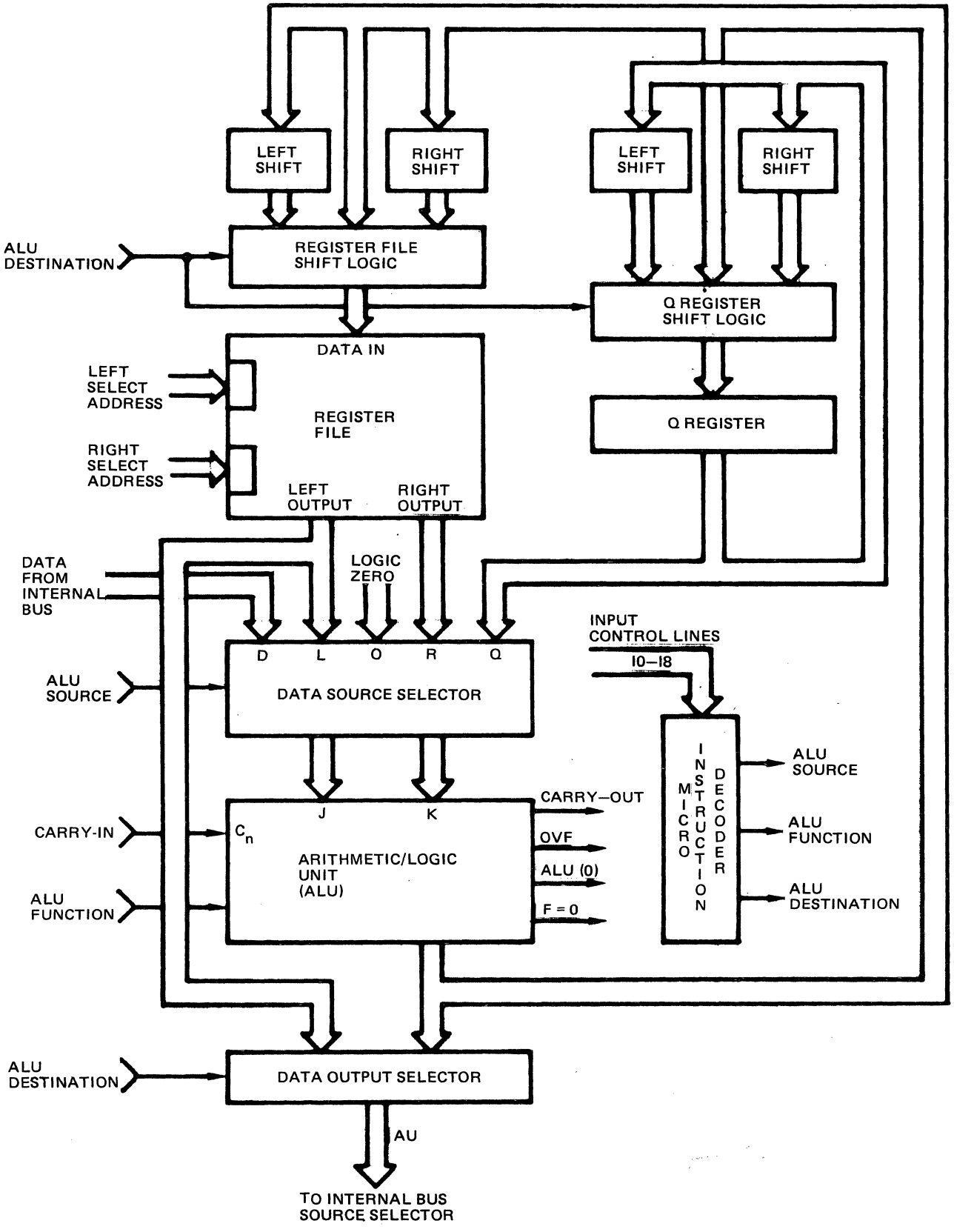


Figure 4-4 Microprocessor Functional Overview

4.4.3 Q Register

The Q register can function as a scratch pad or an extension of any register file register during shift operations and normal transfers of data. This allows, for example, the retention of the least significant half of a double-length product during a multiply operation. Although the Q register is 20 bits wide, only the least significant 16 bits are used during double-precision shift operations.

4.4.4 Data Source Selector

The data source selector is a steering device for data within the RALU, and consists of separate multiplexers for the J and K inputs to the ALU. The sources that serve as inputs to these multiplexers include:

- Register file left and/or right output
- Q register
- Internal bus
- Logical Zero.

The J multiplexer can select any of the above, except the right register file output and the Q register, while the K multiplexer can select any but the internal bus; logical Zero can be selected by either the J or K multiplexer, but not both simultaneously.

4.4.5 Arithmetic/Logic Unit (ALU)

The ALU is the heart of the CPU, performing arithmetic, logical, and compare operations as directed by the firmware. The ALU has two inputs (J and K) that are sourced from the data source selector, and one 20-bit output that may be selected as an input to either the register file or the Q register, and is made available to the internal bus source selector via the data output selector (AU).

The ALU can perform 64 arithmetic and logical operations as directed by RALU input control lines I0 through I5 (see Table 4-1). These operations include the following features:

- Full Carry Look-Ahead
- Overflow Detection
- Result Sign Detection
- All Zeros Detection.

Other ALU outputs that can be tested and/or copied by the firmware include: (1) overflow, (2) carry out, and (3) ALU output equals Zero. These three test signals can relate to an entire 20-bit operation or only the least significant 16 bits.

Table 4-1 RALU Logical and Arithmetic Operations (Sheet 1 of 2)

LOGICAL OPERATIONS							
INSTRUCTION MODIFIER BITS							
I5	I4	I3	I2	I1	I0	GROUP	FUNCTION
0	1	1	0	0	0	OR	L V Q
0	1	1	0	0	1	OR	L V R
0	1	1	0	1	0	Pass	Q
0	1	1	0	1	1	Pass	R
0	1	1	1	0	0	Pass	L
0	1	1	1	0	1	OR	D V L
0	1	1	1	1	0	OR	D V Q
0	1	1	1	1	1	Pass	D
1	0	0	0	0	0	AND	L & Q
1	0	0	0	0	1	AND	L & R
1	0	0	0	1	0	Zero	0
1	0	0	0	1	1	Zero	0
1	0	0	1	0	0	Zero	0
1	0	0	1	0	1	AND	D & L
1	0	0	1	1	0	AND	D & Q
1	0	0	1	1	1	Zero	0
1	0	1	0	0	0	Mask	$\overline{L}$ & Q
1	0	1	0	0	1	Mask	$\overline{L}$ & R
1	0	1	0	1	0	Pass	Q
1	0	1	0	1	1	Pass	R
1	0	1	1	0	0	Pass	L
1	0	1	1	0	1	Mask	$\overline{D}$ & L
1	0	1	1	1	0	Mask	$\overline{D}$ & Q
1	0	1	1	1	1	Pass	0
1	1	0	0	0	0	EX-OR	L $\not\vee$ Q
1	1	0	0	0	1	EX-OR	L $\not\vee$ R
1	1	0	0	1	0	Pass	Q
1	1	0	0	1	1	Pass	R
1	1	0	1	0	0	Pass	L
1	1	0	1	0	1	EX-OR	D $\not\vee$ L
1	1	0	1	1	0	EX-OR	D $\not\vee$ Q
1	1	0	1	1	1	Pass	D
1	1	1	0	0	0	EX-NOR	$\overline{L \not\vee Q}$
1	1	1	0	0	1	EX-NOR	$\overline{L \not\vee R}$
1	1	1	0	1	0	Invert	$\overline{Q}$
1	1	1	0	1	1	Invert	$\overline{R}$
1	1	1	1	0	0	Invert	$\overline{L}$
1	1	1	1	0	1	EX-NOR	$\overline{D \not\vee L}$
1	1	1	1	1	0	EX-NOR	$\overline{D \not\vee Q}$
1	1	1	1	1	1	Invert	$\overline{D}$

DEFINITIONS

X = don't care
+ = addition
- = subtraction
& = logical AND
V = logical OR
$\not\vee$ = exclusive OR
D = ALU input from internal bus
L = ALU input from L latches
R = ALU input from R latches
Q = ALU input from Q register

Table 4-1 RALU Logical and Arithmetic Operations (Sheet 2 of 2)

FUNCTION			SOURCE			ARITHMETIC OPERATIONS			
INSTRUCTION			MODIFIER BITS			C_n = 0 (Low)		C_n = 1 (High)	
I5	I4	I3	I2	I1	I0	GROUP	FUNCTION	GROUP	FUNCTION
0	0	0	0	0	0	Add	L + Q	Add + 1	L + Q + 1
0	0	0	0	0	1	Add	L + R	Add + 1	L + R + 1
0	0	0	0	1	0	Pass	Q	Increment	Q + 1
0	0	0	0	1	1	Pass	R	Increment	R + 1
0	0	0	1	0	0	Pass	L	Increment	L + 1
0	0	0	1	0	1	Add	D + L	Add + 1	D + L + 1
0	0	0	1	1	0	Add	D + Q	Add + 1	D + Q + 1
0	0	0	1	1	1	Pass	D	Increment	D + 1
0	0	1	0	0	0	Subtract	Q - L - 1	Subtract	Q - L
0	0	1	0	0	1	Subtract	R - L - 1	Subtract	R - L
0	0	1	0	1	0	Decrement	Q - 1	Pass	Q
0	0	1	0	1	1	Decrement	R - 1	Pass	R
0	0	1	1	0	0	Decrement	L - 1	Pass	L
0	0	1	1	0	1	Subtract	L - D - 1	Subtract	L - D
0	0	1	1	1	0	Subtract	Q - D - 1	Subtract	Q - D
0	0	1	1	1	1	One's complement	- D - 1	Two's complement	- D
0	1	0	0	0	0	Subtract	L - Q - 1	Subtract	L - Q
0	1	0	0	0	1	Subtract	L - R - 1	Subtract	L - R
0	1	0	0	1	0	One's complement	- Q - 1	Two's complement	- Q
0	1	0	0	1	1	One's complement	- R - 1	Two's complement	- R
0	1	0	1	0	0	One's complement	- L - 1	Two's complement	- L
0	1	0	1	0	1	Subtract	D - L - 1	Subtract	D - L
0	1	0	1	1	0	Subtract	D - Q - 1	Subtract	D - Q
0	1	0	1	1	1	Decrement	D - 1	Pass	D

DEFINITIONS

+ = addition
- = subtraction
& = logical AND
V = logical OR
V = exclusive OR

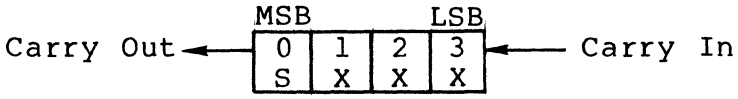
D = ALU input from internal bus
L = ALU input from left output of register file
R = ALU input from right output of register file
Q = ALU input from Q register

4.4.5.1 Look-Ahead Logic

The RALU consists of five stages, with each stage providing a Carry Generate and a Carry Propagate signal. These signals are used in conjunction with an external carry generator (i.e., external to the RALU) to form the look-ahead logic. This logic determines whether or not the Carry Input (CN) to the RALU is propagated through each stage. The determination is based on an interpretation of the input operand rather than awaiting the ripple carry through each stage.

4.4.5.2 Overflow Logic

The overflow logic is internal to the RALU and is used in conjunction with the sign bit during signed arithmetic operations. Overflow reflects the results of an exclusive OR operation between the carry-in and carry-out of the sign bit, and occurs when the result of any addition or subtraction requires more bit positions than the ALU can accommodate. For example, assume that a 4-bit system (three bits plus the sign bit) is used, consisting of one arithmetic unit where the most significant bit is a Zero for positive numbers and a One for negative numbers as shown below:



S = Sign

0: Positive
1: Negative

X = 0/1

Under the preceding conditions, only a maximum value of +7 and a minimum value of -8 is possible. Overflow is detected by comparing the carry-in and carry-out of the sign bit via an exclusive OR operation that is internal to the RALU; if the carries disagree, the overflow signal is not used during logical operations.

4.4.5.3 Zero Detection Logic

The Zero Detect outputs from the least significant four stages of the RALU are effectively tied together, enabling detection of a logical Zero condition for the 20-bit system.

4.4.6 Data Output Selector

The data output selector selects whether the left output from the register file or the ALU output is made available at the output (AU) of the RALU. This output can be used directly in the current firmware step, and is available to the internal bus source selector for distribution to other CPU elements via the internal bus. The output selector also specifies whether the ALU output is to be copied into the register file. If this is done, the register file location modified is the one that supplied the right output to the data input selector.

NOTE

Write operations can be performed only into the right side of the register file.

4.4.7 Shift Logic

The RALU shift logic optionally shifts the output of the ALU (or the ALU and the Q register) before delivering the results to the register file (or the register file and the Q register). The shift logic is capable of shifting single- or double-word operands left or right by one bit position as directed by RALU input control lines I6 through I8 (see Table 4-2).

Table 4-2 RALU Shift Operations

DESTINATION			SHIFT OPERATIONS			
INSTRUCTION MODIFIER BITS			REGISTER FILE FUNCTION		Q REGISTER FUNCTION	
I8	I7	I6	SHIFT	LOAD	SHIFT	LOAD
0	0	0	X	X	None	ALU
0	0	1	X	X	X	X
0	1	0	None	ALU	X	X
0	1	1	None	ALU	X	X
1	0	0	Left	ALU	Left	Q Reg.
1	0	1	Left	ALU	X	X
1	1	0	Right	ALU	Right	Q Reg.
1	1	1	Right	ALU	X	X

X = don't care

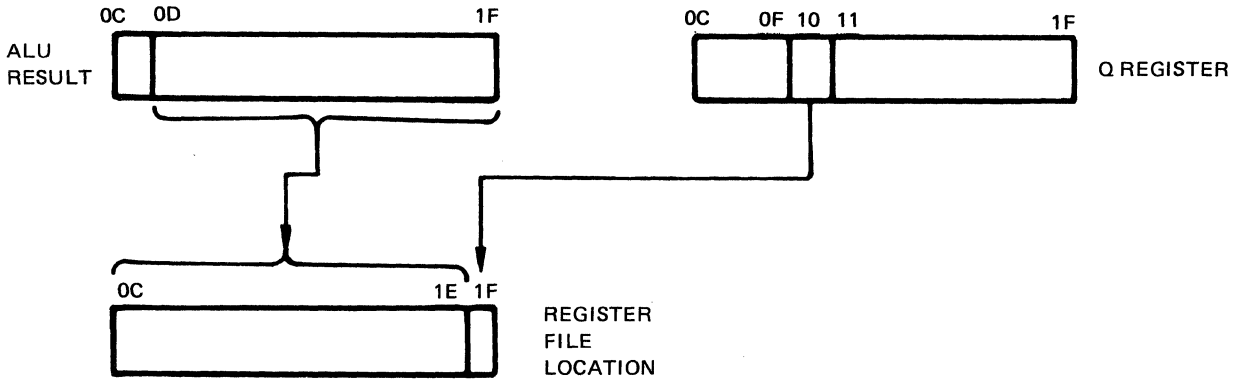
In shift operations, the bit shifted into the vacated bit position is designated as SHIN (shift input), and is controlled by three flip-flops: SHIN1, SHIN2, and MISC (refer to subsection 4.8). The SHIN function is selected as follows:

MISC	SHIN1	SHIN2	SHIN
0	0	0	Internal Bus bit 10
0	0	1	Internal Bus $\overline{\text{bit 10}}$
0	1	0	Zero
0	1	1	Q register bit 1F*
1	0	0	XB register bit 1
1	0	1	Y register bit 7
1	1	0	Zero
1	1	1	Q register bit 1F*

*During shift right operations; otherwise undefined.

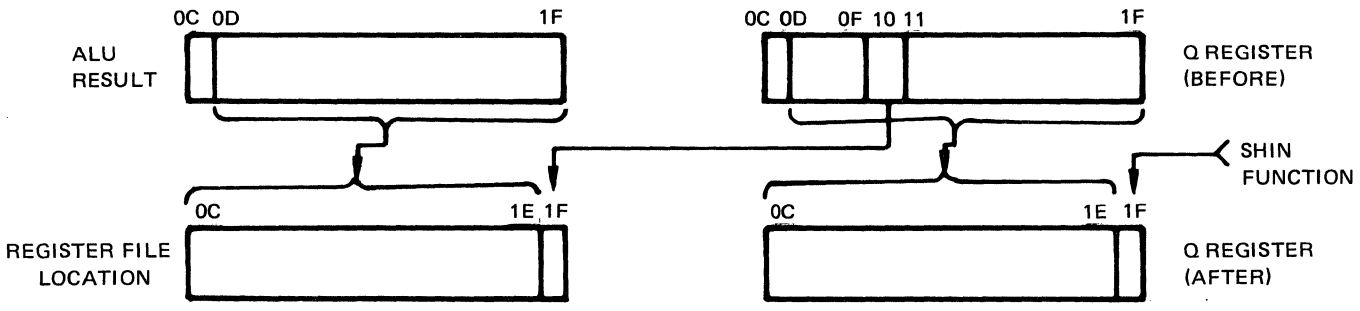
4.4.7.1 Single Left Shift

Bits 0D through 1F of the ALU result are placded in bits 0C through 1E of the selected register file location; bit 1F of the selected register file location receives a copy of Q register bit 10.



4.4.7.2 Double Left Shift

Bits 0D through 1F of the ALU result are placed in bits 0C through 1E of the selected register file location; bit 1F of the selected register file location receives a copy of Q register bit 10. Q register bits 0D through 1F are placed in Q register bits 0C through 1E; Q register bit 1F receives a copy of the SHIN function.

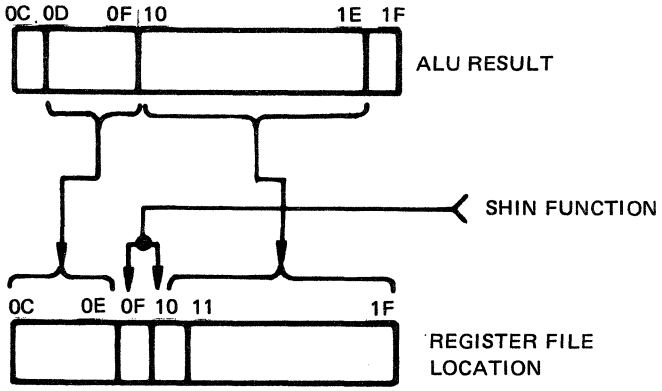


Conceptually, the rightmost 16 bits of the ALU result are concatenated with the rightmost 16 bits of the Q register and shifted left one bit position with the SHIN function shifted in

on the right. The result is placed in the rightmost bit positions of the register file location and the Q register, respectively.

4.4.7.3 Single Right Shift

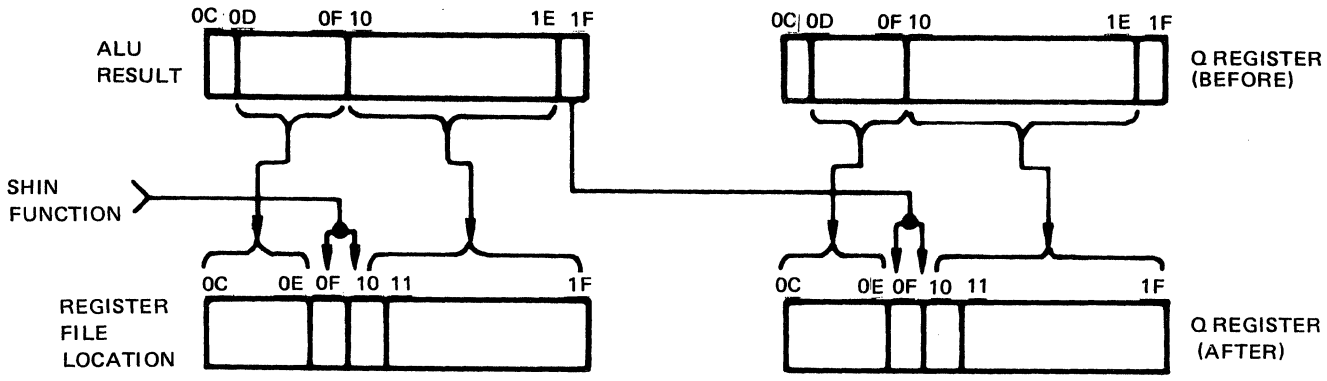
Bits 10 through 1E of the ALU result are placed in bits 11 through 1F of the selected register file location; bit 10 of the selected register file location receives a copy of the SHIN function. Bits 0D through 0F of the ALU result are placed in bits 0C through 0E of the selected register file location; bit 0F of the selected register file location receives a copy of the SHIN function.



Conceptually, the rightmost 16 bits of the ALU result are shifted right one bit position with the SHIN function shifted in on the left. The result is placed in the 16 rightmost bit positions of the register file location.

4.4.7.4 Double Right Shift

Bits 10 through 1E of the ALU result are placed in bits 11 through 1F of the selected register file location; bit 10 of the selected register file location receives a copy of the SHIN function. Q register bits 10 through 1E are placed in Q register bits 11 through 1F; bit 1F of the ALU result is placed in Q register bit 10. Bits 0D through 0F of the ALU result are placed in bits 0C through 0E of the selected register file location; bit 0F of the selected register file location receives a copy of the SHIN function. Q register bits 0D through 0F are placed in Q register bits 0C through 0E; bit 1F of the ALU result is placed in Q register bit 0F.



Conceptually, the least significant 16 bits of the ALU result and the least significant 16 bits of the Q register are concatenated, shifted right one bit position with the SHIN function filling the most significant bit, and the result placed in the least significant 16 bits of the register file location and the Q register, respectively.

4.5 RALU ADDRESSING

The RALU addressing logic can select any one of the 16 registers located within the register file as directed by the LS, RS, and SM fields of the current firmware word. Eight of these registers are directly addressable by the LS and RS fields when the SM field equals zero (see Tables 3-1 and 3-2). Each of the 16 registers can be addressed by using the SM field to modify the LS or RS register address.

The LS, RS, and SM fields are each configured as 3-bit codes that are allocated within the control store as follows:

- LS: bits 1 through 3
- RS: bits 5 through 7
- SM: bits 28 through 30

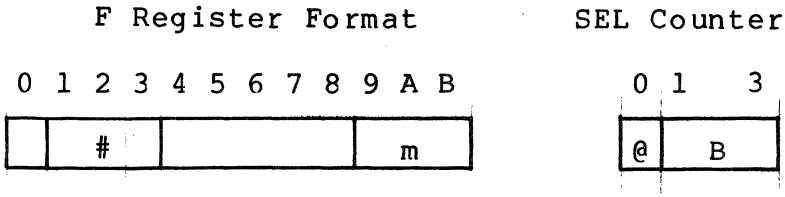
These 3-bit fields generate hexadecimal addresses that deliver the contents of a selected register (or registers) to the left and/or right output of the register file. The register selection performed by the above address fields is defined in Table 3-13.

The LS field also provides addressing for the Random Access Memory (RAM) which is external to the RALU (refer to subsection 4.9.1).

Write operations into the register file can be performed only to a register that is accessed by the right select address lines.

The SM field selects data from one of two sources to modify the LS or RS address:

1. The following F register fields (refer to subsection 1.6.2):



2. One of the following three constants:

   - D: (1,1,0,1)
   - E: (1,1,1,0)
   - F: (1,1,1,1)

These signals are ANDed with the applicable LS or RS field, providing the input signals to the local register for the left and right select addresses.

4.6 INTERNAL BUS

The internal bus (BIXX0C through BIXX1F) provides a 20-bit wide data path that transfers data among elements of the CPU as directed by the BI Field of the current firmware word (see Tables 3-9 through 3-11). The internal bus receives inputs (via the internal bus source selector) from 14 sources, and makes its data available to 12 destinations (see Figure 4-5). These sources and destinations are:

Internal Bus Sources

1. Constant Generators
2. Megabus Data Buffer (BD)
3. Megabus Interrupt Register (RUP)
4. Megabus Procedure 1 Buffer (BP1) or Procedure 2 Buffer (BP2)
5. Control Panel
6. H Register
7. Indicator Register (I)
8. Status/Security Register (S)

9. Program Counter (P), or
   Memory Address Register (Y), or
   MMU Output (Physical Address)
   } Via Address Bus (BA)

10. RAM

11. Hexadecimal Decoder

12. Trap Status (Z-word)

13. Register File (left) Output, or
    ALU Output (AU)

14. Bootload PROM.

Internal Bus Destinations

1. Program Counter (P)
2. Memory Address Register (Y)
   } → Group 1

3. Indicator Register (I) → Group 2

4. RAM → Group 3

5. LINK Register (XL)
6. Instruction Register (F/SEL)
7. Control Panel Display
8. H Register
9. Status/Security Register (S)
10. Megabus Data Lines
11. ALU Data Source Selector
12. Miscellaneous Control Elements
    (Control Flip-Flops, MMU, Test
    Logic, etc.)

→ Group 4

Firmware can select one element (or combinations of several elements) as an internal bus source, and deliver these data to one destination in each of the four groups listed above. The combinations of internal bus elements that are available as internal bus sources are:

1. Two copies of ALU output (bits 0C through 0F) and three interleaving Zeros

2. Eight copies of H register (bit 18) and ALU output (bits 18 through 1F)

3. H register (bits 1C through 1F) and contents of Megabus data buffer.

4. H register (bits 1C through 1F) and contents of Megabus procedure buffer

5. H register (bits 1C through 1F) and control word from control panel

6. H register (bits 10 through 17) right justified and sign extended to 16 bits.

4.7 CPU REGISTERS

The CPU registers, except those contained within the RALU, are described in the following subsections.

4.7.1 Indicator Register (I)

The CPU indicators can be loaded using the firmware controls described in Table 3-11.

4.7.1.1 Arithmetic Indicators

Two indicators can be loaded with the results of arithmetic operations in the CPU; the Overflow (OV) indicator and the Carry (C) indicator. The available inputs to these indicators are:

1. OV Indicator

   - OVFL (ALU overflow signal)
   - Result of exclusive OR operation between internal bus bits 10 and 11

2. C Indicator

   - CRY (ALU carry signal)
   - Internal bus bit 10
   - Internal bus bit 1F
   - Q register bit 1F just prior to right shift in the current firmware step.

4.7.1.2 Bit Test Indicator

Inputs available to the bit test indicator are:

- AUZERO (ALU zero detect signal)
- Internal bus bit 10.

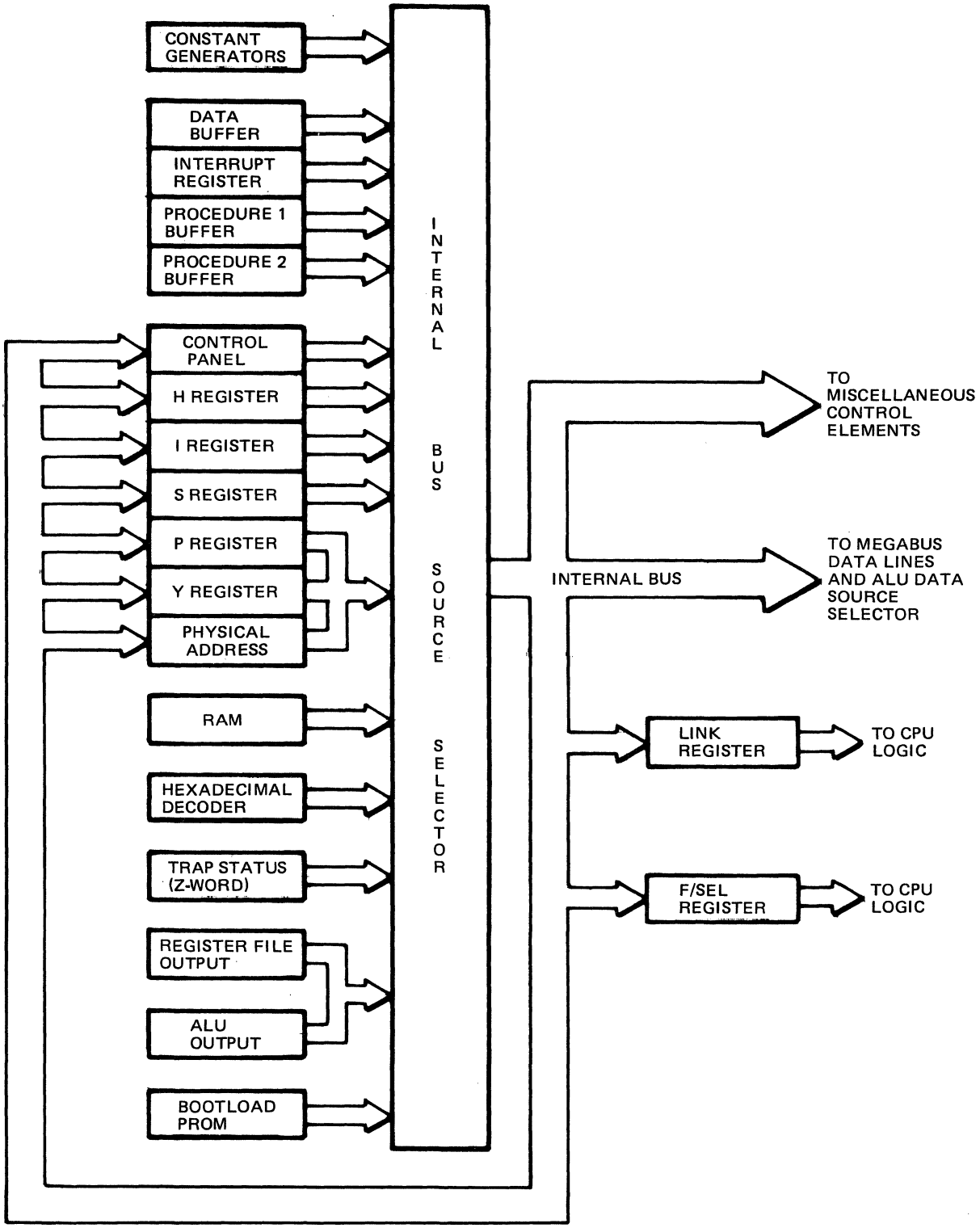


Figure 4-5 Internal Bus Sources and Destinations

4.7.1.3 Input/Output Indicator

This indicator stores the results of the last I/O instruction performed by the CPU. This is accomplished by making the Megabus acknowledge signal available at the input to this indicator. If the I/O instruction is accepted, the indicator is set; otherwise, it is cleared.

4.7.1.4 Comparison Indicators

Three indicators store the results of the last compare operation performed in the CPU; the Greater Than (G), Less Than (L), and Unlike Signs (U) indicators. The inputs available to these indicators are:

1. G Indicator
   - Internal bus bit 10 is zero and the ALU output (16 bits) is not zero
   - Complement of the SIGN flip-flop
2. L Indicator
   - Internal bus bit 10
   - ALU result bit 0C
   - SIGN flip-flop
3. U Indicator
   - Internal bus bit 10.

4.7.2 LINK Register (XL)

Refer to subsection 4.3.3 for a description of the XL register.

4.7.3 Counter Type Registers

Counter type registers are versatile in that they are alternately used as a 4-bit counter or as a 4-bit storage register within their respective circuit configurations.

4.7.3.1 Counter Register (CTR)

The CTR register (RCTR0F through RCTR3F) counts the number of procedure words fetched during instruction execution, and in conjunction with the program counter allows the Trap Handler software to reconstruct the instruction word address in the event an exception condition is detected in the CPU. To accomplish this, firmware initializes the register to a count of 1.

4.7.3.2 Select Register (SEL)

The SEL register (RSEL0F through RSEL3F) generally holds the least significant hexadecimal digit of the current instruction. At times, it is used to count repetitive actions in Shift, Multiply, or Divide operations. Data are received over internal bus bits 1C through 1F. When the SEL register is used as a counter, it is decremented and its contents tested for zero.

4.7.3.3 Byte Indexing Register (XB)

The XB register (RXB0FF through RXB3FF) stores bits shifted out of RALU data registers during half-word, digit, or bit indexing operations. At the start of each instruction, it is cleared to zero. Its output is sent directly to a hexadecimal decoder, and is also available to the internal bus for trap reporting.

4.7.4 Instruction Register (F)

The F register (RF00FF through RF08FF) accepts and stores the most significant three hexadecimal digits of instructions from memory for execution in the CPU.

4.7.5 H Register

The H register (RH10FF through RH1FFF) is configured into two 8-bit segments that accept data directly from the internal bus. Its output is available to the internal bus source selector for delivery to the internal bus, but as its contents are deposited onto the bus, the least and most significant eight bits of the register are swapped.

4.7.6 Status/Security Register (S)

The S register (RS01FF, RS02FF, and RS10FF through RS15FF) retains the system status and security codes for use within the CPU. It is also used for comparing the priority level of an incoming interrupt request with the current CPU operating priority level (refer to subsection 4.11.2). This compare activity is performed to deny acceptance of an incoming request when the request level is equal to or lower than the current CPU priority level. The contents of this register are controlled entirely by firmware, except for the CPU channel number (bits 8 and 9) which are switch controlled.

4.7.7 P Register (Program Counter)

The P register (RP03CF through RP22CF) consists of five 4-bit counters capable of being preset that retain and increment the current instruction address originally obtained from the internal bus.

4.7.8 Memory Address Register (Y)

The Y register (RY03CF through RY22CF) provides operand addresses (via the address bus) for memory or a peripheral device. Its architecture and functionality are basically the same as the P register (refer to subsection 4.7.7). Also included is a logical switching network which, under firmware control, generates a 16-bit address field by isolating the most significant byte (bits 3 through 6) from the 20-bit address. The 16- and 20-bit address fields are defined in Section I as the Short Address Form (SAF) and the Long Address Form (LAF), respectively.

4.7.9 Megabus Registers

The Megabus registers consist of three temporary storage buffers that receive procedure and data words (over the Megabus) from memory. Refer to subsection 4.10.6 for a description of these buffers.

4.7.10 Interrupt Register

Refer to subsection 4.11.3 for a description of the interrupt register.

4.7.11 M Register

The M register accepts pertinent RAM data (see Table 4-3) as the RAM is being updated, and since this data is not immediately available from the RAM, the M register delivers it to the test logic and next address generation logic for instantaneous action. This data is obtained from several sources and loaded into the M register as directed by the GP Field of the current firmware word. These sources include:

- Y register bit 15
- F register bit 0A
- SEL register bits 0 and 2
- H register bits 1A and 1B
- ZERO flip-flop
- AUZERO (ALU Zero Detect signal).

Table 4-3 M Register Format

BIT	SIGNAL	DESCRIPTION
0	RMTRAC	Trace Trap Enable
1	RMSCI1	S1 memory operand length is double-word
2	RMSCI2	S2 memory operand length is double-word
3	RMSCI3	S3 memory operand length is double-word
4	RMSQB6	B6 is in Stack or Queue mode
5	RMSQB7	B7 is in Stack or Queue mode
6	GOTSPU	SIP is part of the processor complex
7	GOTBPU	CIP is part of the processor complex

4.8 CPU CONTROL FLIP-FLOPS

The control flip-flops receive inputs from sources both internal and external to the CPU, permitting modification of firmware actions based on the results of operations performed through the system.

4.8.1 SIGN Flip-Flop

The SIGN flip-flop provides temporary storage of control information during instruction execution, and may be set from:

- Internal bus bit 0
- Internal bus bit 4
- Internal bus bit 19
- One
- Zero.

4.8.2 MISC Flip-Flop

The MISC flip-flop provides temporary storage of control information during instruction execution, and is one of the flip-flops used to select the SHIN function for RALU shift operations (refer to subsection 4.4.6). The MISC flip-flop may be set from:

- Complement of internal bus bit 19
- Internal bus bits 4 through 9 equal to Zero
- CRY - ALU Carry signal
- ACK - Megabus Acknowledge signal
- PROV - MMU Protection Violation signal
- Zero
- One.

4.8.3 SHIN1 Flip-Flop

The SHIN1 flip-flop is primarily used to select the SHIN function for RALU shift operations (refer to subsection 4.4.6), and may be set from:

- I(B) - I register Bit Test indicator
- Zero
- One.

4.8.4 SHIN2 Flip-Flop

The SHIN2 flip-flop is primarily used to select the SHIN function for RALU shift operations (refer to subsection 4.4.6), and may be set from:

- Complement of SIGN flip-flop
- Zero
- One.

4.8.5 ZERO Flip-Flop

The Zero flip-flop is primarily used for temporary storage of an ALU Zero Result condition, and may be set from:

- ALU Zero Detect signal
- QLT active flip-flop from control panel - equals One only if the last CPU Quality Logic Test (QLT) failed
- Zero
- One.

4.8.6 WRAP Flip-Flop

The WRAP flip-flop facilitates the checking of address-arithmetic firmware to detect attempts to exceed the 20-bit capacity of the address registers. If WRAP is On, any access to the Megabus (read request or write operation) will result in the transmission of an illegal address or I/O channel number. This action results in no response via the Megabus, which is interpreted as an "unavailable resource." The WRAP flip-flop may be set from the inequality of the ALU Carry signal (CRY) and the SIGN flip-flop.

4.8.7 NEWXR Flip-Flop

The NEWXR flip-flop distinguishes between reentrant invocations of the XR "splatter" branch. This flop is set when the SEL register is loaded from the internal bus (e.g., during instruction fetch operations); it is cleared when a branch is performed to XR, XE, XW, or XF, and when the WRAP flip-flop is set or cleared.

4.8.8 ACK Flip-Flop

The ACK flip-flop remembers whether the most recent Megabus action was accepted or rejected: if ACK is On, the action was accepted; if ACK is Off, the action was rejected.

4.8.9 YELLOW and PARER Flip-Flops

The YELLOW and PARER flip-flops signal detection of an error in memory or on the Megabus. YELLOW remembers whether at least one data error was corrected by the memory Error Detection And Correction (EDAC) hardware since the last interrogation of this flip-flop; YELLOW is cleared each time it is interrogated by the firmware. PARER remembers whether the most recent Megabus buffer reference (BD or BP) reported either a Megabus parity error or a data error not correctable by memory EDAC hardware. Unless the control panel is in Load, Read, or Write mode, the setting of PARER forces the firmware to control store location 000 for suitable trap generation.

4.8.10 EXTRAP, INTBSY, and TICK Flip-Flops

The EXTRAP, INTBSY, and TICK service request flip-flops are set by hardware to signal a requirement for a break in firmware flow.

EXTRAP is true when one or more external processors (CIP or SIP) has detected a trap condition. EXTRAP becomes false when all external processors with trap conditions have delivered their trap words.

INTBSY is set when an external interrupt of high enough priority is received and accepted by the hardware. No further interrupt, regardless of priority, can be accepted until firmware services buffer RUP, reloads the level field in the S register, and clears INTBSY.

TICK is set every 8-1/3 milliseconds by a crystal-controlled oscillator, signaling the need for service of the YELLOW logic, the RTC and/or WDT, and the control panel.

4.8.11 LOAD, TRAFFIC, and PANOK Flip-Flops

These flip-flops communicate control information between frimware and the operator.

The LOAD flip-flop can be set and cleared both by the operator and by firmware. During system startup operations, LOAD is normally set by the operator and, when bootload action is completed, cleared by firmware. Thereafter, this flip-flop usually remains OFF, but is sometimes set briefly by firmware as a means of preventing a trap to location 000 when a Megabus cycle is addressed to a possibly unavailable resource.

The TRAFFIC flip-flop is loaded by firmware to control the corresponding indicator on the control panel, but is held OFF by hardware unless the control panel is in Run mode. The TRAFFIC flip-flop may be set from the complement of the ZERO flip-flop, which indicates whether or not the instruction op-code just fetched from memory is an HLT (Halt).

The PANOK flip-flop synchronizes the servicing of operator requests. It is set to Zero whenever the CLEAR or EXECUTE pushbutton is depressed and when, in register-change mode, a hexadecimal key is depressed. This flip-flop is set to One by the firmware that services the request, and is used to prevent multiple servicing of a single key-stroke.

4.8.12 EFFRING, NONPROC, NOCHECK, SEGERR, and PROV Controls

These signals and flip-flops support normal MMU operations, permit temporary alteration of access rules, and report errors detected by the MMU.

EFFRING is a two-bit register containing the effective ring number, which the MMU uses to determine the degree of privilege appropriate to the current instruction, and against which memory access requests are tested. Firmware loads EFFRING from the S register RING field at the start of each instruction. EFFRING is modified to decrease its privilege level whenever, in the course of formulating an address, it uses data that might have been generated by a less privileged program.

NONPROC establishes a temporary change in the rules of access. Memory references which use the P register as the address source normally require "Execute" permission; when NONPROC has been set, they require only "Read" permission.

NOCHEK establishes a temporary suspension of the rules of access (it does not affect the mapping of segmented virtual addresses to physical, nor the detection of illegal, non-existent addresses). The intent of this functionality is to remove restrictions on memory access by system firmware (interrupt and trap handlers, RTC/WDT service, panel routines, etc.).

SEGERR signals that the MMU has detected an error in a virtual address; the referenced segment is not valid, or its size has been exceeded, or a protection violation has been detected. If SEGERR occurs during a memory reference, it causes the transmission of an illegal physical address. This action results in no response via the Megabus, which is interpreted as an unavailable resource. If no memory reference or access-rights test is requested, signal SEGERR is ignored.

PROV signals that the MMU has detected a protection violation (failure of access-rights check) on an otherwise legal address (i.e., an address in a valid segment and within the segment size). If a protection violation occurs during a memory reference, the PROV flip-flop is locked in the set state until cleared by the firmware function NOCHEK (this function is normally issued by the trap-generation firmware). If a protection violation occurs during a firmware step that explicitly requests an access-rights test, the next firmware step may copy PROV to the MISC control flip-flop.

4.9 MISCELLANEOUS CPU HARDWARE

The following subsections describe those CPU elements not previously defined under one of the major CPU logic areas.

4.9.1 Random Access Memory (RAM)

The RAM consists of sixteen 20-bit auxiliary storage registers that the CPU uses as work areas, and to maintain selected system status conditions. Addressing the RAM for a read or write operation is performed by the Left Select portion of the RALU addressing logic (refer to subsection 4.5). However, the actual RAM read or write operation is performed as directed by the DI Field of the current firmware word.

The organization of the RAM is illustrated in Figure 4-6.

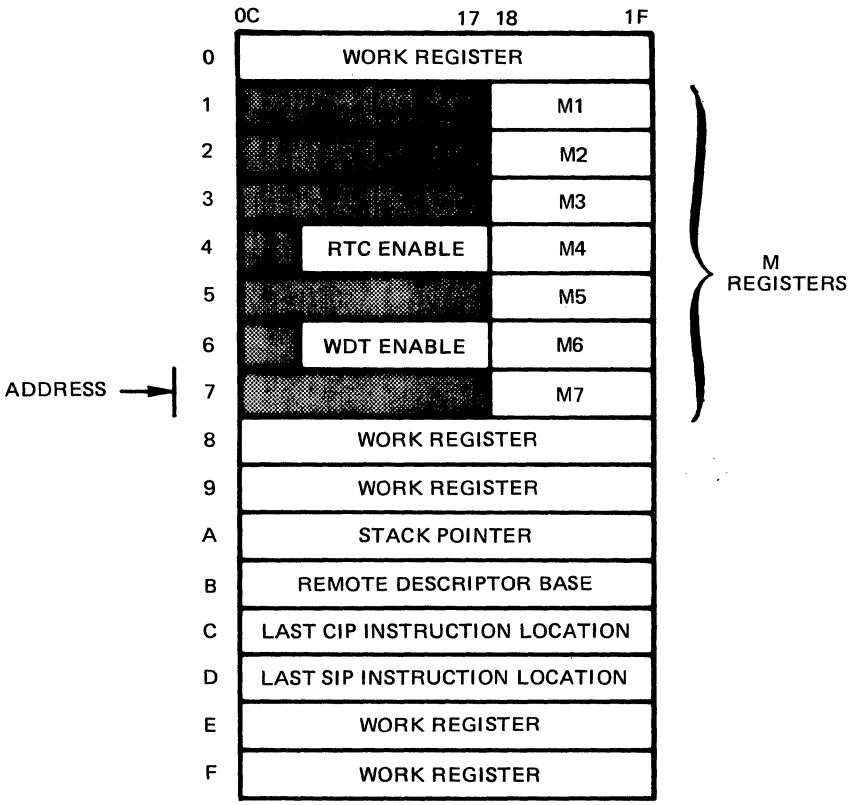


Figure 4-6 RAM Format

4.9.1.1 RAM Location 0

This RAM location contains the instruction word to be reported when a trap occurs.

4.9.1.2 RAM Locations 1 through 3

Bits 0C through 17 of these RAM locations are unused; bits 18 through 1F contain software mode registers M1 through M3.

4.9.1.3 RAM Location 4

Bits 10 through 17 of this RAM location contain the mode information for enabling the Real Time Clock (RTC); bits 18 through 1F contain software mode register M4.

4.9.1.4 RAM Location 5

Bits 0C through 17 of this RAM location are unused; bits 18 through 1F contain software mode register M5.

4.9.1.5 RAM Location 6

Bits 10 through 17 of this RAM location contain the mode information for enabling the Watch Dog Timer (WDT); bits 18 through 1F contain software mode register M6.

4.9.1.6 RAM Location 7

Bits 0C through 17 of this RAM location are unused; bits 18 through 1F contain software mode register M7.

4.9.1.7 RAM Location 8

This RAM location contains a pointer to the next word of procedure.

4.9.1.8 RAM Location 9

This RAM location is unused.

4.9.1.9 RAM Location A

This RAM location contains a stack pointer.

4.9.1.10 RAM Location B

This RAM location contains the Remote Descriptor Base (RDB) register.

4.9.1.11 RAM Location C

This RAM location contains a pointer to the most recently accepted Commercial Instruction Processor (CIP) instruction.

4.9.1.12 RAM Location D

This RAM location contains a pointer to the most recently accepted Scientific Instruction Processor (SIP) instruction.

4.9.1.13 RAM Location E

This RAM location is unused.

4.9.1.14 RAM Location F

This RAM location contains a pointer to the next word of procedure.

4.9.2 Bootload PROM

The bootload PROM actually consists of four 2K PROMs, and is a standard feature of the CPU. Addressing for the boot PROM is provided by bits 14 through 22 of the address bus. The 20-bit output from the boot PROM is delivered to the internal bus either as directed by the BS Field of the current firmware word or when the Load pushbutton on the control panel is depressed.

The bootload operation has the following modes of operation, depending on the contents of the program counter (P):

P CONTENTS	OPERATION
0000	Executes Basic Logic Test (BLT) Reads one physical record from channel 010 (hexadecimal) into memory, startng at location 0100 (hexadecimal); branches to 0100 (hexadecimal) The preceding is the default Bootload procedure*
0002	Does not execute BLT Reads one physical record into memory, starting at location 0100 (hexadecimal), using the channel number previously entered into register R1; branches to 0100 (hexadecimal)
0004	Does not execute BLT Reads one physical record into memory, starting at the address entered into register B1, using the channel number entered into register R1; branches to the address entered into register R1

*This procedure is used when the control panel is locked. If the control panel is unlocked, procedure will halt after the BLT unless Run (R) pushbutton is depressed.

The devices supported and the Boot record file formats are as follows:

DEVICES	RECORD FORMAT
Diskette	Data portion (128 bytes) of track 0, sector 0 (first sector)
Card	The contents of the first card; 80 bytes punches in Bootload format
Paper Tape (ASR)	One record of 256 bytes or less, starting with the character that follows the first non-NUL character and continuing to the first X-OFF or T-OFF character with no escaped data
Cartridge	Data portion (256 bytes) of track 0, sector 0
Magnetic Tape	One record of 256 bytes or less; the record must be the first after BOT

4.9.3 Address Bus

The address bus (MYAD03 through MYAD22) accepts addresses from the P and Y registers (refer to subsections 4.7.7 and 4.7.8, respectively) as directed by the BS Field of the current firmware word. The 20-bit output from the address bus is delivered to the Megabus for I/O and memory read or write operations, or to the internal bus for distribution in the CPU.

4.9.4 Memory Management Unit (MMU)

The MMU checks all memory addresses before permitting them to take part in a memory reference (either over the Megabus to main memory or to the cache memory). These checks ensure that addresses are legitimate and do not violate any software imposed restrictions. If a memory address is rejected by the MMU, a protection violation results. Addresses before being processed by the MMU are called Virtaul addresses, while addresses after being processed by the MMU are called Physical addresses.

4.9.5 Cache Memory

The cache memory contains copies of selected (recently referenced) main memory locations. It has a Megabus interface which allows it to make main memory read references on behalf of the CPU and to monitor the Megabus, copying main memory write data if it currently contains a copy of the main memory location addressed. The cache also has a private interface allowing it to communicate to the CPU to which it is dedicated. It receives main memory read requests across this interface, thereby becoming committed to locate the data for the CPU in its local cache array or in the actual main memory. In either case, the requested data is returned to the CPU with an appropriate handshake.

4.9.6 Hexadecimal Decoder

The hexadecimal decoder (RXBD10 through RXBD1F) generates 16-bit masks for bit test operations (i.e., a single bit within a 16-bit word can be selected). The bit being tested is selected by the 4-bit value from the XB register (refer to subsection 4.7.3.3). This value is used to define which of the 16 bits is zero; the other 15 bits are made all ones.

4.9.7 Subcommand Generator

The 6-bit GP field of the Control Store Word (CSWD) is essentially viewed by firmware as three independent hexadecimal subfields: 10, 20, and 30, which control various hardware operations. Actually, these subfields provide a 6-bit hexadecimal address (0 through 3F) to two PROM devices and a binary-to-decimal decoder. These devices comprise the subcommand generator (see Figure 4-7) which provides control signals (subcommands) for the CPU hardware.

Decoder selection occurs when CSWD 36 through 38 equal 6 (110) and MCLOCK+ is inactive. This disables the two PROMs and enables the least significant three bits of the low-order address field (CSWD 39 through CSWD 41) into decoder 30 for selection of the applicable control signal. Also, when CSWD 36 and 37 equal 3 (11), they enable the control panel for service by the CPU.

Selection between the two PROMs is performed by CSWD 36 and 37 as indicated below:

CSWD		PROM
36	37	
0	0	10/20
0	1	10
1	0	20

This enables the respective hexadecimal address field into the selected PROM for selection of the applicable control signal(s).

NOTE

Some overlap between the two PROMs can occur to provide adequate control signal selection.

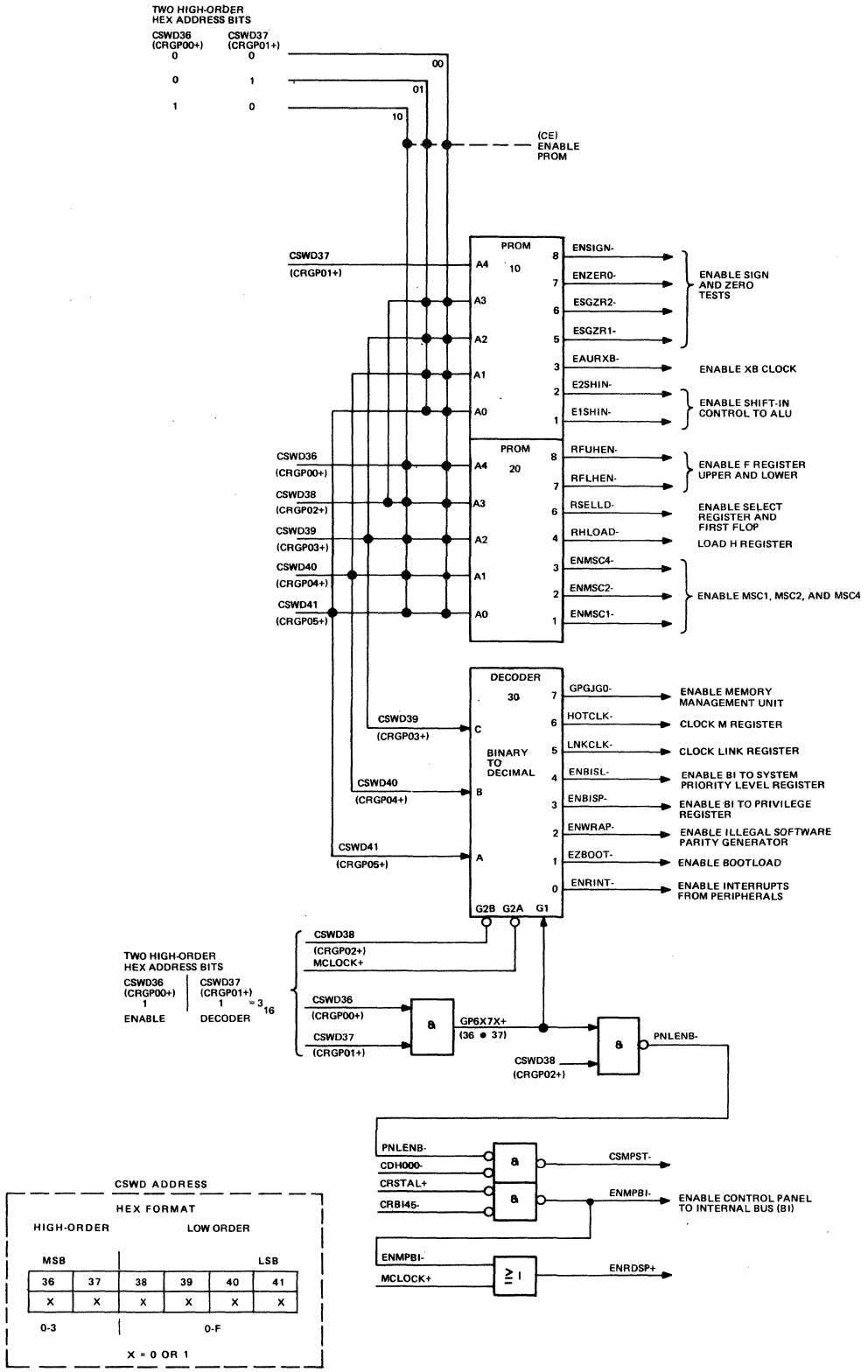


Figure 4-7 Subcommand Generator Logic

4.9.8 Constant Generators

The CPU constant generators provide two types of constants, which are available as internal bus sources as directed by the BI Field of the current firmware word. These constants are:

1. Numeric constants, and

2. Constants used to generate control words for communication with external processors.

4.9.9 Control Panel

Refer to subsection 4.12 for a description of the full control panel.

4.10 MEGABUS NETWORK

The Megabus provides a common communication path among available system units. The design of the Megabus is asynchronous to make communications possible between units of varying speeds.

4.10.1 Interface Logic

The Megabus interfaces with the CPU via a group of transceivers that provide the equivalent electrical characteristics required of all bus connections to allow data, address, and control signals to be routed to and from the CPU. Table 4-4 provides a complete list of the interface signals, while subsections 4.10.1.1 through 4.10.1.6 provide a brief description of each Megabus/CPU interface signal.

4.10.1.1 Timing Signal Lines

The following signals provide the handshake function required by a unit to either initiate, accept, or deny a request for a Megabus cycle from another unit.

Bus Request (BSREQT-)

When true, this signal indicates that one or more of the units connected to the Megabus have requested a bus cycle. When the signal is false, no requests are pending.

Bus Data Cycle Now (BSDCNN-)

When true, this signal indicates that a specific master unit has been granted a requested Megabus cycle and has placed information on the bus for use by a specific slave unit. When this signal is false, the bus is not busy (i.e., between bus cycles).

Table 4-4 Megabus Interface Signals (Sheet 1 of 2)

TYPE	FUNCTION	CPU SIGNAL			BUS SIGNAL
		DRIVER ENABLE	DRIVER INPUT	RECEIVER OUTPUT	
Bus Handshake	Bus Request	GROUND	MYREQT+	BSREQT+	BSREQT-
	Data Cycle Now	GROUND	MYDCNN+	BSDCNN+	BSDCNN-
	Acknowledge	GROUND	MYACKR+	BSACKR+	BSACKR-
	Negative Acknowledge	GROUND	MYNAKR+	BSNAKR+	BSNAKR-
	Wait	GROUND	GROUND	BSWAIT+	BSWAIT-
Data	Data Bit A (unused)	-	-	-	BSDT0A-
	0	MYDCNW-	BIXX10+	BSDT00+	BSDT00-
	1	MYDCNW-	BIXX11+	BSDT01+	BSDT01-
	2	MYDCNW-	BIXX12+	BSDT02+	BSDT02-
	3	MYDCNW-	BIXX13+	BSDT03+	BSDT03-
	4	MYDCNW-	BIXX14+	BSDT04+	BSDT04-
	5	MYDCNW-	BIXX15+	BSDT05+	BSDT05-
	6	MYDCNW-	BIXX16+	BSDT06+	BSDT06-
	7	MYDCNN-	MYDT17+	BSDT07+	BSDT07-
	B (unused)	-	-	-	BSDT0B-
	8	MYDCNN-	MYDT18+	BSDT08+	BSDT08-
	9	MYDCNN-	MYDT19+	BSDT09+	BSDT09-
	10	MYDCNN-	MYDT1A+	BSDT10+	BSDT10-
	11	MYDCNW-	BIXX1B+	BSDT11+	BSDT11-
	12	MYDCNW-	BIXX1C+	BSDT12+	BSDT12-
	13	MYDCNW-	BIXX1D+	BSDT13+	BSDT13-
	14	MYDCNW-	BIXX1E+	BSDT14+	BSDT14-
	Data Bit 15	MYDCNW-	BIXX1F+	BSDT15+	BSDT15-
Main Memory Address	Address Bit 0	MYDCNN-	GJAD00+	BSAD00+	BSAD00-
	1	MYDCNN-	ZGNDAA	BSAD01+	BSAD01-
	2	MYDCNN-	ZGNDAA	BSAD02+	BSAD02-
	3	MYDCNN-	GJAD03+	BSAD03+	BSAD03-
	4	MYDCNN-	GJAD04+	BSAD04+	BSAD04-
	5	MYDCNN-	GJAD05+	BSAD05+	BSAD05-
	6	MYDCNN-	GJAD06+	BSAD06+	BSAD06-
	7	MYDCNN-	GJAD07+	BSAD07+	BSAD07-
	8	MYDCNN-	GJAD08+	BSAD08+	BSAD08-
	9	MYDCNN-	GJAD09+	BSAD09+	BSAD09-
	10	MYDCNN-	GJAD10+	BSAD10+	BSAD10-
	11	MYDCNN-	GJAD11+	BSAD11+	BSAD11-
	12	MYDCNN-	GJAD12+	BSAD12+	BSAD12-
	13	MYDCNN-	GJAD13+	BSAD13+	BSAD13-
	14	MYDCNN-	GJAD14+	BSAD14+	BSAD14-
	15	MYDCNN-	MYAD15+	BSAD15+	BSAD15-
	16	MYDCNN-	MYAD16+	BSAD16+	BSAD16-
	17	MYDCNN-	MYAD17+	BSAD17+	BSAD17-
	18	MYDCNN-	MYAD18+	BSAD18+	BSAD18-
	19	MYDCNN-	MYAD19+	BSAD19+	BSAD19-
	20	MYDCNN-	MYAD20+	BSAD20+	BSAD20-
	21	MYDCNN-	MYAD21+	BSAD21+	BSAD21-
	22	MYDCNN-	MYAD22+	BSAD22+	BSAD22-
	Address Bit 23	MYDCNN-	RXBOFF+	BSAD23+	BSAD23-
Transfer	Memory Reference	MYDCNN-	CIMREF+	BSMREF+	BSMREF-
	Byte (unused)	MYDCNN-	CRCLK0+	BSBYTE+	BSBYTE-
	Bus Write	MYDCNN-	CIREAD-	BSWRIT+	BSWRIT-
	Second Half Read	MYDCNN-	MYSHBC+	BSSHBC+	BSSHBC-
	Lock (unused)	MYDCNN-	MYRMWC+	BSLOCK+	BSLOCK-
	Memory Error (Red)	MYDCNN-	GROUND	BSREDD+	BSREDD-
	Memory Error (Yellow)	MYDCNN-	GROUND	BSYELO+	BSYELO-
	Data Parity Left	MYDCNN-	MYDP00+	BSDP00+	BSDP00-
	Data Parity Right	MYDCNN-	MYDP08+	BSDP08+	BSDP08-
	Address Parity (bits 0-7)	MYDCNN-	MYAP00+	BSAP00+	BSAP00-
	Double Pull	MYDCNN-	XTDBPL+	BSDBPL+	BSDBPL-

Table 4-4 Megabus Interface Signals (Sheet 2 of 2)

TYPE	FUNCTION	DRIVER ENABLE	DRIVER INPUT	RECEIVER OUTPUT	BUS SIGNAL
Verify Bus Continuity	Logic Test Out	-	-	-	BSQLTO-
	Logic Test In	-	-	-	BSQLTI-
	Logic Test Active	-	-	-	BSQLTA+
Establish Positional Priority	Tie-Breaking Network	-	-	-	BSAUOK+
		-	-	-	BSBUOK+
		-	-	-	BSCUOK+
		-	-	-	BSDUOK+
		-	-	-	BSEUOK+
		-	-	-	BSFUOK+
		-	-	-	BSGUOK+
		-	-	-	BSHUOK+
		-	-	-	BSIUOK+
	Tie-Breaking Network	-	-	-	BSMYOK+
Miscella-neous	Master Clear	GROUND	MYMCLR+	BSMCLR+	BSMCLR-
	Power On	-	-	-	BSPWON+
	Resume Interrupt	GROUND	ENRINT+	BSRINT+	BSRINT-
	Spare Line (unused)	-	-	-	BSSPR1-
	Spare Line (unused)	-	-	-	BSSPR3-
	Spare Line (unused)	-	-	-	BSSPR4-
	External Connection (unused)	-	-	-	BSEXTC+

Bus Acknowledge Response (BSACKR-)

When true, this signal indicates to the master unit that the slave unit has received and accepted a specific transfer from the master unit.

Bus Negative Acknowledge Response (BSNAKR-)

When true, this signal indicates to the master unit that a slave unit is refusing a specific transfer (i.e., the slave unit cannot accept the transfer and the master unit should not attempt a retry). For example, a busy memory addressed for a data transfer.

If a situation arises when no unit on the bus recognizes the transfer because of improper addressing or a malfunction, and no response (ACK, NAK, or WAIT) is generated within 5 microseconds, the CPU (which monitors all bus transfers) will issue a Bus Negative Acknowledge Response (BSNAKR-) signal on behalf of the entire system. This is referred to as the Dead Man Time-Out operation.

Bus Wait (BSWAIT-)

When true, this signal indicates to the master unit that the slave unit cannot accept a specific transfer at this time (i.e, the slave unit is temporarily busy and the master unit must initiate successive retries until the transfer is acknowledged).

4.10.1.2 Information Signal Lines

The signals described in this subsection effect the transfer of information during a bus cycle as data or information signals.

Bus Data Lines (BSDT00- through BSDT15-)

The bus data bits can be formatted for a single data word (16 bits), channel number coding (CPU), or the low-order address bits (8 through 23), depending on the operation being performed. Thus, data, address, control, register, or status information can be reflected by the 16 data lines.

Bus Address Lines (BSAD00- through BSAD23-)

The 24 address lines can be formatted for a single 23-bit main memory address to select one of 8M bytes.

The address lines can also be formatted for a channel number code (CPU), and I/O function code on lines 18 through 23, or a combination of all three for an IOLD operation.

4.10.1.3 Information Control Signal Lines

The following signals serve as data, address, and information control signals that effect the transfer and control of such information during a bus cycle.

Bus Memory Reference (BSMREF-)

When true, this signal indicates that bus address lines 0 through 23 contain a complete main memory address from the master unit. When false, the Bus Memory Reference signal indicates that the bus address lines contain a channel number on lines 8 through 17 (with or without a function code on lines 18 through 23), or a main memory module address code on lines 0 through 7; the exact configuration and direction of flow depends on the operation being performed.

Bus Write (BSWRIT-)

When true, this signal indicates that the master unit is transmitting data to the slave unit. When this signal is false, the initial bus cycle signals a read request ($\overline{\text{BSWRIT}}$), while the data lines contain the requesting channel number; the slave unit, if it accepts the request, replies with a read response via a subsequent bus cycle, which is defined as a Second Half Bus Cycle (BSSHBC).

Bus Second Half Bus Cycle (BSSHBC-)

When true, this signal indicates to the master unit that the current information generated by the slave unit is the information previously requested during the initiation cycle.

Bus Double Pull (BSDBPL-)

When true, this signal indicates that the master unit is requesting a double-word operand from the slave unit. During the first Second Half Bus Cycle, BSDBPL- is redelivered to the requesting unit, indicating that another word follows.

NOTE

If a single fetch memory is installed on the system, BSDBPL- is not redelivered during the Second Half Bus Cycle, notifying the requesting unit that only single-word operations will be performed.

4.10.1.4 Status/Error Signal Lines

The following signal lines provide main memory error reporting signals for the available units, and two-way bus parity lines for odd parity signals used with the address and/or information bits placed on the Megabus. Two lines provide for a bus continuity check, combined with a check on the integrity of the resident logic test in each unit. A single line is used to indicate the status of system power.

Bus Red (BSREDD-)

The Bus Red error signal can only be generated by a main memory unit that contains EDAC logic. When true, the signal indicates that the memory has detected an error during a read (Second Half Bus Cycle) operation.

Bus Yellow (BSYELO-)

The Bus Yellow error signal can only be generated by a main memory unit that contains EDAC logic. When true, the signal indicates that memory has detected and corrected an error during a read (Second Half Bus Cycle) operation.

Bus Address Parity (BSAP00-)

The level of the Bus Address Parity signal provides odd parity for address bits 0 through 7 (module address bits).

Bus Data Parity - Left Byte (BSDP00-)

The level of the Bus Data Parity - Left Byte signal provides odd parity for the left data byte (bits 0 through 7).

Bus Data Parity - Right Byte (BSDP08-)

The level of the Bus Data Parity - Right Byte signal provides odd parity for the right data byte (bits 8 through 15).

Bus Quality Logic Test Out (BSQLTO-) and In (BSQLTI-)

The Bus Quality Logic Test Out and In signals are static integrity signals which, if continuously true, indicate that each available unit performed its resident Quality Logic Test (QLT) successfully. The signal is relayed from unit to unit from one end of the bus to the other and back. This signal effectively provides a continuity check for all available units.

Bus Quality Logic Test Active (BSQLTA+)

When true, this signal indicates that at least one unit on the bus has not successfully completed its logic test or that there is a break in continuity on the bus (i.e., a unit is not installed).

Bus Power On (BSPWON+)

When the Bus Power On signal is true, it indicates that all system power supplies are functioning correctly. This signal goes true when power has stabilized and goes false several milliseconds before power fails.

4.10.1.5 Tie-Breaking Control Signals

There are nine tie-breaking signals (BSAUOK+ through BSIUOK+), all of which must be true to provide an enable for any unit requesting a bus cycle. If more than one unit simultaneously requests a bus cycle, the cycle is granted to only one unit on a positional priority basis. The priority extremes are the ends of the bus; memory has the highest positional priority and the CPU has the lowest. Thus, under simultaneous request conditions, the highest positioned requesting unit receives true enables from all nine tie-breaking signals, while the remaining requesting units receive eight or less, depending on the relative position of their decreasing priority.

Bus My OK (BSMYOK+)

When true, this signal indicates to the next lower priority unit that the generating unit, and certain other units of a higher positional priority, have not requested a bus cycle within the last 20 nanoseconds; therefore, a cycle can be granted (if requested) to a lower priority unit.

4.10.1.6 Operational Control Signals

The following control signals are asynchronous in relation to the functions they perform and the normal initiation and control of bus cycles.

Bus Resume Interrupt (BSRINT-)

When true, this signal allows the CPU or memory to reissue an interrupt that has previously been refused.

Bus Master Clear (BSMCLR-)

When true, this signal indicates that the Master Clear (CLR) pushbutton, located on the CPU control panel, has been depressed.

4.10.2 Megabus Network Operation

The information contained in this subsection describes the bus cycle timing and controls in relation to the handshaking techniques used to establish communications between any two units in the system. Bus dialogue is completely asynchronous, and each bus cycle can be considered as an independent handshaking sequence between a master unit and a slave unit. To implement this type of operation, all major logic boards within the units that comprise the system have similar Megabus cycle control logic.

Figure 4-8 depicts the general timing relationships incurred when handshaking techniques are applied between a master unit and a slave unit on request of the master unit. The Bus Cycle Request (BSREQT) signal is common to all available units, and when true, indicates that one or more have initiated a request for a bus cycle.

If a bus cycle request is actually the result of simultaneous requests, the tie-breaking logic (contained in each unit) resolves the priority dilemma by granting the bus to a specific unit on the basis of position. At this point, it is enough to understand that memory resides at the high priority end of the bus and the CPU resides at the low priority end; the remaining units occupy intermediate priority positions.

When a bus cycle is granted, the Bus Data Cycle Now (BSDCNN) signal goes true, indicating that the tie-breaking operation is complete and a specific unit has been granted master status (gained bus control). At this time, the data and address information designated for the slave unit is placed on the bus. Assuming a multi-unit system, the Bus Data Cycle Now signal allows each potential slave unit to internally generate a Data Cycle Now Delayed (BSDCND) signal. The delay (60 nanoseconds) provides bus skew correction time for each unit, allowing the currently active bus address information to be properly examined by the decoding networks available in each unit. Under these conditions, the Data Cycle Now Delayed signal, in conjunction with signal MYCHAN that is derived from the appropriate unit's decoding of the predefined address, allows an internal strobe to be generated in the designated slave unit. The slave unit then returns one of three response signals (BSACKR, BSNAKR, or BSWAIT) to the master unit to complete the handshake and to indicate that the slave unit acknowledges the communication as being accepted, denied, or postponed, respectively.

A bus handshake is an asynchronous operation in which the change of a signal level can only occur in response to the level change of a handshake signal from a defined unit. Since various

units can differ in the length of time required between strobe development and acknowledgment, signal transitions depend on the type of internal functionality. In the case where no acknowledgment is received by a Master Unit (no response signal generated), the CPU Dead Man Time-Out operation allows a delay of 5-microseconds before clearing the bus on behalf of the entire system.

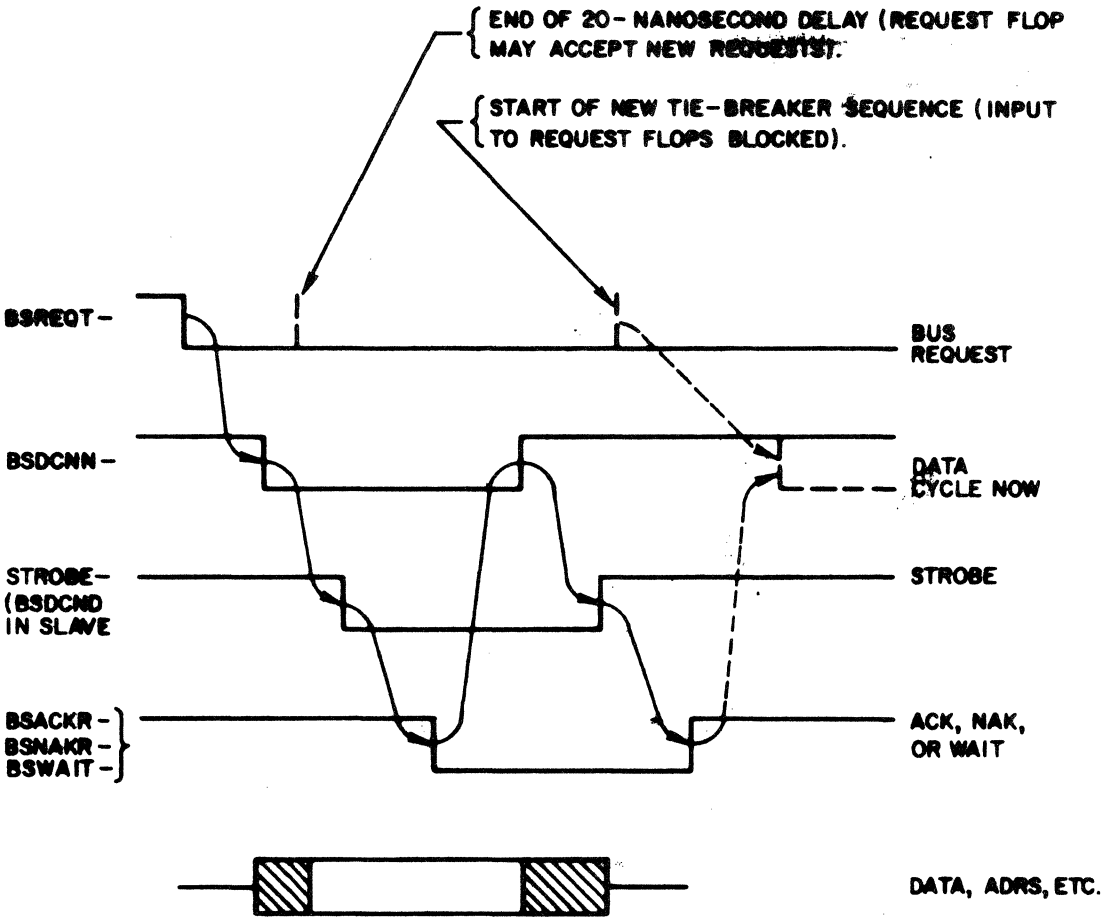


Figure 4-8 Bus Timing Diagram

4.10.3 Megabus Tie-Breaking Function

The tie-breaking function resolves simultaneous Megabus requests and grants bus cycles on a positional priority basis. In any given system, memory has the highest priority and the CPU has the lowest priority; they reside at opposite ends of the bus. Other units occupy intermediate positions and have a priority that increases relative to their proximity to the memory end of the bus.

Each unit on the Megabus contains the logic necessary for granting a Megabus cycle. When the Data Cycle Now signal is valid, the requesting unit gains access to the Megabus.

Referring to Figure 4-9, note that each unit's grant logic monitors the preceding unit's tie-breaking lines (BSAUOK through (BSIUOK). Therefore, only one unit has its grant logic satisfied at any one time. The one primary line that is necessary to satisfy the input structure of the grant logic is BSAUOK. In reality, BSAUOK is the preceding higher priority signal BSMYOK.

NOTE

Main memory always takes priority on Megabus accesses and does not need to monitor the remaining unit's tie-breaking lines.

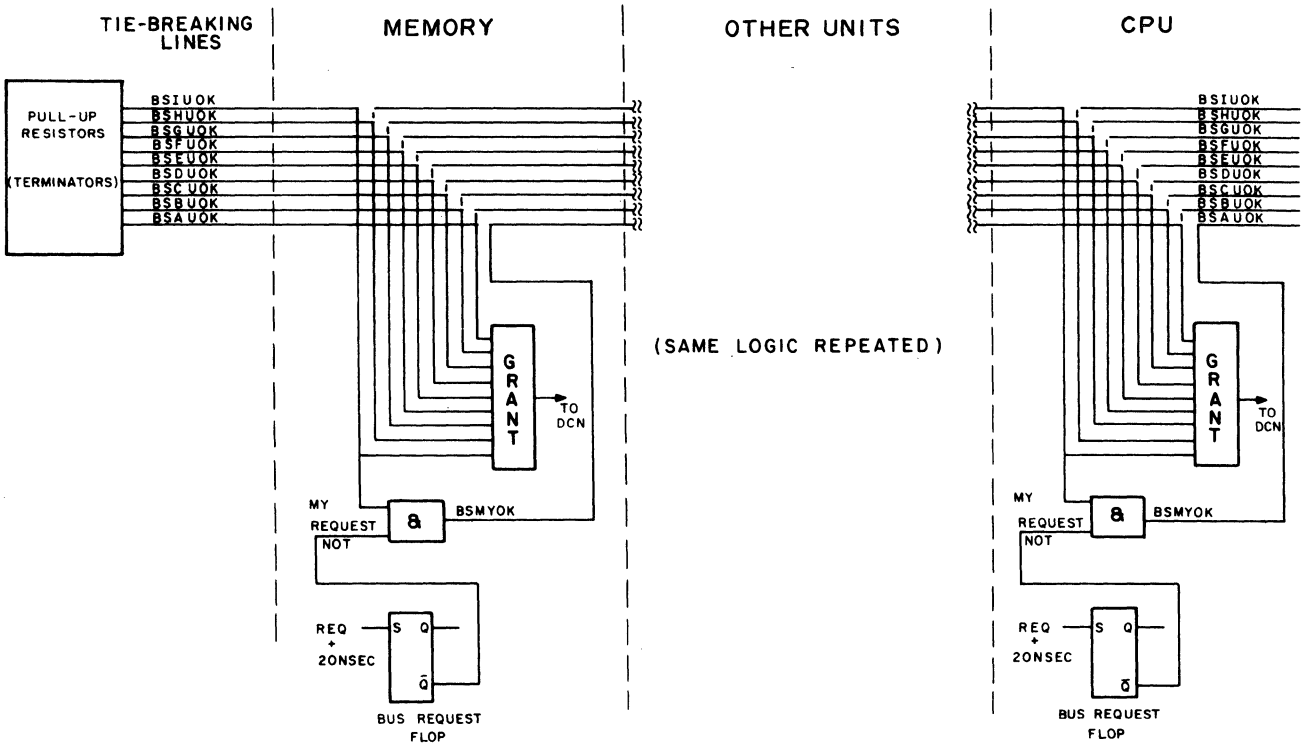


Figure 4-9 Megabus Tie-Breaking Logic

If the CPU requires access to the Megabus (as indicated by control store bit 31), the User flip-flop sets, initiating a CPU start request. When there is no Megabus activity, the Request flip-flop sets, producing a CPU bus request and disabling other units from gaining access to the Megabus during the processor response (see Figure 4-10).

With the Grant flip-flop set, the Data Cycle Now signal generates one of the following three responses: Acknowledge (ACK), Negative Acknowledge (NAK), or Wait (WAIT). When the response

signal is received, the Grant flip-flop clears and the Megabus is available for use by another unit.

4.10.4 Memory Read Request

The CPU can: (1) with a single memory read request, signal that it wants the delivery of two sequential words from memory, and (2) have both a double-word and a single-word request outstanding to two different memory modules simultaneously.

Double-word data is always stored in the P1 and P2 Procedure Buffers whereas single-word data is stored in the Data Buffer (BD). Since both single- and double-word requests may be present at the same time, the processor tags the request in the function code field at the time of the request. Single-word requests are tagged 00_{16}, while double-word requests are tagged 20_{16}. During the request, bus data lines 10 through 15 constitute the tag. During the Second Half Bus Cycle (BSSHBC), address bits 18 through 23 constitute the tag echoed by the memory.

4.10.4.1 Double-Word Fetch

A double-word fetch operation is executed only if: (1) the Procedure Buffers are both empty and (2) memory is not currently processing a double-word pull. Firmware then performs a Procedure Stall (setting the Procedure Buffer Stall flip-flop) until memory transfers either a procedure or the first word of a double-word fetch (see Figure 4-10).

During the double word fetch operation, two flip-flops (Word A Request and Word B Request) sample the Second Half Bus Cycle signal. These two flip-flops set when a memory request is present and test the Second Half Bus Cycle signal when the CPU acknowledges with the My Second Half Response signal. When a double word is received from memory, the Word A Request and Word B Request flip-flops are respectively cleared. During a single word fetch, only one Second Half Bus Cycle signal is received, clearing the Word B Request flip-flop. Two additional flip-flops, Take Word C and Take Word D, track the firmware's usage of the data received from memory. The Take Word C flip-flop clears when firmware uses the first word and the Take Word D flip-flop clears when firmware uses the second word from memory.

4.10.4.2 Single Word Fetch

Single word fetches require at least two firmware steps. The first step generates a Megabus Read Cycle causing the Data Buffer Stall flip-flop to be set when memory (or the I/O) accepts the request cycle. The second step attempts to input the data onto the internal bus. These two firmware steps need not be contiguous. If the Second Half Bus Cycle signal is delayed, the Data Buffer Stall flip-flop remains set until the Second Half Read Data signal is received.

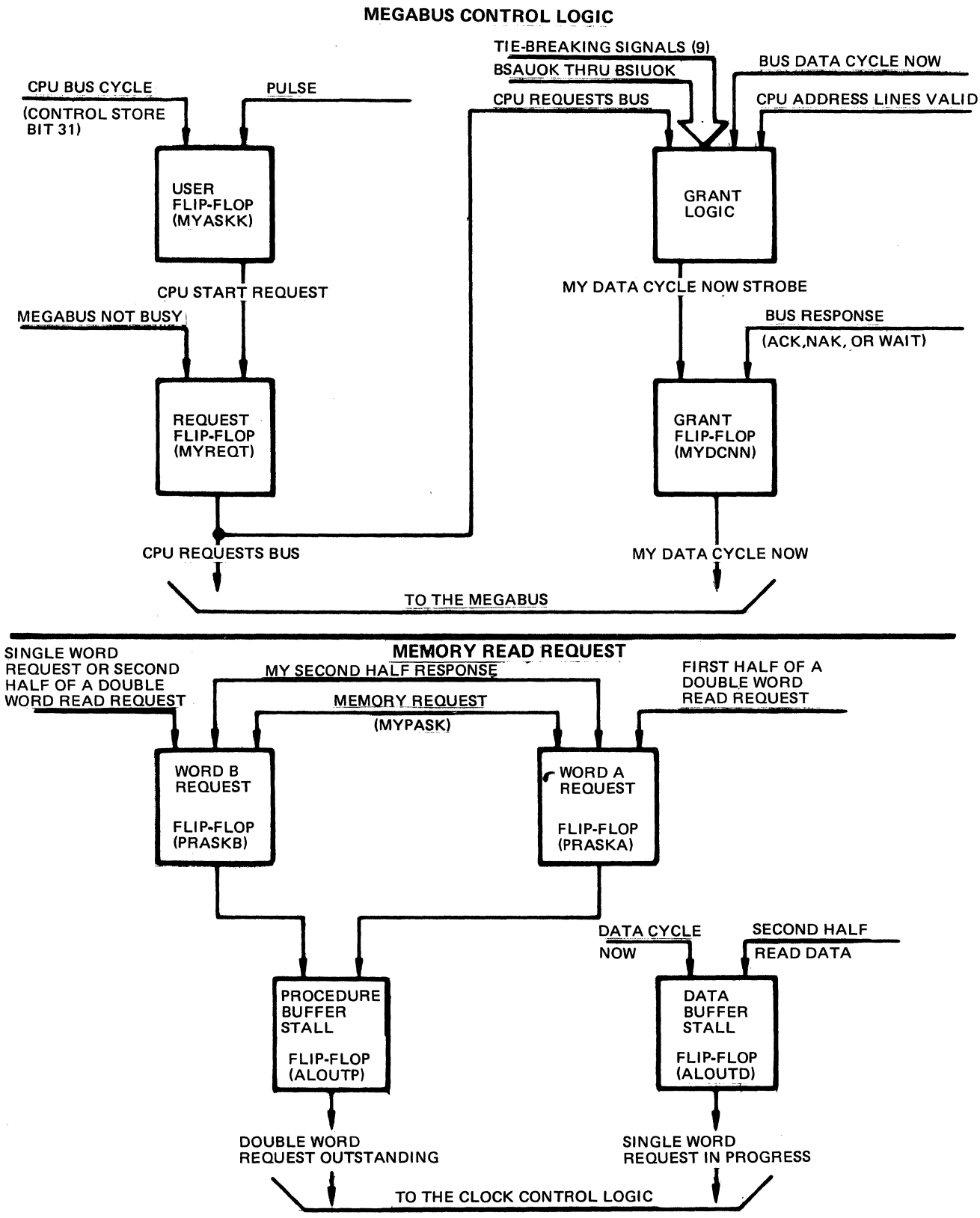


Figure 4-10 Megabus Control and Memory Read Request Logic

4.10.5 Response Summation

The functionality of the CPU logic shown in Figure 4-10 is similar to that of all units operating as a slave unit. However, the specific logic used for the generation of the three response conditions (ACK, NAK, and WAIT) depends on the design requirements of each individual unit. The following information summarizes slave unit response conditions.

Acknowledge Response (BSACKR)

An Acknowledge response (ACK) is generated if the slave unit is currently capable of accepting a bus transfer from the master unit.

Wait Response (BSWAIT)

A Wait response (WAIT) is generated if the slave unit is temporarily busy and cannot accept a transfer, but will be able to accept the transfer after a brief delay. On receipt of the WAIT signal, the master unit generates continual retries until the transfer is completed.

Negative Acknowledge Response (BSNAKR)

A Negative Acknowledge response (NAK) is generated if the slave unit cannot accept a transfer. On receipt of the NAK signal, the master unit does not attempt a retry.

No Response (ACK, NAK, or WAIT)

If no unit on the bus recognizes a transfer, no response occurs. Under these conditions, the CPU (which monitors all bus transfer) awaits a legitimate signal for approximately 5 microseconds, after which time it clears the bus with a NAK signal; the NAK signal is generated by the setting of the Bus Time-Out flip-flop in the CPU logic.

4.10.6 Megabus Registers

The Megabus registers consist of two procedure buffers (P1 and P2) and one data buffer (BD).

4.10.6.1 P1 and P2 Buffers

The P1 and P2 buffers (BIP110 through BIP11F and BIP210 through BIP21F) receive procedure words from the Megabus as a result of double-word requests. Sixteen bits are received from Megabus data lines 0 through 15, while the remaining four bits are loaded from two integrity bits (BSREDD and BSYELO) and two parity bits (BSDP00 and BSDP08).

The P1 and P2 buffers are loaded by the signals My Second Half P1 and P2 (MYSHP1 and MYSHP2, respectively). MYSHP1 is

generated when the first BSSHBC signal is received by the CPU, and MYSHP2 is generated when the second BSSHBC signal is received.

The P1 and P2 buffers are unloaded onto the internal bus by signals Enable P1 to BI or Enable P2 to BI (ENP1BI and ENP2BI, respectively).

The signal PRTAKC controls which register is unloaded onto the bus during a Procedure Stall operation. If PRTAKC is true, ENP1BI is forced true and the P1 buffer is unloaded; if PRTAKC is false, ENP2BI is forced true and the P2 buffer is unloaded.

The 20-bit output of the P1 buffer is labeled Bus Internal P110 through 1F (BIP110 through BIP11F), integrity signals Red and Yellow (BIP1RD through BIP1YL) and Parity signals 8 and 0 (BIP1P8 and BIP1P0). The 20-bit output of the P2 buffer is labeled BIP210 through BIP21F, BIP2RD and BIP2YL, and BIP2P8 and BIP2P0.

4.10.6.2 BD Buffer

The BD buffer, like the P1 and P2 buffers, receives as input from the Megabus BSDT00 through BSDT15, BSREDD and BSYEL0, and BSDP08 and BSDP00. The operand data from the Megabus is distributed throughout the CPU via the internal bus. The BD buffer is loaded by My Second Half Read (MYSHRD) when the BSSHBC signal is received by the CPU during a single fetch operation. It is unloaded onto the internal bus by Enable Data Buffer to BI (ENDTBI), which comes true when the Data Stall (CRDSTL) signal is true.

The 20-bit output from the BD buffer is labeled BIDT10 through BIDT1F, BIDTRD, BIDTYL, BIDT08, and BIDT00.

4.11 INTERRUPT CONTROL LOGIC

The interrupt control logic is divided into three areas:

- Address compare logic
- Level check logic
- Interrupt register.

4.11.1 Address Compare Logic (see Figure 4-11)

An interrupt message on the Megabus is accepted for examination by the CPU if the currently active data cycle (enabled by the interrupt requesting unit) has not been initiated by the CPU itself and does not require a memory reference. If these conditions are satisfied, the Bus Address Compare (BSACMP) signal enables a comparison of the CPU internal channel identification number (supplied by the hexadecimal rotary switch) and Megabus address bits 14 through 17 (least significant four bits of slave unit channel number) to determine if the interrupt message is

intended for this CPU. If the compare operation is successful (A=B) and Megabus address bits 8 through 13 equal Zero (indicating a CPU type channel number), a CPU address compare signal is generated and fed to the strobe select circuits to participate in determining whether the Megabus format reflects a second half bus cycle or an external interrupt. A second half bus cycle generates the Second Half Cycle Strobe signal (MYSHCS) which, in turn, sets the Second Half Read Cycle Flop (MYSHRC). An external interrupt generates the Interrupt Strobe signal (MYINTS) which, in turn, sets the Interrupt Received flop (MYINTR), providing a clock signal for both the negative acknowledge response and interrupt flip-flops (refer to subsection 4.11.2).

At this point, a decision is made to either accept the interrupt for service by allowing the interrupt flip-flop to set, or to delay acceptance by allowing the negative acknowledge response flip-flop to set. The decision is based on a comparison of the interrupt request's priority level to the current operating level of the CPU, and the state of the interrupt busy flip-flop (refer to subsection 4.11.2).

4.11.2 Level Check Logic

The level check logic (see Figure 4-12) compares Megabus data bits 10 through 15 (level of incoming request), S register bits 10 through 15 (current CPU operating level), and the state of the Interrupt Busy flip-flop (MYINTB). If the priority of the incoming request is equal to or lower than the current operating level of the CPU and/or the interrupt busy flip-flop is set (indicating that the CPU is currently servicing a previous interrupt), the output from the comparator sets the No Acknowledge Response flip-flop (MYNAKR), delaying acceptance of the interrupt. However, if the incoming priority level is higher than the current CPU operating level and the interrupt busy flip-flop is not set, the comparator output sets the Interrupt flop-flip (MYINTF). At this point, the interrupt is accepted and the interrupt busy flip-flop is set.

The interrupt busy flip-flop is also set when a power failure occurs and the CPU is not operating at the hghest (zero) priority level.

4.11.3 Interrupt Register

The interrupt register (see Figure 4-13) receives the level of the incoming interrupt request over Megabus data lines 10 through 15 when the interrupt request is accepted by the CPU (i.e., when the interrupt flip-flop is set). This action allows firmware to process the request subsequent to an acknowledgment. Also, if the Megabus format reflects a second half bus cycle rather than an external interrupt request, Megabus data lines 10 through 15 are fed to the Megabus data buffer when the second half read cycle flip-flop is set.

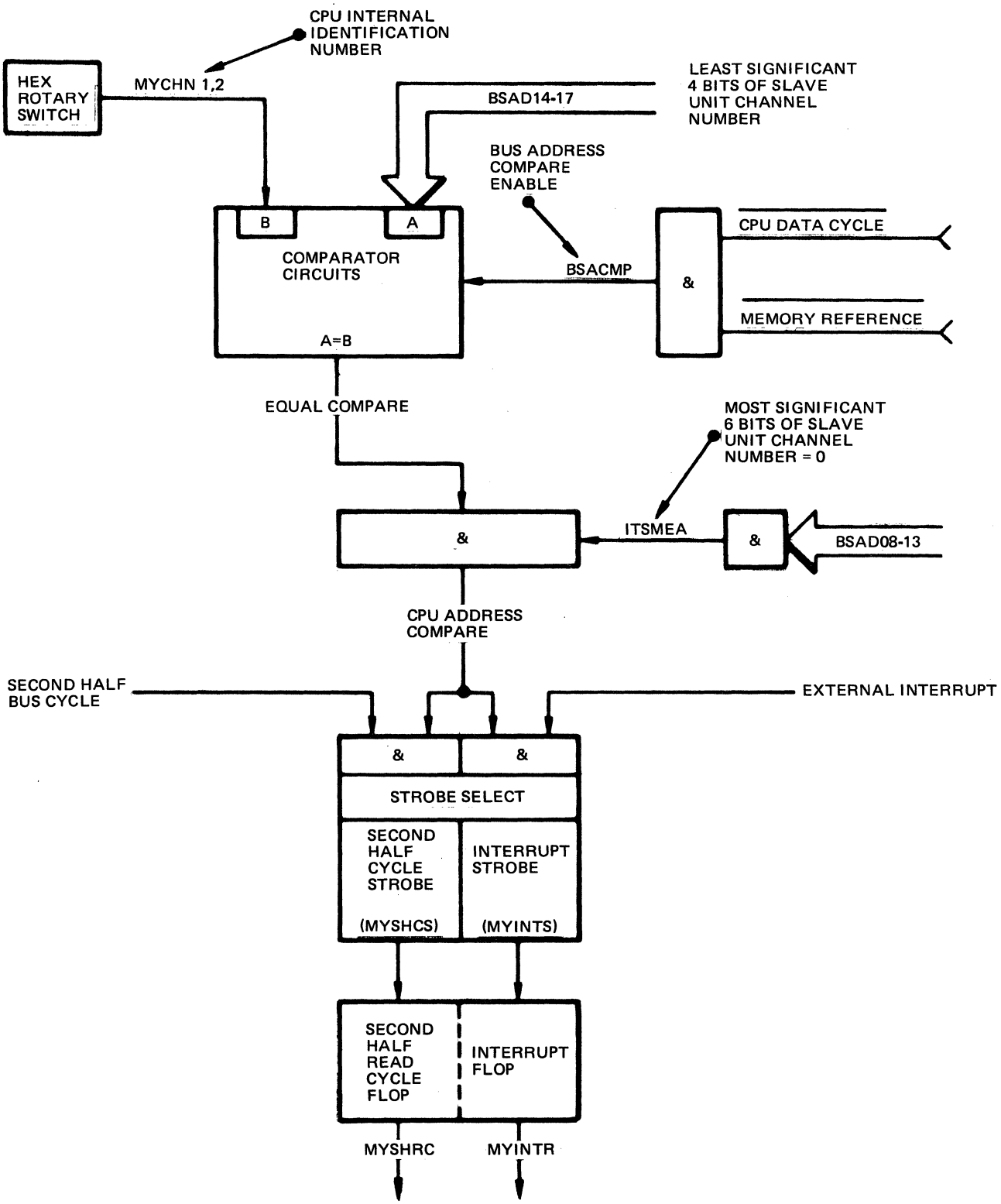


Figure 4-11 Address Compare Logic

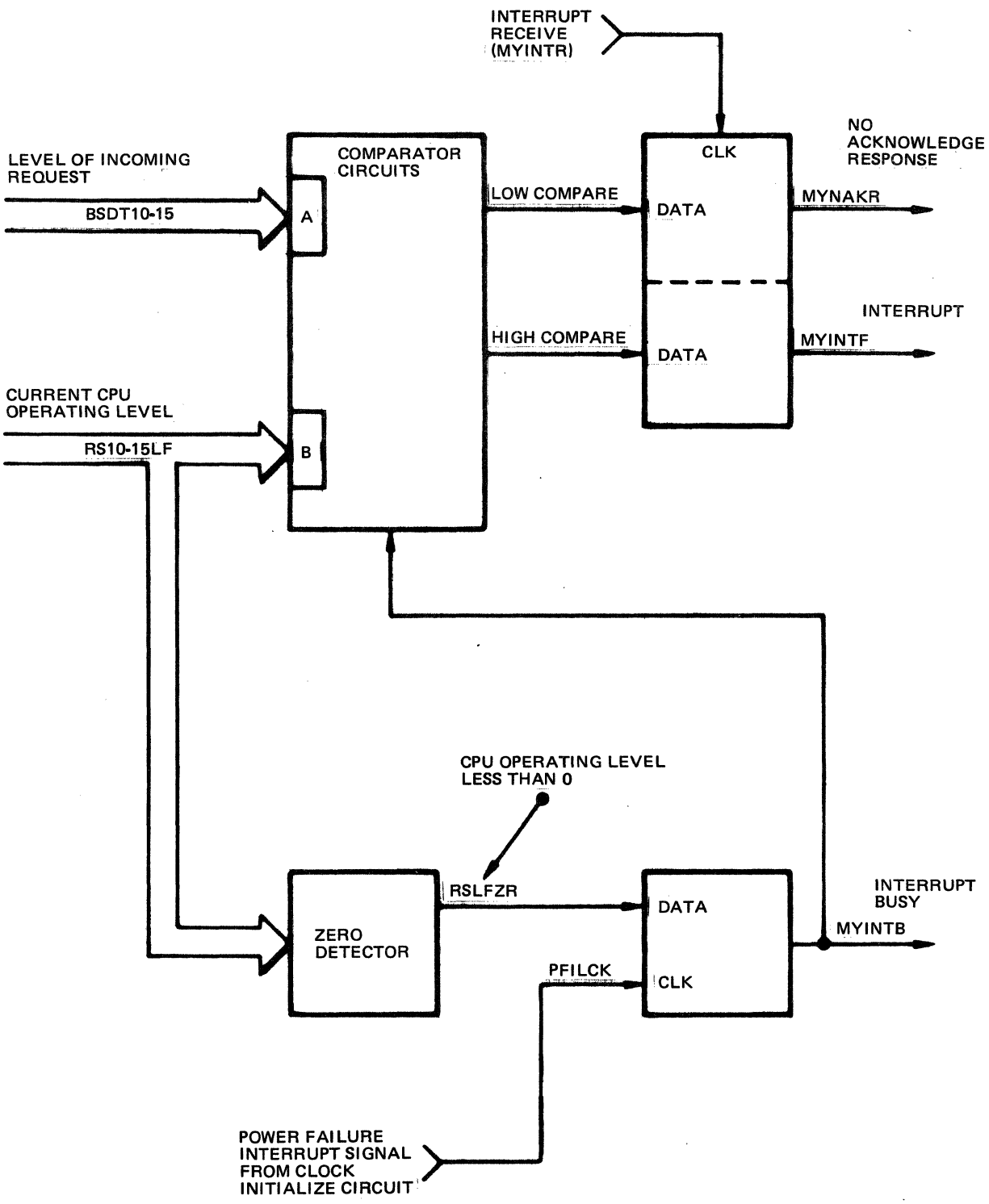


Figure 4-12 Level Check Logic

The output from the interrupt register (BINN10 through BINN1F) is delivered to the internal bus by the Enable Interrupt to BI (ENINBI) signal, indicating that the CPU is ready to service the interrupt.

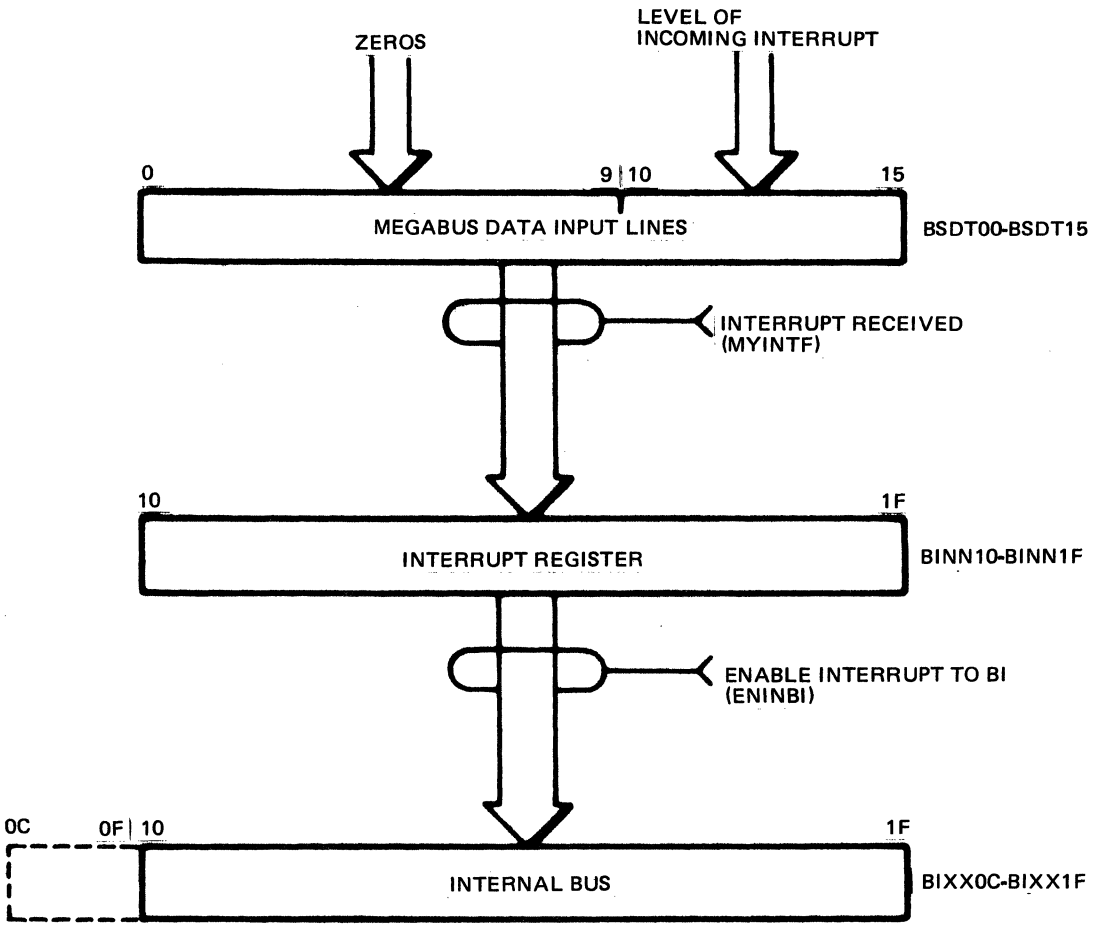


Figure 4-13 Interrupt Register

4.12 FULL CONTROL PANEL

This subsection provides a description of the full control panel physical characteristics and principles of operation. A description of the basic control panel is not included because its limited operating controls and indicators are similar to the corresponding full panel controls and indicators (refer to subsection 2.1.9.9.1). To obtain a comprehensive understanding of each basic panel control and indicator function, the reader should reference the applicable diagrams for the basic control panel and the corresponding text for the full control panel.

NOTE

The material contained herein assumes that a full control panel is connected to an operational CPU, the CPU is powered up, and the panel is active (unlocked).

4.12.1 Physical Characteristics

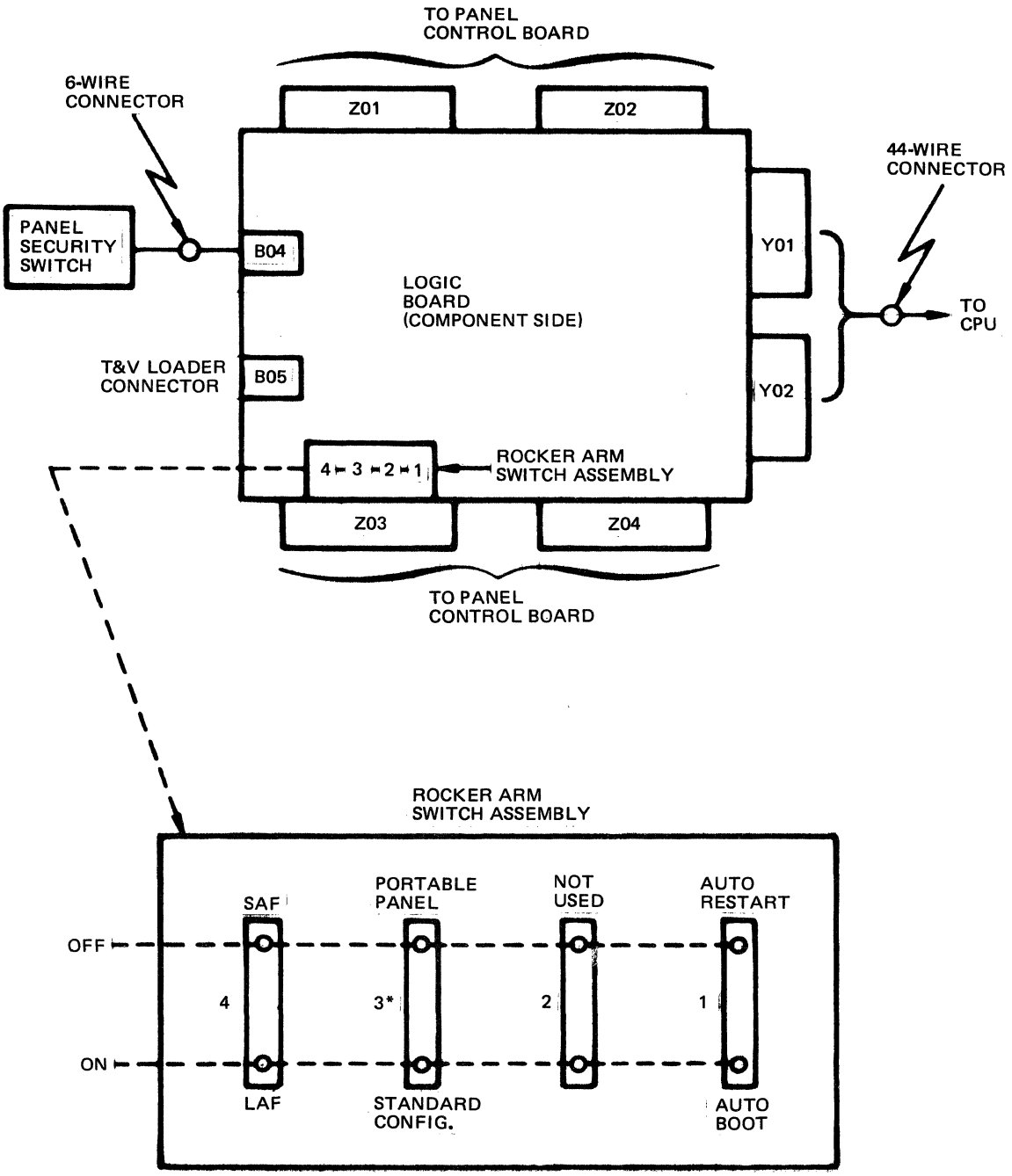
The control panel is a self-contained unit, consisting of one control and one logic board. The control switches, system status, and register display indicators are mounted on the control board; their associated circuit components and configurations are mounted on the logic board. The control switches, status indicators, and display indicators are as follows (see Figure 2-2 for the physical location of each component):

- POWER on/off switch
- PANEL SECURITY lock/unlock
- LOCATION display (register address)
- CONTENTS display (register data)
- CPU controls and indicators
- Key-pad array.

Four connectors provide the direct signal interface connection between both boards (see Figure 4-14). The rocker arm switch assembly is mounted on the logic board (refer to Table 2-1 for correct switch settings). In addition, the logic board connects to the CPU with a 44-wire connector and to the PANEL SECURITY switch with a 6-wire connector. The 44-wire pin-to-pin cable connections are shown in Figure 4-15. The PANEL SECURITY and POWER switches are connected to the panel frame on the left side of the control board.

4.12.2 Principles of Operation

Figure 4-16 illustrates the data flow, control signal development, and signal interface between the CPU and control panel, and should be referred to throughout the text as needed. The principal elements of the panel are the key-pad array and the LOCATION/CONTENTS display. All other elements support each register and key-pad, and produce the control panel internal and CPU interface control signals.



*SWITCH 3 IS NOT IMPLEMENTED ON THE BASIC PANEL.

Figure 4-14 Interconnections and Rocker Arm Switch Assembly

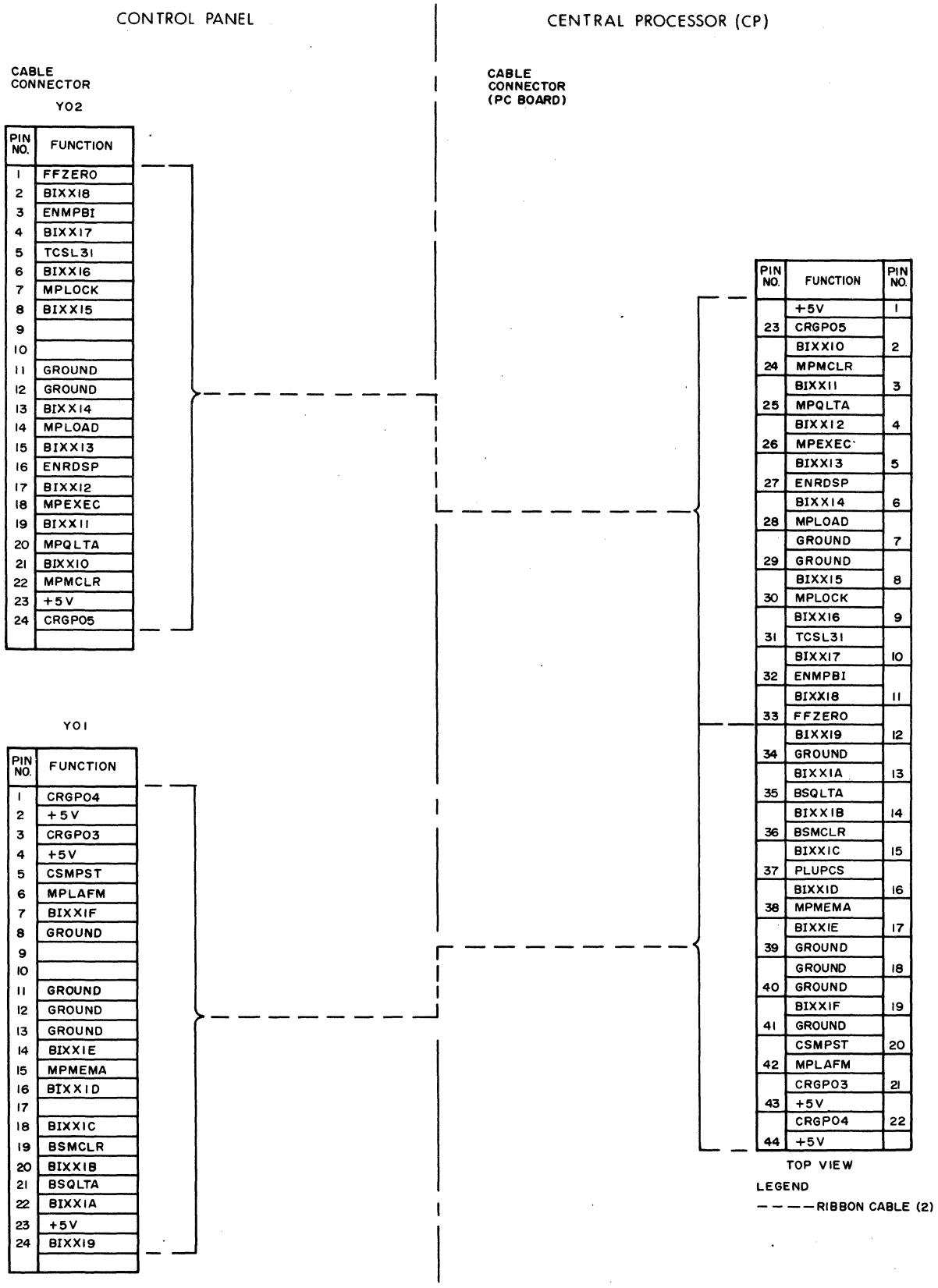


Figure 4-15 Control Panel to CPU Signal Cable Connections

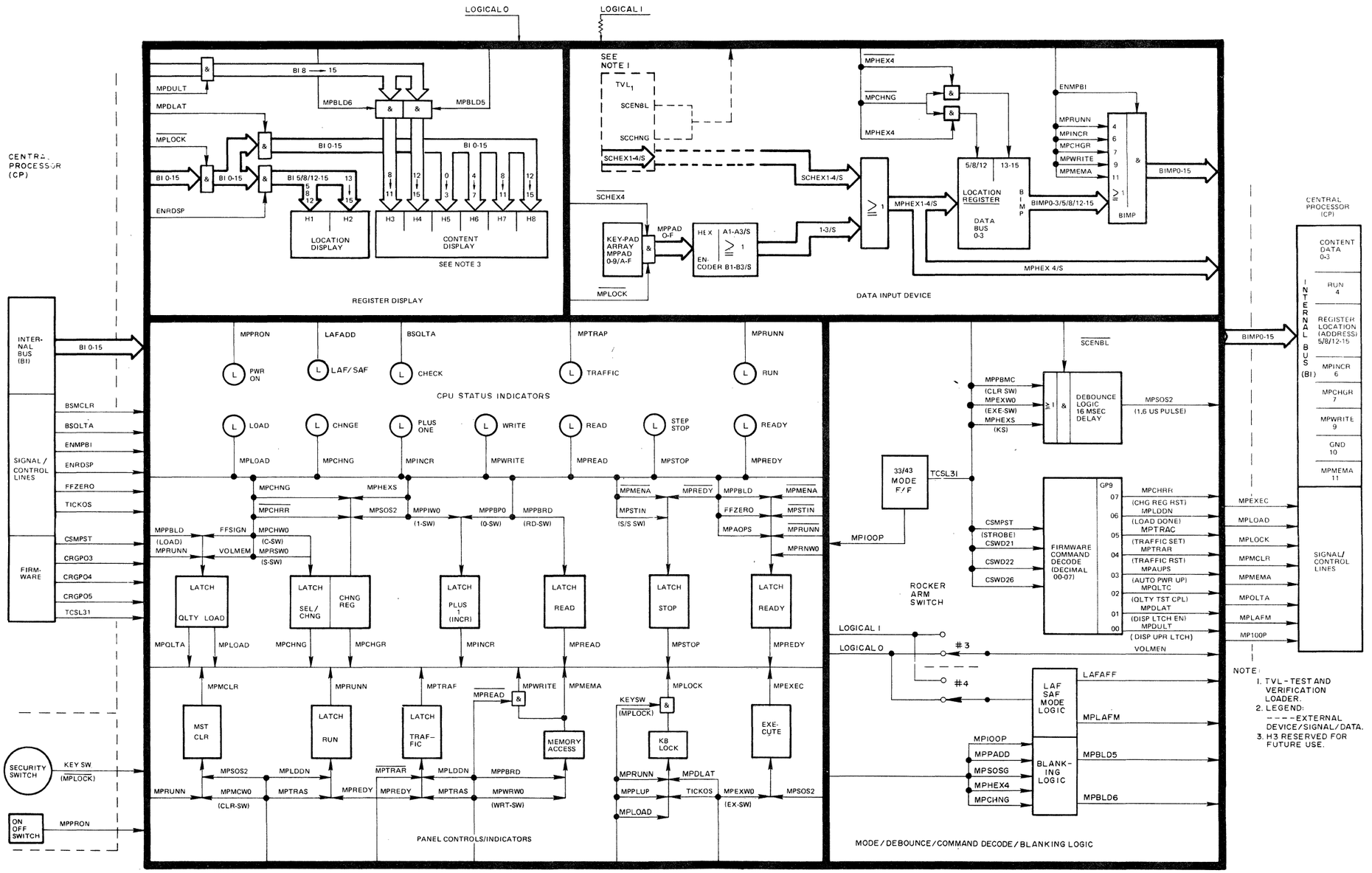


Figure 4-16 CPU Panel Control Logic

4.12.2.1 Control Panel Functionality

All control panel activity is governed by CPU firmware to allow manual operations from the panel when the CPU is in its Stop state, and to automatically update the respective control panel displays and indicators at 8-millisecond intervals when the CPU is in its Run state. Also, with the CPU in its Run state, the firmware acknowledges the following two manual functions from the panel to permit selection of an addressable CPU register for display in the panel displays, or to stop the CPU for manual intervention:

- Select mode: Manual selection via the Select (S) pushbutton
- Stop mode: Manual selection via the STOP (S) pushbutton.

All data and signal communication between the CPU and control panel is through 16 Internal Bus (BI) and 28 unique control panel transmission lines. The CPU gates (ENMPBI) panel data (see Table 4-5) onto the internal bus, and simultaneously gates (ENRDSP) address data from the internal bus into the control panel LOCATION display. All data character transfers between the panel and internal bus are controlled by specific control panel selections (e.g., Read or Write Memory) and the resident CPU control panel firmware.

Table 4-5 Control Panel to CPU Internal Bus Bit Assignments

BIT(S)	FUNCTIONS	REMARKS
0-3	CPU register data	From panel data bus
4	Run mode (true); Stop mode (false)	-
5	Panel location register data	Refer to bits 8 and 12 through 15
6	Increment (true); Nonincrement (false)	-
7	Change mode (true); Select mode (false)	-
8	Panel location register data	Refer to bits 5 and 12 through 15
9	Write mode (true); Read mode (false)	-
10	N/A	Ground
12-15	Panel location register data	Refer to bits 5 and 8

4.12.2.2 Data Input Devices

A key-pad array and a Test and Verification Loader (TVL) provide a user with two optional data input devices to the CPU. The TVL is primarily intended to load diagnostic programs from a tape cassette into main memory. For a complete explanation of the TVL, refer to the Type LDU9101 Test and Verification Loader Manual (Order No. FL97).

Sixteen single-action pushbutton switches (keys) comprise the key-pad array. Each key is assigned a unique hexadecimal character (i.e., 0 through 9/A through F) that is inscribed on its keycap; the keys are used to input data to a hexadecimal encoder. Activating a key (keystroke) generates its corresponding hexadecimal character code (MPHEX1 through MPHEX4) and a unique keystroke (MPHEXS) signal at the encoder output to form a register address or a hexadecimal data character to modify a register. For further details on the key-pad array, refer to the discussion entitled Select/Change Modes in subsecton 4.12.2.4.3.

4.12.2.3 Register Displays

The location and contents indicator fields from the control panel display network enable the user to interrogate or modify any of the selected CPU registers (see Table 2-2).

4.12.2.3.1 LOCATION Display

The LOCATION display consists of two hexadecimal indicator assemblies that are designed to display hexadecimal characters H1 and H2. Both indicators monitor the internal bus to display (ENRDSP) the selected CPU register address that resides in the control panel location register (see Figure 4-16).

4.12.2.3.2 CONTENTS Display

The CONTENTS display consists of five hexadecimal indicators (H4 through H8) as shown in Figure 4-16, and is designed to display the hexadecimal character contents of a CPU addressable 16-bit or 20-bit register (see Table 2-2). Data is stored into the display from internal bus bits 0 through 15 when the Display Latch Enable (MPDLAT) and/or the Display Upper Latch (MPDULT) command is received and decoded in the panel.

4.12.2.4 Panel Controls/Indicators

To implement a panel initiate CPU operation, the CPU at predetermined intervals gates, via ENMPBI, the key-pad output and current panel status (see Table 4-5) onto the internal bus for firmware interrogation and implementation of one of the following CPU operational modes:

- Stop/Ready/Run
- Read/Write Memory
- Select/Change
- Plus 1 (Increment) Plus 0 (Non-Increment).

Concurrent with the preceding operations, the CPU has a fifth operational mode called LAF/SAF. In this mode, the CPU monitors the panel transmission lines to condition itself for or to implement one of the following panel initiated CPU operations:

- Load/Restart
- Execute.

The manner in which each panel status condition, its corresponding operation, and the associated indicators are activated is described in subsequent paragraphs. A description of two specific CPU status indicators (CHECK and DC ON) and the operation of the Clear (CLR) pushbutton is provided in Table 2-1.

4.12.2.4.1 Stop/Ready/Run Modes

Run mode enables the CPU to process instructions. To enter the Run mode, the CPU must first be initialized to its Stop (MPSTOP) state and the panel Ready (MPREDY) signal must be activated. Stop mode allows manual intervention and single instruction execution from the panel, and is activated by depressing the Stop (S) pushbutton (MPPBST). This causes the CPU to complete its current instruction and to perform the following:

- Halt all memory resident program processing activity
- Enter a continuous panel service loop
- Clear the RUN, READY, and TRAFFIC indicators
- Update the Instruction (D0) register with the next instruction for execution
- Equate the program counter to one greater than the next instruction memory location address
- Illuminate the STOP/STEP indicator.

To enter the Ready state, depress the control panel Ready (R) pushbutton (MPRUNN); this illuminates the Ready indicator and enables both the RUN and TRAFFIC indicators. Depressing the EXECUTE (E) pushbutton causes the CPU to process instructions, beginning with the instruction contained in the CPU instruction (F) register. Then the CPU control panel firmware generates a Traffic Set (MPTRAC) command, which is decoded in the panel to illuminate the RUN indicator, and to illuminate the TRAFFIC indicator. The Ready/Run states are also activated when a panel or CPU initiated Load operation is implemented.

4.12.2.4.2 Read/Write Mode

The following two control panel prerequisites are assumed completed by the user:

- The Increment (Plus 1)/Nonincrement (Plus 0) mode is enabled (refer to the following discussion).
- The Memory Address (A0) and Bus Data (B0) registers are initialized (refer to the TVL manual*) for the proper display in the LOCATION and CONTENTS displays, respectively.

Activating a control panel Read (R) or Write (W) operation: (1) clears Increment (MPINCR) and CPU Stop mode (MPSTOP), (2) conditions, via MPMEMA, the CPU for a Read/Write memory cycle through the internal bus and (3) illuminates the selected READ or WRITE indicator. Also, activating Write (MPWRW0) clears the control panel Load/Restart state. Depressing the Execute (E) pushbutton initiates a CPU memory cycle to perform a preselected control panel Read or Write operation. A Read memory (MPWRITE) cycle fetches data from a preselected memory location and stores it in the bus data register (CONTENTS display). A Write memory (MPWRITE) cycle takes data from the bus data register and stores it in a preselected memory location. Each successive depression of the Execute (E) pushbutton repeats a Read or Write memory cycle at the same or next sequential memory location as determined by the Plus 1 (Increment)/Plus 0 (Nonincrement) pushbutton selection.

4.12.2.4.3 Select/Change Modes

The Select/Change modes allow a user to select, display, and modify (from the key-pad array) selected CPU registers as described in the following text. Visual display of a register address and its contents is provided in the LOCATION and CONTENTS displays, respectively.

Select Mode

Activating Select (S) mode: (1) clears change (C) mode (MPCHGR), (2) extinguishes the CHANGE indicator, and (3) notifies the CPU control panel firmware through the CPU internal bus. It also enables, via MPCHNG, the location register to accept key-pad data to form one of 49 register addresses (see Table 2-2 and Figure 4-17).

To form a legitimate CPU register address, the location register is configured to accept alpha hexadecimal characters 8 through F in its leftmost character position (H1), and numeric hexadecimal characters 0 through 7 in its rightmost character

*The TVL manual is entitled: Type LDU9101 Test and Verification Loader Manual (Order No. FL97).

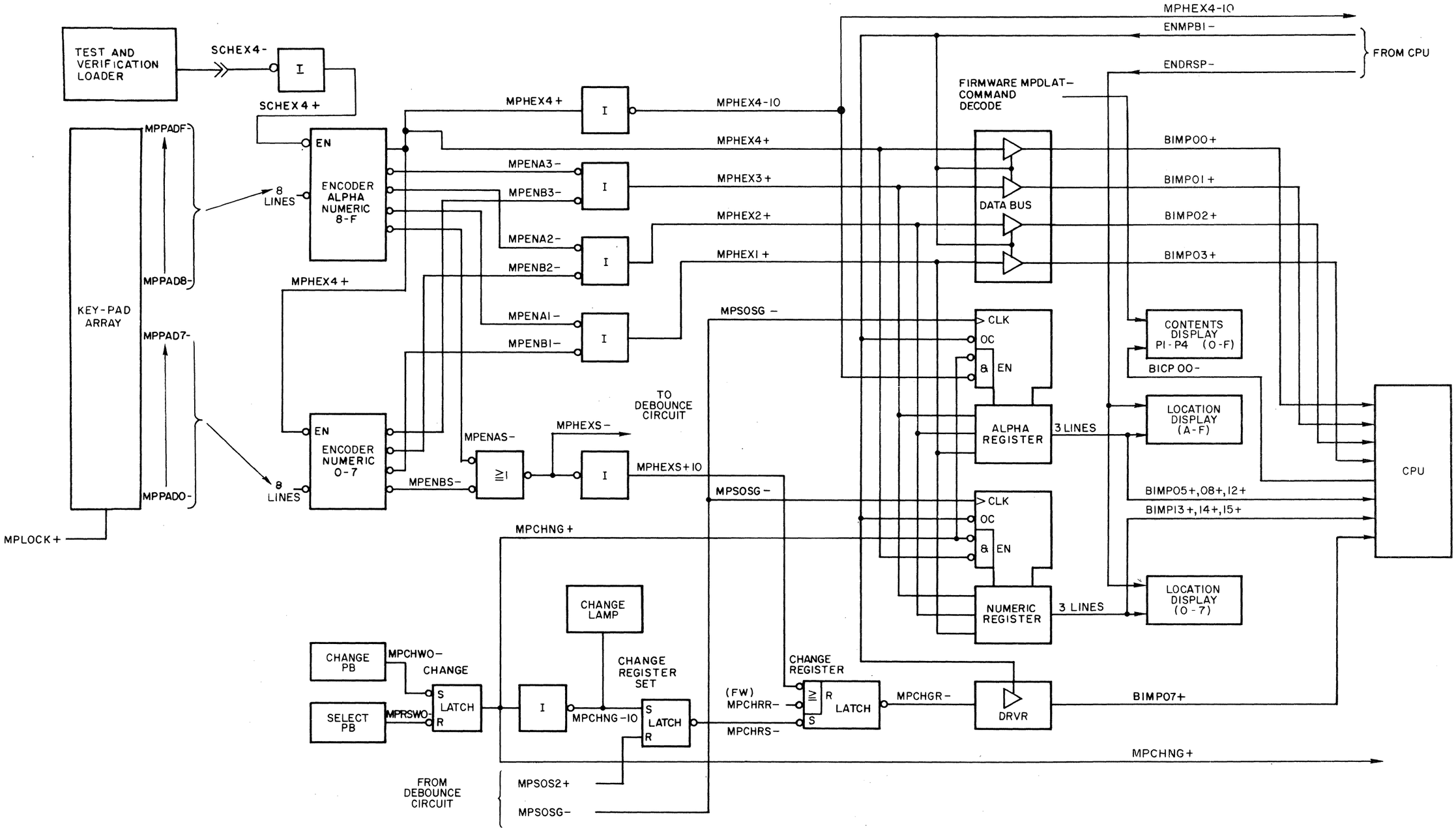


Figure 4-17 Key-Pad Array Logic

position (H2). Each additional depression of an alphanumeric character overlays a character in its respective location register position. This operation is allowed in both the CPU Stop and Run modes.

Change Mode

Activating Change (C) mode (MPCHNG): (1) clears Select mode, (2) illuminates the CHANGE indicator, and (3) notifies firmware through the internal bus. It also allows a user to change the contents of the following addressable CPU registers: A0, B0 through B7, D0 through D7, and E0. All remaining addressable CPU registers (e.g., C0 through C7) are read only type registers. Only 4 characters are needed to fill the data registers (D0 through D7). Base register (B0) requires just 4 characters when used as the source data in a memory panel write operation. The remaining changeable registers (A0, B1 through B7, and E0) are designated as address registers and 5 characters are entered when filling the selected register from the control panel.

Key-pad input activity to a selected CPU register occurs in the following manner and can be observed in the CONTENTS display. Depressing characters 0 through F activates, via MPHEXS, a 6-millisecond debounce network, which generates a 1.6-microsecond (MPS0S2) pulse. This is to allow sufficient time to stabilize any keystroke line transients and to activate a CPU flag (MPCHGR) that is required by the firmware to process a data character. With each depression of a key: (1) the CPU shifts all four/five data characters in the selected CPU register left by one character position, (2) truncates the leftmost character (H4), and (3) stores the corresponding character in the rightmost character position (H8). This process is repeated for each subsequent depression of a key.

NOTE

Refer to Table 2-2 in determining the number of characters required to fill the selected register. To avoid entering the wrong data, exercise care when entering the desired data pattern.

4.12.2.4.4 Increment/Non-Increment Functions

The Increment (MPPOW0+, Plus 1) and Nonincrement (MPPBP0+, Plus 0) panel functions enable firmware to initiate a Nonincrement or Increment Read/Write memory operation with each depression of the EXECUTE (E) pushbutton. Hence, a Read/Write memory cycle is performed at each new sequential memory address in an Increment operation (i.e., Plus 1 is on), or at the same memory address in a Nonincrement operation (i.e., Plus 0 is on).

4.12.2.4.5 Long/Short Address Form (LAF/SAF)

The central processor monitors the address mode (LAF/SAF) which is controlled by switch 4 of the rocker arm switch assembly. Switch 4 set to the LAF position, followed by a master clear (e.g., powering up or depressing CLR pushbutton), will cause the CPU address development to be 20 bits in length. Switch 4 set to the SAF position, followed by a master clear, causes the address fields to be 16 bits in length.

NOTE

With the portable panel and the basic panel installed in the same system, address mode control takes place in the basic panel. When LAF mode is selected, it will be displayed on both the portable panel and the basic panel.

4.12.2.4.6 Load/Restart Operation

Two fundamental CPU start-up operations, called Load and Restart, are included in the system to manually or automatically load a program (Bootstrap), or to restart the CPU from a predetermined memory location. A manual program Load (MPLOAD) is enabled with Stop mode active (MPRUNN) and when the control panel Load (L) pushbutton (MPPBLD) is depressed. Actual execution commences with the depression of the Execute (E) pushbutton. This causes the CPU to load a program from the program designated input device (Default operation) or from a user-selected input device. In a Default operation, a system Quality Logic Test (MPQLTA) is included and is implemented with the first depression of the Execute (E) pushbutton. This illuminates, via BSQLTA, the CHECK indicator, which is extinguished when the test is successfully completed (see Table 2-1). Depressing the Execute (E) pushbutton a second time implements the loading of a program from the default device. A user may select an alternate input device after the Quality Logic Test (QLT) has been completed. First, depress the Stop (S) pushbutton and alter the contents of Data register 1 (D1) to the desired BOOT channel number. Then, in sequence, depress the Ready (R) and Execute (E) pushbuttons. When the preceding load activity is successfully concluded, both the RUN (MPRUNN) and the TRAFFIC (MPTRAF) indicators are illuminated.

When the PANEL SECURITY switch is on (MPLOCK) or a basic panel is installed, an automatic program load or CPU restart sequence is available. The sequence is designed to automatically restart an unattended CPU when ac source power is restored to the CPU after a power failure. In a CPU with a non-volatile memory (e.g., core memory or memory save power supply), program restart commences at memory location 0. When the CPU contains a volatile memory (VOLMEM-), the Default Load operation previously described is performed. When auto boot is performed, the Memory QLT (lower 8K) will be executed. When auto restart is performed, the Memory QLT will be bypassed.

NOTE

If both the portable panel and the basic panel are installed in the same system, auto boot/ auto restart control takes place in the basic panel.

4.12.2.4.7 Execute Operation

The Execute (E) pushbutton is enabled when the Stop mode and the panel are active to initiate one of the following panel selected operations in the CPU:

- Read/Write Memory
- Execute Single Instruction
- Program Load/Restart.

Depressing the Execute (E) pushbutton sends a 1.6-microsecond strobe pulse, 6-milliseconds after its initial depression, to the CPU to allow firmware to implement the selected function.

4.12.2.5 Panel Lock Request

The control panel, when active and requiring service (e.g., display update, register change, mode change), transmits its locked state to the CPU. The 8-millisecond update cycle is controlled internally by the CPU when in the Run mode.

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

M&TO HARDWARE PUBLICATIONS
USER COMMENTS FORM

DOCUMENT TITLE: _____

PART NO.: _____

ORDER NO.: _____

ERRORS:

HOW DO YOU USE THIS DOCUMENT?

THEORY ☐

MAINTENANCE ☐

TROUBLESHOOTING ☐

OTHER: _____

_____

DOES THIS MANUAL SATISFY YOUR REQUIREMENTS?

YES ☐ NO ☐

IF NOT, PLEASE EXPLAIN _____

_____

_____

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

FIRST CLASS
Permit No. 39531
Waltham, Ma.

Business Reply Mail NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

HONEYWELL INFORMATION SYSTEMS INC.
200 SMITH STREET
WALTHAM, MA. 02154

MAIL STATION 872A
HARDWARE PUBLICATIONS, BILLERICA

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

FN28, REV. 1