Prepared for the
GEORGE C. MARSHALL
SPACE FLIGHT CENTER
Huntsville, Alabama

Final Report

Contract Number NAS8-29929

December 31, 1975

IBM Number 76W00013

Technical Description of Space Ultra Reliable Modular Computer (SUMC), Model II B

Prepared for the
GEORGE C. MARSHALL
SPACE FLIGHT CENTER
Huntsville, Alabama

Final Report

Contract Number NAS8-29929

December 31, 1975

IBM Number 76W00013

Technical Description of Space Ultra Reliable Modular Computer (SUMC), Model II B

IBM® Federal Systems Division, Space Systems/Huntsville, Alabama

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

LIST OF FIGURES

LIST OF FIGURES (CONT'D)

LIST OF TABLES

LIST OF TABLES (CONT'D)

APPENDICIES

SUMC-IIB DEVELOPMENT (FINAL REPORT)

1.0 INTRODUCTION

The end result of contract NAS8-29929 is the SUMC-IIB computer, also called the IBM-HTC. This computer is a general purpose digital computer implemented with flexible hardware elements and microprogramming to enable low cost customizing to a wide range of applications. It executes the S/360 standard instruction set to maintain problem state compatibility. Memory technology, extended instruction sets, and I/O channel variations are among the available options.

The design provides the following features:

- Modular
  - Minimum number of part types (LSI chips and functional modules)
  - Easy expansion of data flow
  - Easy changing of architecture
- High reliability via LSI and hybrid packaging
- Small size via LSI and hybrid packaging
- Uses mature logic technology
  - TTL (Schottky) processing
  - Master slice chip organization
  - Customized via single metallization mask
  - Design support by Engineering Design System (EDS) (logic recording, groundrule checking, chip wiring, test pattern generation, and dynamic simulation (chip through system)).
  - Good delay power product
- High logic testability.
- Choice of multiple memory technologies.
- Add memory without changing the design.
- Easy addition of custom instructions.

- Flexible I/O structure
  - All I/O can be on a single channel for simple cabling.
  - DMA can be separated for faster response.
  - CPU can be locked out for burst I/O.
  - Buffered I/O permits simple device design for external controlled I/O.
- Extensive "Built-in-Tests" (hardware and microprogram).
- Memory store protection.

This report provides a detailed description of the SUMC-IIB computer. Section 2 summarizes the system design; Section 3 describes the packaging; Section 4 presents a detailed description of the computer organization and functional design, with Appendix A giving a detailed description of the logic modules used in the design. The functional design of the power supply is in Section 5.

SECTION 2

SYSTEM DESCRIPTION

The SUMC-IIB is a microprogram controlled, general-purpose computer implementing the IBM System/360 instruction set. This feature allows the SUMC-IIB to be Problem State compatible with a widely known ground-based computer system. Potential is thus offered for software development simplification through reduced programmer training and utilization of existing software packages. Problem-state programs written for the System/360 Standard Instruction Set will execute identically on the SUMC-IIB and System/360 or System/370.

The basic SUMC-IIB supports 83 of the 87 instructions in the IBM System/360 Standard Instruction Set, which is a 32-bit fixed-point instruction set. The four instructions not supported are associated with I/O. The SUMC-IIB provides three instructions which control the timers, I/O, and storage protection, which are not a part of the Standard S/360 instruction set. The four I/O instructions not supported and the three additional instructions provided are not used when writing programs in the problem state. They are used by a supervisory program or operating system. Thus, application program modules for the SUMC-IIB computer can be written and debugged on any S/360 - S/370 computer without any special support software.

SUMC-IIB with the S/360 Standard Instruction Set uses 1024 words of microprogram. A fully developed, optional, extended instruction set is available for the SUMC-IIB which requires an additional 512 words of microprogram. The extended instruction set includes three types of instructions; short precision fixed-point, double precision fixed-point, and standard precision floating-point. SUMC-IIB support software supports these optional instructions. The short precision option consists of 53 additional instructions which deal with 16-bit fixed-point operands. These instructions generally execute faster than their counterparts in the basic SUMC-IIB instruction set, which primarily operates on 32-bit operands.

The double precision fixed-point option consists of 10 additional instructions which operate with 64-bit fixed-point operands.

The floating-point option uses microprogramming and fixed-point arithmetic hardware to implement the 22 standard length (32-bit) S/360 floating-point instructions. Full compatibility with S/360 is maintained on these instructions. The four double-length floating-point registers required by S/360 floating-point are implemented (along with the 16 general registers) in the scratch pad memory, SPM.

Problem-state programs can be written in Fortran, PL/1, HAL or any compiler or assembler language which does not use double precision floating-point or decimal instructions. System/360 exception monitoring is fully supported except for the storage protect feature of SUMC-IIB which is implemented differently from the optional System/360 storage protection. The differences are explained in Section 2.2.

2.1 SUMC-IIB CHARACTERISTICS

The hardware developed for SUMC-IIB is modular at three levels, chip, module, and slice. Partitioning of the hardware at all three levels was done to maximize the general utility of the part. Details of the chips and modules are contained in Appendix A. Four functional slices have been developed for SUMC-IIB: CPU/IO, PROM/MEMORY, MEMORY, and POWER SUPPLY.

All of these slices have compatible structures to allow them to be fastened together in a "stack." The ends are closed with covers and the computer is mounted with one edge of each slice attached to a cold plate for efficient cooling. Memory can be added by attaching additional memory slices (See Figure 2.1-1.

As illustrated by System/360 many computer models can be designed around a single architecture, depending upon the performance requirements of the intended application. The SUMC-IIB uses a 16-bit data flow, 134 gate TTL logic chips, N-channel MOS memories, a microprogram control store and hybrid packaging to provide the computer characteristics listed in Table 2.1-1. The hardware organization and data flow can be seen in the block diagram of Figure 2.1-2.

Performance calculations such as KOPS (thousands of operations per second) are application sensitive for a given computer and are architecture sensitive when comparing different computers. Therefore, the SUMC-IIB performance is indicated by Table 2.1-2 which shows the performance of representative instructions in the baseline and optional instruction sets. Appendix B is a complete list of the instructions and execution times.

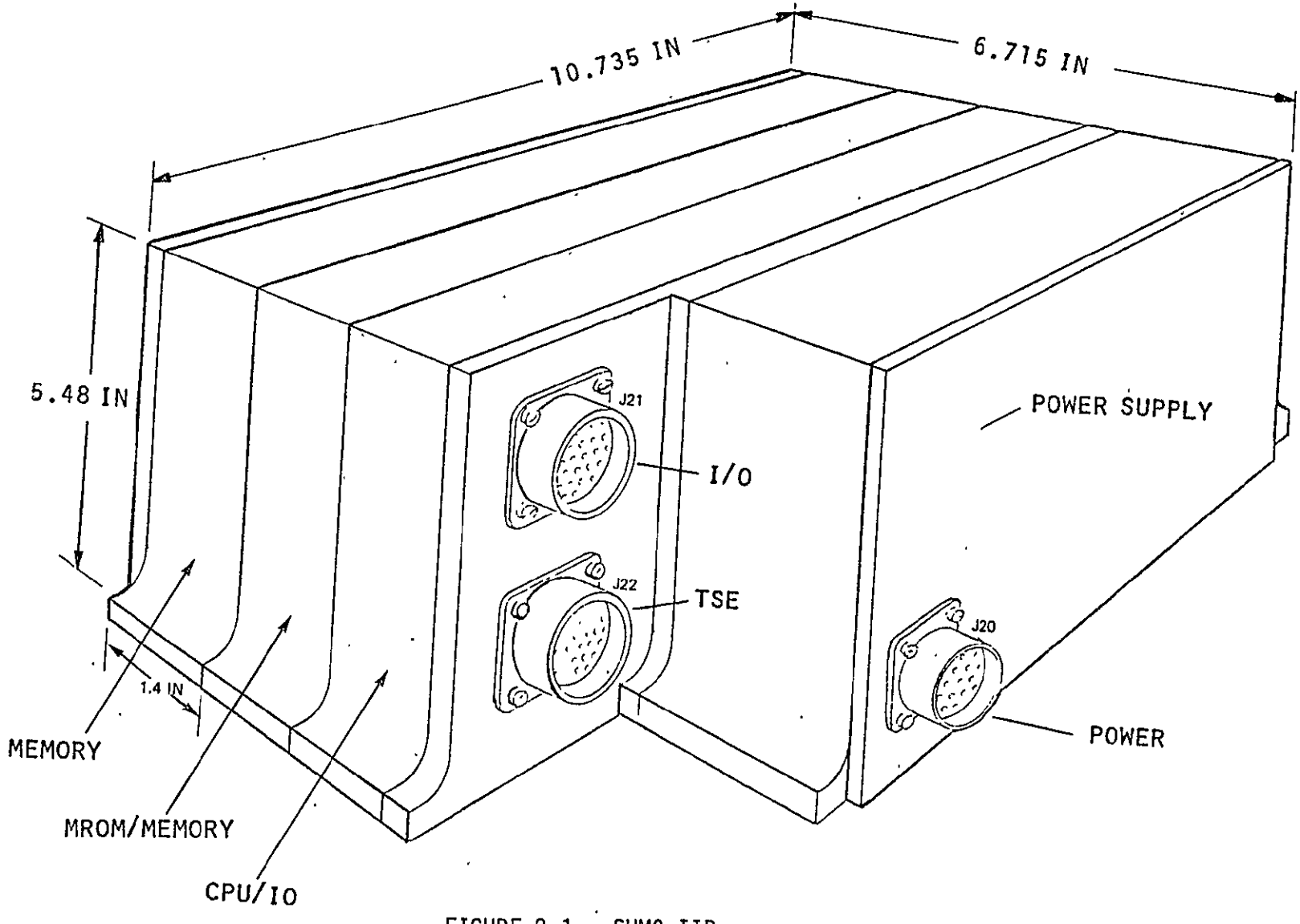


FIGURE 2-1. SUMC-IIB

Table 2.1-1. SUMC-IIB Characteristics

Central Processing Unit	Design	General-register, functionally partitioned in four segments 1. Data Flow 2. Sequence Control 3. Timing and miscellaneous functions 4. Architecture-Dependent functions
Machine Organization	Type	General-purpose, stored program
	Number System	parallel Binary, fixed-point, integer.
	Instruction Word Length	16, 32 and 48 bits
	Fixed-Point Data Word Length	8, 16 and 32 bits (most operations use 32 bits)
	General Register	16 32-bit registers in hardware
	Instruction Set	IBM System/360 standard instruction set
	I/O Interrupt	Single hardware level with multilevel interrupt capability through software mechanization or additional external hardware.
	Execution Time	Instruction (S/360 formats)
		Add (Register-to-Register) (32 x 32 bits) 2.0 μs
		Add (Register-to-Storage) (32 x 32 bits) 2.8 μs
		Multiply (32- x 32-bit operand) 30.4 μs
Main Storage Monolithic N-channel MOS	Density	16,384 x 18 bits per slice (32768 bytes)
	Access Time	550 ns
	Cycle Time	550 ns
	Addressable Unit	8-bit byte
	Capacity	65,536 bytes
	Storage Protect	Has storage protect feature
Small modular[1] core memory	Density	4096 x 17 bits per slice (8192 bytes)
	Access Time	600 ns
	Cycle Time	1.33 μs
	Addressable Unit	8-bit byte
	Capacity	65,536 bytes
	Storage Protect	Has storage protect feature
Input/Output	Externally Initiated	Direct memory access, buffered I/O
	Program Initiated	Direct I/O, external interrupts
	Data Interface	16-bit parallel, data and address
	Data Transfer Rate	150,000 to 750,000 words second
	External Channels	Multiplexing to 16 or more distinct channels

[1]Designed for Core Memory

Table 2.1-1. SUMC-IIB Characteristics. (Continued)

Logic Circuits	Class	Monolithic integrated
	Type	Low-power, bipolar, transistor-transistor logic (TTL); large-scale integration (LSI)
	Package	LSI chips mounted on cermaic, thick-film, multilayer modules.
Power System (Hybrid)	Primary Power	28 ± 4 VDC
	Input Power	o 100 W for 8192 bytes of core memory[2] o 94 W for 16,384 bytes of monolithic memory o 230 W max., 24 to 32 VDC[3] o Output regulated voltages and signals +5 VDC 2% @ 25 A +8.5 VDC 5% @ 3 A -3.15 VDC 2% @ 50 mA o Power-On-Reset - logic level reset at Power On/Off o 9.09 MHz Clock - clock signal to processor logic
	Features	Overvoltage and overcurrent protection, transient power/signal protection, power sequencing, hybrid circuit modules, external pluggable module
Physical	Volume	280 in } Includes P.S. and 16K - Bytes of storage
	Weight	13.9 lb
	Size	5.2 x 5.50 x 10.75 in
	Construction	Pluggable structural slices
	Cooling	Cold plate/heat sink, can be adapted for indirect air, conductive cooling
	Environment	Designed to meet MIL-E-5400, Class 2X

2. Calculated

3. Maximum power supply capability of 150 watts at 65% efficiency.

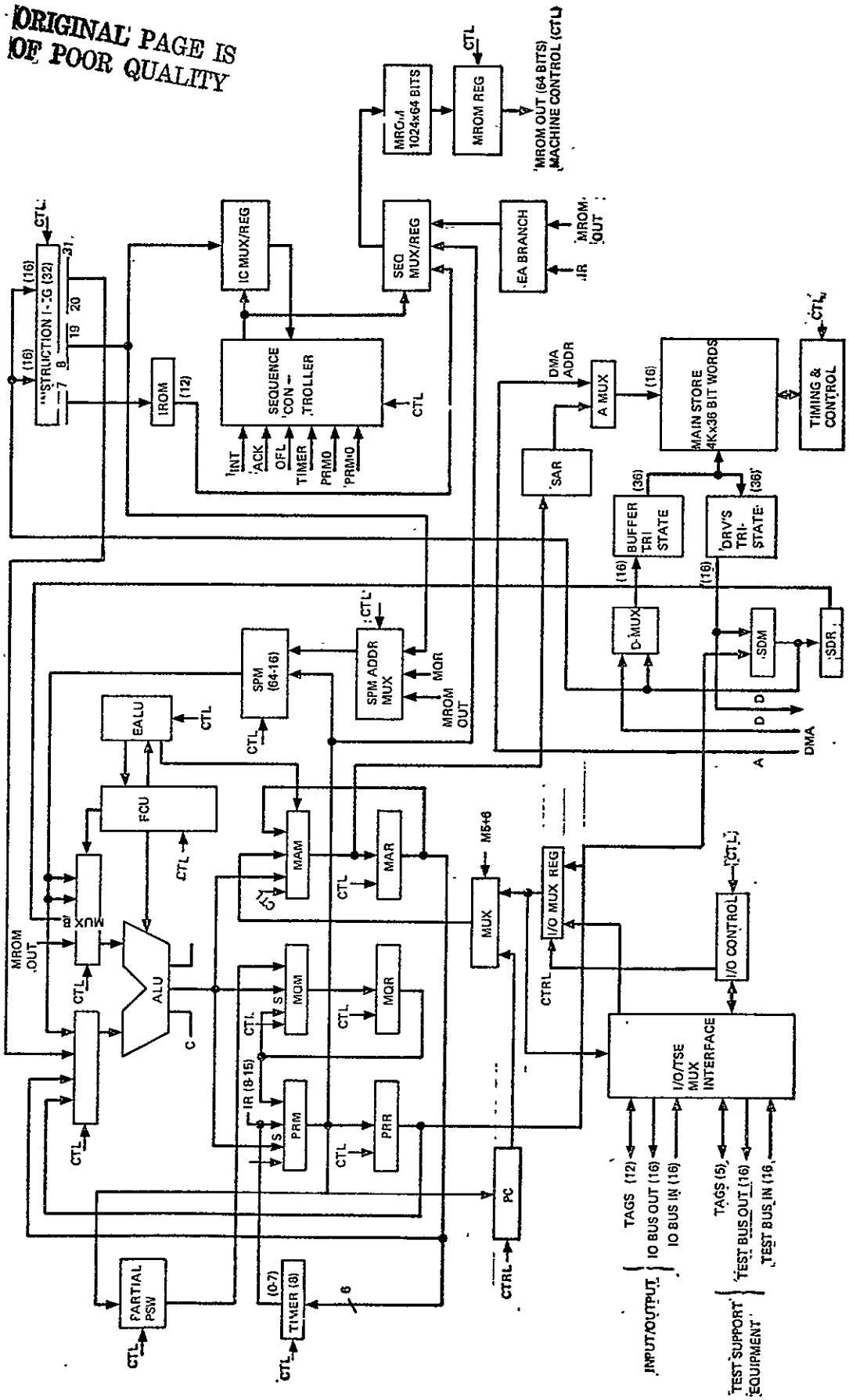


Figure 2.1-2. SUMC IIB Block Diagram

Table 2.1-2. Typical Execution Times of SUMC-IIB Instructions

Instruction	Execution Time (μ sec)			
	Short (16 x 16)	Standard (32 x 32)	Extended (64 x 64)	Floating** (32 x 32)
Add/Sub/Logic* (RR)	1.7	2.0	6.1	20.8+
Add/Sub/Logic (RX)	2.2	2.8	6.1	21.8+
Multiply (RX)	7.8	30.4	-	33.8+
Divide (RX)	15.1	51.8	-	48.6+
Branch and Link	-	3.4	-	-
Shift (Avg 3-6 bits)	3.3	3.7	-	-
Compare (RX)	2.6	3.1	6.4	16.2+

*No logical operations are provided in the extended precision or floating-point instructions.

**The + in the execution times are for exponent alignment and/or normalization (typically in the range 1 to 3 μ seconds extra).

2.2 SYSTEM ARCHITECTURE

The System/360 architecture has been defined as baseline for the SUMC. The following discussion delineates the SUMC-IIB consistency with the baseline architecture and represents the architecture to be applied to the Tug application.

2.2.1 System/360 Compatibility

The SUMC-IIB is problem state compatible with the IBM System/360 Standard Instruction Set. Short floating-point instructions are supported along with 16-bit and 64-bit fixed-point instructions in an optional microprogram extension. Decimal instructions and long format floating-point instructions are not currently supported. Problem state programs written for the S/360 standard instruction set will execute correctly on the SUMC-IIB without change. Programs written in high level languages for S/360 such as FORTRAN and PL/1 can be executed on the SUMC-IIB provided that long format floating-point and decimal variables are not used.

The SUMC-IIB is Supervisor State compatible with the IBM System/360 with the following exceptions.

- I/O - The I/O portion of the SUMC-IIB provides the means of communication between the system I/O and test support equipment (TSE) and the CPU/main store (MS). In the SUMC-IIB the I/O is implemented as a 16-bit parallel channel providing direct I/O, buffered I/O, and external interrupts, and stand-alone (SA) direct memory access (DMA) interface. There is only one I/O instruction, the SIO (Start I/O) instruction, which controls direct I/O. All other I/O is device controlled. The SIO instruction fetches a 16-bit I/O command from main memory and transmits a 16-bit data word to/from one of the general registers.

- Timer - The SUMC-IIB has a real time clock and an interval timer, each containing both hardware and microprogrammed elements. Both are accessed by using the TMRS instruction. The S/360 interval timer in memory location 80 is not supported.

  The interval timer is a 16-bit decrementing counter, that is decremented every 112.64 microseconds. It has a maximum of 7.38 seconds. The real time clock is a 32-bit incrementing counter, that is incremented every 112.64 microseconds. It has a maximum value of 5 days, 14 hours, 23 minutes and 5 seconds. The TMRS instruction is used to read either of the timers into a general register. When the TMRS instruction is used to load either of the timers from main storage, the old value of the timer is placed into a general register so that the timer may be read and loaded without an intervening step. Problem programs may read either of the timers directly, but only the supervisor is permitted to load the timers. Duration of the timer can be extended by programming.

- Storage Protect - The size of the storage protect blocks in the SUMC-IIB is 1024 bytes (512 halfwords) and the bit configuration of the instruction SSK (Set Storage Key) operand has been changed. The instruction ISK

(Insert Storage Key) has not been implemented in the SUMC-IIB. A two bit storage protect key is used for each block; bit 30 of the designated register indicates Central Processing Unit storage protect and bit 31 indicates Direct Memory Access storage protect.

The SUMC-IIB supports all S/360 exception monitoring for the instructions implemented.

2.2.2 Exception Monitoring

The exceptional conditions are primarily programming errors; all exceptions cause a program interruption. The errors monitored are:

1. Operation Exception. Execution of an unassigned operation code was attempted.

2. Privileged Operation. A privileged operation was encountered in the problem state.

3. Protection Exception. An instruction tried to store into a protected location.

4. Addressing Exception. An address specifies data, an instruction or a control word outside of the available storage.

5. Specification Exception. A data, instruction or control word address does not specify an integral boundary for the unit of information.

6. Parity Error. A memory parity error has occurred.

7. Fixed Point Overflow. An overflow has occurred during an arithmetic operation.

8. Execution Exception. The object of an execute instruction was another execute instruction.

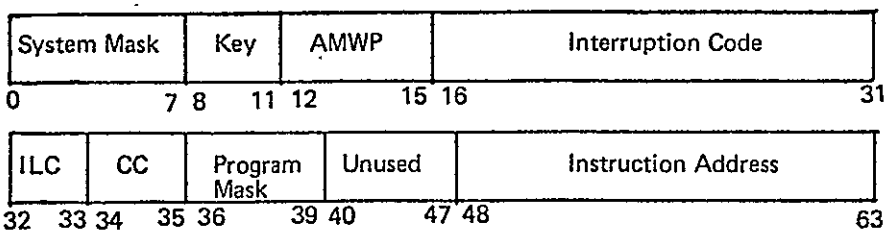
9. Fixed-Point Divide Exception. The quotient cannot be expressed in a 32-bit signed fixed-point number.

10. Data Exception. Data of the second operand in a CONVERT TO BINARY instruction is not in packed decimal format.

If the floating-point instructions are supported there are four additional exceptions monitored to maintain S/360 compatibility (exponent-overflow, exponent-underflow, loss of significance, and floating-point divide).

2.2.3 Program Status Word

A double word, the program status word (PSW), contains the information required for proper program execution. The PSW includes the instruction address, condition code, and other fields to be discussed. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. The active or controlling PSW is called the "current PSW." By storing the current PSW during an interruption, the status of the CPU can be preserved for subsequent inspection. By loading a new PSW or part of a PSW, the state of the CPU can be initialized or changed. Figure 2.2-1 shows the PSW format.



0-7	System Mask
0	I/O Mask
1	
2	
3	Unused
4	Don't Care
5	
6	
7	Timer Mask
8-11	Must be 0
12	ASCII(A)
13	Machine-Check Mask (M)
14	Wait State (W)
15	Problem State (P)
16-31	Interruption Code
32-33	Instruction Length Code (ILC)
34-35	Condition Code (CC)
36-39	Program Mask
36	Fixed Point Overflow Mask
37	Unused
38	Don't Care
39	
40-47	Unused
48-63	Instruction Address

Figure 2.2-1. Program Status Word Format

Interruptions are taken only when the CPU is interruptable for the interruption source. The system mask, program mask, and machine check mask bits in the PSW may be used to mask certain interruptions. When masked off, an interruption either remains pending or is ignored. The system mask may cause I/O and timer interruptions to remain pending, and the machine-check mask may cause machine hard stops. The program mask may cause fixed point overflow interruptions to be ignored. Other interruptions cannot be masked off.

An interruption always takes place after one instruction execution is finished and before a new instruction execution is started. However, the occurrence of an interruption may affect the execution of the current instruction. To permit proper programmed action following an interruption, the cause of the interruption is identified and provision is made to locate the last executed instruction.

2.3 SYSTEM STRUCTURE

2.3.1 Main Storage

The SUMC-IIB has a 16,384 byte (4,096 word) storage that is expandable to 65,536 bytes (16,384 words). Each word is 4 bytes (32 bits) long. The system transmits information between main storage and the CPU in units of eight bits (plus parity), or a multiple of eight bits at a time. Each eight bit unit of information is called a byte, the basic building block of all formats.

When the length of a field is not implied by the instruction operation code, but is stated explicitly, the information is said to have variable field length. This length can be varied in one-byte increments.

2.3.2 Addressing

Byte locations in storage are consecutively numbered starting with 0; each number is considered the address of the corresponding byte. A group of bytes in storage is addressed by the leftmost byte of the group. The number of bytes in the group is either implied or explicitly defined by the operation. The addressing arrangement uses a 16-bit binary address to accommodate a maximum of 65,536 byte addresses. This set of main storage addresses includes some locations reserved for special purposes. Several techniques are being evaluated for extending the addressing capability beyond the current 64K-bytes.

An addressing exception is recognized when any part of an operand is located beyond the maximum available capacity of an installation. The addressing exception causes a program intervention.

2.3.3 Central Processing Unit

The Central Processing Unit (CPU) contains: the facilities for addressing main storage, for fetching or storing information, for arithmetic and logical processing of data, for sequencing instructions in the desired order, and for initiating the communication between storage and external devices.

The system control section provides the normal CPU functions necessary to execute the instructions.

The CPU provides 16 general registers for fixed-point operands.

The general registers can be used as index registers, in address arithmetic and indexing, and as accumulators in fixed-point arithmetic and logical operations. The registers have a capacity of one word (32 bits). The general registers are identified by numbers 0-15 and are specified by a four-bit R field in an instruction Some instructions provide for addressing multiple general registers by having several R fields.

2.3.3.1 Arithmetic and Logic Unit

The arithmetic and logic unit can process binary integers of fixed length and logical information of either fixed or variable length.

- Fixed-Point Arithmetic - The basic arithmetic operand is the 32-bit fixed-point binary word. Sixteen-bit halfword operands may be specified in most operations for improved performance or storage utilization. To preserve precision, some products and all dividends are 64 bits long.

  Additions, subtractions, multiplications, divisions, and comparisons are performed upon one operand in a register and another operand either in a register or from storage. Multiple precision operation is made convenient by the two's-complement notation and by recognition of the carry from one word to another. A word in one register or a double word in a pair of adjacent registers may be shifted left or right. A pair of conversion instructions -- CONVERT TO BINARY and CONVERT TO DECIMAL -- provides transition between decimal and binary radix (number base) without the use of tables. Multiple register loading and storing instructions facilitate subroutine switching.

- Logical Operations - Logical information is handled as fixed or variable length data. It is subject to such operations as comparison, translation, bit testing, and bit setting. When used as a fixed length operand, logical information can consist of either one, four, or eight bytes and is processed in the general registers.

2.3.3.2 Program Execution

The CPU program consists of instructions, index words, and control words specifying the operations to be performed. This information resides in main storage and general registers, and may be operated upon as data.

- Instruction Format - The length of an instruction format can be one, two or three halfwords. It is related to the number of storage addresses necessary for the operation. An instruction consisting of only one halfword causes no reference to main storage. A two halfword instruction provides one storage address specification; a three halfword instruction provides two storage address specifications. All instructions must be located in storage on integral boundaries for halfwords. Figure 2.3-1 shows five basic instruction formats.

  The five basic instruction formats are denoted by the format codes RR, RX, RS, SI, and SS. The format codes express, in general terms, the operation to be performed. RR denotes a register-to-register operation; RX, a register-and-indexed storage operation; RS, a register-and-storage operation; SI, a storage and immediate-operand operation; and SS, a storage-to-storage operation. An immediate operand is one contained within the instruction.

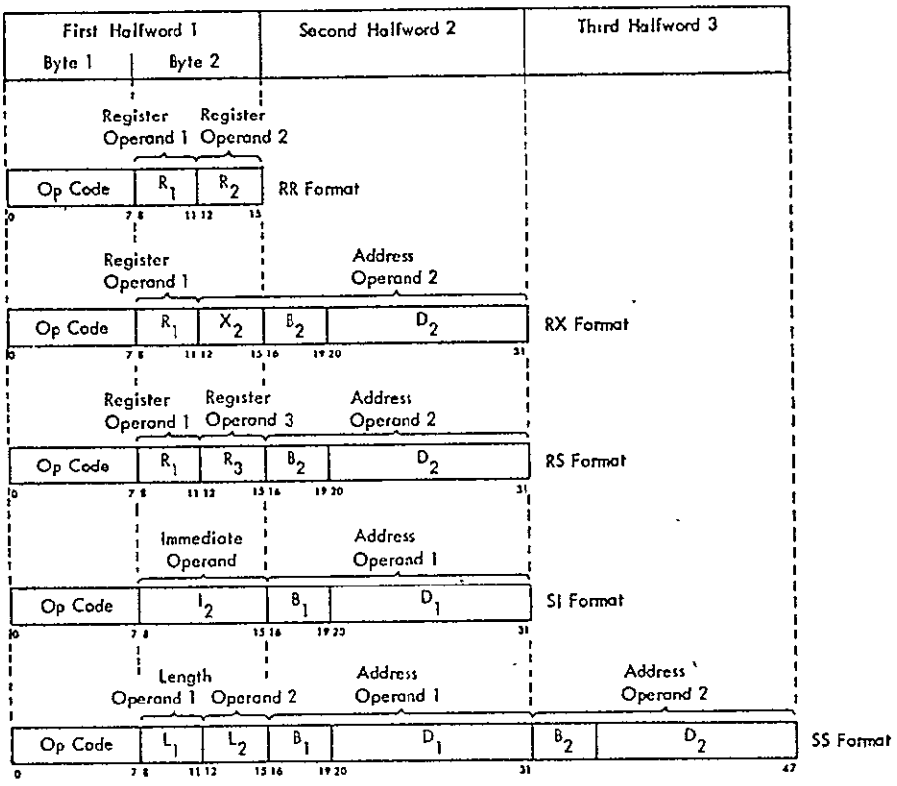
  For purposes of describing the execution of instructions, operands are designated as first and second operands and, in the case of branch-on-index instructions, third operands. These names refer to the manner in which the operands participate. The operand to which a field in an instruction format applies is generally denoted by the number following the code name of the field, for example, R_1, B_1, L_2, D_2.

  In each format, the first instruction halfword consists of two parts. The first byte contains the operation code (op code). The length and format of an instruction are specified by the first two bits of the operation code.

- Address Generation - For addressing purposes, operands can be grouped in three classes: explicitly addressed operands in main storage; immediate operands placed as part of the instruction stream in main storage; and operands located in the general registers.

To permit the ready relocation of program segments and to provide for the flexible specifications of input, output, and working areas, all instructions referring to main storage have been given the capacity of employing a full address.

The address used to refer to main storage is generated from the following three binary numbers.



INSTRUCTION LENGTH RECORDING

INSTRUCTION LENGTH	INSTRUCTION FORMAT
One halfword	RR
Two halfwords	RX
Two halfwords	RS or SI
Three halfwords	SS

Figure 2.3-1. Five Basic Instruction Formats

Base Address (B)

Base Address (B) is a 16-bit number contained in a general register specified by the program in the field of the instruction. The B field is included in every address specification. The base address can be used as a means of static relocation of programs and data. In array-type calculations, it can specify the location of an array and, in record-type processing, it can identify the record. The base address provides for addressing the entire main storage. The base address may also be used for indexing purposes.

Index (X)

Index (X) is a 16-bit number contained in a general register specified by the program in the X field of the instruction. It is included only in the address specified by the RX instruction format. The RX format instructions permit double indexing; i.e., the index can be used to provide the address of an element with an array.

Displacement (D)

Displacement (D) is a 12-bit number contained in the instruction format and is included in every address computation. The displacement provides for relative addressing up to 4095 bytes beyond the element or base address. In array type calculations the displacement can be used to specify one of many items associated with an element. In the processing of records, the displacement can be used to identify items within a record.

In forming the address, the base address and index are treated as unsigned 16-bit positive binary integers. The displacement is similarly treated as a 12-bit positive binary integer. The three are added as 16-bit binary numbers, ignoring overflow. Since every address included a base, the sum is always 16 bits long. The address bits are numbered 16-31 corresponding to the numbering of the base address and

The program may have zeros in the base address, index, or displacement fields. A zero is used to indicate the absence of the corresponding address component. A base or index or zero implies that a zero quantity is to be used in forming the address, regardless of the contents of general register 0. A displacement of zero has no special significance. Initialization, modification, and testing of base addresses and indexes can be carried out by fixed point instructions, or by BRANCH AND LINK, BRANCH ON COUNT, or BRANCH-ON-INDEX instructions.

The logic for forming the effective address (EA) is shown in flow chart form in Figure 2.3-2. The basic address in an indexed, base displacement is calculated as:

$$EA = D + (B) + (X)$$

where ( ) denotes "the contents of".

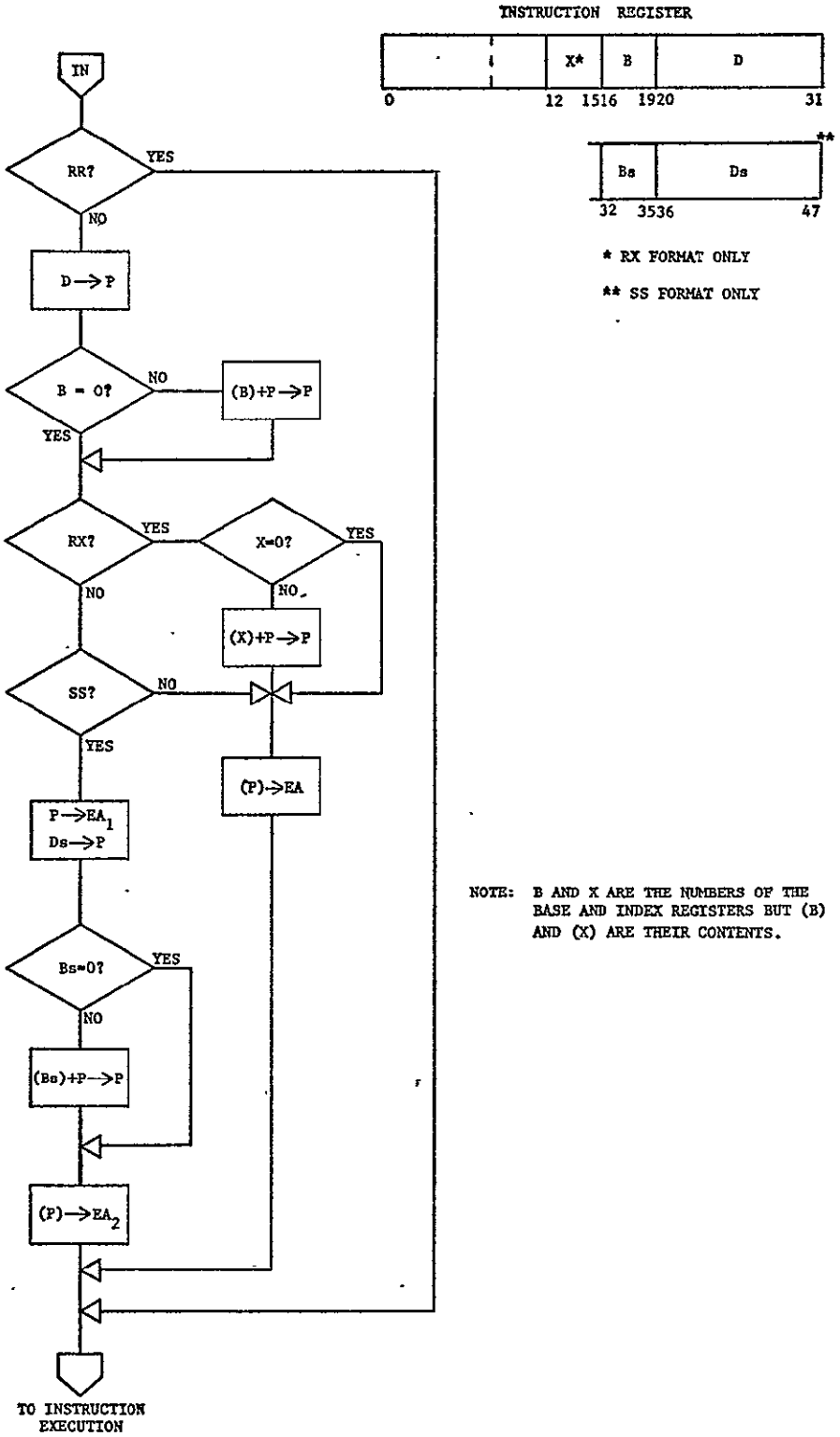


Figure 2.3-2. Representative Flow Diagram of the S/360-370 Effective Address Calculation

Sequential Instruction Execution

Normally, the operation of the CPU is controlled by instructions taken in sequence An instruction is fetched from a location specified by the instruction address in the current PSW. The instruction address is held in the SPM. The instruction address is increased by the number of bytes in the instruction fetched to address the next instruction in sequence. The instruction is then executed and the same steps are repeated using the new value of the instruction address.

A change from sequential operation may be caused by branching, status switching, interruptions, manual intervention, or by the execute instruction.

Branching

The normal sequential execution of instructions is changed when reference is made to a subroutine, when a two way choice is encountered, or when a segment of coding such as a loop is to be repeated. All these tasks can be accomplished with branching instructions. Provision is made for subroutine linkage, permitting not only the introduction of a new instruciton address but also the preservation of the return address and associated information.

Loop control can be performed by the conditional branch when it tests the outcome of address arithmetic and counting operations. For some particularly frequent combinations of arithmetic and tests, the instructions BRANCH ON COUNT and BRANCH ON INDEX are provided. These specialized branches provide increased performance for these tasks.

2.3.3.3 Machine States

Running or Waiting State

In the running state, instruction fetching and execution proceed in the normal manner. The wait state is normally entered by the program to await an interruption for example, an I/O interruption. In the wait state, no instructions are processed the timer is updated, and the I/O and external interruptions are accepted, unless masked. Running or waiting state is determined by the setting of bit 14 in the PSW.

Masked or Interruptible State

The CPU may be interruptible or masked for I/O, timer, machine-check, and some program interruptions. When the CPU is interruptible for a class of interruptions, these interruptions are accepted. When the CPU is masked the I/O and timer interruptions remain pending, whereas program interruptions are ignored. The interruptible states of the CPU are changed by changing the mask bits of the PSW.

Supervisor or Problem State

In the problem state, I/O and a group of control instructions are invalid. In the supervisor state, all instructions are valid. The choice of problem or supervisor state is determined by bit 15 of the PSW. The supervisor state (priviledged) instructions are: LOAD PSW, SET SYSTEM MASK, START IO, TIMER SET, SET STORAGE KEY, and SET PROGRAM MASK.

2.3.4 Interruption

The interruption system permits the CPU to change state as a result of conditions external to the system, in input/output units, or in the CPU itself. Five classes of interruption considitions are possible: I/O, program check supervisor call, external, and machine check.

Each class has two related PSWs called "old" and "new" in unique main storage locations (Table 2.3-1). In all classes, an interruption involves merely storing the current PSW in its "old" position and making the PSW at the "new" position the current PSW. The "old" PSW holds all necessary status information of the system existing at the time of the interruption. If, at the conclusion of the interruption routine, there is an instruction to make the old PSW the current PSW, the system is restored to the state prior to the interruption and the interrupted routine continues.

Table 2.3-1. Permanent Storage Assignments

	ADDRESS	LENGTH	PURPOSE
0	0000 0000	Double-Word	Initial Program Loading PSW
8	0000 1000	Double-Word	Unused
16	0001 0000	Double-Word	Unused
24	0001 1000	Double-Word	External old PSW
32	0010 0000	Double-Word	Supervisor call old PSW
40	0010 1000	Double-Word	Program old PSW
48	0011 0000	Double-Word	Machine check old PSW
56	0011 1000	Double-Word	Input/Output old PSW
66	0100 0000	Half-Word	Buffered I/O Status Word
72	0100 100-0	Word	Channel Address Word (Buffered I/O Table)
76	0100 1100	Word	Unused
80	0101 0000	Word	Unused
84	0101 0100	Word	Unused
88	0101 1000	Double-Word	External new PSW
96	0110 0000	Double-Word	Supervisor call new PSW
104	0110 1000	Double-Word	Program new PSW
112	0111 0000	Double-Word	Machine check new PSW
120	0111 1000	Double-Word	Input/Output new PSW

Interruptions are taken only when the CPU is interruptible for the interruption source. The system mask, program mask, and machine check mask bits in the PSW may be used to mask certain interruptions. When masked off, an interruption either remains pending or is ignored. The system mask may cause I/O and timer interruptions to be held pending, and the machine-check mask may cause machine hard stops. Other interruptions cannot be masked off.

An interruption always takes place after one instruction execution is finished and before a new instruction execution is started. However, the occurrence of an interruption may affect the execution of the current instruction. To permit proper programmed action following an interruption, the cause of the interruption is identified and provision is made to locate the last executed instruction.

External Interrupts

External interrupts from two sources can occur: timer interrupts (when the interval timer underflows) and interrupts from the interrupt key on the test console. These interrupts are serviced between instructions.

These two types of external interrupts may be masked off. Timer interrupts are masked by bit 7 of the system mask (PSW bit 7), as usual. A 1 enables timer interrupts; a 0 masks them and the interrupt remains pending. External interrupt key interrupts are masked by system mask bit 0 (PSW bit 0) which is also used to mask I/O interrupts (refer to paragraph 2.3.5). If a key interrupt is disabled, it remains pending and the channel is Hung. Other bits of the system mask are ignored and need not be zero upon PSW load or in the SSM instruction.

In SUMC-IIB, the two types of external interrupts are not presented simultaneously if they occur simultaneously of if they are enabled simultaneously. In either case, the timer interrupt is taken and the key interrupt remains pending. The timer interrupt has interrupt code X'0080' and the external key interrupt has interrupt code X'0040'.

Program Check Interrupts

The following program exceptions are monitored in SUMC-IIB.

Operation
Privileged operation
Execute
Protection (storage only)
Addressing
Specification
Data
Fixed-point overflow
Fixed-point divide

These exceptions were discussed in paragraph 2.2.2. More than one cause of a program interruption may occur at once, but only one program interrupt is taken. In SUMC-IIB, the following priorities apply when this occurs:

Instruction Fetch:

Addressing and specification exceptions may co-occur. If the instruction address (address of the first halfword of the instruction) is out of the bounds of implemented memory, an addressing interrupt will occur. If, however, specification is bad (not on halfword boundary) and the second, third, or fourth halfword of the instruction has a bad address, a specification interrupt occurs.

Instruction Execution:

Occurrence of an operation exception (invalid op code) rules out other interruptions. However, privileged operation, protection, addressing, and specification may co-occur. Privileged instructions are dealt with below. Barring other factors, also discussed below, the priority of these interrupts is:

Addressing
Specification
Storage Protection

a) Privileged operations

There are six privileged instructions in the SUMC-IIB. If a privileged operation exception occurs together with a memory reference exception (one of the three above) the following exception has priority and causes the interrupt:

i) SSK - privileged operation
ii) SSM - memory reference, in the above order
iii) Diagnose - privileged operation
iv) SIO - memory reference, in the above order
v) TRMS - For the interval timer, the memory reference takes priority. This is also the case for a bad RTC address. If only the second halfword of the RTC address is bad, the privileged operation exception will take precedence over the addressing exception (the RTC address need not be fullword aligned). A memory reference exception can take place even in cases of timer read only, but only for the first halfword address. (The second halfword is not read when the RTC is to be read only.)
vi) LPSW - If the new PSW address is not on a halfword boundary or is an invalid address, the addressing or specification exception will have priority over privileged operation. If the new PSW address is not on a double wrod boundary, the privileged operation interrupt will occur if in problem state.

b) In the instruction D and M, a memory reference exception (for the second operand) takes precedence over a specification exception caused by improper (odd) register specificaiton for the first operand.

c) In SS instructions, memory reference exceptions for the second operand take precedence over those for the first operand.

The only program interruption which can be masked off is fixed-point overflow. If bit 36 of the PSW (first bit of the program mask) is zero, no interruption occurs. Only one bit of the program mask is used, the other bits are ignored and need not be zero upon PSW load or in the SPM instruction.

2.3.5 Input/Output Modes

The SUMC-IIB I/O provides three types of device initiated information transfer as well as program controlled transfers. The three types are:

- o Buffered I/O
- o Direct Memory Access (DMA)
- o External Interrupts

Program initiated I/O is provided by Direct I/O. Two interfaces are provided to the SUMC-IIB; a general I/O channel and a direct memory interface. A four-bit device identification code permits up to 16 system devices to be attached directly to the I/O channel. The DMA interface must be shared by device cooperation or be used by only one device as it appears to the computer like a single device interface. As an option the DMA function can be handled over the channel rather than as a separate interface.

2.3.5.1 Buffered I/O

Buffered I/O allows a device to transfer single or multiple words of data to/from a table in main memory without the device knowing the location of the table. The CPU hardware is used to keep track of table word count and address incrementing. When the table is full/empty the device is notified by a signal on the ZERO COUNT line. Separate input and output tables are maintained for each buffered device code 16 codes).

It should be noted that even though the CPU hardware is shared to handle the buffered I/O transfers, the program is not interrupted and the time consuming save operations associated with program interrupt are not required. Buffered I/O operations are handled between instructions and do not use any register visible to the programmer. The table starting address and size are controlled (initialized) by programming.

2.3.5.2 Direct I/O

Direct I/O provides a means for the programmer to send a command or data word to an I/O device or request a data word (or status) from a device. Each direct I/O instruction sends a 16-bit command word out on the System I/O channel and can send or request a data word to/from the addressed I/O device. The channel is attached and relinquished for each Direct I/O instruction.

2.3.5.3 Input/Output Interruption

An I/O interruption provides a means by which the CPU responds to conditions in the I/O units.

An I/O interruption can occur only when the mask bit associated with I/O is set to one. The status and address of the I/O unit involved are recorded in bits 16-31 of the I/O old PSW.

2.3.5.4 Direct Memory Access

Direct Memory Access provides the fastest means of transferring data to or from main storage. Both waiting time and transfer rates are faster than other I/O modes. The CPU hardware does not participate in the DMA operations and, therefore, they can take place in the middle of an instruction. If the CPU is not requesting memory service at the time of a DMA request there will be no impact on CPU performance. In the case of conflicts between CPU and DMA, memory service is alternated so that neither will be locked out.

DMA operations are not under direct program control, however, a programmer may prevent the DMA from modifying any particular block of memory by setting the DMA storage protect indicator for the block. The DMA feature is provided with a separate interface to operate concurrently with buffered or direct I/O.

2.3.5.5 I/O Channel Rate

The I/O channel burst-mode data rates are shown in Table 2.3-2.

Table 2.3-2. I/O Data Rates

Mode	Rate (Maximum)	
Direct I/O	100K	16-bit words/second
Buffered I/O	150K	16-bit words/second
DMA I/O	750K	16-bit words/second

2.3.6 Input/Output Operations

The SUMC-IIB I/O interface provides a 16-bit parallel channel for support of two classes of I/O equipment. These are:

1. System I/O devices and
2. Test support equipment (TSE) I/O devices.

A particular I/O device is classified based on whether it is attahced directly to the HTC or indirectly via the TSE. Further provision has been made to allow both program and device initiated information transfer which includes I/O commands, data words, and external I/O interrupts.

This portion of the manual describes the programmed control of I/O devices by the channel and central processing unit (CPU) including formats for the various types of I/O control information. Although certain information, formats, etc., may be applicable to both system and TSE I/O, each type is described individually for simplicity.

Buffered I/O allows a device to transfer single or multiple words of data to/from a table in main memory without knowing the location of the table. The CPU keeps track of table word count and address incrementing. When the table is full/empty the device is notified by a signal on the ZERO COUNT line. Separate input and output tables are maintained for each buffered device code (16 codes).

The Channel Address Word (CAW) at memory location 72, points to the first location of a table that consists of sixteen (16) eight (8) BYTE entries that contain the input storage address and count, and output storage address and count of each of the (possible) 16 Buffered I/O devices (see Table 2.3-3).

The programmer controls Buffered I/O by initialization of the I/O address and word count in the Buffered I/O Control Table.

To initialize a buffered I/O sequence the programmer must:

1. Set the CAW (location 72) to the address of the start of the Buffered I/O table.

2. Set the device I/O word count (in the buffered I/O table) to the number of 16-bit data words to be transferred.

3. Set the device I/O word address (in the Buffered I/O table) to the memory address of the beginning of the data to be written out (or to a location for the data to be written in).

4. Start the I/O device so it will request a buffered I/O "interrupt". This is usually done by giving a direct out command to the device via a SIO instruction.

Table 2.3-3. Buffered I/O Device Table

DEVICE NO.	INPUT		OUTPUT	
	2 BYTES	2 BYTES	2 BYTES	2 BYTES
*0	I/O WORD COUNT	I/O WORD ADDRESS	I/O WORD COUNT	I/O WORD ADDRESS
1	"	"	"	"
2	"	"	"	"
3	"	"	"	"
4	"	"	"	"
5	"	"	"	"
6	"	"	"	"
7	"	"	"	"
8	"	"	"	"
9	"	"	"	"
15	"	"	"	"

*CAW = ADDRESS OF THIS LOCATION

The I/O word count is updated by one and the I/O address is updated by two in the I/O Control Table for each 16-bit (two byte) word that is transferred to or fro memory by the CPU, unless the I/O word count is in TWOs complement form. If the I/O word count is in TWOs complement form, the I/O word count and I/O word address are not updated at the end of a Buffered I/O transfer. Therefore, the I/O address and word count start from the initial value each time the I/O device initiates a data transfer. This method of data transfer is useful for devices that send a burst of data periodically. Once a device initiates a transfer, the I/O channel is tied up until the device releases it.

If an I/O device requests a data transfer and the I/O Word Count is zero, an I/O error interrupt will be generated. The I/O Channel Code word is furnished as the interruption code in the I/O old PSW upon most I/O interrupts including error interrupt. The SUMC-IIB channel code word is shown in Figure 2.3-3.

It should be noted that even though the CPU hardware is interrupted to handle the buffered I/O transfers, the program is not interrupted and the time consuming save operations associated with program interrupt are not required. Buffered I/O operations are handled between instructions and do use any register visible to the programmer.

2.3.6.1 Buffered I/O Status Word

The Buffered I/O status word (location 66) is set to the current Buffered I/O address during Buffered I/O operations and is cleared to zero when a Buffered I/O operation is completed successfully.

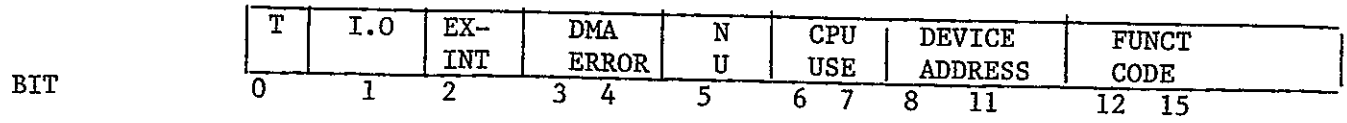
If an addressing exception, memory protect exception, or parity error occurs during Buffered I/O, a program exception interruption will be generated with the Buffered I/O status word set non zero; the contents will indicate the address of the Buffered I/O word in use at the time the error occurred.

2.3.6.2 Service Interrupt

Interrupts permit a device to interrupt the normal program sequence. A single level of interrupt is provided. Programmed priorities may be implemented. In the interrupt sequence an I/O Channel Code word is sent from the device and stored as the Interruption Code in the old I/O PSW (see Figure 2-6).

The new I/O PSW is used as the current PSW on all I/O interrupts except Buffered I/O. A Buffered I/O interrupt is not visible to the programmer. He will never see bit 1 set in the I/O interrupt code.

A Direct Memory Access (DMA) error will cause a normal I/O interrupt, except the only bits set in the old I/O PSW interrupt code will be bits 3 or 4 indicating DMA error 1 or 2 (see Figure 2-6).



IOTSE BITS	USE	
0	LOGIC 0 - I/O SERVICE LOGIC 1 - TSE INTERRUPT	NORMAL CONVENTION
1	LOGIC 1 - BUFFERED INPUT LOGIC 2 - BUFFERED OUTPUT	
2	LOGIC 1 - EXTERNAL INTERRUPT LOGIC 0 - BUFFERED I/O	USE ONLY FOR BIT 0 = 0
3	DMA ERROR NO. 1	
4	DMA ERROR NO. 2	
5	UNUSED	
6-7	CPU USE	
8-11	TAG (DEVICE ADDRESS) OR INTERRUPT CODE	
12-15	DEVICE FUNCTION CODE (MUST NOT BE ZERO)	

NOTE: The device or tester is responsible for generating the code word in the format shown above, but inverted bit-by-bit (at the cable).

Figure 2.3-3. I/O Channel Code Word

2.3.6.3 TSE I/O Devices

The TSE has a Typewriter and Paper Tape Reader, which are both direct I/O. All commands are sent to the TSE equipment and all data received by using the SIO instruction. The command word (see Figure 2.3-4 for TSE commands) is placed at the effective address (EA); the output data word is placed in the register designated by R1; and the data word read will be in the register designated by R3 after instruction completion. To write data to the typewriter:

1. Send a command to put the typewriter in the output mode.

2. Send each character (byte) to the typewriter by placing it right justified in R1 and sending a write typewriter command.

3. Check the condition code after each SIO instruction to ensure the I/O interface was not busy and the instruction was completed successfully.

4. The Typewriter will give a typer cycle complete interrupt (see Table 2.3-4) after each character is complete and it is ready to receive another command.

NOTE:

It is possible to preclude the typer cycle complete interrupt by immediately generating another Direct Out command to send the next character to the typewriter. The SIO instructions may be strung together in this manner and all I/O interrupts will be locked out as the typewriter will have control of the I/O channel for the whole period. Note that this type of operation will prevent the Clock and the Timer from being updated while the channel is tied up.

The typewriter input cycle is exactly the same as the output cycle except the typewriter must be placed in the input mode and each character of input is the register designated by R3 at the completion of each SIO instruction. To read from the paper tape:

1. Send a Start Tape command.

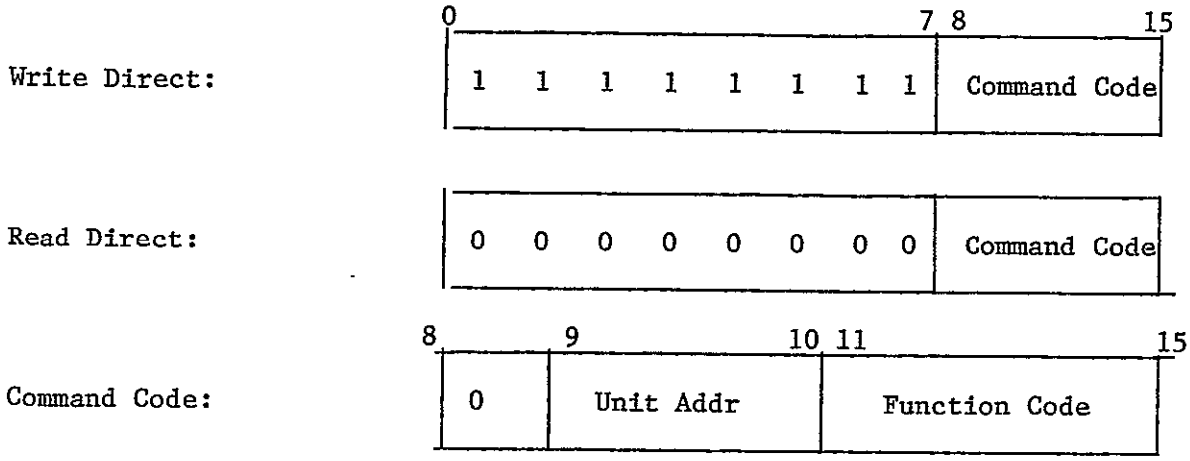
2. When a tape character has been read and is ready to be transmitted to the CPU a Tape Data Ready I/O-TSE interrupt will be generated.

3. Read each character by sending a read tape command. Each character will be right justified in the register designated by R3 after the SIO instruction.

4. When the last desired character is read in, a stop tape command is sent to the tape reader.

NOTE:

It is possible to preclude the Tape Data Ready Interrupt by immediately generating another Direct in command to fetch the next character from the tape. The SIO instructions may be strung together in this manner and I/O interrupts will be locked out as the Tape Reader will have control of the I/O channel for the whole period. Note that this type of operation will prevent the Timer and Clock from being updated while the channel is tied up.



Command	Code
Typewriter Output Mode	1111 1111 0110 0001
Typewriter Input Mode	1111 1111 0110 0010
Read Typewriter	0000 0000 0110 0011
Write Typewriter	1111 1111 0110 0100
Start Tape	1111 1111 0100 0001
Stop Tape Advance	1111 1111 0100 0010
Read Tape	0000 0000 0100 0011
Read 16 right-most bits of panel address register	0000 0000 0000 0100
Read 8 left-most bits of panel address register	0000 0000 0000 0101
Read 16 left-most bits of panel data register	0000 0000 0000 0111
Read 16 right-most bits of panel data register	0000 0000 0000 0110
Display Registers	1111 1111 0000 1000

Figure 2.3-4. TSE Commands Words

Table 2.3-4. Tester Interrupts

INTERRUPT	CODE	CAUSE
Enter Soft Stop	1000 0000 0000 0001	Depression of STOP switch
Read SPM	1000 0000 0000 0010	Depression of READ SPM switch
Write SPM	1000 0000 0000 0011	Depression of Write SPM switch
Read Main Memory	1000 0000 0000 0100	Depression of READ Memory switch
Write Main Memory	1000 0000 0000 0101	Depression of WRITE Memory switch
Exit Soft Stop	1000 0000 0000 0110	Depression of START switch
External Interrupt	1000 0000 0000 0111	Depression of external interrupt switch
Tape Load	1000 0000 0000 1000	Depression of IPL Program Load switch
PSW Restart	1000 0000 0000 1001	Depression of PSW restart switch
Attention *	1000 1000 0110 1011	Depression of attention key on typewriter
Clear Memory	1000 0000 0000 1010	Depression of clear memory switch
Tape Data Ready *	1000 0000 0100 1011	Reading a character on paper tape
Typer Cycle Complete *	2000 0000 0110 1011	Completion of typing an input or output character on the typewriter.

*These interrupts are visible to the SUMC-IIB program. The other interrupts in this table are intercepted and acted upon by the microprogram.

Direct I/O provides a means for the programmer to send a command or data word to an I/O device or request a data word from a device. Each Direct I/O instruction sends a 16-bit command word out on the System I/O channel and may send or request a data word to/from the addressed I/O device. Figure 2.3-5 shows the format of the command word. The channel is attached and relinquished for each Direct I/O Instruction.

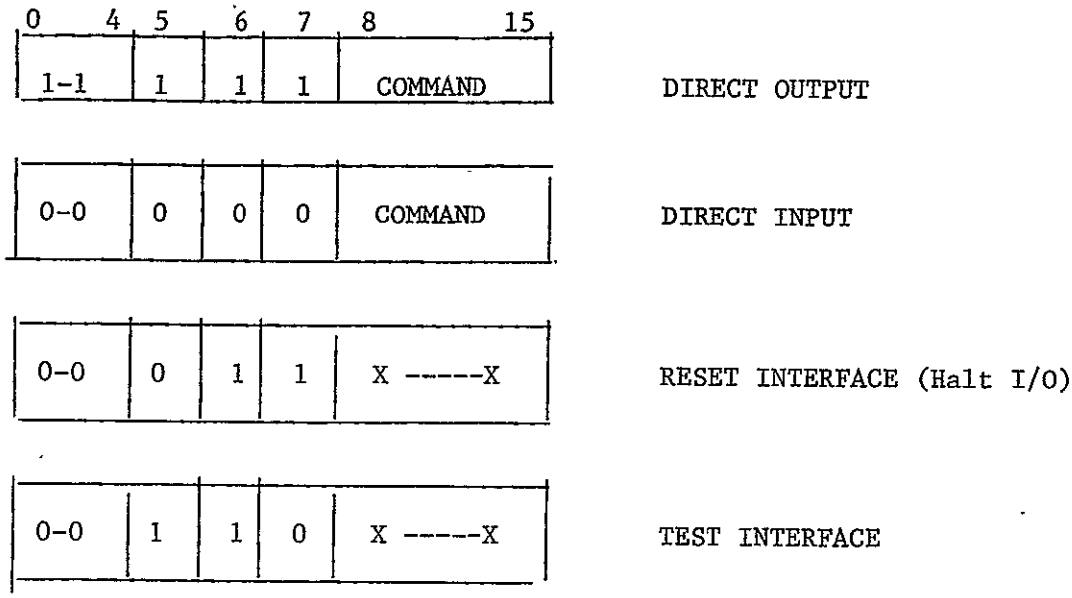


Figure 2.3-5. Direct I/O Command Word and CPU to I/O Command Word

The start I/O Instruction (SIO) is used to generate all Direct I/O commands.

If the I/O Interface is busy, the condition code is set to 1 without performing the I/O operation. A condition code of 0 indicates successful completion of the SIO Instruciton.

The Direct I/O command word is also used for CPU to I/O commands. Figure 2.3-5 shows those commands. Reset Interface immediately halts any I/O operation and clears the I/O channel by sending the Service Acknowledge signal and holding it on for 10 microseconds minimum.

Test Interface tests for channel busy and sets the condition code (1 if busy, 0 if not busy).

Electromechanical devices such as typewriters, perforated tape readers, and punches will have a special operation under Direct I/O. Direct Out (DO) will be as follows:

The HTC I/O places the command word and data word on the line normally.

The addressed device takes the command and data word and starts to perform the indicated operation (type a character, etc.).

The DO sequence is terminated and the channel freed up. (All standard so far).

Programmer option: Normally during system operation the program would perform useful work while the device is executing the command.

When the device has completed its task and is ready for the next task (such as type another character), it will generate a standard I/O interrupt to indicate device ready.

If the program had more tasks another DO would be generated and the sequence repeated.

2.3.7 Soft-Stop

The SUMC-IIB normally operates in the wait and running states, handling interrupts, executing instructions, etc.; or it can operate in "soft-stop" mode. In soft-stop, instructions are not executed and interrupts are ignored. SUMC-IIB just waits for requests from the test support equipment (TSE). When the system reset button or the stop button is pressed, SUMC-IIB is put into the soft-stop mode.

In soft-stop:

a) TSE requests are enabled.

b) The real time clock is incremented, but the interval timer is not decremente

c) All interrupts are ignored except parity, which causes the microprogram to hang up.

d) Buffered I/O requests are ignored.

2.4 MICROPROGRAM CONTROL WORD

The microprogram control word for the SUMC-IIB is comprised of five major fields and many sub-fields as shown in Figure 2.4-1. A brief explanation of the control word follows.

Scratch Pad Memory Control, Bits 1-10 (S1-S10)

Scratch pad memory control controls the SPM address to be read or written (S1-S6), the source of the address (S7-S9) (sources are IR BITS 8-11, 12-15, 16-19, and MROM BITS S1-6) and whether a read or read/write cycle (S10) is to be taken.

ALU Control, Bits 11-22 (A1-A12)

Arithmetic Logic Unit (ALU) input and function control are specified by these bits. These bits control the source of the two ALU input multiplexers (A and B) and the function performed by the ALU. Bits 11-14 (A1-A4) control the "A" multiplexer (MXA) source which can be one or more of the following:

SCRATCH PAD MEMORY (SPM)

PRODUCT REMAINDER REGISTER (PRR)

INSTRUCTION REGISTER BIT 20-31 (IR) 20-31

MEMORY ADDRESS REGISTER (MAR)

ZERO

Bits 15-17 (A5-A7) control the "B" multiplexer (MXB) source which can be one or more of the following:

STORAGE DATA REGISTER (SDR)

SPM

MAIN READ ONLY MEMORY (MROM)

ZERO

Bits 18-20 (A8-A10) control the ALU function which can be any one of the following:

LOGICAL AND

SUBTRACT (B-A)

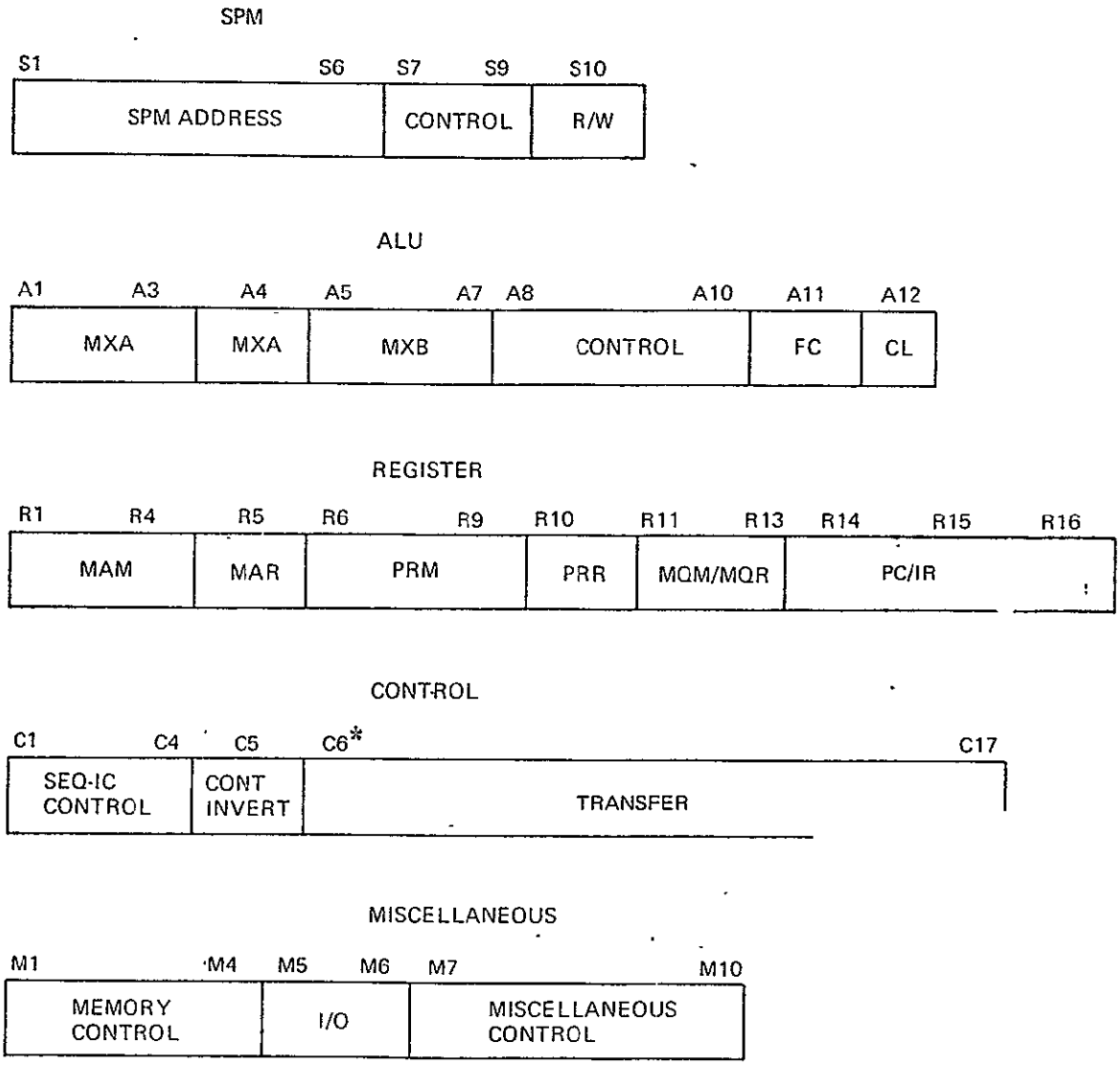
SUBTRACT (A-B)

MULTIPLY, DIVIDE, SQUARE ROOT (MDS) *

LOGICAL OR

LOGICAL EXCLUSIVE OR

* Square root is not supported in the SUMC-IIB microprogram.



*C6 – NOT USED FOR 16 BIT HTC

Figure 2.4-1. Microprogram Control Word Format

Bit 21 (A11) is the force carry bit. Bit 22 (A12) is used to save the carry out (overflow) from one arithmetic operation to the next.

Register Control, Bits 23-38 (R1-R16)

Bits 23-26 (R1-R4) are used to control shifting in and the source of the Memory Address Multiplexer (MAM). Both left and right shifts of 1, 2 and 4 are available. Possible sources are MAR, 0, ALU and IO/TSE buss.

Bit 27 (R5) is used to control the setting of the Memory Address Register from the MAM.

Bits 28-31 (R6-R9) are used to control shifting in and the source of the Product Remainder Multiplexer (PRM). Both left and right shifts of 1, 2 and 4 are available. Possible sources are the ALU, IR and the MQR.

Bit 32 (R10) is used to control the setting of the Product Remainder Register (PRR) from the PRM.

Bits 33-35 (R11-R13) are used to control shifting in and the source of the Multiplier/Quotient Multiplexer (MQM).

If anything is gated into the MQM, the Multiplexer/Quotient Register (MQR) is set to the resulting output of the MQM.

Bits 36-38 (R14-R16) are decoded eight ways to provide control of the program counter, instruction register and reading the hardware timer. Section 4.9 lists the complete usage of the field.

Sequence Control, Bits 39-55 (C1-C17)

Bits 39-42 (C1-C4) control the action of the Sequencer and Iteration Multiplexers. This conditionally controls the sequence of microinstructions depending on a variety of test conditions that may be specified.

Bit 43 (C5) reverses the branch (sequence) conditions specified by Bits C1-C4.

Bits 44-55 (C6-C17) contain the transfer address of the next micro-instruction for conditions when the next instruction to be executed is not sequential (may be conditional). This field may also be used to emit constants, etc., when a transfer address is not required.

Miscellaneous Control, Bits 56-65 (M1-M10)

Bits 56-59 (M1-M4) are used to control the action of the Main Storage Device (MEMORY) including such functions as Read, Write, Fullword, Halfword, etc.

Bits 60-61 (M5-M6) are used to control I/O. M5 is turned on to signify a command has been generated. M6 is turned on to acknowledge a signal being received from the I/O channel

Bits 62-65 (M7-M10) are the miscellaneous control bits and are used in conjunction with other fields to vary specific instructions. Examples are:

Load Timer

Load IC

Load PSW, etc.

A full definition is given in subsection 4.7.

2.5 SUMC-IIB SUPPORT SOFTWARE

2.5.1 SUMC-IIB Assembler

The assembler language used for the SUMC-IIB is a symbolic programming language similar to the IBM 360 assembler language. It enables the programmer to use System/360 machine functions, as if he were coding in System/360 machine language. The assembler program translates symbolic instructions into machine-language instructions, assigns storage locations, and performs auxiliary functions necessary to produce an executable machine-language program.

COMPATIBILITY

The SUMC-IIB Assembler uses the S/360 instruction set with the following exceptions:

1. The SUMC-IIB I/O is different from S/360 and only uses the SIO instruction. The SIO instruction format has been changed from SI to an RS format. The S/360 TIO, HIO, and TCH instructions are not supported by the SUMC-IIB assembler.

2. A new instruction, Timer Read and Set (TMRS) has been added for the SUMC-IIB. The TMRS instruction has an RS format and the storage operand must be aligned on a halfword boundary.

3. The SUMC-IIB assembler does not support the long or extended precision S/360 Floating Point Feature instructions, the Decimal Feature instructions, the Direct Control Feature instructions, the Channel Command Word (CCW) assembler instruction, or the Insert Storage Key (ISK) instruction.

4. All extended instructions, i.e., instructions in the short precision option, the double precision fixed-point arithmetic option, and the single precision floating point are supported by the assembler. However, only the floating point are in the S/360 instruction set.

THE ASSEMBLER LANGUAGE

The basis of the assembler language is a collection of mnemonic symbols which represent:

1. System/360 machine-language operation codes.

2. Operations (auxiliary functions) to be performed by the assembler program.

The language is augmented by other symbols, supplied by the programmer, and used to represent storage addresses or data. Symbols are easier to remember and code than their machine-language equivalents. Use of symbols greatly reduces programming effort and error.

Machine Operation Codes

The assembler language provides mnemonic machine-instruction operation codes for all machine instructions implemented for the SUMC-IIB with extended mnemonic operation codes for the conditional branch instruction.

Assembler Operation Codes

The assembler language also contains mnemonic assembler-instruction operation codes, used to specify auxiliary functions to be performed by the assembler. These are instructions to the assembler program itself and, with a few exceptions, result in the generation of no machine-language code by the assembler program.

Macro Instructions

The assembler language enables the programmer to define and use macro instructions. Macro instructions are represented by an operation code which stands for a sequence of machine and/or assembler instructions. Macro instructions used in preparing an assembler language source program fall into two categories: system macro instructions, provided by IBM, which relate the object program to components of the operating system; and macro instructions created by the programmer specifically for use in the program at hand, or for incorporation in a library, available for future use.

THE ASSEMBLER PROGRAM

The assembler program, also referred to as the "assembler," processes the source statements written in the assembler language.

Basic Functions

Processing involves the translation of source statements into machine language, the assignment of storage locations to instructions and other elements of the program, and the performance of the auxiliary assembler functions designated by the programmer. The output of the assembler program is the object program, a machine-language translation of the source program. The assembler furnishes a printed listing of the source statements and object program statements and

additional information useful to the programmer in analyzing his program, such as error indications. The object program is in the format required by the linkage editor component of the SUMC-IIB support software.

The amount of main storage allocated to the assembler for use during processing determines the maximum number of certain language elements that may be present in the source program.

PROGRAMMER AIDS

The assembler provides auxiliary functions that assist the programmer in checking and documenting programs, in controlling address assignment, in segmenting a program, in data and symbol definition, in generating macro instructions, and in controlling the assembler itself. Mnemonic operation codes for these functions are provided in the language.

Variety in Data Representation: Decimal, binary, hexadecimal, or character representation of machine-language binary values may be employed by the programmer in writing source statements. The programmer selects the representation best suited to his purpose.

Base Register Address Calculation: As discussed in "IBM System/360: Principles of Operation," the System/360 addressing scheme requires the designation of a base register (containing a base address value) and a displacement value in specifying a storage location. The assembler assumes the clerical burden of calculating storage addresses in these terms for the symbolic addresses used by the programmer. The programmer retains control of base register usage and the values entered therein.

Relocatability: The object programs produced by the assembler are in a format enabling relocation from the originally assigned storage area to any other suitable area.

Sectioning and Linking: The assembler language and program provide facilities for partitioning an assembly into one or more parts called control sections. Control sections may be added or deleted when loading the object program. Because control sections do not have to be loaded contiguously in storage, a sectioned program may be loaded and executed even though a continuous block of storage large enough to accommodate the entire program may not be available.

The assembler allows symbols to be defined in one assembly and referred to in another, thus effecting a link between separately assembled programs. This permits reference to data and transfer of control between programs.

Program Listings: A listing of the source program statements and the resulting object program statements may be produced by the assembler for each source program it assembles. The programmer can partly control the form and content of the listing.

Error Indications: As a source program is assembled, it is analyzed for actual or potential errors in the use of the assembler language. Detected errors are indicated in the program listing.

2.5.2 Linkage Editor

The Linkage Editor Program prepares the output of the SUMC-IIB assembler for execution. The Linkage Editor prepares a load module that is to be brought into main storage for execution.

The linkage editor used for the SUMC-IIB is similar to the OS/360 Linkage Editor.

The Linkage Editor provides several processing facilities such as creating overlay programs, and aiding program modification. (The Linkage Editor is also used to build and edit system libraries.)

2.5.3 Tape Formattor

The SUMC-IIB Tape Formattor Program modifies and reformats the load module generated by the Linkage Editor such that it will be executable on the SUMC-IIB.

The program will provide a data set which will contain the executable program in a paper tape format that is acceptable to the Initial Program Load Sequence.

Data in the same format can be stored on magnetic tape for use with the Field Tester which uses magnetic tape for an IPL device.

2.5.4 SUMC-IIB Simulator

The SUMC-IIB simulator is known as the Emulator System (ES) and is designed to provide dynamic program analysis, modification and control of programs written for execution on the SUMC-IIB computer. The ES functionally duplicates the operation of the Central Processor Unit (CPU) of the computer at the programmable register level and provides for associated I/O device handling as well as extensive user control over the simulation. The ES aids in furnishing a near real-life situation and greatly assists the problem programmers in debugging the operational program(s) written for the SUMC-IIB machine.

The ES operates under OS/360 and OS/370 and is designed as a series of subroutines available to a User written Fortran Control Program (UCP). These subroutines form the interface between the simulated program and the user to enable the user to analyze, modify and control the simulation run. The ES simulates the SUMC-IIB computer at the programmable register level rather than the micro logic level. The simulated program execution is directed by the UCP while information from the simulated memory map is conveyed back to the user. Thus the flexibility of static or closed loop dynamic simulation is available.

Complete instruction-by-instruction or branch instruction only lists can be printed from the simulator run. Execution times and storage mapping are also available from the simulator.

2.6 SUMC-IIB TEST EQUIPMENT

2.6.1 General Features

Two basic test systems have been developed for the SUMC-IIB processor. These are a factory test system used to support the design, manufacturing and product improvement effort related to the SUMC-IIB processor and a field test system designed to support the installed unit in the field.

The factory test system consists of a basic CPU tester, a read only memory (ROM) simulator, a paper tape reader, an IBM Selectric I/O typewriter, a 32-bit display interface and an auxiliary MROM/IROM display panel. The factory tester is shown in Figure 2.6-1.

The field test system consists of a basic CPU tester and a 16-bit display interface. Optionally available with this system is an IBM S/360 compatible Magnetic Tape Unit, a 100 to 165 character per second line printer and a CPU power and cooling unit. The field tester is shown in Figure 2.6-2.

2.6.2 Factory Test System

The factory test system provides the total capability required for detailed check-out of the SUMC-IIB processor and its associated memories. Key elements of this system were given in Paragraph 2.6.1 and are discussed in the following paragraphs.

Basic Tester

The basic tester provides the capability to interface with and to exercise the SUMC-IIB processor. The capability is provided to manually control the processor through such features as Single Instruction operation, Single Micro-instruction operation, Stop on Compare and Display on Compare. In addition, capability is provided for manually loading or reading both the CPU Main Store and the Scratch Pad memory. Also provided are a Main Memory clear feature and several operator controlled interrupts such as PSW Restart and an External I/O interrupt.

32-Bit Display Interface

The factory tester display interface is designed to provide a display capability for a full 32-bit SUMC machine with an I/O interface that is compatible with that of SUMC-IIB. The displays provided are the following:

- Main Memory Storage Address Register (SAR)
- Main Memory Storage Data Register (SDR)
- Memory Address Register (MAR)

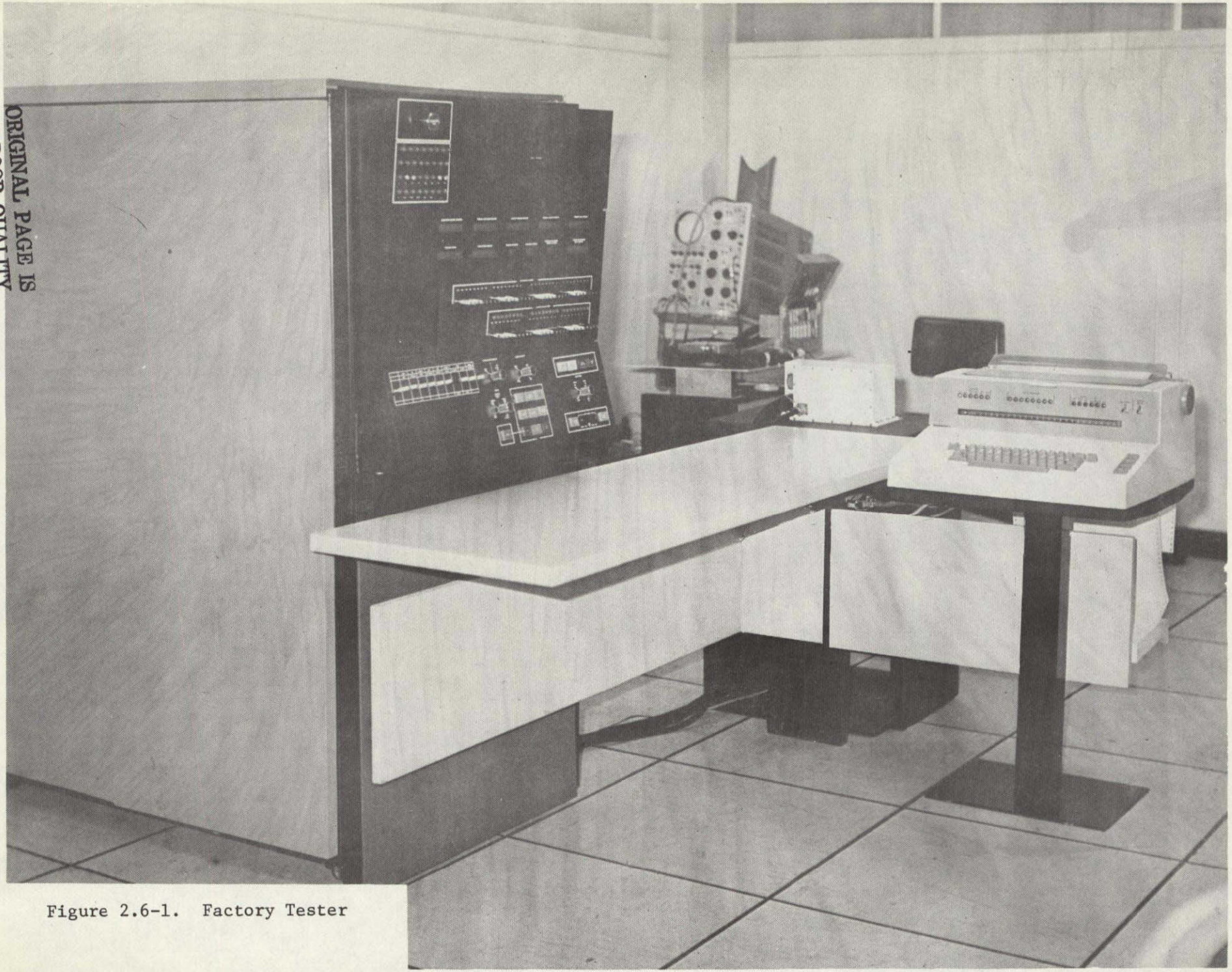


Figure 2.6-1. Factory Tester

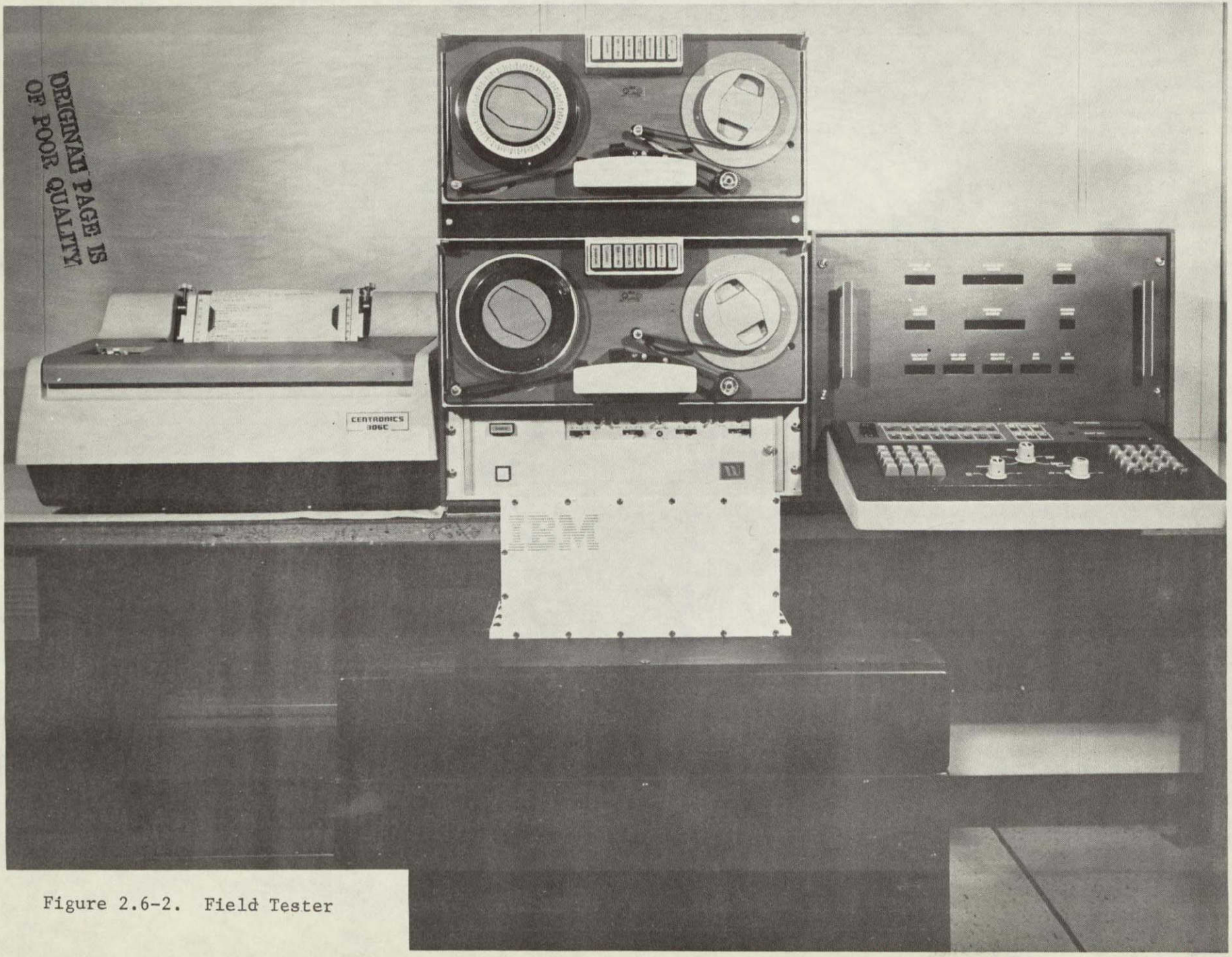


Figure 2.6-2. Field Tester

- Product Remainder Register (PRR)
- Multiplier Quotient Register (MQR)
- Instruction Register (IR)
- Iteration Counter (IC)
- Sequence Counter
- Program Counter (PC)
- Scratch Pad Memory (SPM) Address Register
- Scratch Pad Memory (SPM) Data Register
- Program Status Word

These displays operate in conjunction with the basic tester logic to provide complete visibility into processor operation.

Read Only Memory Simulator

The factory tester contains a 4K by 99-bit read/write memmory for use as a ROM simulator. This memory is segmented into two parts; one for simulating the Micro-instruction ROM and one for simulating the Instruction Address ROM. These memories may be collectively or individually substituted for the actual internal processor ROM. This feature provides a means for debug of the processor microcode before this code is committed to PROM Burn-In.

Auxiliary MROM/IROM Display Panel

This feature provides for the bit for bit display of the contents of the Micro-instruction and Instruction Address Read Only Memories. This feature is only operable when the processor is in the "fan-out" test fixture.

Paper Tape Reader

The factory tester contains a paper tape reader which is used to load both the CPU Main Memory and the MROM/IROM simulator memories. Load of CPU Main Memory is under program control while load of the simulator memories is under tester control. The tape unit provided in this system is a Remex Model No. RRS1150 BC1/651/G-B/U000. This unit can load data at a maximum rate of 150 characters per second.

IBM Selectric I/O Typewriter

The factory test system provides an IBM Model 1052 I/O Typewriter for use as a real-time program interface device and as an I/O printer. This device will operate at a maximum rate of 15 characters per second.

2.6.3 Field Test System

The Field Test System provides the capability to functionally test and to operate the SUMC-IIB processor in the field where detailed test

capability is not required. This system consists of the basic tester, an optional Magnetic Tape Unit, an optional line printer and an optional power and cooling stand for the processor. The complete system with all of the options would provide a basic test and debug capability equal to that of the factory test system, but without the ROM simulator or the 32-bit display capability. For this system the displays are consistent with the SUMC-IIB's internal 16-bit structure. Key features of the field test system are given in the following paragraphs.

Basic Tester

The basic test functions provided in this system are identical to those provided by the factory test system and described in Paragraph 2.6.2.

Display Interface

The display interface for the field test system provides for a 16-bit SUMC-IIB processor interface. The displays provided are identical to those listed in Paragraph 2.6.2.

Magnetic Tape Unit

This optional feature of the field test system provides a 9-track IBM S/360 compatible NRZI magnetic tape capability. This option may be selected with one or two tape decks as desired or required by the application.

Line Printer

This optional feature consists of a Centronics Model 306C line printer adapted to the SUMC-IIB interface. This feature provides a high speed (up to 165 characters per second) hard copy output capability.

Power and Cooling Stand

This optional feature provides a central source for processor power and cooling when such is not readily available at the test site.

SECTION 3

MECHANICAL DESIGN

3.1 GENERAL PACKAGING

The computer unit is formed by assembling modular sub-assemblies called slices, with the appropriate end plates, then bolting this assembly to a heat exchange interface (cold plate or air plenum). The basic SUMC-IIB is formed from three slices: power supply, CPU/IO, and MROM/Memory. Figure 3.1-1 shows the basic unit and Figure 3.1-2 typifies the individual slices. Memory expansion can be provided by attaching additional memory slices to the end of this basic structure. Each add on slice is contained in a volume of .047 $ft.^3$, increases the width of the unit by 1.4 in. and has a weight of 3.44 lb.

The SUMC-IIB with 16K-bytes of memory is contained in a volume of 0.17 $ft.^3$ has case dimensions of 10.74 in. long, 5.48 in. high, and 5.32 in. wide, and has a weight of 13.87 lb.

All slices are pluggable by way of flexible interconnection harnesses. Multilayer interconnection boards (MIBs) are used to support and electrically interconnect components. Each MIB is constructed from several layers of etched copper-clad epoxy-laminates which are bonded together under heat and pressure, using polymer impregnated epoxy glass. Layer-to-layer interconnection is made through copper plated holes.

Thermal control of the unit is maintained by attaching each slide directly to an auxiliary heat exchanger or cold plate. The internal heat is primarily transferred through the shortest possible conduction paths to maintain desirable junction temperatures. The SUMC-IIB has been analyzed to determine the typical and worst case component temperatures. The results indicate that the computer can meet its thermal design goals and that none of the hybrid modules will be thermally overstressed.

The HTC is designed to meet the requirements of MIL-E-5400, Class 2X equipment. The following typical environmental design boundaries are defined:

- Temperature

Continuous Operation*	$-54^{\circ}C$ to $+71^{\circ}C$
Intermittent Operation*	$-54^{\circ}C$ to $+95^{\circ}C$
Non-Operating	$-62^{\circ}C$ to $+95^{\circ}C$

*The upper limits apply only when the computer is attached to an appropriate heat exchanger maintained at $+55^{\circ}C$ maximum.

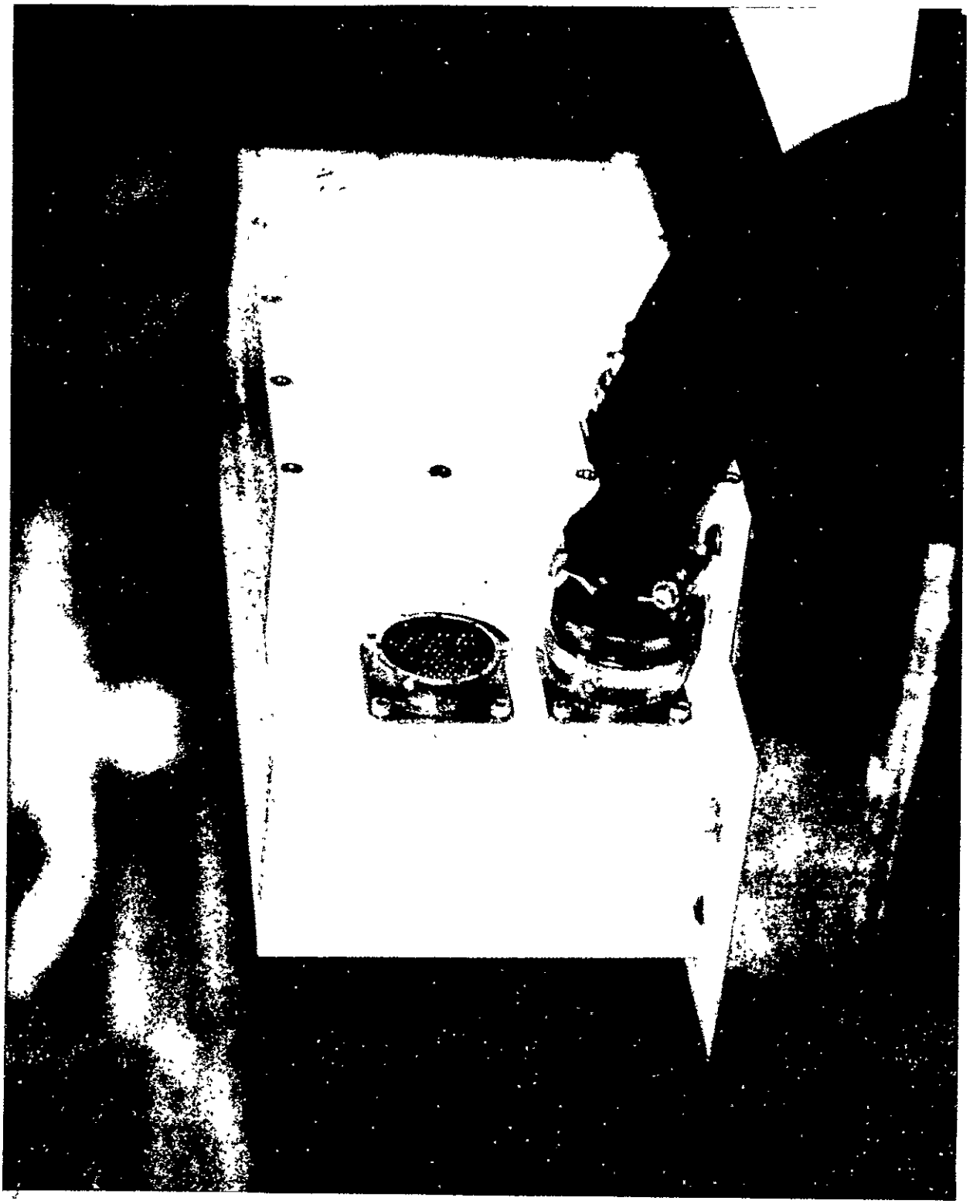


Figure 3.1-1. SUMC-IIB Computer

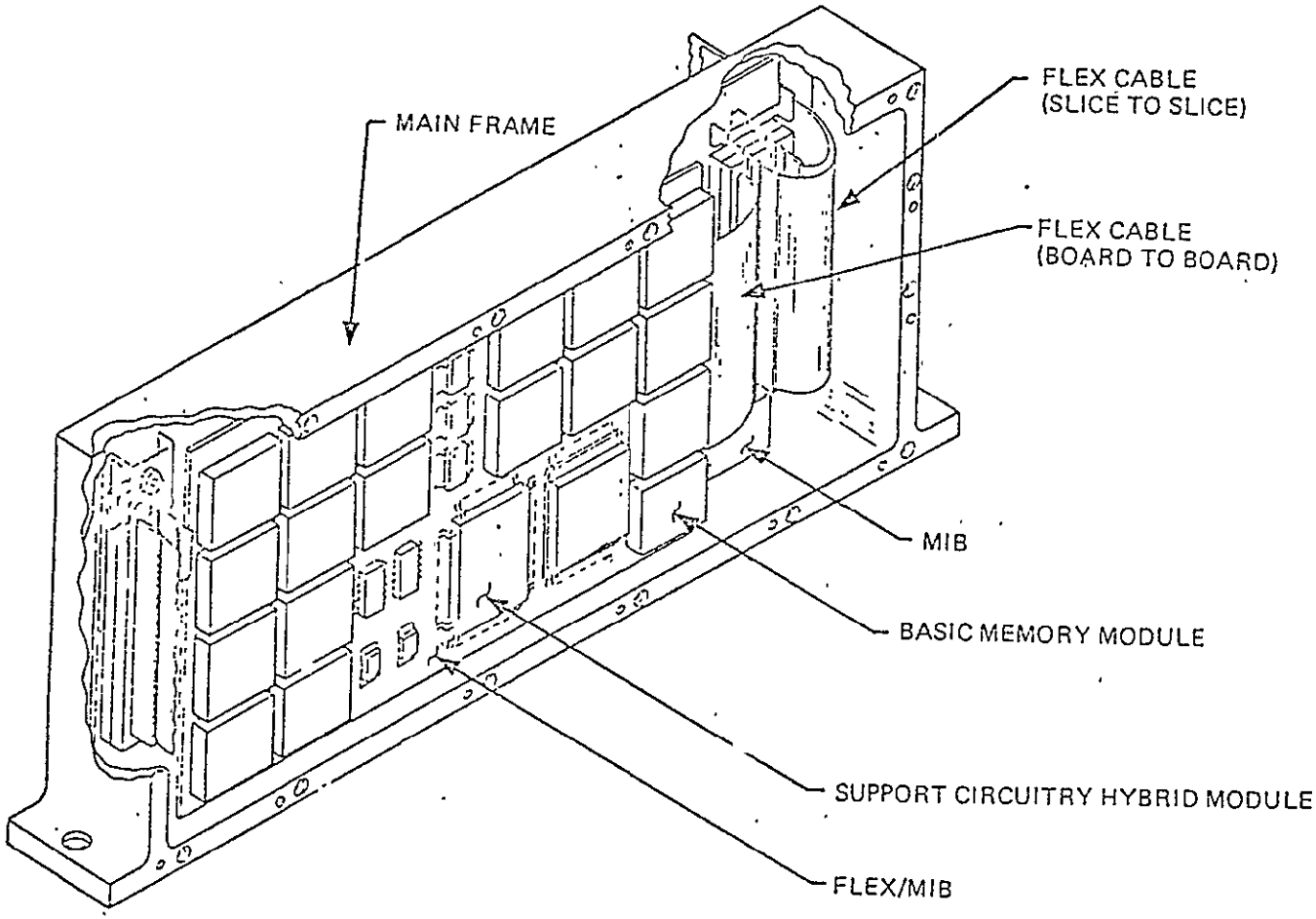


Figure 3.1-2. Memory Slice

- Vibration

  10G sine level from 20 to 2000 Hz.

  12.6G rms random from 20 to 2000 Hz.

  Qualification of the SUMC-IIB is planned but has not been conducted. The following sections describe in more detail the construction and characteristics of the modular subassemblies utilized, the interconnection technique, and the hybrid modules.

3.2 POWER SUPPLY SLICE

The structure is an aluminum frame machined to an I beam cross section from rectangular plate stock. This produces a center web section for mounting a MIB, high power dissipating components, and an I/O connector. Appropriate cutouts are provided in the web for components mounted on one side to interconnect with the MIB bonded to the other side.

The MIB is fabricated with a layer of flexible printed wiring cable laminated internally during the layer bonding process. During profiling this flexible material is exposed by a milling operation. With the flexible portion located in the middle of the MIB, the component mounting plane can be folded 180° to allow half of the plane to be bonded to the center web and the other half to be attached to the frame with fasteners. Various components are mounted to either side of the MIB such as transformers, hybrid packages, and discrete devices. The components dissipating the most power are mounted directly to the frame using a thermal mating compound interface for optimum heat conduction. The slice is shown in Figure 3.2-1.

Prime power input is made through a Bendix JT series subminiature cylindrical filter connector which is intermateable with MIL-C-38999 and MIL-C-27599 connectors. The contact pins have printed circuit tails to interface directly with the MIB eliminating the need for an internal harness.

Regulated power is distributed internally through a 120 pin Burndy type ML connector plug. This connector is mechanically attached to a 'U' shaped support bracket with threaded fasteners. The contact extensions are soldered to pads provided on the MIB surface and potted for additional reinforcement. This connector arrangement is located on the end of a flexible printed wiring cable laminated as an integral part of the MIB.

3.3 CPU/IO AND MEMORY SLICES

The mechanical components include a flexible/multilayer interconnection board (flex/MIB) and a main frame similar to those described in the Power Supply section. Minor differences occur in the flex/MIB support due to the low profile and decreased weight of the components. A 40 mil copper heat frame is bonded to the flex/MIB between the board surface and the bottom surface of the components. When the populated flex/MIB is folded and attached, the copper heat frame makes

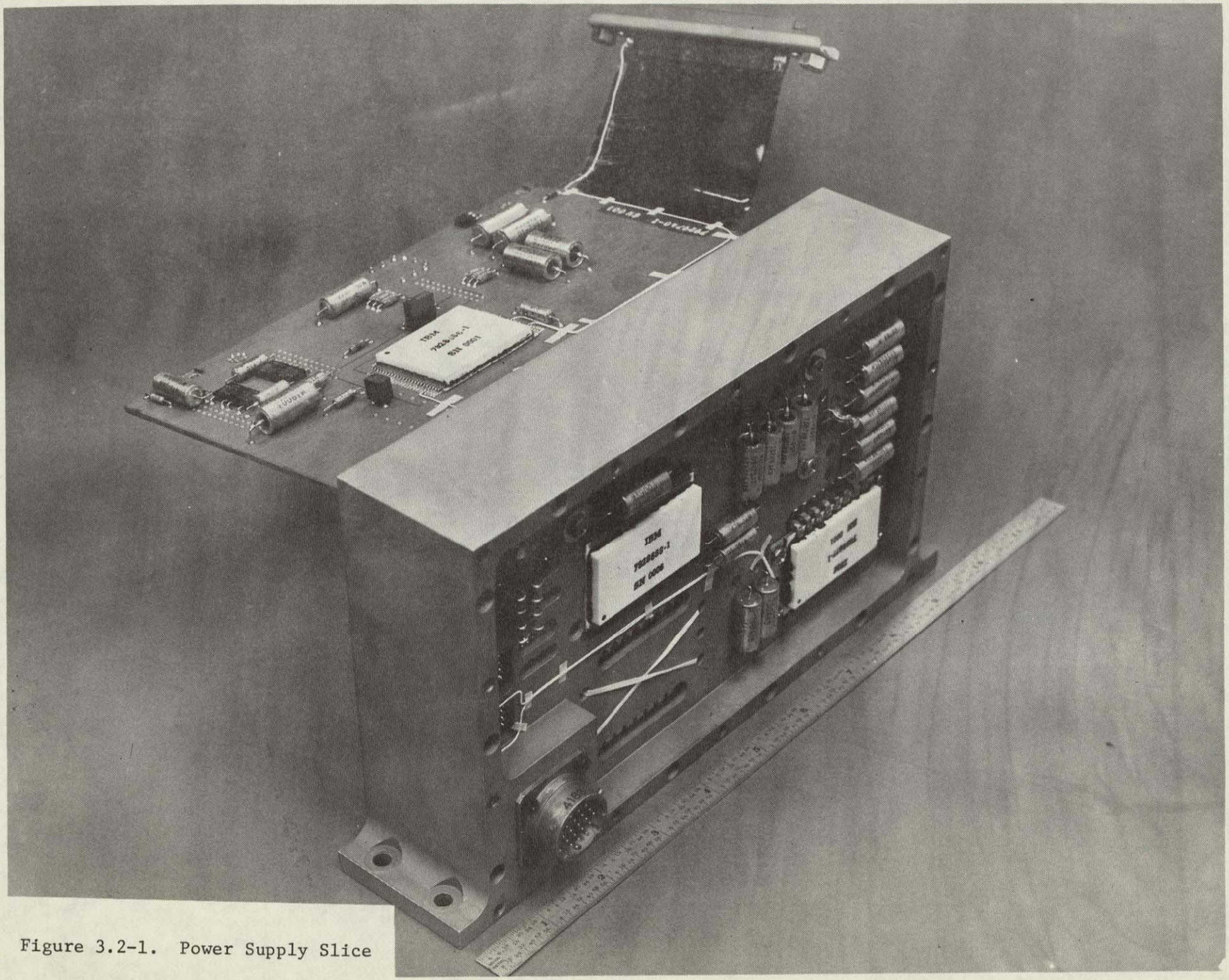


Figure 3.2-1. Power Supply Slice

directly contact with the center web of the main frame. With the use of thermal mating compound to fill the void between the components and the copper frame, the conductive heat transfer path from chip junction to main frame has been optimized.

The I/O and TSE signals are brought in/out through two 128 pin Bendix JT series cylindrical connectors which are intermateable with MIL-C-38999/27599 connectors. These connectors are supported on an auxiliary frame bonded to the flex/MIB and the contacts have round pins in place of the usual solder cup to interface directly with the CPU flex/MIB plated through holes. Figures 3.3-1 and 2 show these CPU/IO and MROM/MEMORY slices.

3.4 INTERCONNECTION TECHNIQUES

Slice-to-slice interconnections are made through several 120 pin Burndy type ML connector receptacles. For example, the internal pluggable interface described in the power regulator section mates with a receptacle on the CPU/IO MIB. Captivated hardware on the plug support frame threads onto the MIB mounting fasteners to provide the mechanical force required to engage and disengage the connectors. Similar pluggable interfaces distribute the power and signals throughout the unit.

3.5 HYBRID LOGIC MODULES

LSI components are packaged in hybrid modules of two different configurations. The basic module has a ceramic substrate with single layer thick film platinum-gold (PE-Au) conductor patterns providing the interconnection media between two chips and the module I/O leads. The chips are electrically and mechanically attached via an array of solder bumps on the active surface of the chip. Chip joining is accomplished by a solder reflow process to a matching pattern on the ceramic substrate. One hundred copper alloy I/O leads are lap soldered and epoxy supported to a surface pattern symmetrically spaced around the perimeter of the substrate. This module is shown in Figure 3.5-1. A ceramic cap is epoxy-bonded to the substrate, providing a hermetically sealed enclosure for the chip. The hermeticity provided has been demonstrated to be compatible with the MIL-STD-883, methods, test procedures, and test criteria.

The second module is of similar construction except the conductor patterns are applied in multiple layers providing the interconnections between four or five chips and 140 module I/O leads. The perimeter of the substrate has increased to provide the necessary area for the additional lap solder joints. An example of this module is shown in Figure 3.5-2. Unlike the two-chip module, this type of module has chip-to-chip interconnections to make functional units such as data flow modules, I/O interface modules, etc. Five different "personalities" of this substrate type are used in the SUMC-IIB.

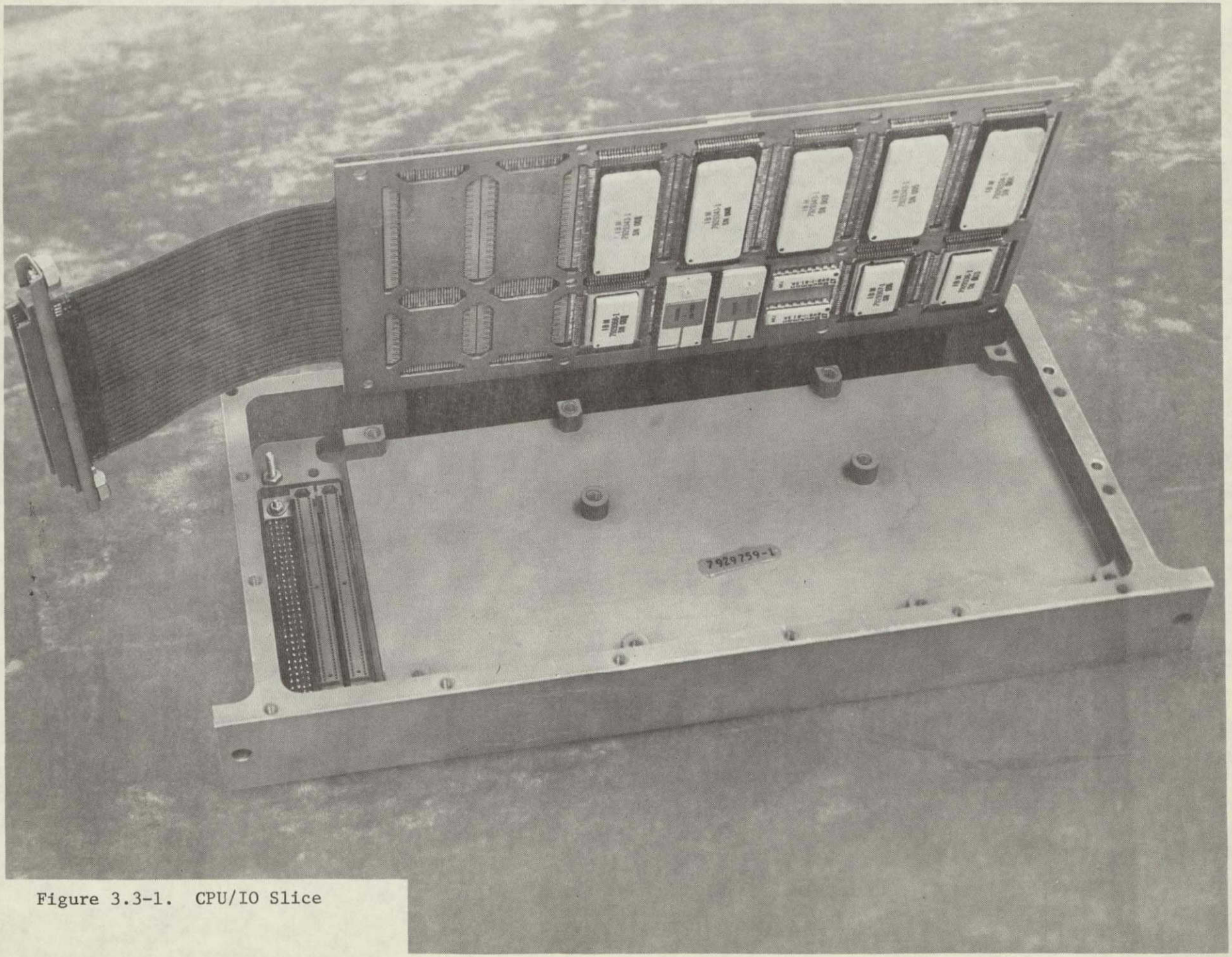


Figure 3.3-1. CPU/IO Slice

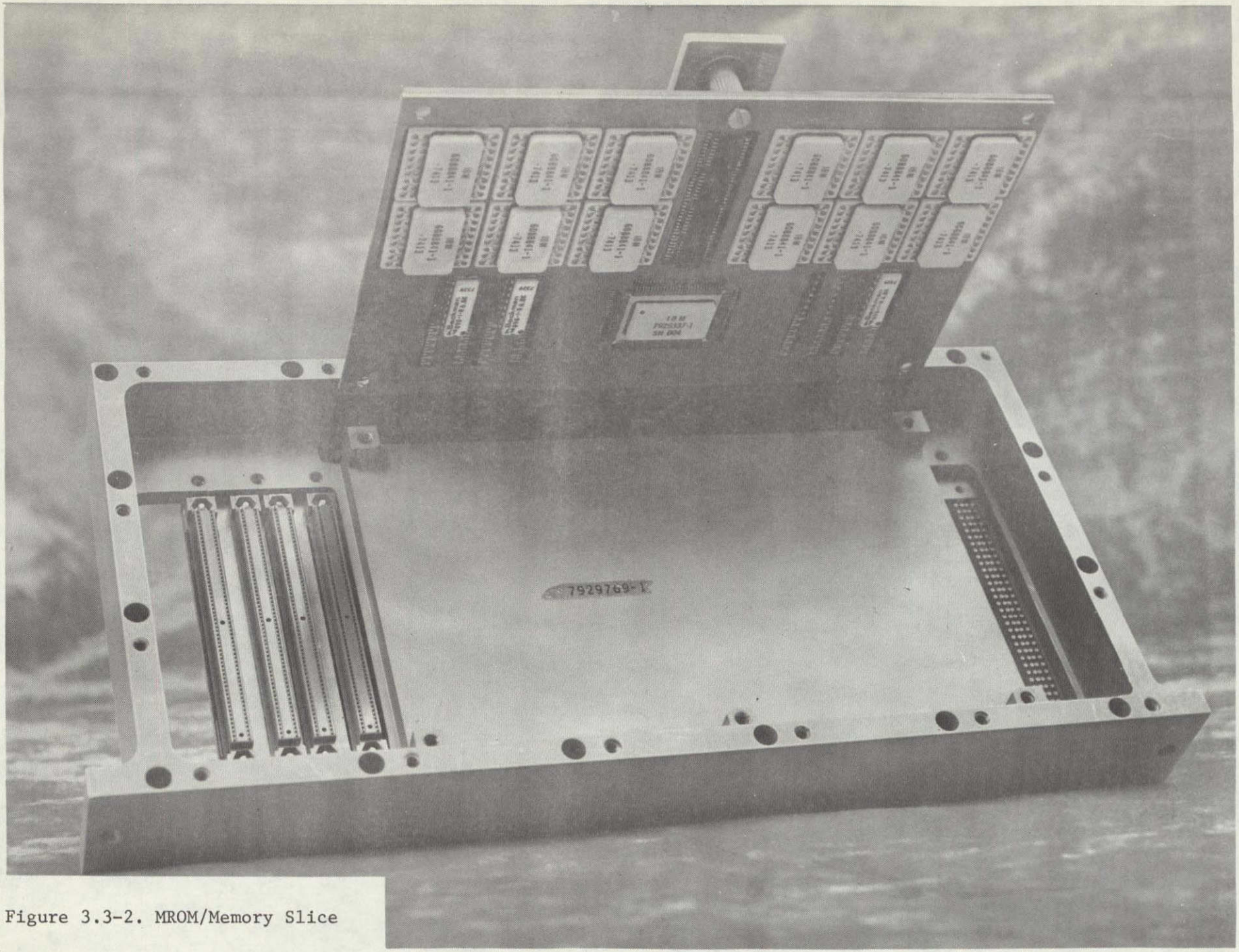


Figure 3.3-2. MROM/Memory Slice

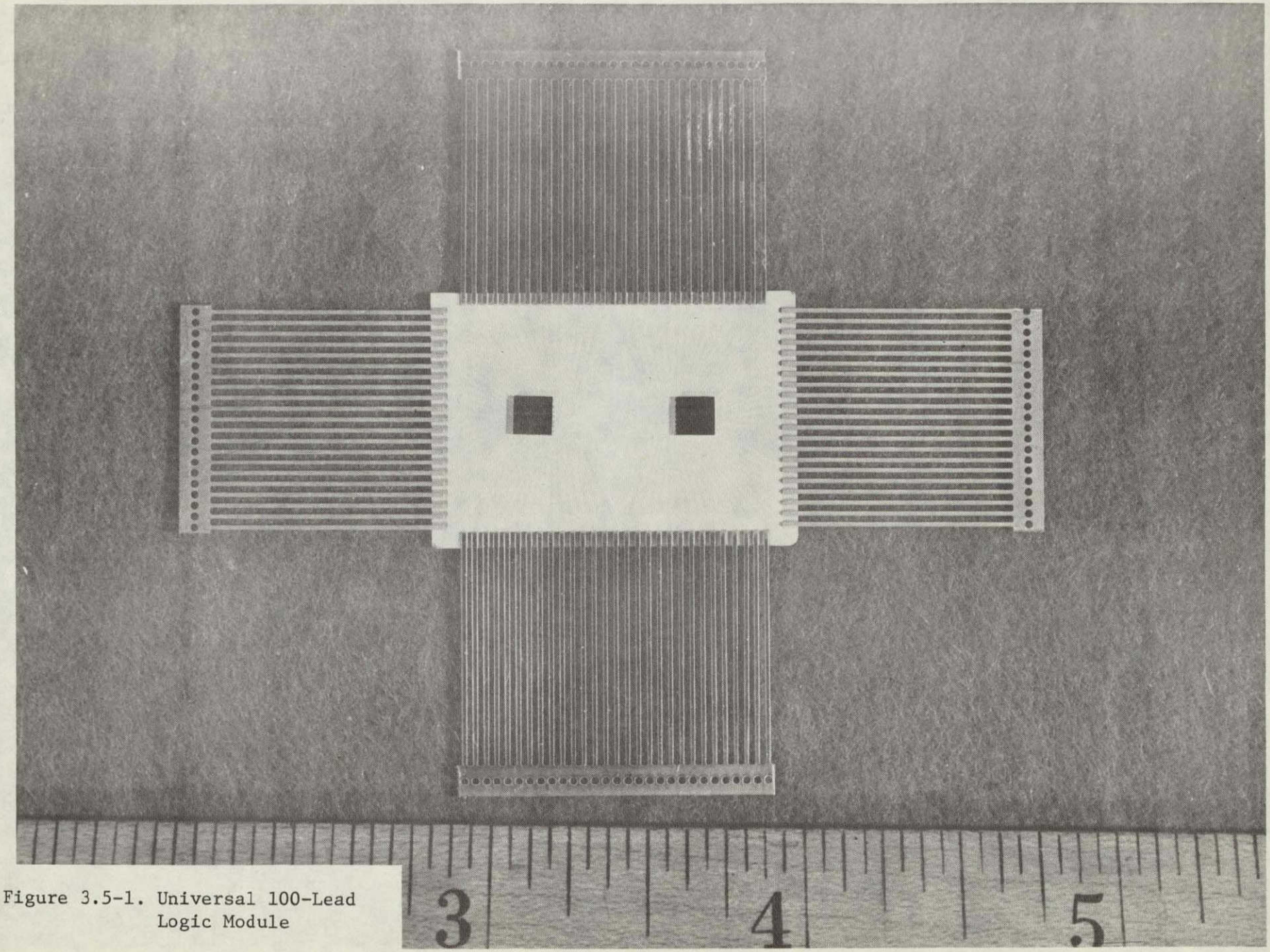


Figure 3.5-1. Universal 100-Lead Logic Module

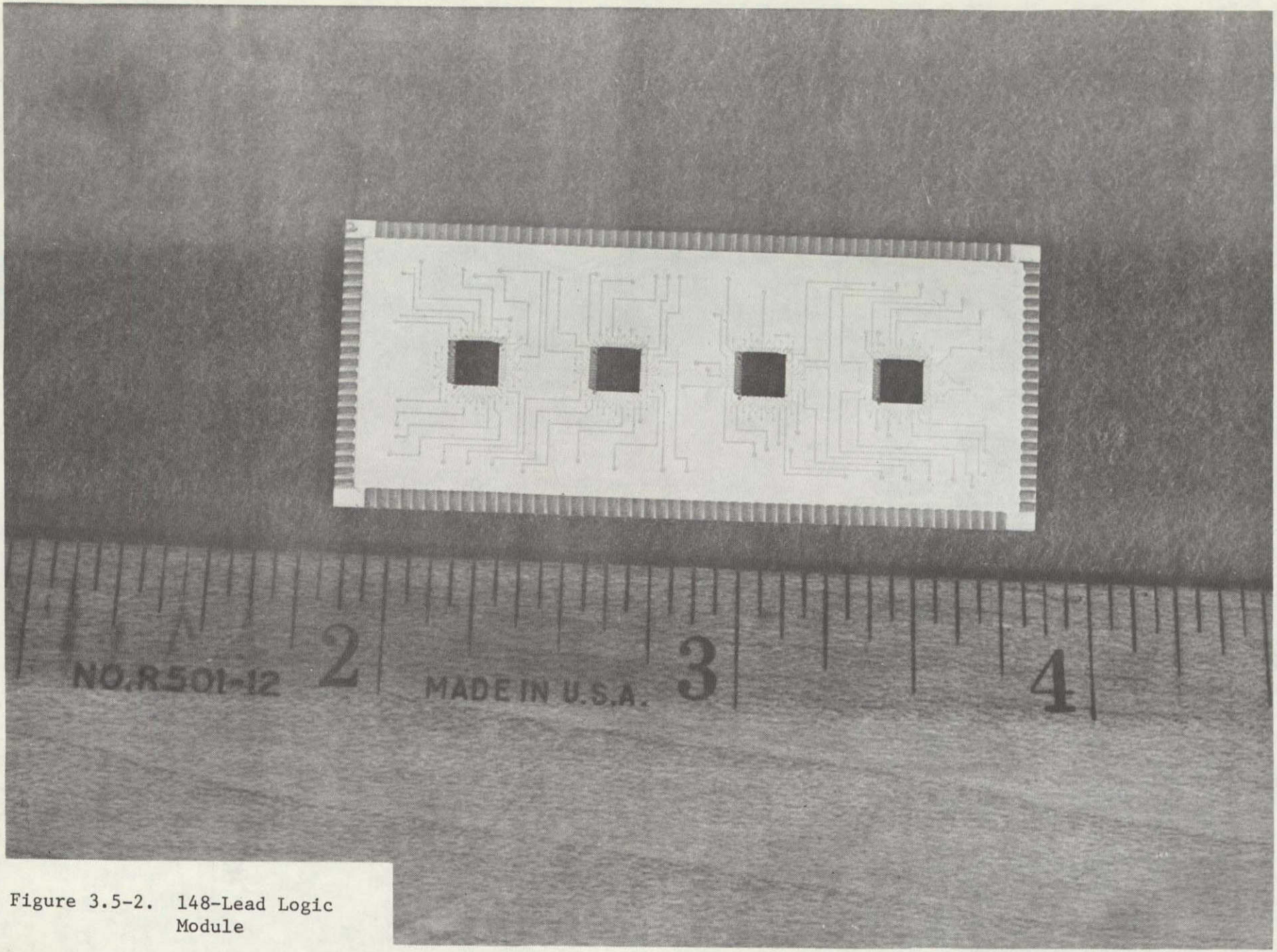


Figure 3.5-2. 148-Lead Logic Module

These hybrid modules become over sized flat packs attached to the solder tabs on the surface of the flex/MIB. The thermal support frame has suitable openings etched to allow the module I/O leads to be formed and hand soldered to the MIB surface. After testing and cleaning all solder joints, the exposed MIB surfaces are conformally coated to provide environmental protection. The coating material can be removed using a soldering iron for piece part replacement.

3.6 BASIC MEMORY MODULES (BMMs)

Each memory module contains four high-performance memory chips and two sense-amp chips attached to a metallized alumina ceramic substrate with chromium-copper-chromium (Cr-Cu-Cr) conductor patterns providing the interconnections between the chip solder pads and the module I/O pins. The conductor patterns are tinned and protected with a polyurethane coating. Thirty-six copper I/O pins are swaged in place and soldered. A rectangular ceramic cap is attached, with an epoxy preform, to provide a hermetically sealed enclosure for the chip circuitry. This module, without its cap, is shown in Figure 3.6-1.

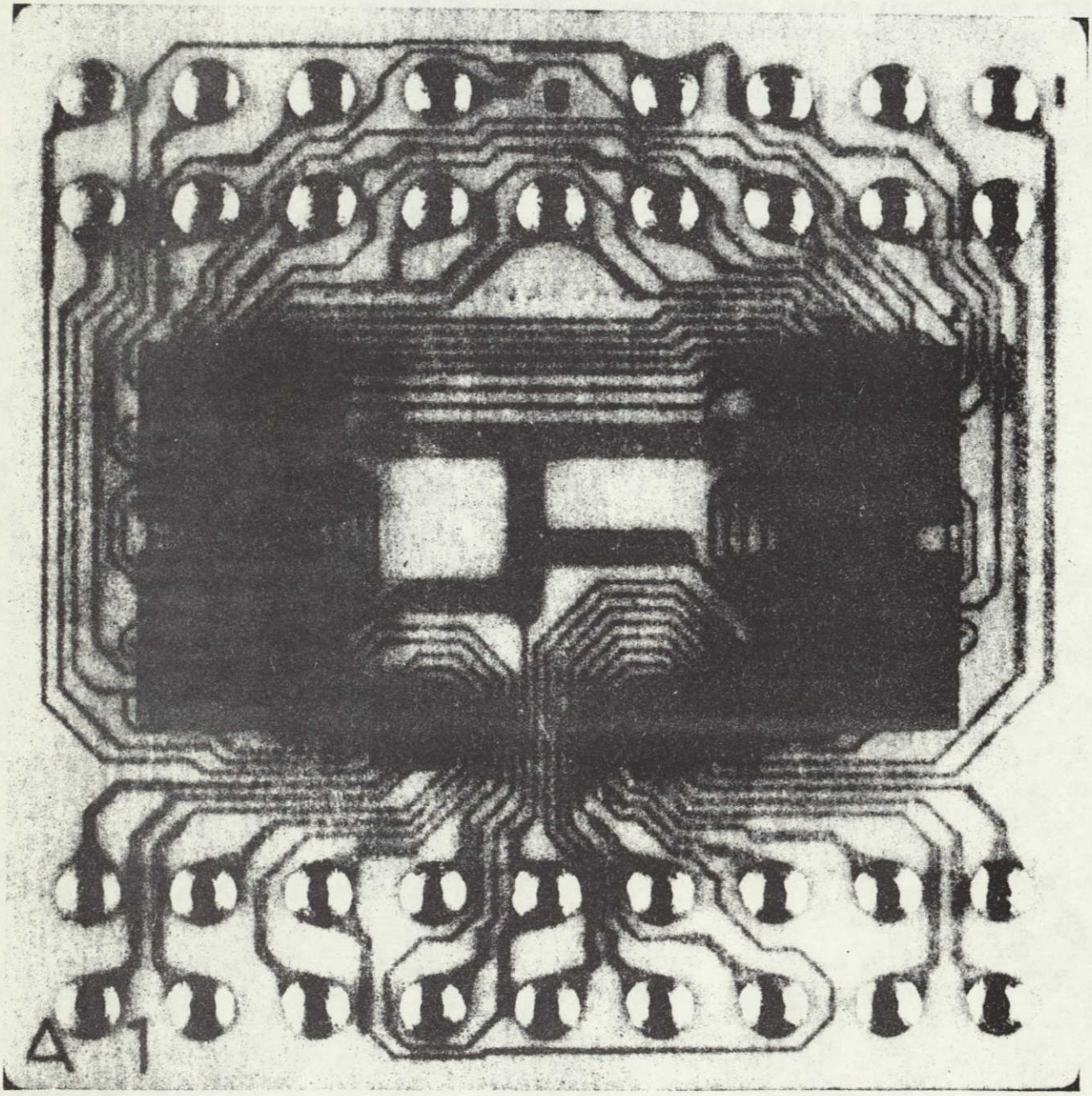


Figure 3.6-1. Basic Memory Module

3.7 POWER REGULATOR HYBRID MODULES

Three of the six hybrid modules used in the SUMC-IIB power regulator contain power semiconductors. There are four silicon NPN power transistors in one hybrid (Figure 3.7-1) and four silicon NPN power rectifiers in the second hybrid. A third diode module is populated with 14, one amp diode chips. Basically these modules have ceramic substrates with thick film conductor patterns providing the interconnections between the semiconductor chips and the I/O leads. The chips are attached to the substrate by an eutectic bonding procedure. Connections between the semiconductor and the conductor patterns are made using redundant wire bonds. Copper-alloy I/O leads are soldered and epoxy supported to an array of surface patterns located on the long sides of the substrate. A ceramic cap is sealed with an epoxy preform providing a hermetically sealed enclosure.

The three remaining modules are control hybrids namely, the Series Dissipative Regulator (SDR), the Pulse Width Modulator (PWM), and the Internal Voltage (IV). Figure 3.7-2 is typical of the control modules. The SDR module contains two series dissipative regulators capable of delivering up to seven amperes of current at output voltages of from 5 to 15 volts. Positive or negative outputs may be obtained from either regulator. Fold back current limiting is provided for both outputs. Fault detectors monitor both outputs and issue a warning signal if an undervoltage condition is sensed. The PWM module contains the control, timing and drive required to implement a voltage regulator using a pulse width modulated technique. The IV module performs the turn-on and turn-off sequencing of the HTC power supply and provides +12, -6, and -3.15 volt regulated outputs.

These modules contain several chip semiconductors and integrated circuits which are bonded and interconnected similar to the power hybrids. In addition, required resistors are screened and fired using thick film pastes with appropriate resistive values. The regulated voltages and other values are set using active trimming procedures. The I/O pins are lap soldered and epoxy supported to surface patterns on two sides of the substrate. Epoxy preforms are used to bond a ceramic cap for hermeticity.

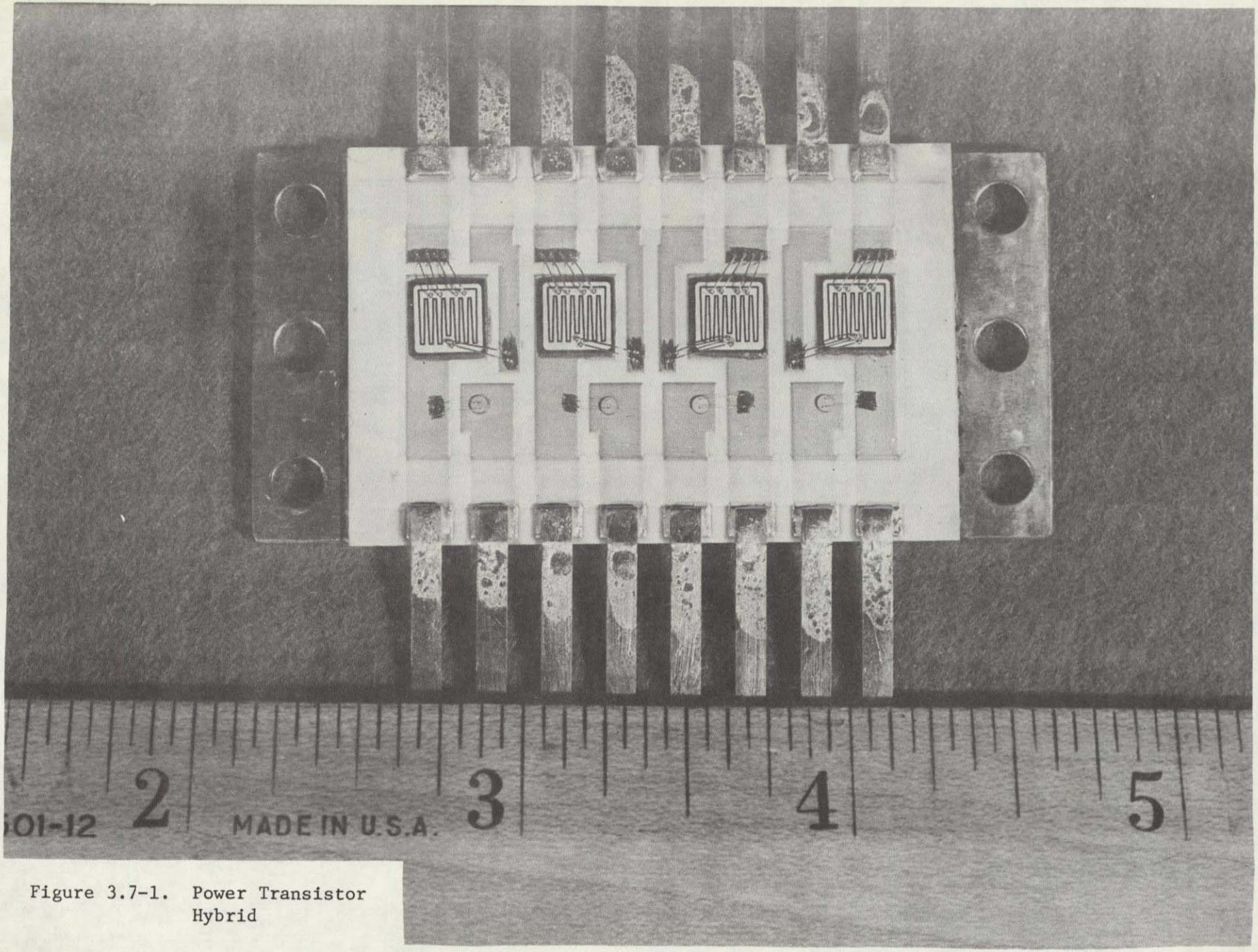


Figure 3.7-1. Power Transistor Hybrid

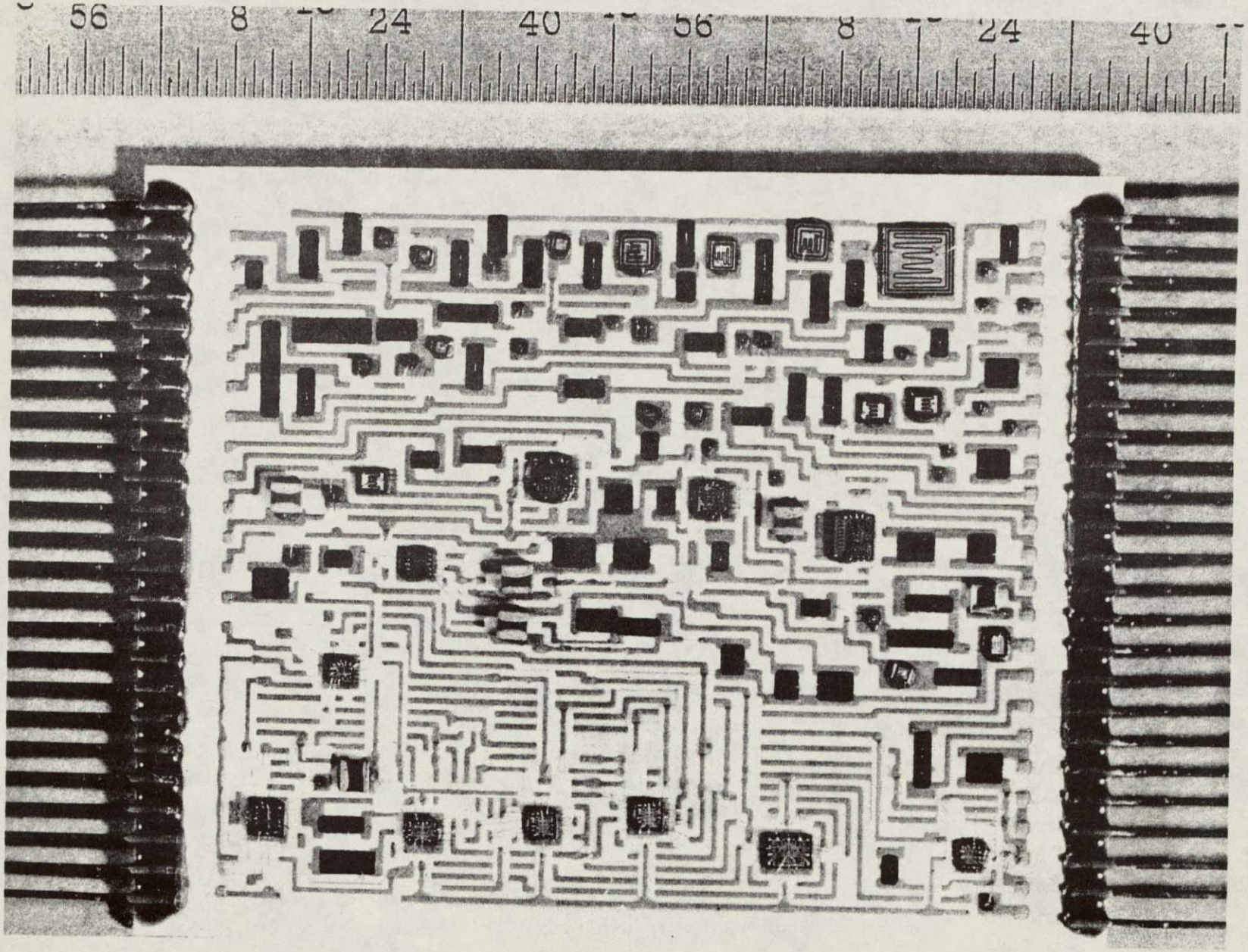


Figure 3.7-2. Internal Voltage Module

SECTION 4

FUNCTIONAL IMPLEMENTATION

The SUMC-IIB is implemented in three flexible levels of modularity: (1) functional slices (CPU/IO, Power Supply, MROM/Memory, and Memory only); (2) functional modules such as data flow, storage interface logic, storage array, and I/O interface; and (3) functional chips such as MUX/ALU, register, MUX/register, and timer. Each modular element is implemented to enhance general utility, wherever practical. This section of the report describes the functional implementation of the computer except for the power supply which is described in Section 5. Appendix A of the report describes the modules and chips used in the logic of the SUMC-IIB.

Table 4.0-1 lists the 13 logic module types used in the computer and identifies the complement of chips in each module type. There are 15 different chip types with a total of 71 chips used in the SUMC-IIB. Chip usage has a high of 25 for the MUX REG chip which is used in three different types of module. See Table 4.0-2. The REG chip is used nine times in four different module types. A total of 23 logic modules are used in the SUMC-IIB.

The organization of the SUMC-IIB can be seen in the block diagram of Figure 4.0-1. Reference to this diagram will add perspective to the detailed discussions constituting the remainder of this section.

The SUMC-IIB uses several technologies for the logic, memories, and registers. The principal technologies will be summarized here.

LSI Logic

Most of the logic in the computer is implemented in TTL, master slice chips, packaged in multichip modules as described below:

- Mature TTL Technology
- 100 Internal Gates
- 34 Off-Chip Driver Gates
- Low Delay/Power Product
- 45 Signal I/Os
- Standard 5400 TTL Compatibility
- Master Slice Customizing
- Flip-Chip Attachment

Table 4.0-1. Module and Chip Usage on SUMC-IIB

Module/Chip	ALD Sh.	Quantity		Part Number
		Module Req'd	Chip/Module	
1. Data Flow (148)		4		7929343-1
(a) Mux ALU	AB		2	7928744
(b) Mux Reg.	BC		3	7928751
2. SCU (148)		1		7930435
(a) Seq. Mux	DD		3	7928747
(b) Seq. Control	EF		1	7929711
3. TSE/SDR (148)		3		7929333-1
(a) Mux Reg.	BC		4	7928751
4. SIL (148)		1		7930436-1
(a) Mux Control 1	HH		1	7929702
(b) Mux Control 2	RS		1	7930274
(c) Mem Timing	KS		1	7930272
(d) Mem Control	JS		1	7930270
(e) Reg.	FG		1	7928752
5. I/O Interface (148)		1		7929342-1
(a) Seq. Mux	DD		3	7928747
(b) AOTC	GK		2	7929708
6. Register (100)		3		7929369-1
(a) Reg	FG		2	7928752
7. FCU/EALU (100)		1		7929368-1
(a) Mux ALU	AB		1	7928744
(b) FCU	CC		1	7928746
8. SPM Address Mux/Timing (100)		1		7929367-1
(a) Timing	QQ		1	7929707
(b) Mux Reg	BC		1	7928751

Table 4.0-1. Module and Chip Usage on SUMC-IIB (Continued)

Module/Chip		Quantity		Part Number
		Module Req'd	Chip/Module	
9. Arch (100)		1		7929336-1
(a) Arch	LL		1	7928753
(b) Reg	FG		1	7928752
10. Timer (100)		2		7929335-1
(a)	TT		2	7929710
11. I/O (100)		1		7929339-1
(a) I/O	PP		1	7929706
(b) Register	FG		1	7928752
12. AOTC (100)		3		7929341-1
(a) AOTC	GK		2	7929708
13. Mem Support (100)		1		7930437
(a) Mem Timing	KS		1	7930272
		23		

NOTE: The number in parentheses in the left column identifies whether the module uses a 100 lead universal 2-chip carrier or a 148 lead multi-layer substrate.

15 - Chip Types (71 Total)
5 - 148 Pin Module Types (10 Total)
8 - 100 Pin Module Types (13 Total)

Table 4.0-2 LSI Chip Usage

Chip Name	No. of Times Used	No. of Module Types Used In
MUX REG	25	3
MUX ALU	9	2
REG	9	4
AOTC	8	2
SEQ MUX	6	2
TIMER	4	1
MEMORY TIMING	2	2
SEQ CTRL	1	1
MUX CTRL 1	1	1
MEM CTRL	1	1
FCU	1	1
TIMING	1	1
ARCH	1	1
I/O (CTRL)	1	1
	71	

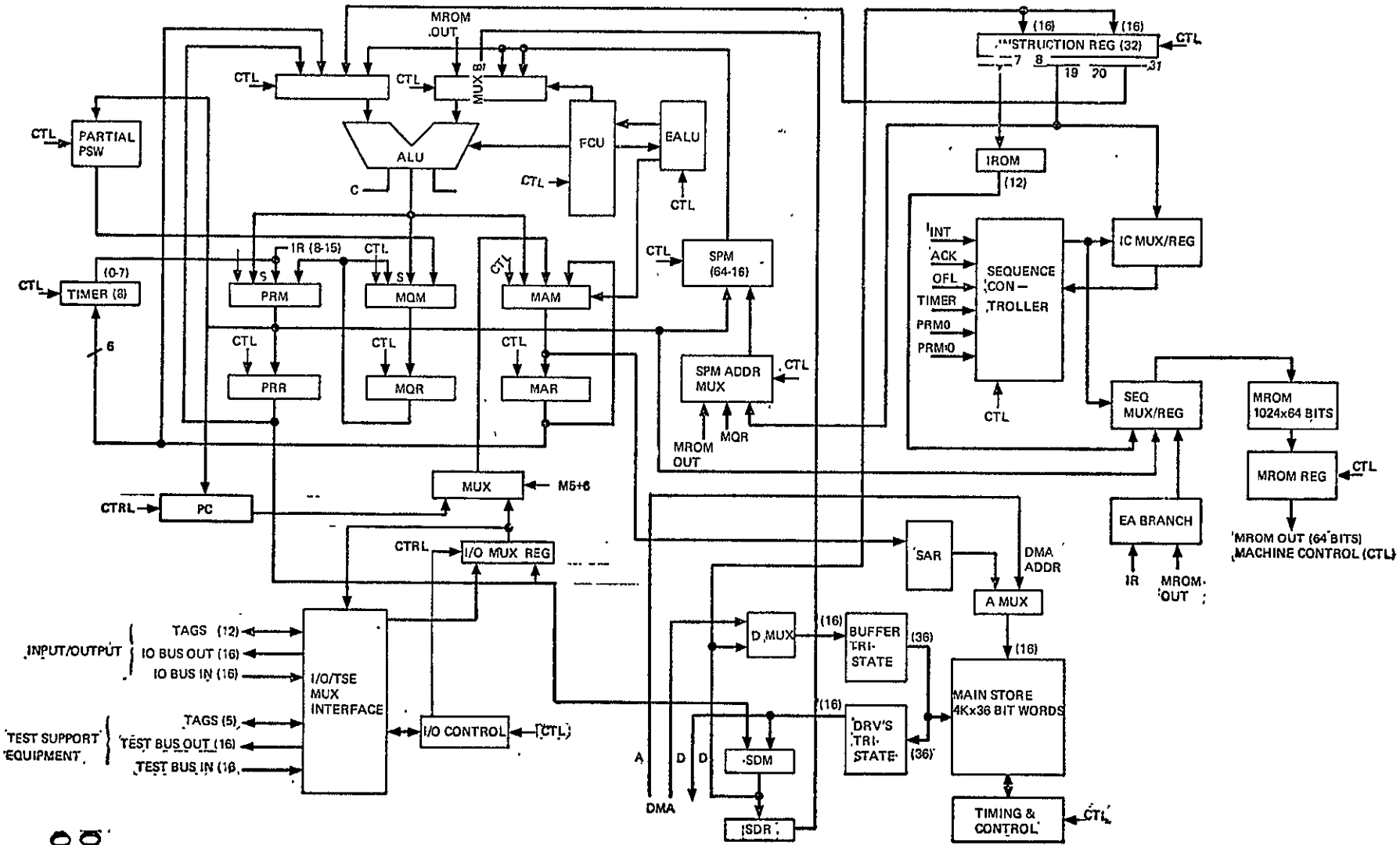


Figure 4.0-1. SUMC IIB Block Diagram

SPM Registers

The general registers, floating point registers, and some work space for micro-program use is implemented in a register chip as follows:

- Organized 64 words by 9 bits
- 60 ns access
- Full 5400 TTL compatibility
- 750 mw/chip
- 28 pin DIP
- Inverts data

PROMs (MROM & IROM)

The ROMs used to hold the microprogram (MROM) and for OP code decode (IROM) are of the field programmable type commonly called PROMs. The circuit used for this function is organized 512 words by 8 bits and comes in a 24 pin DIP. Several manufacturers offer pin compatible parts for this PROM including MMI, Fairchild, and Harris. The part has a typical access time under 100ns and power dissipation about 600mw. All interfaces are fully 5400 TTL compatible.

Memories

The principle main store technology for the SUMC-IIB is the basic memory module (BMM) developed for the SUMC program and described in detail in IBM Report Number 74-585-006 dated 30 June 1974. This memory module contains four array chips and two driver/sense amplifier chips in a configuration which can be used either 4K-words by two bits or 8K-words by one bit. The module has an access time less than 200 ns, a cycle time less than 250ns and an operating power around 500 mw depending upon usage.

Core memories and Read Only Monolithic memories are also available for applications which would prefer them although the specific SUMC-IIB configuration has not been fully developed to date.

4.1 DATA FLOW

The SUMC-IIB data flow consists of four Data Flow modules. Each module contains a MUX/ALU and three 4-bits, MUX/REG's.

The right input to the ALU is driven by Mux A choosing the contents of the PRR, SPM, MAR or IR bits 20-31. The left input to the ALU is driven by Mux B, selecting the contents of the SDR, MROM bits C7 thru C17, SPM or SPM shifted right arithmetic one (called 1/2 SPM).

The standard arithmetic and logic operations are performed under direct microprogram control in a single pass through the ALU. Special operation such as multiply and divide are performed under control of the Functional Control Unit (FCU) as explained in paragraph 4.5.1. These special operations require several passes through the ALU for completion and are data sensitive.

The working MUX/REG's are identified below:

A. PRM - Product Remainder Multiplexer
   PRR - Product Remainder Register

B. MAM - Memory Address Multiplexer
   MAR - Memory Address Register

C. MQM - Multiples Quotient Multiplexers
   MQR - Multiplies Quotient Register

Each register input is supplied by the associated three input multiplexer, with one of the three inputs providing shift capability. These registers are invisible to the computer programmer but are used by the microprogram for instruction execution, address manipulation, and "housekeeping" functions. The contents of the registers are not carried over from one instruction to the next.

4.1.1 ALU Mux

Four properly interconnected Data Path modules provide the SUMC-IIB with a 16-bit ALU which has A and B inputs provided from two independent multiplexers. Multiplexer A is a four-input mux with control derived from MROM bits A1, A2 and A3. This three-bit subfield selects the data to be applied to ALU input A as shown in Table 4.1-1.

Table 4.1-1 ALU MuxA

MROM A1	MROM A2	MROM A3	ALU MUXA Operation
0	0	0	PRR + SPM (Logical OR)
0	1	1	PRR
0	0	1	PRR + MAR (Logical OR)
0	1	0	PRR + IR20-31 (Logical OR)
1	0	0	SPM
1	1	1	Zero
1	0	1	MAR
1	1	0	IR20-31

Multiplexer B is a three input mux with one input having the capability to be shifted right one and the MSB replaced with an input from the module (SPMSG)*. A three-bit MROM mux control subfield (A5, A6 and A7) combined with the MROM ALU control subfield (A8, A9, and A10) selects the ALU input B.

When the ALU operation is not multiply, divide, or square root (MDS) ALU input is selected as shown in Table 4.1-2.

*The SPMSG input is wired to SPM0 (Sign bit).

Table 4.1-2 ALU MUX B

MROM A5	MROM A6*	MROM A7*	ALU MUXB Operation
0	1	1	SDR + MROM C7-C17 (Logical OR)
0	0	0	SDR
0	1	0	SDR + SPM (Logical OR)
0	0	1	SDR + 1/2 SPM** (Logical OR)
1	1	1	MROM C7-C17
1	0	0	Zero
1	1	0	SPM
1	0	1	1/2 SPM **

* The physical control signals SEL3N and SEL4N which are generated in the FCU actually control operation of the ALU. Paragraph 4.5.1.8 explains the SEL signals.

**1/2 SPM is the current output of the SPM shifted right arithmetic 1.

Mux B controls for MDS Functions are defined in the following paragraph.

4.1.2 ALU

The ALU performs the arithmetic operations identified in Table 4.1-3.

Table 4.1-3 ALU Functions

MROM A8	MROM A9	MROM A10	ALU Function
0	0	0	Logical AND
0	0	1	Subtract (B-A) *
0	1	0	Subtract (A-B) *
0	1	1	Special MDS
1	0	0	Logical OR
1	0	1	Logical Exclusive OR
1	1	0	Add
1	1	1	MDS

The SUMC-IIB contains special logic to simplify execution of the multiply, divide, and square root operations (MDS). When A8-A10 are 111 or 011, MROM A6 and A7 become extensions of the function code. Control of the ALU operation, as well as selection of data for the B side of the ALU, is exercised by the special MDS logic. MROMA6 and A7 specify multiply, divide, or square root operation operations as shown in the Table 4.1-4.

Table 4.1-4 MDS Control

MROM A6	MROM A7	MDS Functions
0	0	Square Root Sign
0	1	Multiply
1	0	Divide/Square Root
1	1	Divide Sign

To utilize (μ program) this special logic the multiply, divide and square root algorithms must be understood. In the SUMC-IIB the special MDS operation (011) operates identically with the MDS (111). The MDS controls described below are for multiply and divide operations of 16-bit numbers because the data path can handle that size. The microprogramming of the computer must perform double precision operations to handle the 32-bit parameters required by the S/360 architecture.

ALU Control for MDS

1) Square Root Sign (SWRT SIGN)
Since square root is not implemented in the SUMC-IIB this operation will not be described.

2) Multiply Algorithm
For each step of the algorithm three bits of the multiplier (MQR 14, 15 and 16) are tested and the multiplicand (held in SPM) is scaled and added to (or subtracted from) a running sum of partial products held in the PRR. Each step results generation of two bits (starting with the least significant) of the product. These two bits are shifted into the MAR as they are generated. To scale each step of the algorithm all three registers (PRR, MAR, MQR) are right shifted two bits at each pass except the last; one the last the MAR is shifted only one bit to adjust the location of the binary point. Note that the product generated is 31-bit two's complement number with the sign and most significant half in the PRR and a significant bit of data in the sign location of the MAR which holds the low order half of the number. The LSB of the MAR is always zero.

To implement multiply and divide, the EALU (ALU 16 and 17) are used as least significant extensions of the ALU. Control of multiply is shown in the Table 4.1-5. For proper operation MROMA4 must be a 1 to select MAR 0 and 1 for A inputs to ALU 16 and 17. Also MROMA5 must be a 1 to prevent selection of memory data for the ALU. See Table 4.1-6 for a Multiply example.

Table 4.1-5. Multiply Algorithm

MQR 14	MQR 15	MQR 16	ALU B INPUT 0-15	ALU B INPUT 16	ALU B INPUT 17	ALU 18	CARRY* 18	ALU FUNCT 0-17	ALU A INPUT 16	ALU A INPUT 17
0	0	0	0's	0	0	MAR 2	0	ADD	MAR 0	MAR 1
0	0	1	1/2 SPM**	SPM 15			0	ADD		
0	1	0	1/2 SPM**	SPM 15			0	ADD		
0	1	1	SPM	0			0	ADD		
1	0	0	SPM	0			1	SUB		
1	0	1	1/2 SPM**	SPM 15			1	SUB		
1	1	0	1/2 SPM**	SPM 15			1	SUB		
1	1	1	0's	0			1	SUB		

* Forces a carry into ALU 17 to get 2's complement for subtract. C16 is the normal carry from bit 16 to bit 15.

** 1/2 SPM is the SPM with a one-bit right arithmetic shift.

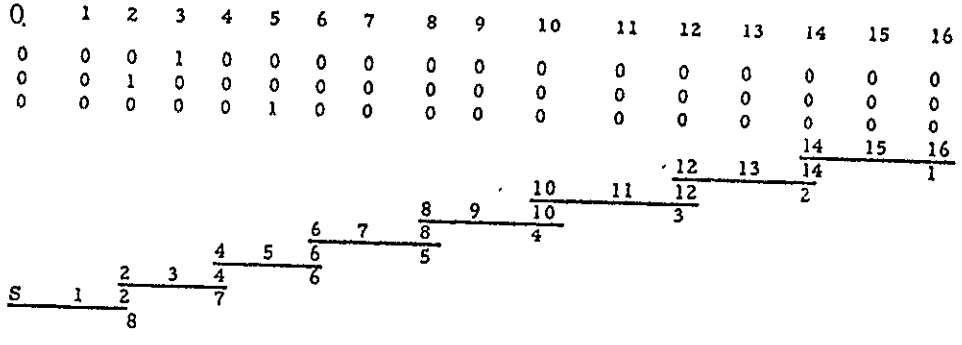
Notes:

1) MROMA Codes FOR MULTIPLY ALGORITHM

1	2	3	4	5	6 7 8	9	10	11	12
1	0	0	1	0	01	1	1	X	0
PRR			MAR	$\overline{\text{SDR}}$	MPY	MDS			$\overline{\text{CL}}$

2) Other controls for programming the multiply algorithm: PRM, MAM, and MQM all shift right two bits. All three registers are loaded. NO SPM write occurs. The microprogram step is repeated until IC = 0(7 passes for a 16-bit by 16-bit multiply).

Table 4.1-6 Multiply Algorithm Example



Multiplicand (1/8) SPM
Multiplier (1/4) MQR
Product (1/32) PRR/MAR

1. Zero PRR, MAR, MQR and MQR16
2. Multiplier MQR
3. Multiplicand SPM
4. Test MQR 14-15-16 and execute per Table 4.1-5
5. Shift PRR, MAR and MQR right 2
6. Repeat above 6 more times (Total of 7)
7. For 8th pass no shift

Note: If Multiplier or Multiplicand is negative number, they must be 2's complimented before multiply is started.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		
	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0		
	S	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		
			2		4		6		8		10		12		14		16		
			8		7		6		5		4		3		2		1		
First Pass Shift Right 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
									"										
									"										
									"										
Sixth Pass Shift									"										
Seventh Pass (100)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRR-SPM	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2's Comp	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shift Right 2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Eighth Pass (001)																			
PRR + 1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1/2 SPM	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0 = 1/32 Product

ALU 18 bits wide

3) Divide/SQRT Algorithm

Since the SUMC-IIB does not implement square root (SQRT), only the divide feature will be described. Each step in the divide process forms one bit of the quotient. The divide function divides a 32-bit dividend held in the PRR/MAR registers by a 16-bit divisor held in SPM. The 16-bit quotient is formed in the MQR. Dividend, divisor and quotient are two's complement numbers. MAR 0 contains a significant bit of the divident. The micro-program for divide sign must be executed once before the normal divide microprogram step. The divide step is executed 15 times to form the quotient. For divide (all MDS operations) the ALU is 18 bits wide. The output of ALU 16 and 17 enters the MAM and PRM as shown in MAM and PRM operations.

The divide algorithm is a non-restoring algorithm. Each time the algorithm step is executed the following operations take place:

a) The sign of the PRR is compared with the stored sign of the SPM (Divisor). (The SPM sign is stored at CKZ time when the MQR load is enabled.

b) If the signs are the same, the divisor (held in SPM) is shifted right one bit and substracted from the PRR/MAR. Shifting of the SPM is under control MROM A11 (shift if ROM A9 = 0). If the signs are different the divisor is shifted right one and added to the PRR/MAR.

c) A bit (called Q) is generated which is a 1 if the ALU sign is the same as the divisor sign. The ALU sign is a result of the subtraction/addition of step b).

d) The PRM and MAM are shifted left one-bit and loaded into the PRR and MAR. At the same time the MQR is shifted left one-bit and the function Q is entered into its least significant bit. (Step (d) is controlled by normal use of the ROM R field).

After 15 iterations of the divide step the quotient is in the MQR. If the quotient is negative a one (1) must be added to the quotient by a subsequent microprogram step. Also if a remainder is to be formed the divisor must be added to or subtracted from the PRR depending on the sign of the divisor. (Add if positive.)

4) Divide sign

The divide sign microprogram step must be executed one time before executing the divide algorithm step just described. The operation of the special MDS logic for divide sign is identical to divide algorithm except that the function Q is 1 when the stored sign of the SPM (divisor) is different from the sign of the PRR (divident). It should be noted that the divisor must be read from SPM sign stored. During each step of the divide or divide sign the SPM sign is re-stored. See Table 4.1.7 for a typical Divide example utilizing a 16-bit dividend and an 8-bit divisor. This example requires 7 iterations of the Divide Algorithm and one of the Divide Sign.

Table 4.1-7. Divide Example

Dividend (PRR & MAR) = 0.1001000 00000000 = 9/16

Divisor (SPM) = 0.101000 = 10/16

	0	1	0	0	1	0	0	0	0	0	0	0	
-	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR
	0	0	1	0	0	0	0	0	0	0	0	0	1/2 SPM
	0	1	0	0	0	0	0	0	0	0	0	0	QS = 0
-	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left. 1
	0	0	0	1	1	0	0	0	0	0	0	0	1/2 SPM
	0	0	1	1	0	0	0	0	0	0	0	0	Q1 = 1
-	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	0	0	0	0	1	0	0	0	0	0	0	0	1/2 SPM
	0	0	0	1	0	0	0	0	0	0	0	0	Q2 = 1
-	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	1	1	1	0	1	0	0	0	0	0	0	0	1/2 SPM
	1	1	0	1	0	0	0	0	0	0	0	0	Q3 = 1
+	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	1	1	1	1	1	0	0	0	0	0	0	0	1/2 SPM
	1	1	1	1	0	0	0	0	0	0	0	0	Q4 = 0
+	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	0	0	0	1	1	0	0	0	0	0	0	0	1/2 SPM
	0	0	1	1	0	0	0	0	0	0	0	0	Q5 = 0
-	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	0	0	0	0	1	0	0	0	0	0	0	0	1/2 SPM
	0	0	0	1	0	0	0	0	0	0	0	0	Q6 = 1
-	0	0	1	0	1	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	1	1	1	0	1	0	0	0	0	0	0	0	1/2 SPM
	1	1	0	1	0	0	0	0	0	0	0	0	Q7 = 1
+	0	1	0	1	0	0	0	0	0	0	0	0	PRR/MAR Shift Left 1
	0	0	1	0	0	0	0	0	0	0	0	0	SPM

Remainder

MQR

Force Carry

The MROMA11 bit has two uses. It is normally used to force a carry into ALU bit position 15 (to give a 2's complement or increment a register). This function is performed if MROMA9. MROMA10 = 0 (all functions other than M.D.S.). For MDS functions, ROMA11 is used in determining the source of data for the B input to the ALU as shown in Table 4.1-8.

Table 4.1-8 ALU B Source for MDS Functions MROM A Field

8	9	10	6	7	11	ALU B SOURCE
1	1	1	0	1	x	FUNCT OF MQR 14-16
1	1	1	1	0	0	1/2 SPM*
1	1	1	1	0	0	
1	1	1	1	1	0	
1	1	1	0	0	1	SPM **
1	1	1	1	0	1	
1	1	1	1	1	1	

* ALU_B 16 = SPM 15, ALU_B 17 = 0

** ALU_B 16 = ALU_B 17 = 0

Overflow Latch

The MROMA12 bit can be used to generate C16 (the carry into ALU15) under the following conditions. For ALU operations other than MDS, C16 is a 1 if MROMA12 = 1 and there was a carry on the previous microinstruction cycle. A function OFL is generated in the ALU which is a 1 if the carry into and out of the sign-bit are not the same.

For MDS operations, the MROMA12 bit must be zero to allow proper generation of C16 through normal carry propagation from the lower order bits of the ALU.

Multiplexer (EALU 16-17) (Ref. Section 4.4)

This one-bit subfield (ROMA4) selects the A input to ALU bits 16 and 17, by controlling the mux input gating as follows:

Table 4.1-9 EALU A Inputs

MROM	ALU Input	
A4	Bit 16	Bit 17
0	PRR0	PRR1
1	MAR0	MAR1

Since the PRM and MAM operations use the ALU 16 and 17 outputs, this bit is used in conjunction with the MROM R-bits as well as with MROM A-bits. The B input to ALU 16-17 is as follows:

- For operations other than MDS it is controlled by ROMA 6 and 7.

MROM A		EALU B IN		ALU B IN
6*	7*	16	17	0 - 15
0	1	SPM15	0	1/2 SPM
0	0	0	0	All zeros
1	0	0	0	SPM
1	1	0	0	MROM (C7-C17)

- For MDS operations the B input to ALU 16-17 is somewhat complex.
  - For multiply they are as previously described in the Multiply Algorithm Table 4.1-5.

    For the divide/square root algorithm, divide sign, and square root sign, the inputs are controlled by MROMA11 as shown:

	EALU B Input	
MROMA11	Bit 16	Bit 17
0	SPM 15	F25A**
1	F25A**	0

Where F25A = 1 when the sign of the PRR equals the stored scratch pad memory sign for square root sign and 0 otherwise.

* Physical control signals are SEL3N and SEL4N from the FCU Chip.

**F25A Input to EALU is wired to logic 0 for SUMC-IIB.

4.1.3 Mux/Reg

The four interconnected data flow modules provide the SUMC-IIB with three 16-bit working registers (PRR, MAR and MQR). Each register input is supplied by a 3 input multiplexer, with one of these inputs providing the shift operations required for the SUMC-IIB.

The following MROM register subfields are used to control loading data into the three registers. The MROM bit identified by Table 4.1-10 and the trailing edge of CKZ causes the respective register to be loaded.

Table 4.1-10. Register Load Control

REG	CONTROL BIT OR FIELD
PRR	MROMR10 = 1
MAR	MROMR5 = 1
MQR	$\overline{\text{MROMR11}} \cdot \overline{\text{MROMR12}} \cdot \overline{\text{MROMR13}}$

The MROM subfield identified below for each mux is used to control the select inputs and shifting inputs to the three respective multiplexers (see Table 4.1-11, 4.1-12, and 4.1-13).

TABLE 4.1-11 PRM Operation

A = ALU
M = MAR
I = IR
R = MQR

MROM BITS				PRM OPERATION																OPERATION
R6	R7	R8	R9	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 (Null)
0	1	0	0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	ALU
1	0	1	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	Left 1 Logical Long
1	1	1	0	A0	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	Left 1 Arithmetic Long
1	0	0	1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	Left 2 Logical Long
1	1	0	1	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	M2	M3	Left 4 Logical Long
1	1	1	1	A0	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	M2	M3	Left 4 Arithmetic Long
1	0	1	1	0	A0	A1	A2	A3	A14	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Right 1 Logical
0	0	1	1	A0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Right 1 Arithmetic
0	1	0	1	0	0	0	0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	Right 4 Logical
0	0	0	1	A0	A0	A0	A0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	Right 4 Arithmetic
0	1	1	1	A15	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	Right 1 Rotate
0	0	1	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	0	Left 1 Logical
0	1	1	0	A0	A0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	Right 2 Arithmetic
1	1	0	0	0	0	0	0	0	0	0	0	I8	I9	I10	I11	I12	I13	I14	I15	Instruction Register
1	0	0	0	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	MQR

Table 4.1-12 MAM Operation

A = ALU
M = MAR

MROM BITS				MAM OUTPUT															
R1	R2	R3	R4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	OPERATION
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 (Null)
0	1	0	0	A16	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	MAR
1	0	1	0	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	0	Left 1 Logical
1	1	1	0	A16	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	0	Left 1 Arithmetic
1	0	0	1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	0	0	Left 2 Logical
1	1	0	1	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	0	0	0	0	Left 4 Logical
1	1	1	1	A16	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	0	0	0	0	Left 4 Arithmetic
1	0	1	1	A15	A16	A17	M2	M3	M14	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	Right 1 Double
0	0	1	1	0	A16	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	Right 1 Logical
0	1	0	1	A12	A13	A14	A15	A16	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	Right 4 Double
0	0	0	1	0	0	0	0	A16	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	Right 4 Logical
0	1	1	1	0	A16	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	Right 1 Logical
0	0	1	0	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	MQRO	Left Rotate Double
0	1	1	0	A14	A15	A16	A17	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	Right 2 Double
1	0	0	0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	ALU
1	1	0	0	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	IO/TSE/P. CTR*

*If M5 + M6 = 1, The IO/TSE is selected.

Table 4.1-13 MQM Operation

A = ALU
R = MQR
Q = Quotient

MROM BITS*			MQM OUTPUT															
R11	R12	R13	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	OPERATION
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 (Null)
0	1	0	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	MQR
1	0	1	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	Q	Left 1 Quotient
0	1	1	0	0	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	Right 2 Logical
1	0	0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	ALU
1	1	0	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	PSW

* MQR Load Enable Controlled By: $\overline{R11 \cdot R12 \cdot R13}$

4.2 REGISTERS

In addition to registers which are integrated into multiplexers (such as the MUX REG just described in the data flow discussion), there are several registers used in a relatively stand-alone manner. The Instruction Register (IR) and control word holding register are examples which use the two-chip register module. Another example is the 64 word x 16 bit register stack called the scratch pad memory (SPM) which holds the S/360 general and floating-point registers as well as a number of registers used by the microprogram.

The Register Module contains two 16-bit register chips, which provide eight individual 4-bit registers. Each register is controlled by its gate and clock inputs. Register loading occurs at the trailing edge (fall) of the clock when the gate input line is at a logic one state.

4.2.1 MROM Register

The MROM Register utilizes two Register modules (four chips) to configure a 64-bit register. Loading of this register is controlled by clock CKX. The output of this register specifies which control lines are to be active during each micro cycle. Each bit specifies the status of a control line during each micro cycle. The 64 bits are divided into the five major subfields identified in Figure 4.2-1 and operate either individually or in small groups to control some part of the machine.

4.2.2 Instruction Register

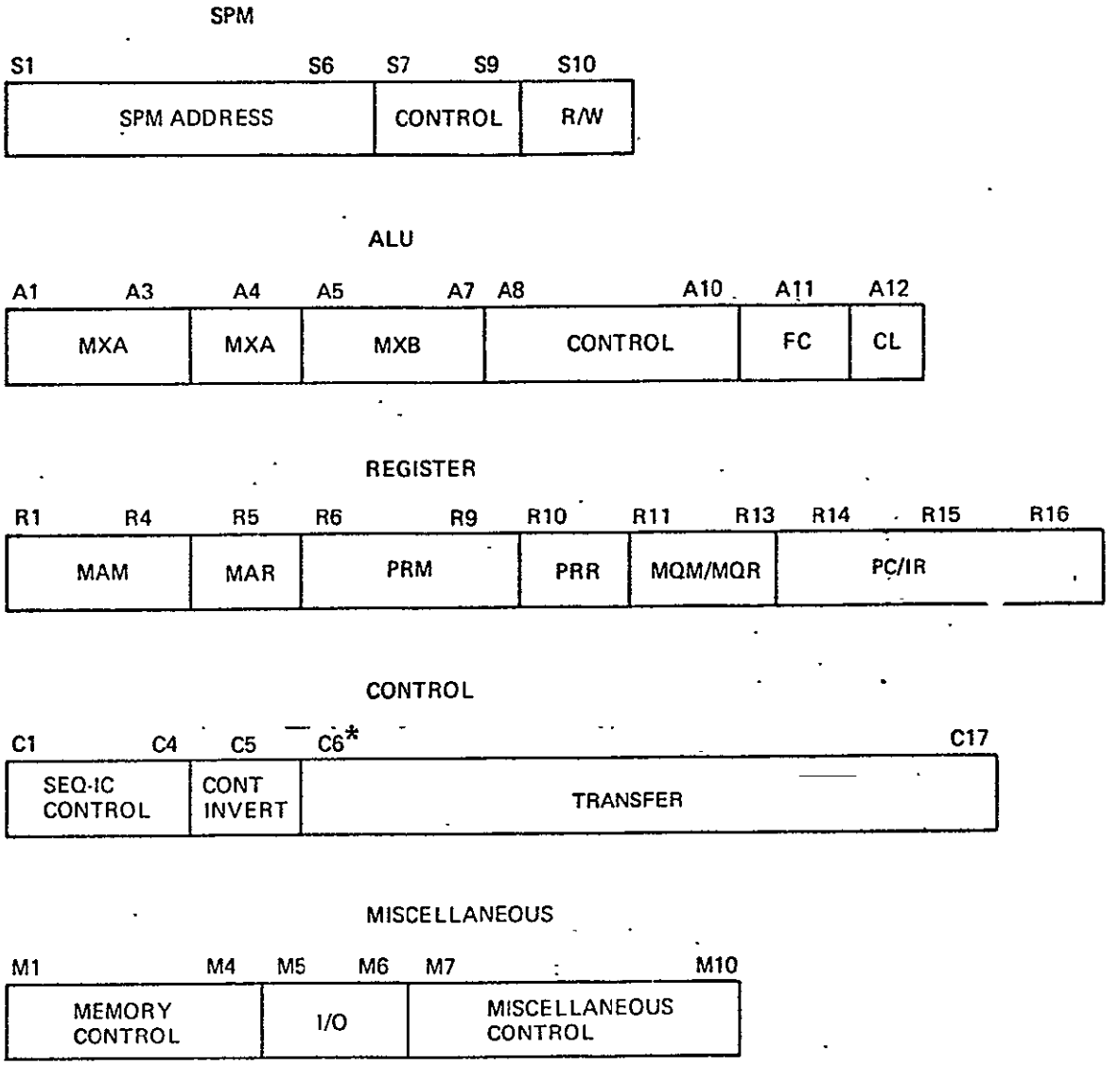
The Instruction Register (IR) for the SUMC-IIB uses a single Register module to provide two 16-bit registers independently controlled by micro code (MROM). Register loading is controlled by CKX and the following MROM control bits.

LOAD ACTION	MROM 14-16
IR (Left)	0 1 1
IR (Right)	1 0 0

4.2.3 Register Chip

The register chip is divided into four 4-bit registers each with its separate gate, clock and reset inputs.

The chip provides a NOR output whose four inputs are tied internally to register outputs from two of the 4-bit registers. Figure 4.2-2 shows the NOR configuration and general chip functioning.



*C6 – NOT USED FOR 16 BIT HTC

Figure 4.2-1 Microprogram Control Word Format

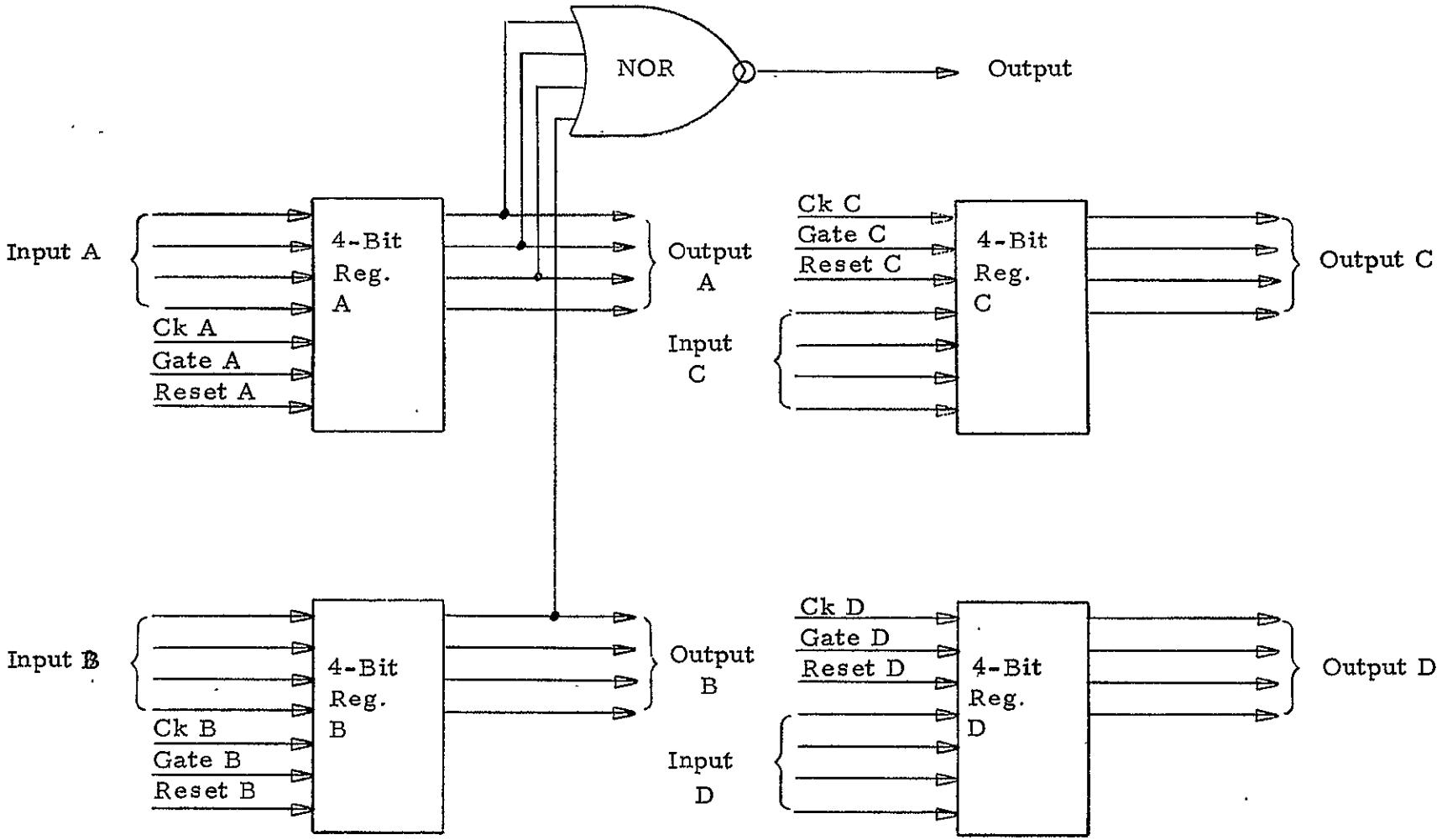
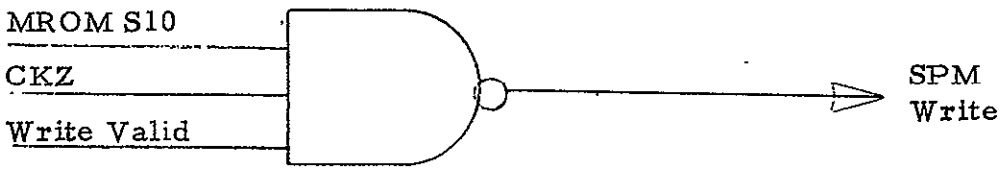


Figure 4.2-2 Register Chip

4.2.4 Scratch Pad Memory (SPM)

The SPM is a 64 word by 16 bit register stack. Addressing of the SPM is partially controlled by a SPM address multiplexer located in the SPM ADDR/Timing Module (see subsection 4.6). MROM bits S1 and S2 directly control the two address MSBs but the four address LSBs are controlled by the MUX. These two source combine to provide a 6-bit SPM address. The SPM is configured so that it reads the location specified by the address except when a write pulse is generated. Generation of the SPM write pulse occurs in the timing chip and causes the PRM data on the SPM input lines to be stored in the location specified by the address. Writing into SPM occurs at CKX time. The write pulse is generated on the timing chip when MROM S10 equals a logical 1, AND CKS occurs, AND the "write valid" signal equals a logic 1. The write valid signal is generated by the Arch Module's exception monitoring logic as described in 4.7.3. Figure 4.2-3 shows the functional implementation of gates in the timing chip which generate the SPM write pulse.



NOTE: Gate contained on MISC Module Timing Chip

Figure 4.2-3. SPM Write Pulse Generation

Output data from the SPM is the complement of data written into the SPM, therefore the output of the SPM is inverted before it is made available to the data path. The Write Valid signal is generated in teh ARCH module. This signal goes to ZERO when the ARCH functions detect errors which might result in bad data being loaded into the SPM. The same signal is used to enable memory write operations.

4.3 TIMING

Timing within the SUMC-IIB consists of a set of three clocks (CKX, CKY, and CKZ) and the load pulse for the sequence register. Other timing signals required in the CPU are derived from the clocks and the appropriate functional signals by the "User". A basic CPU cycle or microcycle is from the fall of CKZ to the fall of the next CKZ. At this time the basic data path registers are loaded with the results from the current computer cycle and the MROM register is loaded with microprogram control word which will control the CPU for the next machine cycle. Figure 4.3-1 shows the basic CPU timing.

The computer cycle time is dependent upon the operation being performed as delineated below.

- Logical and add/subtract operations take four clock cycles of 110 ns each (440 ns).
- MDS functions take five clocks or 550 ns.
- Operations overlapped by memory cycles will be stretched until completion of the memory cycle.
- Microprogram sequence decisions based on the ALU output (called late decisions) stretch the normal cycle by three clock or 330 ns.
- A computer modification has been defined but not implemented which will allow the logic operations to be performed in three clocks or 330 ns.

The timing chip provides a load pulse for the SCU register contained in the SCU module (see paragraph 4.4.1). This load pulse is generated once each micro cycle when a MROM prefetch is valid (Reference Table 4.4-1). The load pulse is generated twice each micro cycle in which an MROM conditional branch does not specify "plus-one" for the sequencer. Generation of this second load pulse is controlled by an early/late decision signal. With this signal in the reset state, an early decision load pulse is generated and with it in the set state a late decision load pulse is generated. The signal reset is controlled by the trailing edge of CKZ thus causing an early decision load pulse to occur every micro cycle except when the MROM C1 -- C5 indicates the sequencer is to hold, then the early load pulse is inhibited. The signal set condition is controlled by the trailing edge of the early load pulse and the prefetch decision controlled by MROM C1 -- C5. (Again refer to Table 4.4-1) The late decision load pulse is generated only for conditional branches and when the data field requires the ALU output.

A state called LIMBO provides time for late sequencer decisions (References Figure 4.3-1.) LIMBO also provides a means for a memory cycle to stop the CPU until the cycle has been completed. This is controlled by a memory busy signal from main store. With the signal present the CPU timing will stop in LIMBO until the signal is removed and then continue. An external input also provides the test support equipment a means of stopping the CPU timing.

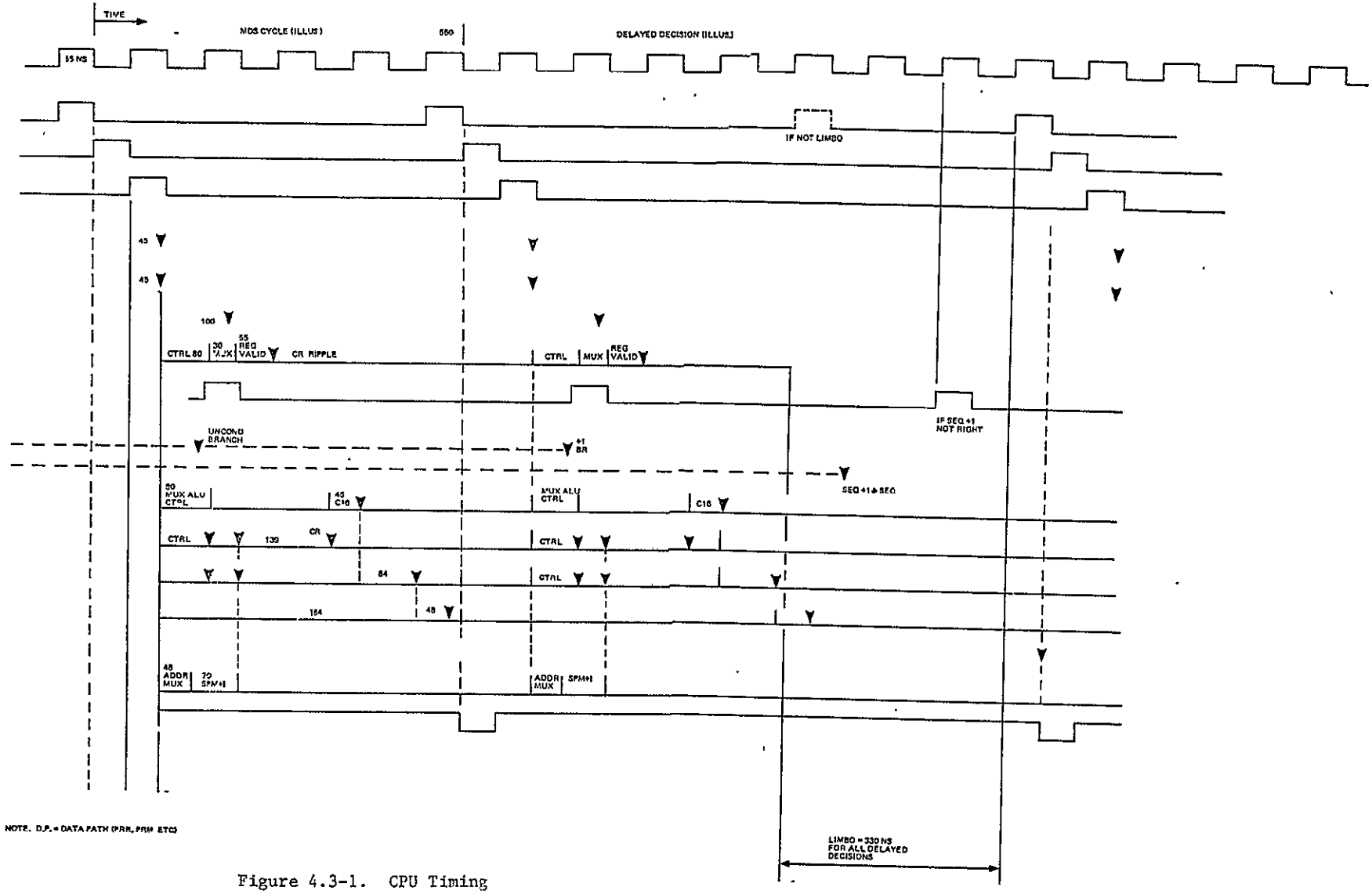


Figure 4.3-1. CPU Timing

FIGURE 4.3-1 CPU TIMING

4.3.1 SEQUENCER SUPPORT (PRM = 0)

The timing chip generates a PRM = 0 signal which indicates all PRM bits are logic zeros. This is accomplished by taking the PRM = 0 signal from each Data Path module and combining these into one signal. Each PRM = 0 signal from a Data Path module indicates that a group of four PRM bits equal zero. The timing chip provides for 8 inputs (to support a 32-bit data flow). Unused inputs for the SUMC-IIB are tied to a logic one.

4.4 CONTROL

The basic control of the SUMC-IIB is provided by a microprogram residing in a read-only memory called MROM. Sequencing logic selects the next micro-program control word (micro-code) based upon the current control word and current machine conditions. Four possibilities exist:

- Select the next sequential control store word
- Hold the present control word and use it again in the next computer cycle. (This would be accompanied by decrementing the iteration counter and breaking the "hold loop" when the counter goes to zero.)
- Transfer to the location identified by the transfer field of the current control word
- Or under certain error conditions the sequencer is forced to MROM word zero.

4.4.1 SEQUENCE CONTROL UNIT

The Sequence Control Unit (SCU) governs the flow of micro instruction execution in the machine. The SCU is composed of the following parts:

- MROM Address Register - Holds the address of the next micro instruction.
- Sequence Multiplexer - Selects the input to the MROM Address Register.
- Sequence Control Logic - Provides multiplexer control of the micro instruction address register and iteration counter multiplexer.
- Iteration Counter - Six bit counter for micro loops.
- Iteration Counter Multiplexer - Loads iteration counter from MROM bits C12 thru C17, PRM bits 10 thru 15, IR bits 10 thru 15, or the present contents of counter incremented by -1 or -4.

4.4.1.1 Sequence Control

The control chip decodes five MROM bits in order to determine what action is required by the sequencer and iteration multiplexers. Table 4.4-1 specifies the MROM control bits and their associated actions. The fifth control bit MROM C5 is used to reverse the branching conditions for sequencer only i.e. for 1010 and C5=0 the microprogram will branch if the PRM output is zero and step through if not equal to zero (as shown in the table). If, however, C5=1 the transfer will be taken for PRM outputs not equal to zero and the microprogram will increment if PRM = 0. Control action is controlled by the five MROM bits and not by clock signals.

Table 4.4-1. Sequencer and Iteration Counter Actions

MROM C1-C5	TEST CONDITION	SEQUENCER ACTION	ITERATION COUNTER	PREFETCH	COMMENTS
00000	INT = 0	+1	HOLD		INT is an 'OR' of the following conditions: (1) I/O interrupt, (2) Fixed Point Overflow Error Latch, (3) Interval Timer Interrupt Latch.
	INT = 1	MT → SEQ	HOLD	YES	
00001	INT = 0	MT → SEQ	HOLD		
	INT = 1	+1	HOLD		
00010	–	MT → SEQ	HOLD	YES	Unconditional Branch
00011	–	+1	HOLD		Unconditional +1
00100	REQ = 0	+1	HOLD		REQ = I/O data request line
	REQ = 1	MT → SEQ	HOLD	YES	
00101	REQ = 0	MT → SEQ	HOLD		
	REQ = 1	+1	HOLD		
00110	IC ⩾ 4	HOLD	-4		
	IC < 4	MT → SEQ	HOLD		
00111	IC ⩾ 4	MT → SEQ	-4	YES	
	IC < 4	HOLD	HOLD		This condition is not useful.
01000	IC ⩾ 4	+1	-4		
	IC < 4	MT → SEQ	HOLD		
01001	IC ⩾ 4	MT → SEQ	-4	YES	
	IC < 4	+1	HOLD		
01010	–	+1	IR → IC	YES	IR = Instruction Register bits 10-15.
01011	–	+1	IR → IC		
01100	IC ≠ 0	HOLD	-1		
	IC = 0	MT → SEQ	HOLD		
01101	IC ≠ 0	MT → SEQ	-1	YES	
	IC = 0	HOLD	HOLD		This condition is not useful.

Table 4.4-1. Sequencer and Iteration Counter Actions (Cont'd)

01110	IC ≠ 0	+1	-1		
	IC = 0	MT → SEQ	HOLD	YES	
01111	IC ≠ 0	MT → SEQ	-1		
	IC = 0	+1	HOLD		
10000	INT.IRR=1**	+1	HOLD		IROM bit 15 = 0 indicates RR instruction format. IROM output must be stable at the beginning of the cycle; i.e., Instruction Register Left must have been set at least two cycles earlier.
	I15.IRR=1	MT → SEQ	HOLD	YES	
	OTHERWISE	IROM → SEQ*			
10001	INT.IRR=1**	+1	HOLD		IROM output need not be stable at the beginning of the cycle; i.e., Instruction Register Left may have been loaded by the previous microword.
	I15.IRR=1	MT → SEQ	HOLD	NO	
	OTHERWISE	IROM → SEQ*	HOLD		* Substitutes IROM bits 13-14 for MROM bits M1-M2 for memory control. IROM bits 2-12 are gated into the sequence register.
					**INT.IRR=1 is dominant over I15.IRR=1
10010	–	PRM → SEQ	HOLD	NO	PRM bits 4-15 are gated into the sequence register.
10011	–	PRM → SEQ	HOLD		
10100	PRM Bit 0 = 0	MT → SEQ	HOLD		PRM positive.
	PRM Bit 0 = 1	+1	HOLD	NO	PRM negative.
10101	PRM Bit 0 = 0	+1	HOLD		
	PRM Bit 0 = 1	MT → SEQ	HOLD		
10110	PRM = 0	+1	HOLD		PRM bits 0-15 all 0.
	PRM ≠ 0	MT → SEQ	HOLD	NO	
10111	PRM = 0	MT → SEQ	HOLD		
	PRM ≠ 0	+1	HOLD		
11000	–	+1	MT → IC	NO	MROM bits C12-C17 are gated into the iteration counter.
11001	–	+1	MT → IC		
11010	–	+1	PRM → IC	YES	PRM bits 10-15 are gated into the iteration counter. This function may also be accomplished by the MROM miscellaneous field, M7-M10 = 0111.
11011	–	+1	PRM → IC		

Table 4.4-1. Sequencer and Iteration Counter Actions

11100	NOT HCINT	MT → SEQ	HOLD		HCINT = hardware interval timer interrupt latch. If this latch is set, the INT input to the sequencer is also activiated.
	HCINT	+1	HOLD	NO	
11101	NOT HCINT	+1	HOLD		
	HCINT	MT → SEQ	HOLD		
11110	ALU Overflow = 0	+1	HOLD		An ALU overflow is defined to have occurred if the carry-into-ALU bit 0 differs from the carry-out-of-ALU bit 0.
	ALU Overflow = 1	MT → SEQ	HOLD		
11111	ALU Overflow = 0	MT → SEQ	HOLD	NO	
	ALU Overflow = 1	+1	HOLD		

I = IROM Output
MT = Modified Transfer Field - The field is modified for effective address (EA) branches (C1-CY = 0000). If an EA is not specified the MT = the C7-C17 of the MROM.

The control chip determines when a prefetch can occur as specified by Table 4.4-1. When the MROM C1 thru C4 bits specify a condition in which a prefetch can occur the appropriate action for the sequencer and iteration multiplexers are taken and the SCU Register load pulse from the Timing chip loads the SCU register. This then causes the MROM to fetch the word specified by SCU register (MROM Address), thus making the MROM word available to the input of the MROM register. This completes the prefetch of MROM thus making the new word available for the next computer cycle. For conditions which require a branch decision or a wait for data (i.e., PRM), the sequencer action is plus one and the interation counter action is "hold" anticipating the final condition. The final decision is made at the end of the micro cycle but prior to the loading of any data path registers. The time for this decision occurs by stopping the clock prior to CKZ.

4.4.1.2 Sequence and Iteration "Counters"

Two of the sequence Mux/Reg chips are used for sequencing. Each chip provides a 6-bit path. The 12-bit output of the sequencer mux/reg is utilized to provide MROM addressing capability of 4,096 words. Sequence register load is controlled by a signal from the timing chip. This load signal is generated by logic which determines when a MROM fetch is to occur.

The sequence register also provides a reset capability which is utilized to start the computer at MROM Location Zero for power on reset, system reset from TSE, and error latch reset.

One sequence Mux/Reg chip is used for the iteration counter (IC) providing a 6-bit wide path. Like the sequence register the IC register load is controlled by a signal from the MISC module.

The PRM inputs to the IC mux will be force selected under the conditions specified in Table 4.4-2. These conditions override those specified in Table 4.4-1 for IC action.

Table 4.4-2. Forcing the Iteration Counter

MROM				IC Inputs
M7	M8	M9	M10	
0	1	1	1	PRM

4.4.2 READ ONLY MEMORY (ROM)

As already indicated, machine operation is controlled by a microprogram held in a ROM. The ROM is divided into two parts: Instruction Read Only Memory (IROM) and Microprogram Read Only Memory (MROM). The IROM provides instruction decode and consist of 256 16-bit ROM locations. The MROM consists of 1024 64-bit ROM locations (Expandable to 2048). The MROM field is utilized for machine control.

4.4.2.1 IROM

The IROM 256 x 16-bit words are under control of eight address lines (IR-07) from the instruction register. The IROM provides instruction decode, special memory control and the instruction length code.

The following is a description of the 16-bit word.

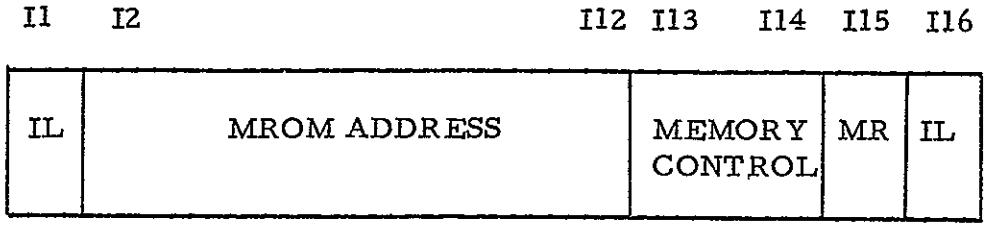


Figure 4.4-1. IROM Word Format

- MROM Address (I2-I12) points to a MROM location via SCU which contains the micro code for the instruction specified by the IR.
- MEMORY CONTROL (I13-I14) specifies memory operation, data width, and alignment when an IROM Branch occurs.
  - no memory operation
  - fullword on fullword boundary
  - halfword on halfword boundary
  - byte on byte boundary
- MEMORY REFERENCE (I15) specifies that the instruction references memory. This bit will be off for all RR instructions and it will be on for RX, RS, SI and SS instructions.
- IL bits I1 and I16 contain the instruction length code required for the PSW (see paragraph 4.7.4).

IROM outputs are open collector circuits thus providing access to the test support equipment (TSE) through a DOT-OR for IROM simulation and verification. Control of the DOT is provided by the TSE, when connected.

4.4.2.2 MROM

The MROM 1K x 64-bit words are under control of ten address lines (SEQ 3-12) from the SCU. Control of the address lines are specified in paragraph 4.4.1. For MROM definition reference "HTC Microword Word" document IBM Number 74W-00087.

The MROM outputs are open collector circuits thus providing for TSE simulation and verification as described for the IROM.

4.5 DATA PATH SUPPORT

The Function Control Unit (FCU) provides the control signals to the ALU. In addition to the standard ALU functions, the FCU supplies the necessary control to perform Multiply, Divide and Square Root steps. When MDS (Multiply/Divide/Square Root) is specified in the micro word (MROM), the FCU controls the ALU operation to carry out a step in the routine. Therefore, the micro program only needs to loop on this operation until the multiply (for example) is complete.

The EALU (Extended ALU) provides a two-bit extension of the standard ALU to support MDS and shift functions.

4.5.1 FCU

The FCU chip provides control for the ALU control, MDS and non MDS functions.

Control for the ALU is provided by the output signals from the FCU specified in Table 4.5-1.

Table 4.5-1. ALU Control

ALU Functions	CONT	SUB 2	SCAR	SUB 1	C16
AND	0	1	1	1	0
OR	1	0	1	0	0
XOR	0	0	1	0	0
ADD	0	0	0	0	0
SUB (A-B)	0	0	0	1	1
REV. SUB (B-A)	0	1	0	0	1

4.5.1.1 CONT, SUB2, SCAR

CONT, SUB2 and SCAR ALU control signals are generated from MROM A8, A9 and A10 as specified in Table 4.5-2.

Table 4.5-2. FCU Operation

MROM A8	MROM A9	MROM A10	FCU OUTPUTS CONT	FCU OUTPUTS SUB2	FCU OUTPUTS SCAR
0	0	0	0	1	1
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	0

4.5.1.2 SUB1

SUB1 control is a function of the MDS and non MDS function. The following equation identifies the generation of SUB1.

$$SUB1^{*} = \overline{A6} \cdot A7 \cdot A9 \cdot A10 \cdot MQR14 + \left[(A6 + \overline{A7}) \cdot (A9 \cdot A10) \cdot \overline{ALUOS \oplus SOS} + \overline{A8} \cdot \overline{A10} \right]$$

*A = MROMA

SOS signal is SPMO stored, with the store operation controlled by trailing edge of CKZ and MQRL equal one.

ALUOS is ALUO stored, with the store operation controlled by trailing edge of CKZ and MROMR10 equal one.

4.5.1.3 C16

C16 signal is the carry into ALU15. For ALL operations other than MDS, C16 is a 1 if MROM A12 = 1 and there was a carry (C) on the previous micro cycle. For MDS operations, the MROM A12 bit must be zero to allow proper generation of C16 as shown by the following equation.

$$C16^* = \left[(\overline{A10} \cdot A11) + (A11 \cdot \overline{A9}) + (CS \cdot A12) + (X7 + X6) \cdot (\overline{SUB1L} + \overline{A12})\right] \cdot \left[(X7 + X6 + \overline{SUB1L})\right]$$

*A = MROMA

CS signal is C, from ALU, stored, with the store operation controlled by the trailing edge of CKZ.

X6 and X7 signals are generated by ALU extension for ALU bits 16 and 17 with (X6 + X7) equals to the carry out of the extended ALU.

4.5.1.4 SUB1L

SUB1L is a function of SUB1 as shown by the following equation. It provides the control for ALU extension for bits 16 and 17. Table 4.5-3 identifies the functions for the ALU extension.

$$SUB1L^* = SUB1 \cdot A9 \cdot A10$$

*A = MROMA

Table 4.5-3. EALU Control

ALU Extension	SUB1L
Add	0
Sub (A-B)	1

4.5.1.5 C(N+4)

C(N+4) signal is the carry into ALU 17. For ALU operations other than MDS, C(N+4) equals zero. For MDS operations (CN+4) is generated as shown in the following equation.

$$C(N+4)^* = A9 \cdot A10 \cdot (SUB1 + SQRD) \cdot (\overline{SQRD} + MAR2)$$

*A = MROMA

4.5.1.6 SQRD

SQRD signal is the input to the ALU extension B mux. The following equation identifies the function. SQRD equals zero for all operations except square root sign.

$$SQRD^* = \overline{A6} \cdot \overline{A7} \cdot A9 \cdot A10$$

*A = MROMA

4.5.1.7 F25A

F25A signal is the second input to the ALU extension B mux. Where F25A equals a one when the sign of PRR equals the stored scratch pad memory sign for square root sign and equals zero for all other functions. The following identifies F25A generation.

$$F25A = SQRD \cdot \overline{SUB1}$$

$$F25A = SQRD \cdot \overline{SUB1}$$

4.5.1.8 SEL3N/SEL4N

SEL3N/SEL4N signals are utilized in mux B input selection for the ALU B inputs. These signals along with MROMA5 provide the following selections for mux B.

Table 4.5-4. ALU B-Mux Control

MROM			
A5	SEL3N	SEL4N	ALU MUX B SELECTION
0	0	0	SDR + MROM (C7-C17) (Logical OR)
0	1	1	SDR
0	0	1	SDR + SPM (Logical OR)
0	1	0	SDR + 1/2 SPM (Logical OR)
1	0	0	MROM (C7 - C17)
1	1	1	Zero
1	0	1	SPM
1	1	0	1/2 SPM

The following equation identifies SEL3N and SEL4N generation.

$$SEL3N^* = (\overline{A11} + \overline{A6}\ A7 + \overline{A9} + \overline{A10})\ (A9 \cdot A10 + \overline{A6})$$

$$\left[\overline{(MQR14 \oplus MQR16)} + (MQR15 \oplus MQR16) + A6 + \overline{A7} + \overline{A9} + \overline{A10}\right]$$

$$SEL4N^* = \left[\overline{A6} \cdot A7 + A11 + \overline{A9} + \overline{A10}\right]\ \left[A9 \cdot A10 + \overline{A7}\right]$$

$$\left[\overline{(MQR15 \oplus MQR16)} + (A6 + \overline{A7} + \overline{A9} + \overline{A10})\right]$$

*A = MROMA

MQR16 signal is generated by storing MQR14 at the trailing edge of CKZ when MQRL equals a one.

4.5.1.9 SPMSG

SPMSG is a signal generated to provide the MSB position of the B Mux when selecting 1/2 SPM. SPMSG equals zero for special MDS and equals SPMS for all other functions. The following equation identifies the generation of SPMSG.

$$SPMSG^* = (SPM0 \cdot A8) + SPM0\ (\overline{A9} + \overline{A10})$$

*A = MROMA

4.5.1.10 Quotient Generation

Q is current bit of the quotient generated during a divide step. The following equation identifies the generation of Q.

$$Q = \left[(A7 + SUB1) + SQRD\right] \quad \left[(\overline{ALU0} + \overline{SQRD})\right]$$

where A7 = MROMA7, and + means exclusive or

4.5.1.11 ALU18

ALU18 is generated for use with Square Root function as defined by the following equation

$$ALU18 = MAR2 + SQRD$$

4.5.1.12 MQRL

MQRL Load Enable is a signal generated which controls the loading of the MQR. The following equation identifies the generation of MQRL.

$$MSRL = \overline{MROMR11 \cdot MROMR12 \cdot MROMR13}$$

4.5.2 Mux/ALU Extension (EALU)

The Mux/ALU extension provides for ALU bits 16 and 17 which are used in all MDS and some shift operations involving the PRM and MAR (see Section 4.1).

The A input to the ALU extension is controlled by MROMA4 in the following manner.

MROMA4	ALU INPUT 16	17
0	PRR0	PRR1
1	MAR0	MAR1

The B inputs for the ALU extension are controlled as shown in Table 4.5-5.

Table 4.5-5. EALU B Input Control

SEL3N	SEL4N	ALU B INPUT 16	17	5-15
1	0	SPM15	F25A*	1/2 SPM
1	1	0	0	All zero's
0	1	F25A*	SQRD*	SPM
0	0	0	0	I/O

*For the SUMC-IIB these ALU inputs wired to ground.

EALU control is exercised by the SUBIL signal, from the FCU, as indicated below (see Paragraph 4.5.1.4).

EALU Operation	Control (SUBlL)
Add	0
Sub (A-B)	1

Carry into ALU 17 is controlled by C(N+4) signal, from the FCU, (see paragraph 4.5.1.5).

4.6 SPM ADDRESS MUX

The SPM Address Mux provides for multiplexing the four low order bits of the SPM address. The four multiplexed bits plus MROM S1 and S2 bits are used to address the 64 word SPM. Multiplexer control is accomplished by the SPM address control bits, MROM S7 thru S9.

The timing chip derives the computer timing from one of the following sources: oscillator or external oscillator. The basic oscillator is physically packaged as part of the power supply. The external oscillator and selection for which oscillator is to be used is supplied from the test support equipment (when connected) otherwise the internal oscillator is selected.

4.6.1 SPM ADDRESS MUX

The three control bits, MROM S7-S9, selects the source for the least significant four bits of the SPM address. Many of these address inputs are intended to support S/360 general register addressing requirements. (see Table 4.6-1.)

Table 4.6-1. SPM Address Source Control

	SPM ADDRESS BITS			
S7-S9	3	4	5	6
000	0	0	0	0
001	IR8	IR9	IR10	S6
010	S3	S4	S5	S6
011	MQR12	MQR13	MQR14	MQR15
100	IR8	IR9	IR10	IR11
101	S5	S6	IR16	IR17
110	IR12	IR13	IR14	IR15
111	IR16	IR17	IR18	IR19

SPM Address MSB's 0 and 1 are controlled directly by MROM S1 and S2 respectively.

4.7 ARCHITECTURE SUPPORT FUNCTIONS

The Architecture (Arch) module implements functions which are sensitive to the machine architecture being emulated. This module is a two chip carrier which contains Arch and Register chips for S/360 emulation. Figure 4.7-1 identifies Arch module functions. The architecturally dependent portion of the SUMC-IIB is comprised of:

- EA Branch - For effective address generation, the three least significant bits of MROM address are modified according to instruction format (e.g. RR, RS, SI) and the Base and Index operations required for the EA. This provides a multiway branch to the proper EA generation microprogram routine.
- Condition Code (CC) and PSW Logic - Partial PSW function and condition code setting is provided by this logic.
- Mask Gates - Overflow logic and PSW masks.

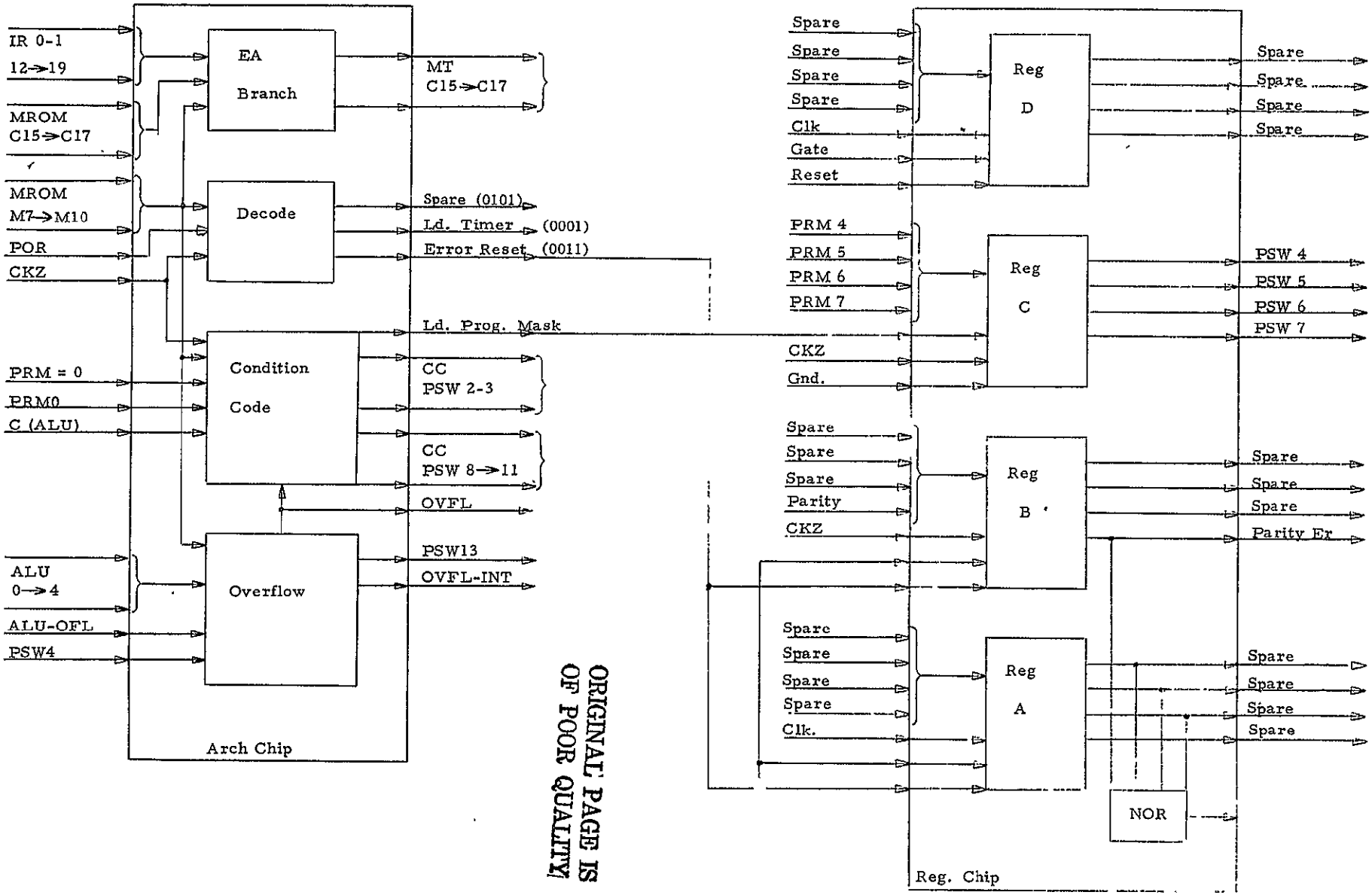
4.7.1 EFFECTIVE ADDRESS (EA) BRANCH

MROM M7, M8, M9, and M10 bits are decoded to determine when an EA Branch (0100) is to be generated. EA Branch generation is accomplished by modifying the three Transfer Field LSBs (MROM C15, C16, and C17). The LSBs are modified as specified in Table 4.7-1 when MROM M7, M8, M9, and M10 equals 0100 respectively. For all other conditions of MROM M7, M8, M9, and M10 the EA Branch equals the MROM Transfer Field MROM C6-C17.

Table 4.7-1 S/360 EA Branch Conditions

Branch Conditions (MROM M7-M10 Equal 0100)	Modified
RR	110
RX, RS, SI, and NO BASE and NO INDEX	000
RX, RS, SI and BASE and NO INDEX	001
RX, and NO BASE and INDEX	010
RX and BASE and INDEX	011
SS and NO BASE	100
SS and BASE	101

Figure 4.7-2 functionally describes the derivation of EA Branch.



ORIGINAL PAGE IS
OF POOR QUALITY

Figure 4.7-1
ARCH Module

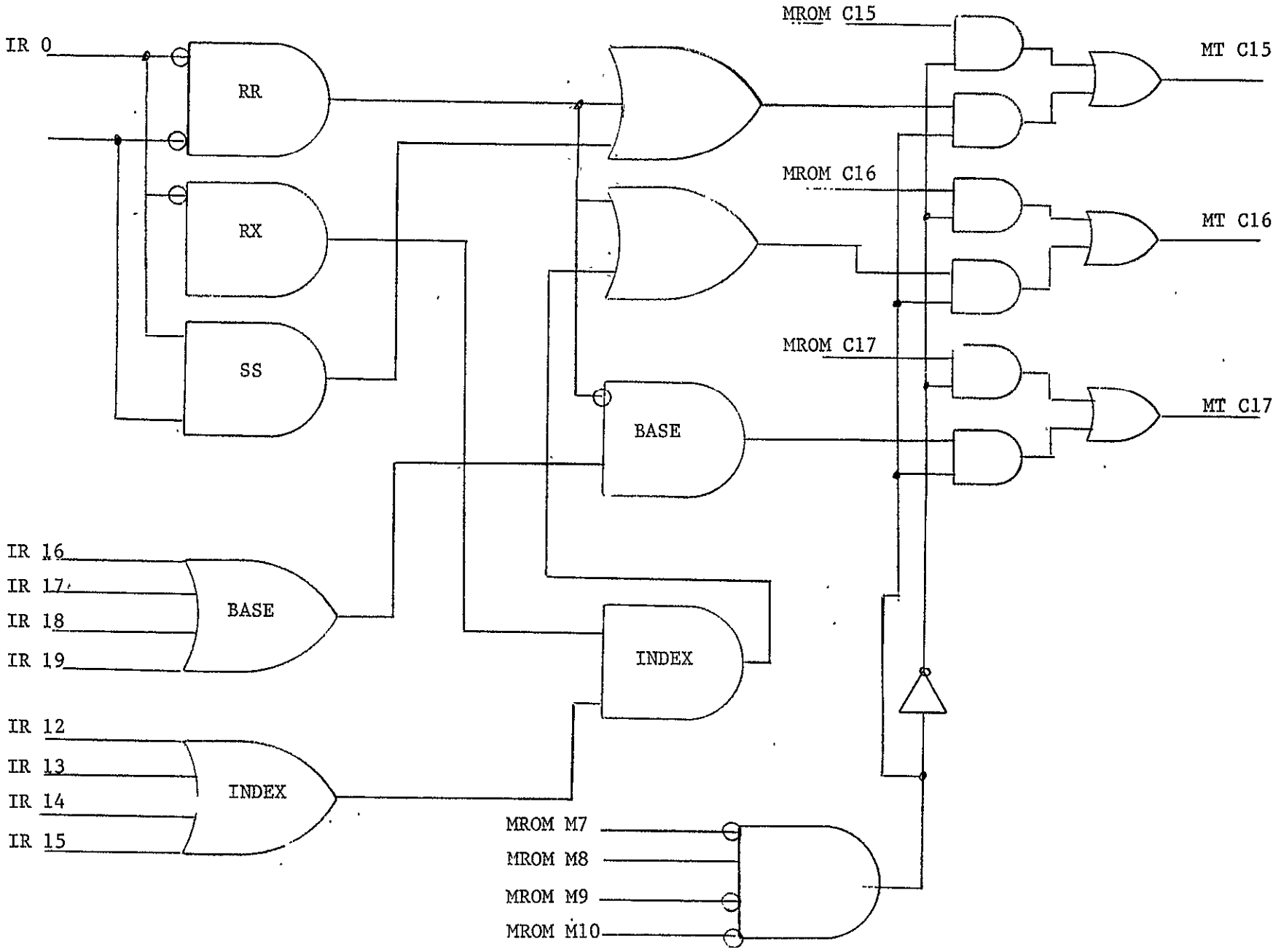


Figure 4.7-2. EA Branch Logic

4.7.2 Condition Code

To support a S/360 compatible condition code, the conditions must be capable of being set as described in Table 4.7-3. The selection of a particular set conditions is determined by MROM M7, M8, M9 and M10. For each condition only one of the four bits shall be set at any one time. The setting of any one of the four bits is controlled by a priority selection of the condition code bits. Priority level are bit 3, 0, 1, and 2 with bit 3 having the highest priority and bit 2 the lowest. Example; if PRM equals a plus and an overflow condition from the ALU occurs for MROM M7-M10 code only the overflow (bit 3) will be set.

The condition code is maintained in the Program Status Word (PSW) described in Paragraph 4.7.4 in both a four-bit code and a two-bit code. The two-bit code is specified by Table 4.7-3 and the four-bit code is derived from the two-bit code as specified in Table 4.7-2.

Table 4.7-2. 2-Bit vs 4-Bit Condition Code

2-Bit Code		4-Bit Code			
0	1	0	1	2	3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

The PRM=0 is generated by the Timing chip and made available for use at the Arch Module.

PRM = minus and PRM = plus are determined by testing the PRM 0 bit; PRM 0 = 1 is minus indication and PRM 0 = 0 is plus indication. "Carry" is generated by the ALU and made available to the Arch module.

The previous condition code (CC) setting is derived from the value of the two-bit code contained in the PSW.

The overflow condition (OVFLO) is the ALU overflow from ALU ORed with a PRM overflow. The PRM overflow is generated for one and four left arithmetic shifts. For a left one arithmetic shift PRM overflow is generated if ALU 0 and 1 bits disagree. For a left four arithmetic shift PRM overflow is generated if ALU 0 disagrees with any ALU 1, 2, 3, or 4 bits. Figure 4.7-3 functionally describes the PRM OVFLO functions. The OR'ed condition of the ALU and PRM OVFLO signals is made available for use with the SCU (Reference paragraph 4.4.1).

Table 4.7-3. Condition Code Generation

MROM				S/360 Condition Code			
M7	M8	M9	M10	0	1	2	3
1	1	0	0	PRM=0	PRM=-	PRM=+	
1	1	0	1	PRM=0	PRM=-	PRM=+	ALU OVFLO
1	1	1	0	PRM=0, NC	PRM≠0, NC	PRM=0, C	PRM≠0, C ALU
1	0	0	1	PRM=0, CC=0	PRM=-	PRM=+or CC≠0	ALU OVFLO or CC=3
1	1	1	1	PRM=0, CC=0	PRM=-	PRM=+or CC≠0	OVFLO or CC=3 (PRM Shift Left 1 Arith.) + ALU
1	0	1	1	PRM=0, CC=0	PRM=-	PRM=+or CC≠0	OVFLO or CC=3 (PRM Shift Left 4 Arith.) + ALU
1	0	0	0	PRM=0, CC=0	PRM=-	PRM=+or CC≠0	
1	0	1	0	PRM=0, CC=0, NC	(PRM≠0 or CC≠0), NC	PRM=0, CC=0, C	(PRM≠0 or CC≠0), C
0	0	1	0	Load the condition code register from ALU 2, 3			

OVFLO - ALU overflow condition (arithmetic operations) or PRM overflow (Shift operations)

C - Carry out of ALU 1

NC - No carry out of ALU 1

CC - Previous condition code setting of 2 bit form

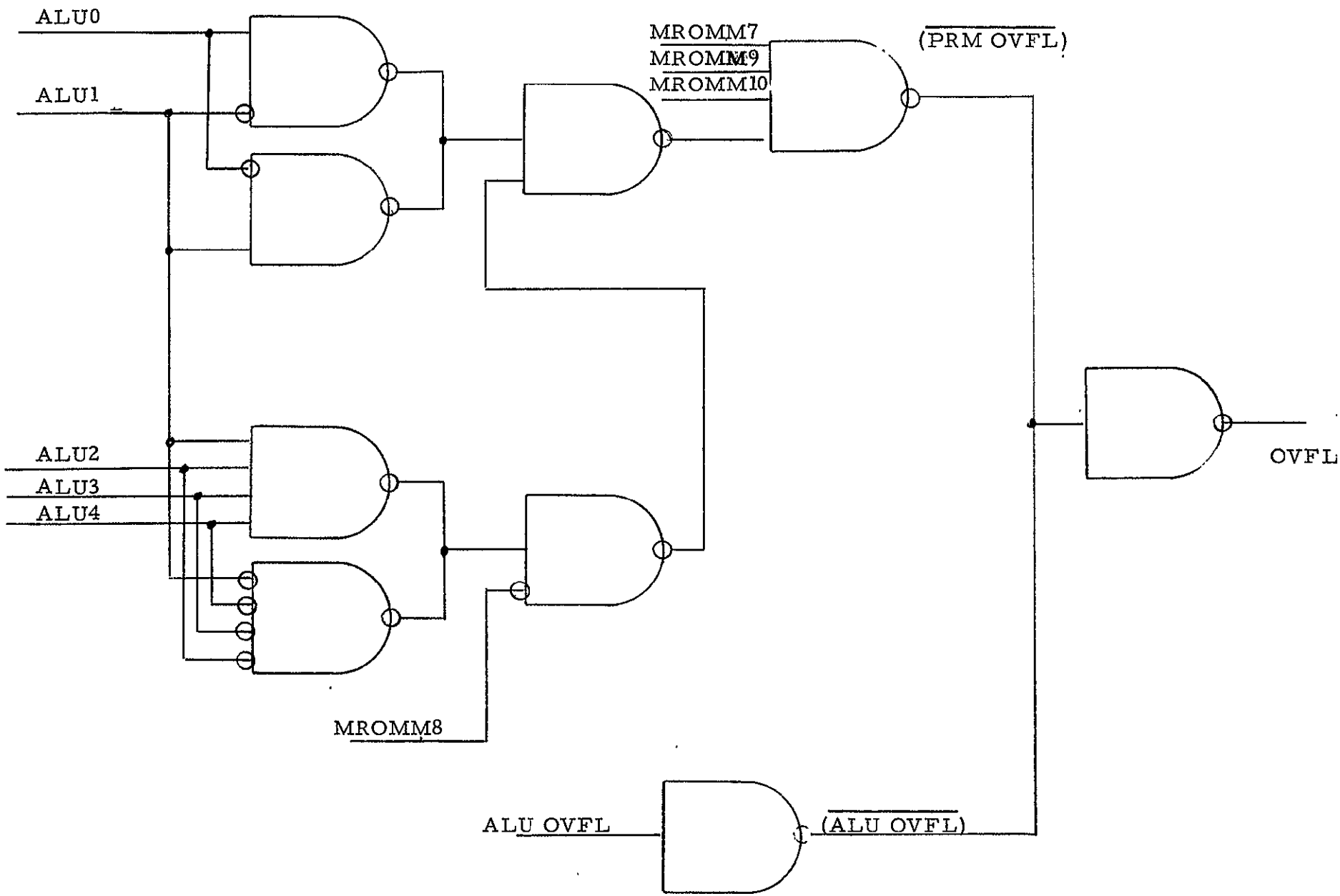


Figure 4.7-3. Overflow Generation

The Data Path Condition Code input provides a means for external input to the condition code logic. This input is in the form of the two bit condition code with the input from ALU 2 and 3. This two-bit code is then utilized to provide the associated 4-bit code.

4.7.3 Exception Monitoring

S/360 architecture entails a significant amount of exceptions monitoring as described in S/360 Principles of Operation, pages 156 thru 161.

Five exceptions are monitored by the SUMC-IIB hardware: (1) Addressing, (2) Memory Specification, (3) Storage Protect, (4) Parity, (5) Overflow. Each of these exceptions will set its respective error latch. The setting of the Addressing, Specification, Storage Protect or Parity error latch will prevent subsequent writing into main memory or SPM and will cause a reset signal to be generated that forces the timing chip to initialize the machine to MROM location zero. This initialization and write inhibit is accomplished by a Write Valid signal. The write valid signal equals a logic one when the above error latches are reset. The setting of any of the above error latches will cause the write valid signal to equal a logic zero thus inhibiting writing into both SPM and Main Memory. This change of state also fires a single-shot which causes the timing chip to initialize by forcing a reset to the sequence register (MROM Address) located in the SCU. The WRite Valid at a logic zero also prevents any further change of state in the above four error latches.

The setting of the overflow latch generates an interrupt signal which can be tested by the SCU utilizing MROM bits C1-C4 (Reference Section 4.3). This interrupt signal shall be negative.

The error latches are reset by clock CKZ ANDed with the decode of MROM bits M7-M10 = 0011.

4.7.3.1 Addressing Exceptions

Addressing exceptions are monitored by Storage Interface Logic (SIL). A CPU address exception latch is provided to the CPU from SIL. Setting of the latch is controlled by an error clock signal from SIL.

An addressing exception occurs when memory is addressed with an address greater than implemented memory.

4.7.3.2 Memory Specification

Memory specification exceptions are monitored by SIL. When a half-word or full-word memory operation is requested the memory address is checked to verify that the LSB or two LSBs are zero, respectively.

The CPU memory specification latch is contained on the SIL Module. Setting of the latch is controlled by an error clock signal from SIL.

4.7.3.3 Storage Protect

Storage protect errors are monitored by Storage Interface Logic (SIL). A CPU storage protect error latch is provided by SIL and is set under the control of the error clock signal from SIL.

Storage protect errors occur when a store operation is requested in a memory/ location within a block whose store protect bit is ON.

4.7.3.4 Parity

Parity errors are monitored by Storage Interface Logic (SIL). A CPU parity error signal is provided to Arch for input to the parity error latch. This latch, contained on the register chip, is set by the trailing edge of clock CKZ.

4.7.3.5 Overflow

Overflow exception occurs when MROM bits M7-M10 request the condition code to be set by an overflow (see Table 4.7-3), then an overflow exception is generated if the overflow program mask is set to a logic one (PSW bit 4; see Paragraph 4.7.4). The overflow exception will set the overflow error latch. This latch contained on the Arch chip is set by the trailing edge of clock CKZ.

4.7.4 Program Status Word (PSW)

The SUMC-IIB PSW is maintained partly in hardware and partly in a SPM location used only by the microprogram. Whenever the PSW is to be "manipulated", a partial PSW (which is 16 bits long) is read into the data path through the MQM so that the 64-bit PSW can be assembled (see paragraph 2.2.3). The format of this partial PSW is shown in Figure 4.7-4.

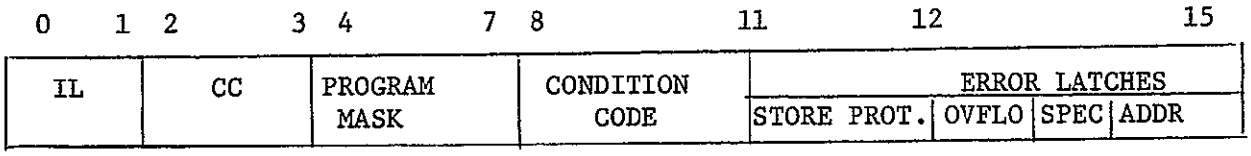


Figure 4.7-4. Partial PSW Format

The two-bit instruction length code (IL) comes from IROM bits 1 and 16 (for PSW 0 and 1 respectively). These two bits are loaded into a register at the fall of CKZ if the MROM R14, R15, R16 bits are not 000 and an IROM Branch is taken by the SCU (see Table 4.4-1).

For convenience to the microprogram, the condition code is maintained in both a two-bit and four-bit form. The two-bit form matches the needs of the PSW format and the decoded form matches the branch on condition instruction's condition mask format (see the HTC Principles of Operation, IBM No. 74W-00026).

The condition code and program mask registers are loaded from the PRM at the fall of CKZ when MROM M7-M10 = 0110.

The four errors (address, specification, storage protection, and overflow) were described in paragraph 4.7.3. Each of these conditions has a latch which is set when the corresponding error is set and a program or microprogram branch is taken. After the PSW has been read into the data flow the error latches are reset by microprogram action (MROM M7-M10 = 0011).

A complete listing of the actions taken by the M7-M10 field is given in Table 4.7-4.

4.8 TIMER

The hardware timer supports the microprogram maintenance of a real-time clock an an interval timer. Both are accessed by the programmer via the TMRS instruction

The interval timer appears to the programmer as a 16-bit decrementing counter which is decremented every 112.64 microseconds. It has a maximum interval of 7.38 seconds. The real-time clock appears as a 32-bit incrementing counter which is incremented every 112.64 microseconds. It has a maximum value of 5 days, 14 hours, 23 minutes and 5 seconds. The TMRS instruction is used to read either of the timers into a general register. When the TMRS instruction is used to load either of the timers from main storage, the old value of the timer is placed into a general register so that the timer may be read and loaded without an intervening step. Problem programs may read either of the timers directly, but only the supervisor is permitted to load the timers. Duration of the timer can be extended by programming. When the interval timer goes to zero an external interrupt (timer) is generated. When the real-time clock overflows, however, no action is taken by either the hardware or microprogram. It merely resets to zero and continues counting.

4.8.1 Supporting Hardware

The SUMC-IIB timer hardware consists of two Timer chips of 8 bits each, which are wired as a 16-bit ripple counter. The 6 MSB's are loadable from the MAR bits 10 thru 15, and an eight-bit gated output is made available to the data path by the PRM. The clock source is provided by the basic machine oscillator which has a 9.09 MHZ frequency.

The timer hardware sets an interrupt request latch when the counter overflows from all 1's to 0's and CKZ occurs. This interrupt provides an input to the SCU (Reference paragraph 4.4.1).

Table 4.7-4. Miscellaneous Field Bits M7-M10

MROM				
M7	M8	M9	M10	Operation
0	0	0	0	No Operation
0	0	0	1	Load Hardware Timer
0	0	1	0	Load Condition Code Register from ALU 2 and 3
0	1	0	1	Spare
0	1	0	0	EA Branch Condition (Ref Para. 4.7.1)
0	1	1	0	PSW load: Condition Code, Program Mask (Ref. Para. 4.7.4)
0	1	1	1	IC Load from PRM (Ref. Para. 4.4.1.2)
0	0	1	1	Reset PSW Error Latches (Ref. Para. 4.7.3)
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	Condition Code (see Table 4.7-3)
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

The eight MSBs of the timer are read into the data path (PRM bits 8-15) by a decode of MROM bits R14-R16 = 101. The six MSB's of the timer are loaded from PRM 8-13 when MROM bits M7-M10 = 0001 (gated at CKZ). Since the timer load is not synchronized with the timer counting, each load is preceded and followed by a read to see if a "tick" occurred in the least significant bits. If so one is added to the load value and a new load is generated. The sharing of the function between microprogram and hardware is invisible to the programmers.

4.9 PROGRAM COUNTER

The SUMC-IIB has a 16-bit counter which operates under microprogram control to point to the next instruction to be executed. The counter uses two timer chips in a timer module and is controlled by the MROM R14-R16 field as indicated in Table 4.9-1.

Table 4.9-1. Program Counter Control

MROM R14	R15	R16	Action
0	0	0	NULL (No-Op)
0	0	1	Load Program Counter from PRM
0	1	0	Increment Program Counter (+2 Bytes)
0	1	1	Load IR Left
1	0	0	Load IR Right
1	0	1	Read Hardware Timer
1	1	0	Special P. C. Increment *
1	1	1	Unused

* The special program counter code increments the PC by the same as code 010, but it also controls the ALU B MUX to select zeros if I REG 16-19 = 0000 (no base reg); otherwise it selects SPM as indicated by the MROM A5-A7 field.

For branch and link type instructions it is necessary to be able to read the program counter into the data path and store it. The SUMC-IIB uses an AOTC module which is an 18-bit, two way multiplexer to select either the program counter or the I/O register as an input to the MAM. The multiplexer selects the PC if MROM M5, M6 = 00; otherwise it selects the I/O register.

4.10 MEMORY SUBSYSTEM DESCRIPTION

The memory subsystem consists of storage interface logic on the CPU slice and one to four independent sections of storage, herein called storage pages. See Figure 4.10-1. Each section of storage contains 16,384 bytes of memory and the necessary support circuitry to make a functioning memory system for the SUMC-IIB. The storage interface logic provides an interface for direct memory access (DMA) and for CPU access. The CPU interface provides full-word, half-word, and byte operations, but the DMA uses only half-words.

The full-word operations are provided in microcode by two half-word operations. the logic is designed for expansion to accommodate a 32-bit data flow interfacing with the standard 32-bit memory. The storage page is designed to interface as a technology independent memory element and two technologies have been implemented. Wiring is provided for a 32-bit data interface which can be controlled by the storage interface logic to be either 16 or 32-bits wide. (In the SUMC-IIB it operates as 16-bits). No changes to the storage page are required for 32-bit operation. The storage logic on the CPU slice generates and checks parity for all storage operations and implements a storage protection feature which protects blocks of 512 half-words against incorrect storage operations. Separate protection is provided for CPU and direct memory access.

For CPU operation the memory mode is controlled by the miscellaneous field of the microprogram control word (bits M1 thru M4). Bits M1 and M2 specify: no operation, byte operation, half-word operation, or full-word operation. Bits M3 and M4 specify: read, read-and-hold, write, and write-move. Details of the memory operations are shown in Tables 4.10-1 and 4.10-2. In addition to the memory modes which cause the memory to fetch or store, there are two memory "operations" which do not actually cycle the memory. The no-op code (M1 and M2 both equal to zero) is used in combination with bits M3 and M4 to specify these non-memory cycling operations. One operation results in the storage data register (SDR) being loaded from the PRR without a memory operation taking place. The other results in the storage protect register being loaded from the PRR.

4.10.1 Memory Operation

CPU initiated memory operations are controlled by the micro control word whose contents are sampled at clock CKZ time of each CPU cycle. Any indicated memory operation will take place during the subsequent micro cycle. When the memory starts to operate, a memory busy signal is generated to indicate that the memory is in use. During a CPU initiated memory cycle, when the CPU is ready for a CKZ timing pulse AND memory busy is active, the computer will go into a state called "limbo" and wait for memory busy to fall. Thus the microcycle is stretched to accommodate the memory operation. During CKZ time M1 thru M4 of the microcode word are examined to determine whether a CPU memory operation will take place in the subsequent microcycle or not. If a memory operation is to take place, it will be initiated at the fall of clock CKZ.

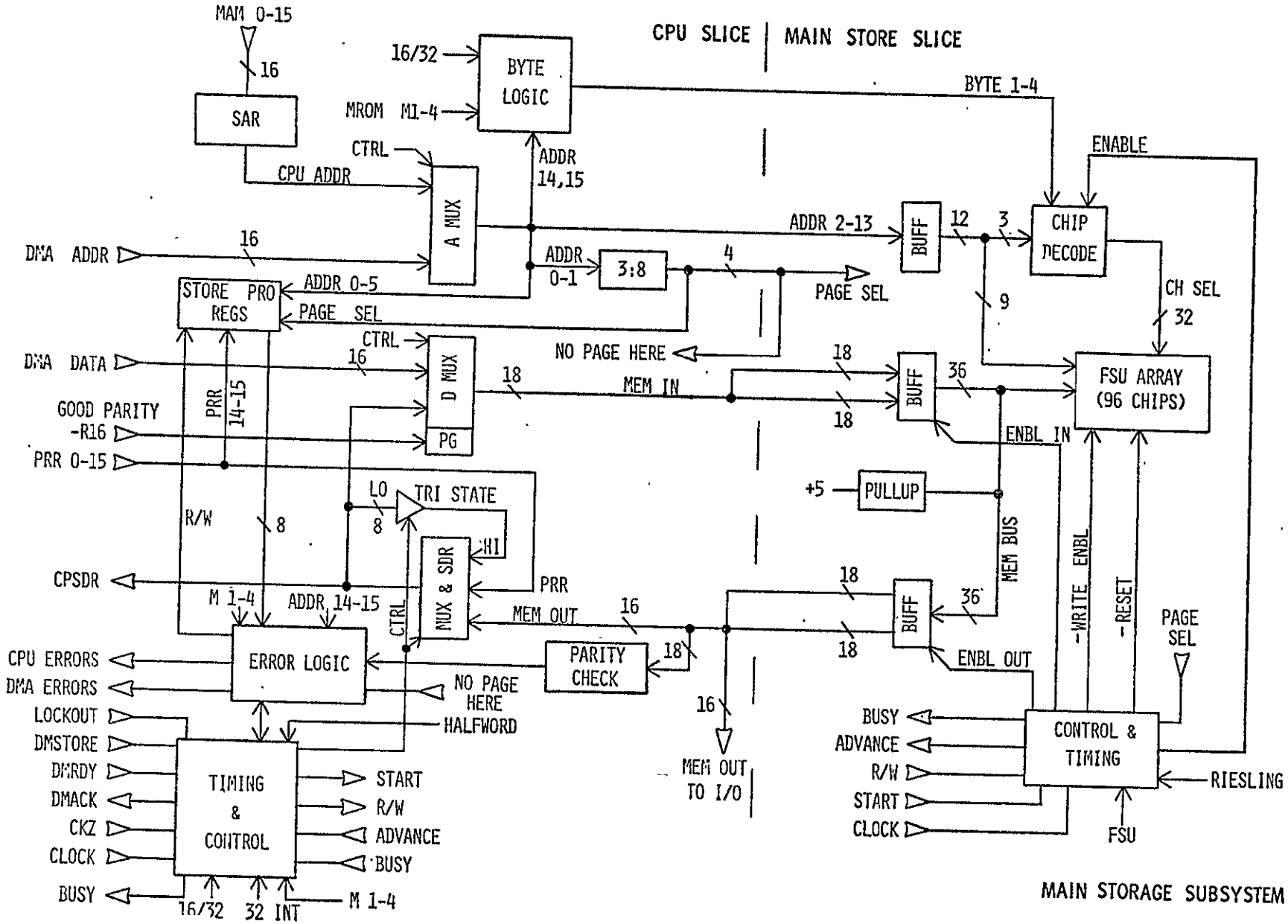


Figure 4.10-1. Main Storage Subsystem

Table 4.10-1. Memory Operation and Data Size Control

M1-2	Operation	Description
0 0	No Operation	This code in conjunction with control bits M3 and 4 specify a true NO-OP and two other operations not causing either a read or write. M3-4 = 00 is a true NO-OP. M3-4 = 01 NO-OP M3-4 = 10 causes the SDR to be loaded from the PRR but no memory write is initiated. This facilitates loading the IR from the data path. M3-4 = 11 causes the storage protect mask register to be loaded from the PRR. MAM 0 - 5 specify the sector to be protected (for up to 64K-bytes of memory).
0 1	Full-word	The memory operation initiated at "Z" time is for 32-bit information and the address shall be on a full-word boundary (MAM 14 and 15 = 0). On the 16-bit machine this code will only be used for the lower (odd) half-word and the storage interface logic (SIL) must force a 1 in address position 14.
1 0	Half-word	The memory operation is for half-word (16-bit) information and shall be on a half-word boundary (MAR 15 = 0). For a 32-bit machine this data shall be aligned properly for a half-word either in the SDR (read) or the PRR (write). If the machine is fractional the half-word will be left justified (bits 0 - 15); if integer the bits will be right justified (bits 16 - 31).
1 1	Byte	The memory operation initiated is for an eight bit operand and will be right justified in either the PRR or SDR.

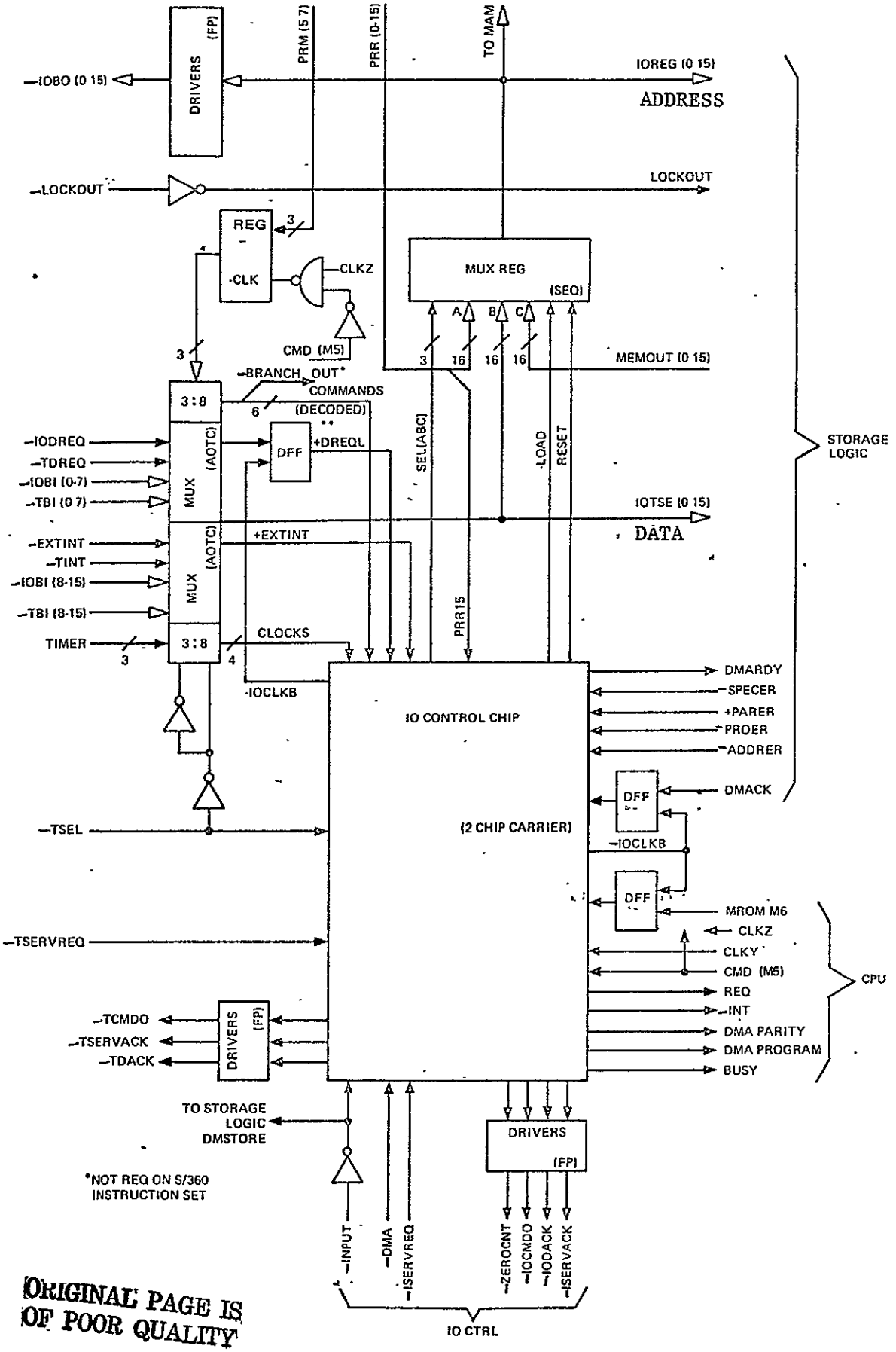
Table 4.10-2 Memory Read/Write Control

M3-4	Operation	Description
0 0	Read	Initiates a memory read operation at "CKZ" time using the address at the MAM output. Information will be in accordance with M1 and M2, and will be loaded into the SDR by the SIL at the subsequent CKZ clock even if CPU timing has to be suspended. The SIL will generate a busy signal starting at CKZ time and ending with the following CKZ time. If a destruction readout (DRO) memory is used, a restore is implied as a part of read.
0 1	Read-and-Hold	Identical to read except that the memory operation is not terminated at the subsequent CKZ time, and no restore is performed in a DRO memory. This is the first part of a read/modify data/write. Memory operation is terminated after the subsequent write operation.
1 0	Write	This is either a stand alone write operation or the conclusion of a read/modify data/write. The SAR is loaded from the output of the MAM and the SDR is loaded from the PRR (at CKZ time). If the write is the completion of a read/modify data/write, the SAR will not be loaded. The CPU need not be held up waiting for completion of the write unless there is another memory operation in the following program word. Information in the PRR is in accordance with M1, M2, and alignment might be required of the SIL before storing. If a write operation is indicated in the MROM word following a read the contents of the SDR will be ambiguous for both the read and write operations.
1 1	Write -Move	Identical to the write operation above except that the SDR is not loaded. This can follow a read to provide a move.

On both read and write operations, the contents of the memory address multiplexer are loaded into the CPSAR. If the operation is a store, the contents of the PRR will also be loaded into the CPSDR, however, the M1 thru M4 bits and least significant two address bits must be examined to determine what multiplexing is required. If a byte operation is being performed, the data in the PRR is right justified and must be moved to the appropriate location according to the byte address bit. Address specification is checked by the control logic in the storage interface logic module for all read or write operations except byte. If a half-word operation is indicated the least significant address bit must be a zero. For the 16-bit machine a special meaning is given to the code corresponding to the full-word operation, however, as in S/360 the address for a full-word memory operation must have zeros for the two least significant bits of address. In the 16-bit machine the full-word memory operation code is used to read or store the least significant half-word of a full-word operand. The storage interface logic will force a 1 in address bit 14 to cause the proper half-word operation to take place. In the case of a store, the data is made available to the storage page on a 32-bit bus from the CPU to the storage page. In the case of the 16-bit machine the 32 bits are made up by a replication of the 16 bits from the CPSDR.

Address and specification errors will inhibit the memory cycle by suppressing the "byte" signals going to the storage page. Storage protection errors are detected too late to completely stop the storage page operation so a write is converted to a read. Contents of the SDR are undetermined at the end of a storage protect violation. Storage protection is checked on a read-and-hold operation as well as a store operation since a read-and-hold must be followed by a write.

The sequence of events leading to a DMA memory operation is significantly different from that of a CPU operation. However, once the actual operation starts they are identical. In the SUMC-IIB direct memory access can either be handled via the I/O Channel (see Figure 4.10-2) or by a separate DMA Interface. A sophisticated I/O unit such as a multiplexer or selector channel would operate on the separate DMA interface. The separate interface is considered standard. The DMA feature cannot be supported both as a stand-alone interface and integrated into the I/O channel. The IO/DMA interface shares a single input bus for address and data. The address is sent first to indicate to the computer that a memory operation is desired and is stored in the IO register. A store operation will be identified by a true state on the store line. After the address has been stored, data will be provided (if a store operation is desired). After both address and data are ready DMA service is requested of the SIL and the next memory cycle will be given to the DMA. In case of overlapping requests for service from the CPU and direct memory interface, service will be alternated. The direct memory interface can lock the CPU out of memory by use of a signal called "lockout". The lockout signal must be true prior to initiation of the memory operation (rise of the address valid signal). When lockout is used the CPU will not be given memory service until the lockout line is dropped. Lockout does not become effective until service is requested by and granted to the DMA. Only read and store half-word operations can be performed through the direct memory interface. Once, however, a memory cycle is initiated for direct memory access, it will be performed in identically the same manner as a CPU request would be except address specification check is performed for a correct half-word address, i.e. address bit 15 = 0.



ORIGINAL PAGE IS OF POOR QUALITY

Figure 4.10-2. SUMC-IIB I/O Block Diagram (with Integrated DMA)

4.10.2 Memory Data Flow

The basic flow of data in the storage subsystem can be seen in the block diagram of Figure 4.10-1. For the CPU the source of data is the PRR, and the source of the address is the MAM. For the IO/DMA interface option the DMA address and data inputs are by the IO channel as described in the IO Section 4.11. For the separate DMA port option an Address Bus, Data In Bus, Data Out Bus and controls are provided at the IO connector.

The storage data interface portion of the CPU contains two separate data paths to memory, one for CPU use, labeled CPSDR, and one for direct memory use, labeled DMA Data. Because the CPU operations involve byte, half-word, and full-word operation, shifting of the data is required into the storage data register. For byte storage operations the data in the PRR is right justified and must be relocated in the storage data register according to the least significant bits of the storage address. Similarly, for a byte read, the byte coming from the memory interface must be placed in the lower bits of the storage data register for right justification on the bus into the CPU. This register and multiplexer is implemented by using four of the mux register chips used in the Data Flow Module. Each chip provides a six way by four-bit multiplexer and register with outputs both from the register and the multiplexer. Controls for the multiplexer are located in the storage interface logic module within the storage interface section of the CPU. The address register for the CPU is also located in the storage interface logic module.

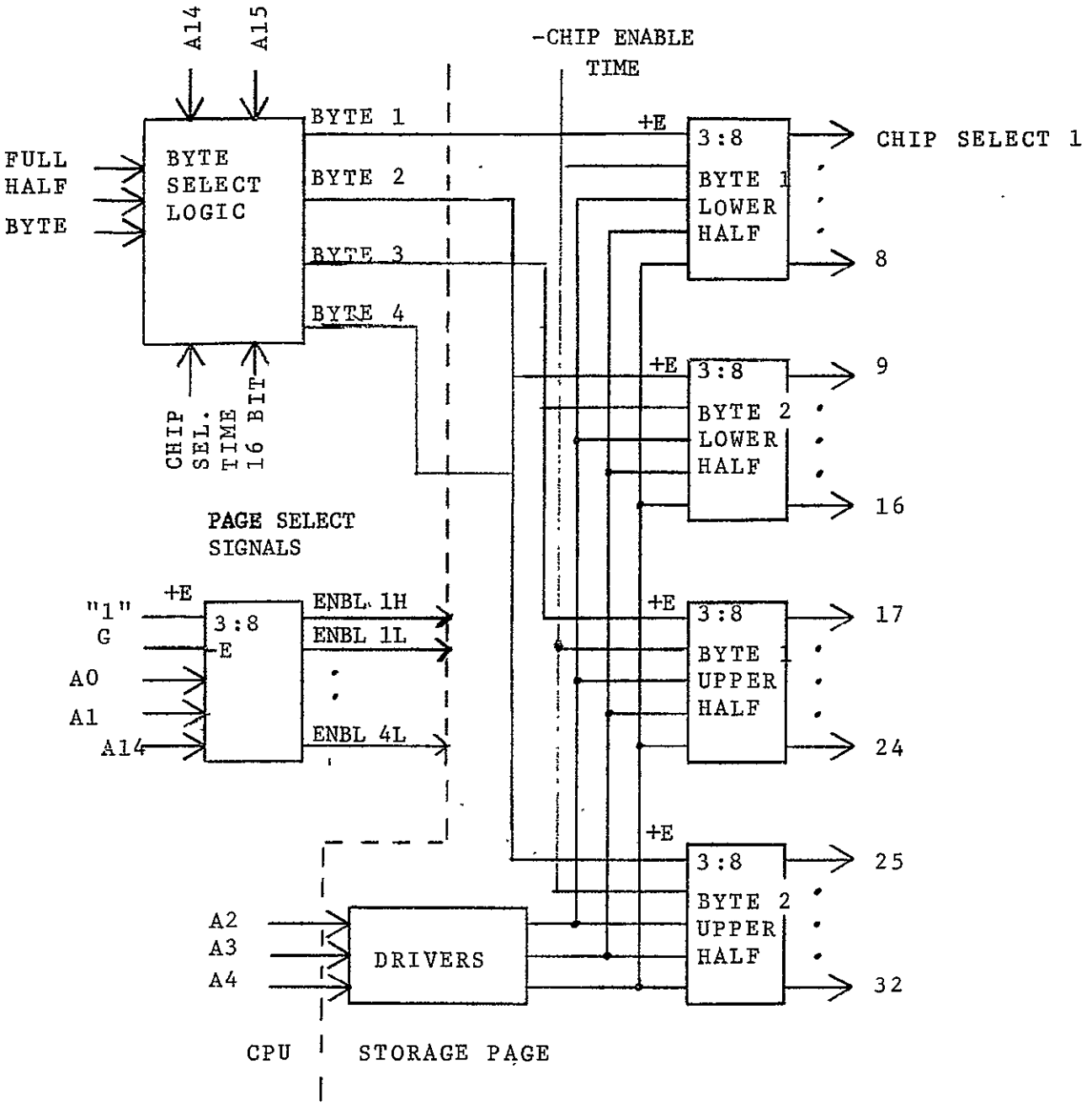
The storage interface portion of the CPU is implemented with four modules: (1) The TSE/SDR module contains four mux registers chips. These chips are interconnected to provide data selection and alignment for read and write operations. (2) D-Mux module utilizes two AOTC chips to provide multiplexing the CPU or DMA data onto the Memory Input Bux. (3) A-Mux module contains two AOTC chips to provide multiplexing the CPU or DMA memory addresses onto the memory address bus. The A-Mux and D-Mux modules are identical. (4) The Storage Interface Logic (SIL) module contains five chips: 1 Register, 1 Memory Control, 1 Memory Timing, 2 Mux Control. The module provides CPU address storage, the logic necessary to control all data multiplexing, partial decoding of address, and the handling of exceptional conditions.

4.10.3 Address Decoding

The storage page is designed to work as a 16-bit, byte addressable memory when working with a 16-bit CPU, and as a 32-bit, byte addressable memory when working with a 32-bit CPU. Therefore, the page is organized as four 4K-byte sections. In the 16-bit machine one or two bytes of memory can be operated simultaneously, but in the 32-bit machine one, two, or four bytes can be operated at the same time.

Referring to Figure 4.10-1, the address handling can be seen to be divided between the SIL function on the CPU page and the supporting logic on the storage page.

The address logic is shown in Figure 4.10-3. The byte select logic in the SIL module is controlled by an input "16-BIT" to perform as a 16-bit memory. (See Table 4.10-3) Only two Byte signals are used in the 16-bit machine. The Chip Select Time signal is generated in the timing chip on the storage page and is used by the SIL module to time the generation of the Byte 1 - 4 signals.



NOTES:

1. The upper two inputs on the 3:8 decoder are enable inputs (one positive enable and one negative enable).

2. Each chip select line goes to 3 of the 96 chips.

Figure 4.10-3 Chip Select Logic

REV A.

HALFWORD	H1		H2	
BYTE	B1	B2	B3	B4
A14, 15	00	01	10	11

DATA WORD (NORMAL)

Table 4.10-3. Byte Select Truth Table

				16 Bit				32 Bit				
M1	M2	A14	A15	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 1	BYTE 2	BYTE 3	BYTE 4	OPERATION
0	0	0	0	X	X	X	X	X	X	X	X	NO OP
0	0	0	1	X	X	X	X	X	X	X	X	
0	0	1	0	X	X	X	X	X	X	X	X	
0	0	1	1	X	X	X	X	X	X	X	X	
0	1	0	0	0	0	1	1	1	1	1	1	FULL WORD *
0	1	0	1	SPEC ERROR				SPEC ERROR				
0	1	1	0	SPEC ERROR				SPEC ERROR				
0	1	1	1	SPEC ERROR				SPEC ERROR				
1	0	0	0	1	1	0	0	1	1	0	0	HALF WORD
1	0	0	1	SPEC ERROR				SPEC ERROR				
1	0	1	0	0	0	1	1	0	0	1	1	
1	0	1	1	SPEC ERROR				SPEC ERROR				
1	1	0	0	1	0	0	0	1	0	0	0	BYTE
1	1	0	1	0	1	0	0	0	1	0	0	
1	1	1	0	0	0	1	0	0	0	1	0	
1	1	1	1	0	0	0	1	0	0	0	1	

*For the full word operation code in the 16-bit machine, the storage logic forces a ONE in address location A14 and proceeds as for half word operation.

NOTE: Byte 1 - 4 signals are gated with the chip select time signal for memory timing. For address and specification errors the Byte signals are suppressed.

Each AOTC chip contains a three-bit decoder with 2 enable lines. One of these decoders is used to generate the Page Select signals. Two different implementations of storage page have been developed for the SUMC-IIB. The first used the FSU 1.5K-bit chips packaged 8 per module. Figures 4.10-1 and 4.10-3 show the overall storage subsystem and the chip select signal generation as they are used for the FSU memory technology. Four decoders are used on the storage page to generate the 32 chip-select signals. Each select signal goes to three of the three bit chips thus implementing the byte organization required by S/360. Figure 4.10-4 shows the organization of the eight chip FSU module.

Each chip is 512 words by 3 bits. The data in and data out lines are tied together on the substrate to permit module wiring on a single layer. By use of the chip select lines and dotting of data lines the FSU module can be used as 4, 1KX3 sections; 2, 2KX3 sections or a 4KX3 section. The SUMC-IIB memory uses the 2K x 6/3.

The second storage page implementation uses the faster 2K-bit chips in 8K x 1 bit basic memory modules (BMMs). This module was developed for SUMC use under contact NAS8-30460 and is described in detail in IBM Report Number 74-585-006 dated 30 June 1975 written by F. C. Tietze. The BMMs are then packaged, two "pages" per memory slice as described in IBM Report No. 75W000 dated 15 April 1975.

The BMM is made up of four storage chips and two driver/sense amplifier chips as shown in Figure 4.10-5. Figure 4.10-6 shows the typical BMM addressing to use the module in the 8K x 1-bit configuration. The overall memory "page" (16K bytes = 1/2 SLICE) is shown in block diagram form in Figure 4.10-7.

4.10.4 Exceptional Condition Monitoring

The storage subsystem performs selective checks to detect abnormal memory conditions when memory service is requested either by the CPU or DMA. For both CPU and DMA operation the memory subsystem (SIL) makes address checks, parity checks, storage protect violation checks and specification violation checks. An address specification error will be generated if the address provided by the CPU or DMA exceeds the capacity of storage provided. The main store has parity generation and checking logic for each byte. On a read operation the 4 byte-parity bits are ORed together to provide a single line which indicates the current status of the parity checkers. A storage protect violation signal will be generated if either the CPU or DMA tries to write into a location which has been protected against storage operations from that source. Specification violations are as follows: When a full-word operation is requested, the two least significant bits of the memory address must be zeros. If the requested operation was for a half-word, or was from the DMA the least significant bit of the address must be zero. Violation of either of these conditions will result in a specification check error signal.

Since both the CPU and direct memory access can have simultaneous requests for service, only the storage subsystem knows whether a current memory operation is associated with the CPU or the DMA. Therefore, error signals are gated internal to the storage interface so that signals going to the CPU will only be associated with CPU memory operation and those going up to the DMA will be associated only with DMA operations.

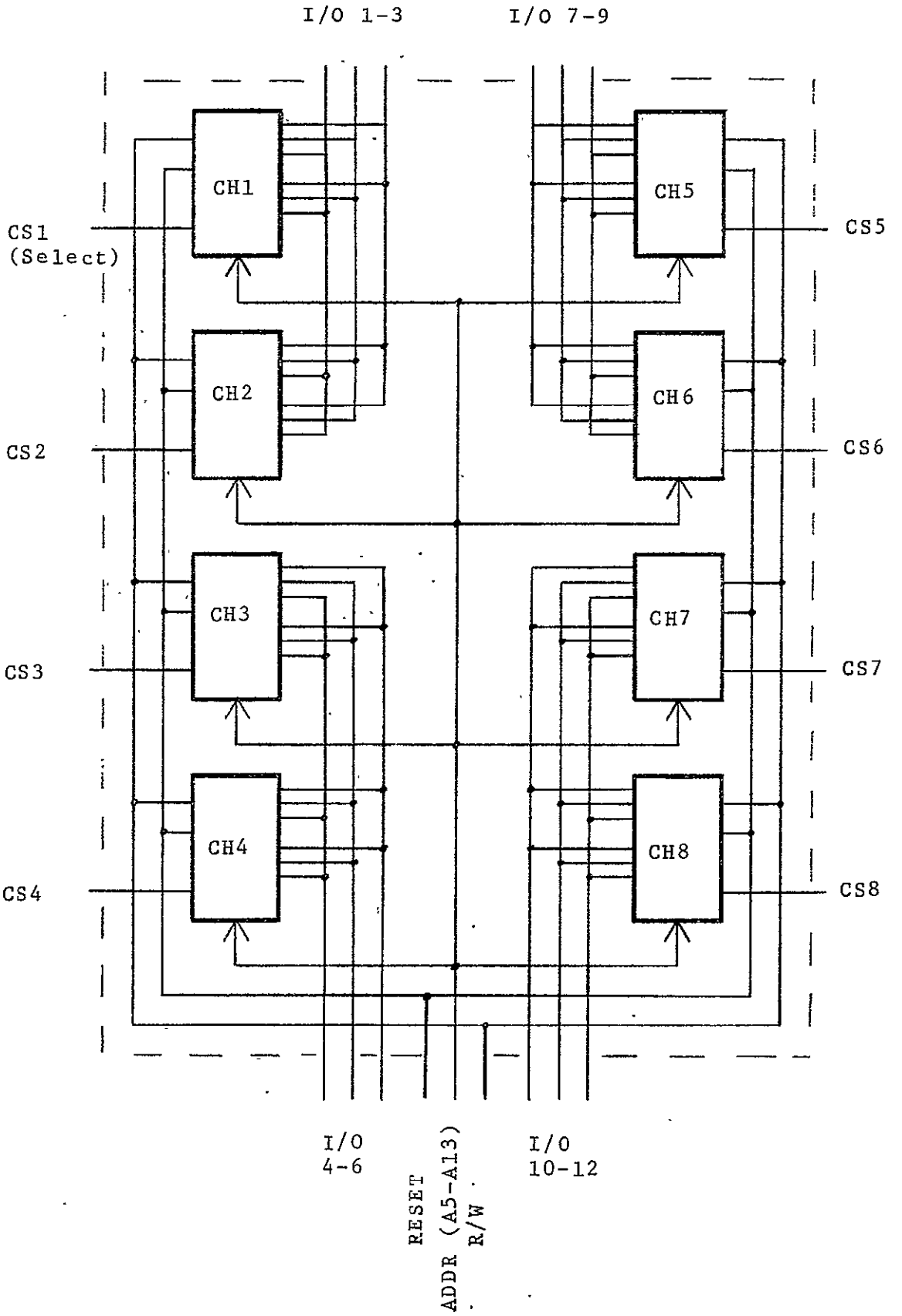
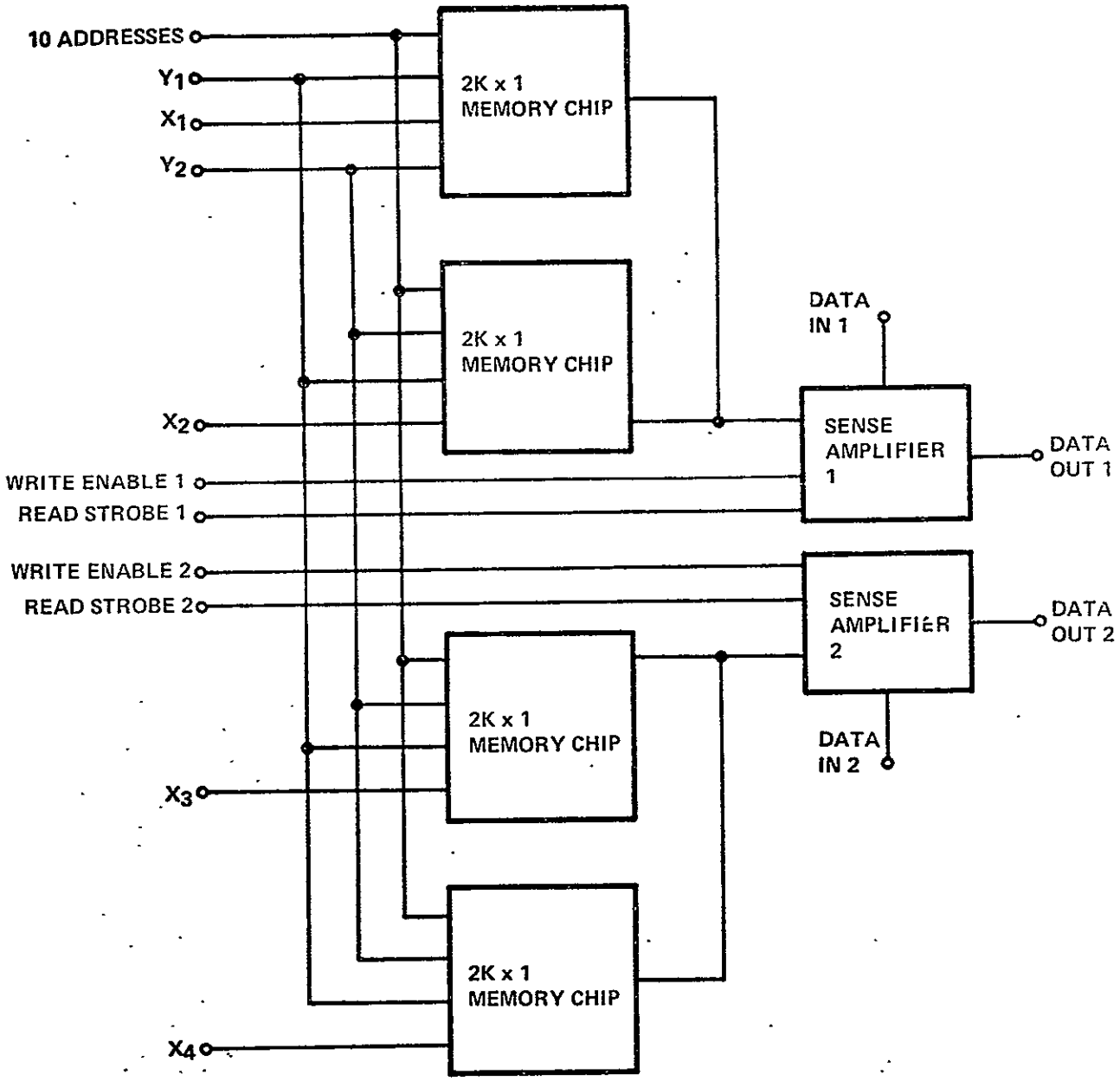


Figure 4.10-4. FSU Memory Module



4K x 2 CONNECT X_1 & X_3
CONNECT X_2 & X_4

8K x 1 CONNECT IN 1 & IN 2
CONNECT OUT 1 & OUT 2

X_{1-4} ARE CHIP SELECT LINES

$Y_{1,2}$ SELECT 1K ARRAYS WITHIN EACH 2K CHIP

Figure 4.10-5. Basic Memory Module Block Diagram

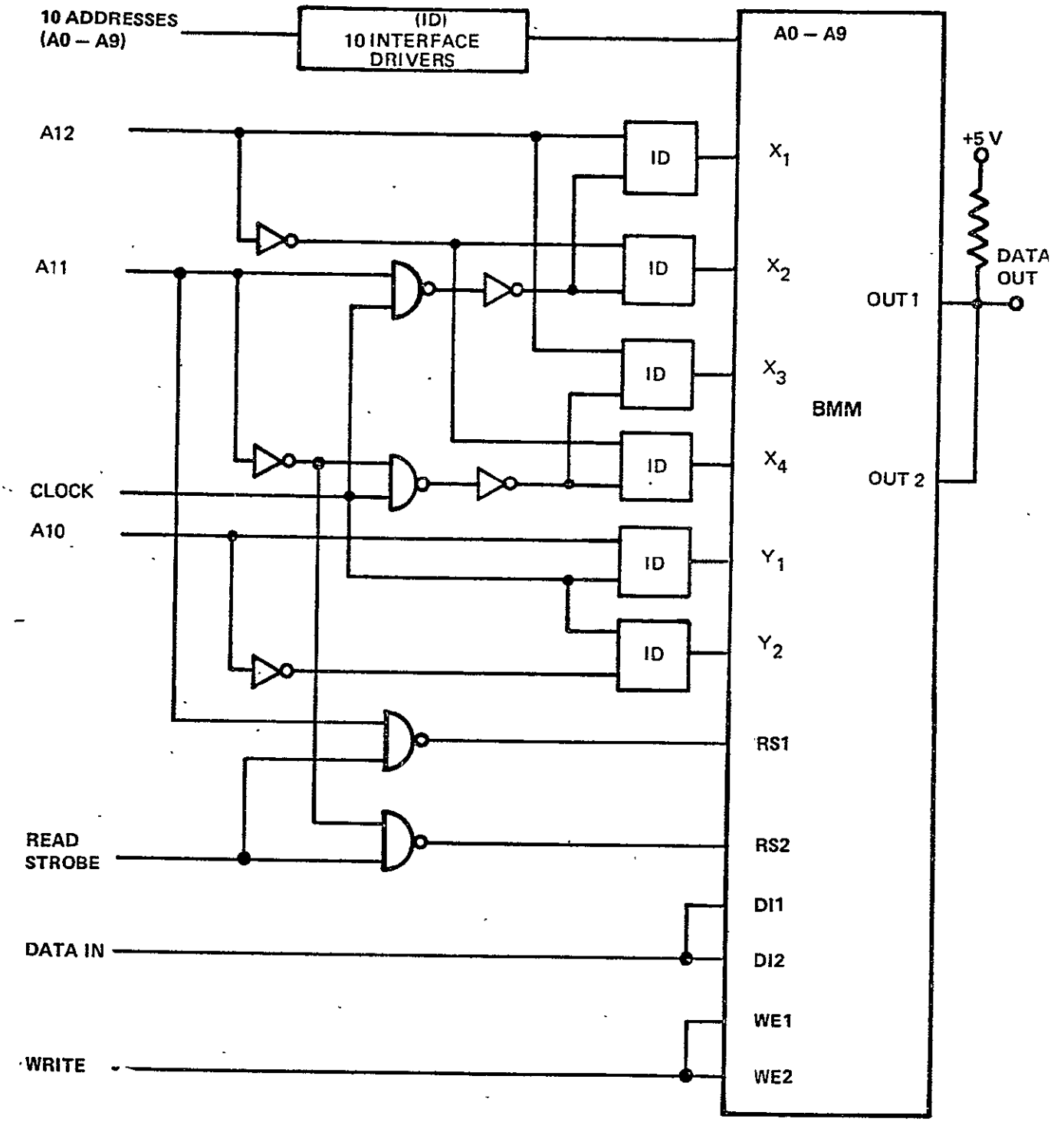


Figure 4.10-6. 8K x 1 Application

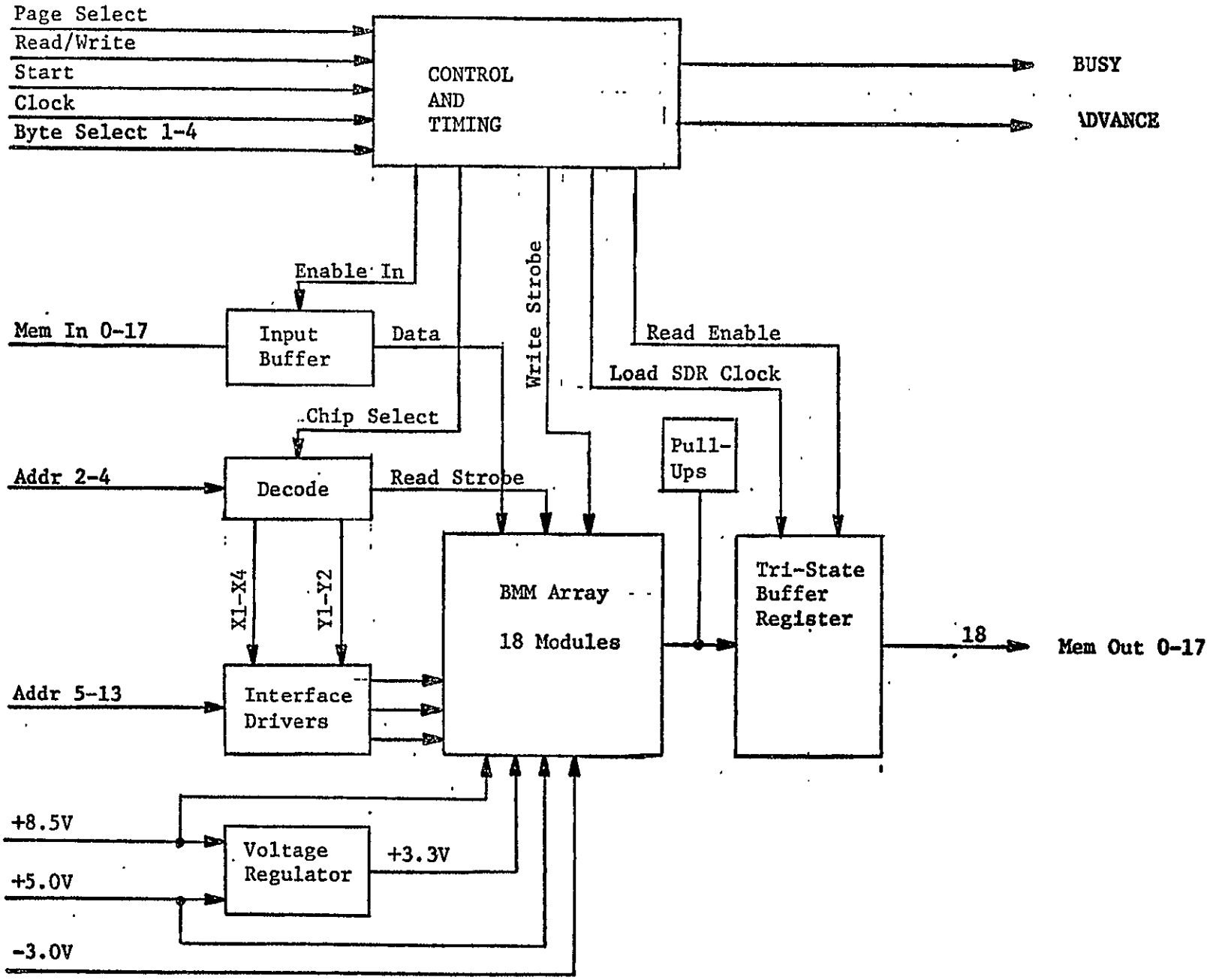


Figure 4.10-7. Riesling Storage Page Block Diagram

The AOTC chips contain parity trees which are used in the generation of parity. If the input to the ninth bit of the parity generator is tied to a one, the output of the parity network for that chip will be the proper value to store as a parity bit in the memory. If the ninth input is a ground (zero) signal, false parity will be generated. The SUMC-IIB provides the ability to force a parity error by tying the ninth input bit to the complement of the Load IR Right signal.*

Thus, whenever the load is a 1R RIGHT signal and a store operation is indicated, false parity will be stored. This will be used by the diagnose instruction to allow the parity generation and checking network to be checked for proper operation. To ensure proper storing of false parity, both the micro program control word which initiates a memory operation and the following it must indicate a Load IR Right.

Parity checking is performed on a byte basis with these parity signals sent to the SIL module where they are ORed together before being sent either to the CPU or the DMA.

Storage protection is provided to prevent inadvertent writing over permanent data or instructions which should not be modified during the normal program operation. The storage protect feature divides the memory into segments of 512 half-words. Each segment can be protected or not under program control (priviledged instruction). Thus the CPU slice has four 16-word registers to hold the storage protect words for 64K-bytes of memory. The CPU can load the storage protect register by initiating special memory operations (refer to Table 4.10-1). If the no-op code is used in combination with the write-move mode of operation the contents of the PRR will be loaded into the appropriate storage protect register. When a store operation is requested by CPU or DMA, the most significant bits (bits 2 thru 5) are used to address the storage protect register. If the address corresponds to a protected area, a storage protect violation signal is generated and sent back to the SIL module. Write operations will be converted to read operations when a storage protect violation occurs. The storage protect violation signal is valid at the rise of chip select time. Due to the way the SDR multiplexer is controlled, the contents of the SDR at the end of a memory operation involving storage protect violation will be some combination of the data loaded in from the PRR at the beginning of the write and the data read from the array in the converted read operation. At this time, the contents of the SDR, at the end of such an operation, will not be specified.

On a CPU operation, the exceptional conditions result in forcing an MROM branch to location zero and setting an error latch in the CPU. While this error latch is set, scratch pad memory store operations and memory read or write operations will be suppressed by a signal called Write Valid being in a zero state. When Write Valid is a zero, it forces all memory operations which are requested to perform as if M1 and M2 bits of the microprogram word were zeros. Hence, some form of no-op will be performed. Thus the SDR can be loaded and the storage protect registers can be loaded but the memory cannot be cycled in either a read or a write mode.

*Load 1R Right is a decode of MROM R14, R15, R16 = 100.

4.10.5 Timing

To facilitate changes to the memory technology without changes to the CPU logic, an asynchronous interface is provided. The CPU requests memory service by the state of the M1 - M4 bits of the microprogram word at CKZ time. The fall of CKZ starts the memory operation and the memory informs the CPU that the operation is complete by dropping its busy signal. For a read-and-hold operation the advance signal indicates completion of the read portion. The busy does not fall until a subsequent write operation occurs. If the CPU is ready for data before the data is ready, the CPU will hold in a state called LIMBO until the memory is ready. Conversely, the Memory will hold the data until the CPU uses it. See Figure 4.10-8.

4.10.6 Interfaces

The storage interfaces are listed and described below. The drive capability of the memory interface is shown in Table 4.10-2.

Start	Starts a memory cycle
Busy	Indicates a memory operation is in progress
Advance	Fall of advance is used to restart the CPU timing during a storage operation
Addr (2-13)	In conjunction with page select and byte signals, provides addressing of the storage pages.
Pg Sel (1-4)	Uniquely selects one of four pages dependent upon the two most significant address bits A0 and A1
Byte (1-4)	Selects the proper byte or bytes for a particular memory operation (dependent upon A14 and A15)
Write	Logic 1 indicates the current memory operation is to be a STORE operation. Logic 0 indicates a READ.
MEM OUT (0-15)	Contains the data for a store operation
CLK	A 9 MHz clock signal for use by the storage pages as required.

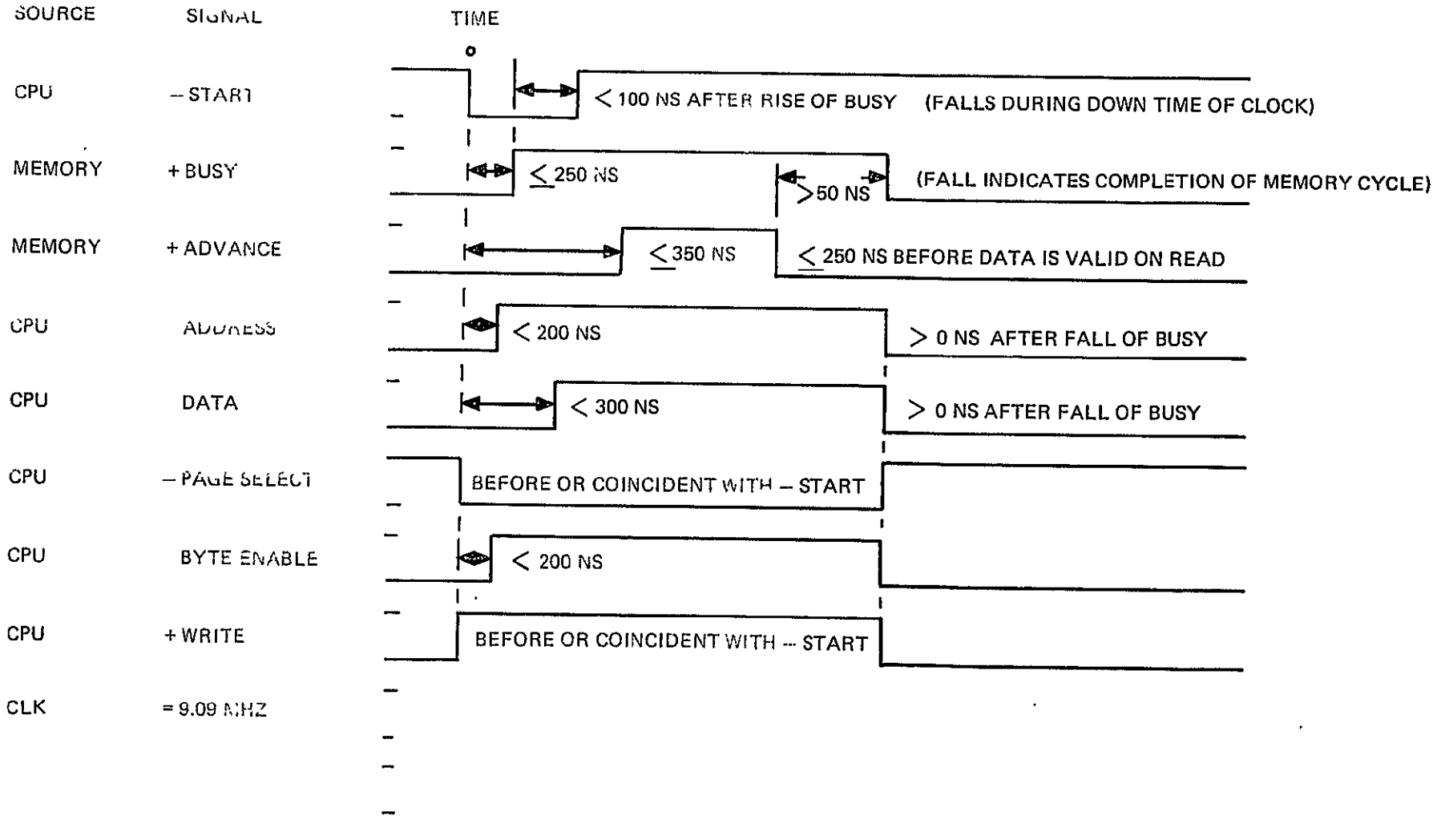


FIGURE 4.10-8. MEMORY TIMING

Table 4.10-2. Memory Interface Drive

NAME	EQUIVALENT TTL LOADS
Start	8
Busy	4
Adv.	4
Addr. (2-13)	4
Pg Sel (1-4)	7
Byte (1-4)	8
Write	8
Clk	S/N 54128W
Mem Out	4

4.11 SUMC-IIB I/O DESCRIPTION

The I/O section of the SUMC-IIB provides the means of communication between system I/O, test support equipment (TSE), and the CPU/main store (MS). In the SUMC-IIB the I/O is implemented as a 16-bit parallel channel providing direct I/O, buffered I/O, external interrupt, and (optionally) direct memory access (DMA). The standard DMA facility is provided as a separate interface. Figure 4.11-1 is an I/O block diagram showing the integrated DMA. Figure 4.11-2 shows the interfaces between the I/O and all other hardware.

The I/O channel is switched between system I/O and TSE, under TSE control. Under normal operation the tester will not switch the input multiplexer unless the I/O has acknowledged the tester's service request. With manual intervention, however, an operator can gain system control by use of a manual override switch.

The DMA is handled directly between I/O and main store. The only impact on the CPU is the slowing of program execution due to DMA "stolen" memory cycles. If the CPU and DMA are both making continuous requests for service, the MS will give cycles alternately to the CPU and DMA. For this purpose, a read/modify/store operation is considered to be one memory cycle.

The TSE to computer interface is made up of two separate paths. Commands and data from the CPU come directly from the CPU via the TSE MUX. Since this path is required for monitoring registers within the computer there is no need for an additional data path. Data and channel code words to the CPU go via the I/O interface and follow the format of the System I/O. Using the I/O interface allows a single input in the data path to serve both the TSE and System I/O.

4.11.1 System I/O (Integrated Channel)

This section describes the interface of the SUMC-IIB with other system equipment. The interface provides a 16-bit parallel channel which can be used in several operating modes. A four-bit device identification code permits up to 16 system devices to be attached directly to the I/O channel but sub-multiplexing can be provided on a single device ID for additional devices, if necessary. An alternate approach to system I/O is to attach an Input Output Adapter IOA to the I/O channel and handle all I/O through it. This approach could facilitate analog I/O, serial I/O, and implement interrupt masking and priorities. Such an IOA could provide a S/360 standard interface to peripheral devices.

The integrated I/O channel provides three types of device initiated information transfer: (1) Buffered I/O, (2) Direct Memory Access (DMA)*, and (3) External Interrupts. Program initiated I/O is provided by the SIO Instruction (Direct I/O). For device initiated I/O, the channel is tied up from the time the device requests service until it deactivates the service request line. Therefore, it is important that each device use the I/O mode best suited for its I/O usage and not tie up the channel for unnecessary time. A device must deactivate all channel control lines INPUT, DMA, etc., before detaching from the channel. The service

*Available as part of the integrated I/O channel or as a stand alone interface.

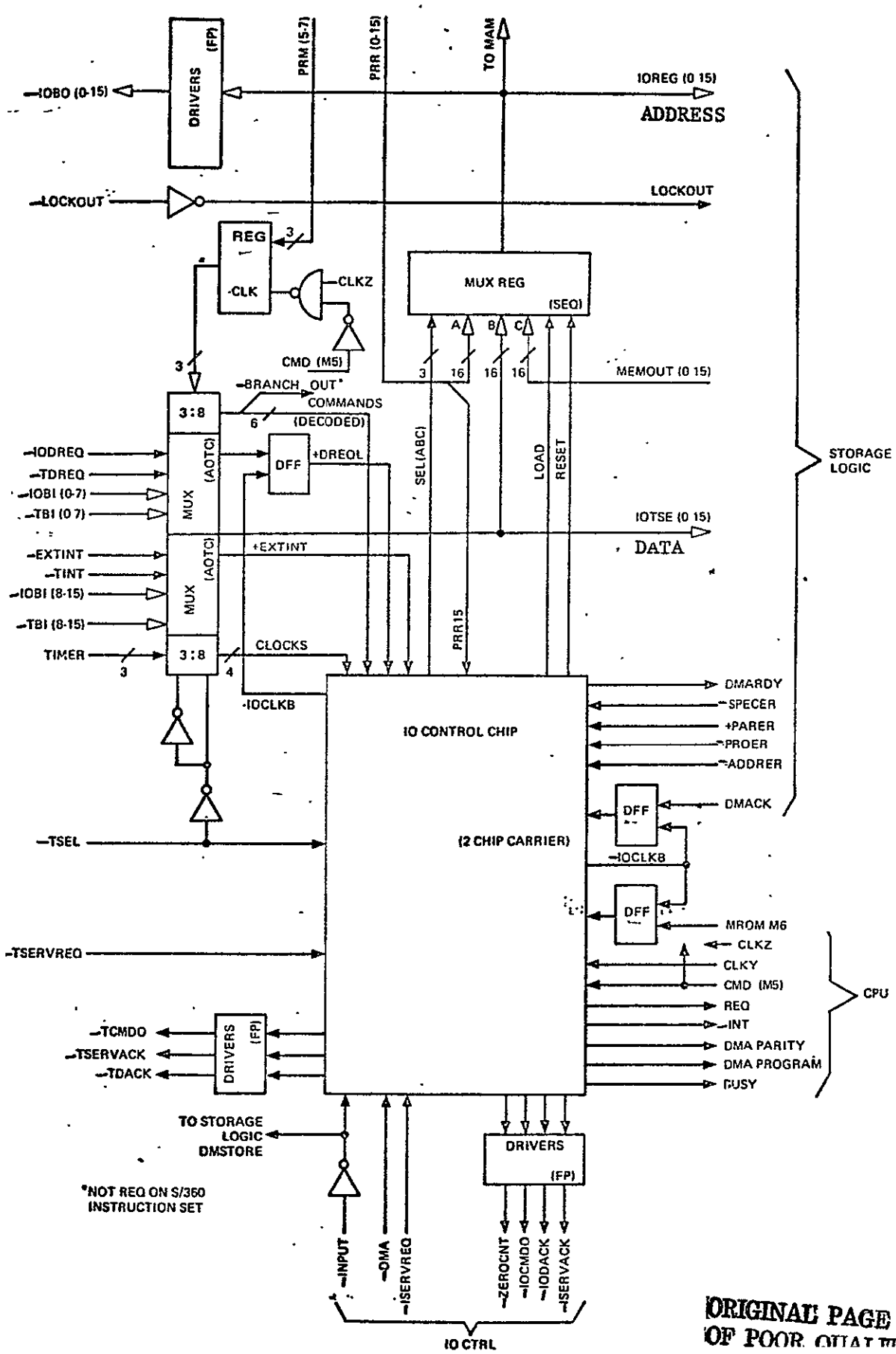
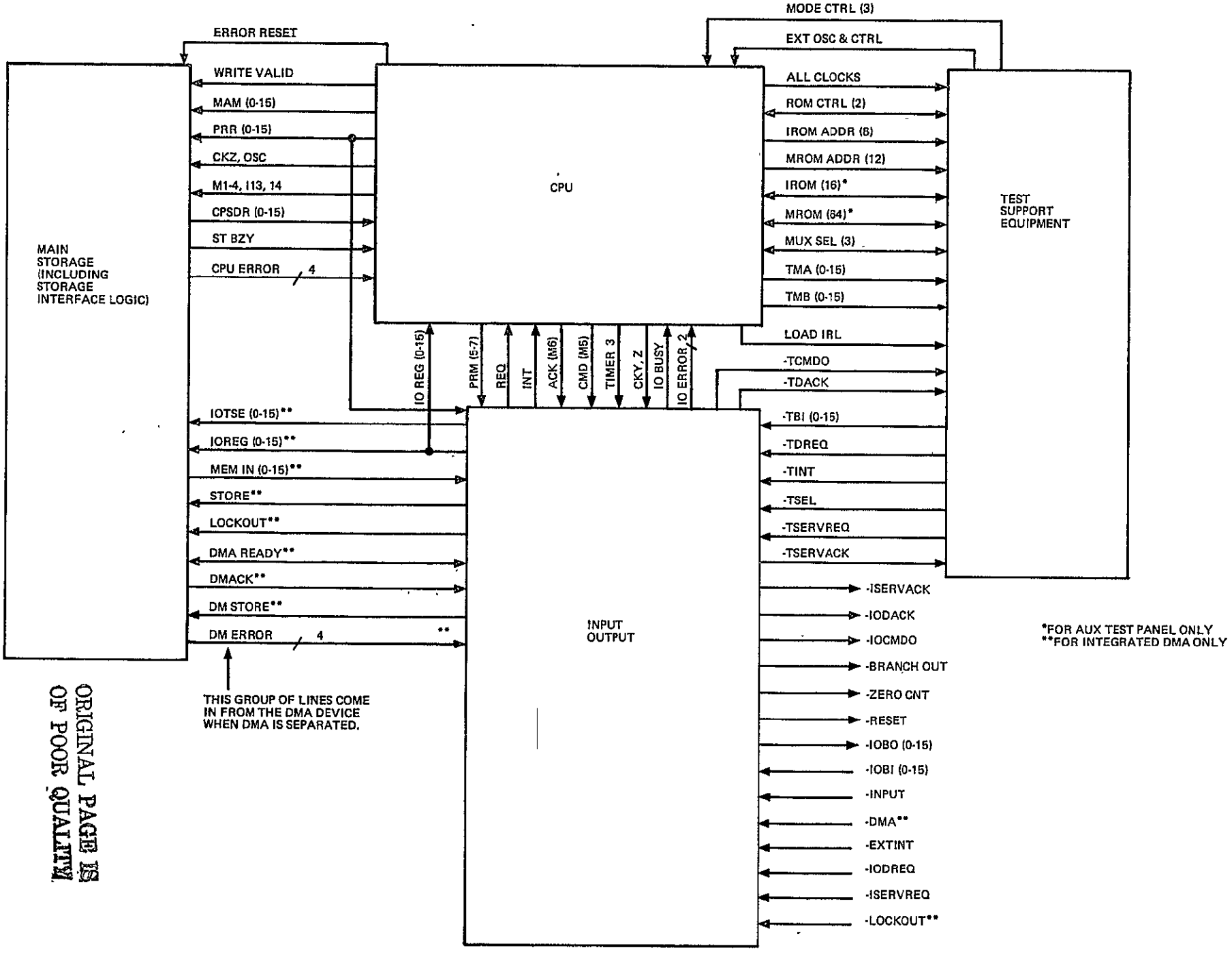


Figure 4.11-1. SUMC-IIB I/O Block Diagram (with Integrated DMA)



ORIGINAL PAGE IS OF POOR QUALITY

FIGURE 4.11-2 MT INTERFACE

request line shall be controlled by the devices so that a new service request is not generated until the I/O deactivates service acknowledge.

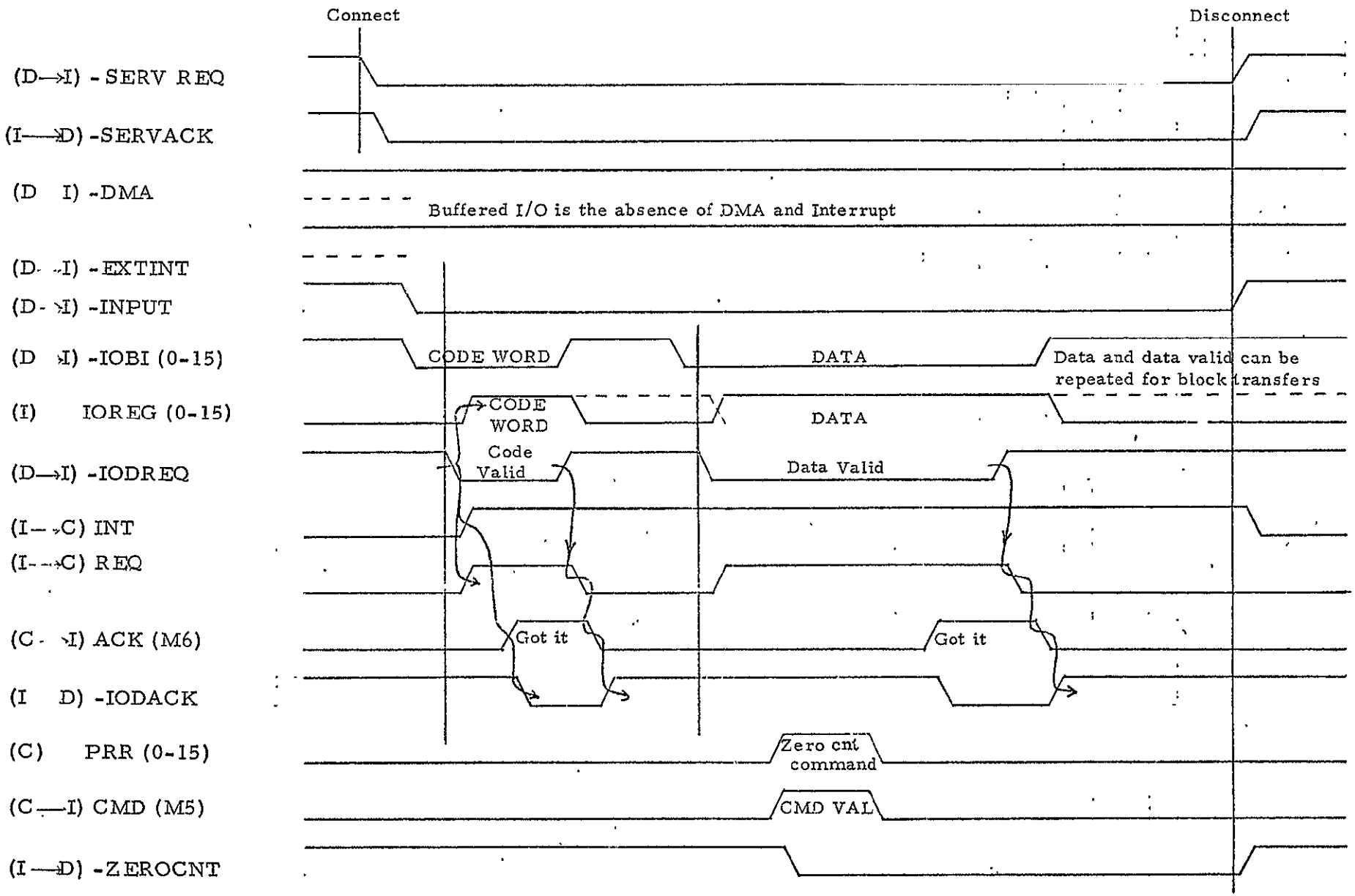
4.11.1.1 Buffered I/O allows a device to transfer single or multiple words of data to/from a table in main memory without knowing the location of the table. The CPU hardware and microcode keep track of table word count and address incrementing. When the table is full/empty the device is notified by a signal on the -ZERO COUNT line. Separate input and output tables are maintained for each Buffered device code (16 codes). The microcode uses four, 16 word tables located by the programmer in memory to keep track of input storage address and count, and output storage address and count for the 16 devices. The fixed memory location 12 (S/360 CAW) is used as a pointer to the location of the 64 word Buffered I/O table. Buffered I/O sequences are shown in Figures 4.11-3 and 4.11-4. The device provides an I/O Channel Code word to the CPU to identify the service desired (see Table 4.11-1).

The programmer controls buffered I/O by initialization of the address and word count memory locations and can start an I/O device by use of a direct output command which is described later in this section. Alternatively the device can use an external interrupt to notify the program that the buffer table has been filled/emptied. (Interrupt is also described later in this section.)

It should be noted that even though the CPU hardware is interrupted to handle the buffered I/O transfers the program is not interrupted and the time consuming save operations associated with program interrupt are not required. Buffered I/O operations are handled between instructions and do not use any register visible to the programmer.

4.11.1.2 Direct Memory Access provides the fastest means of transferring data to or from main storage. This description is for the DMA as an integrated feature of the I/O channel. DMA as a separate interface is described in paragraph 4.11.2. Both waiting time and transfer rate are faster than other I/O modes. The CPU hardware does not participate in the DMA operations and, therefore, DMA operations can take place in the middle of an instruction. If the CPU is not requesting memory service at the time a DMA operation takes place, there will be no impact on CPU performance. In case of conflicts between CPU and DMA, memory service is alternated so that neither can lock the other out (except for the I/O use of LOCKOUT which will be explained later). DMA service is device initiated by seizing the channel (connect). After seizure, one or more words can be transferred. The channel is tied up until the device releases it by deactivating service request.

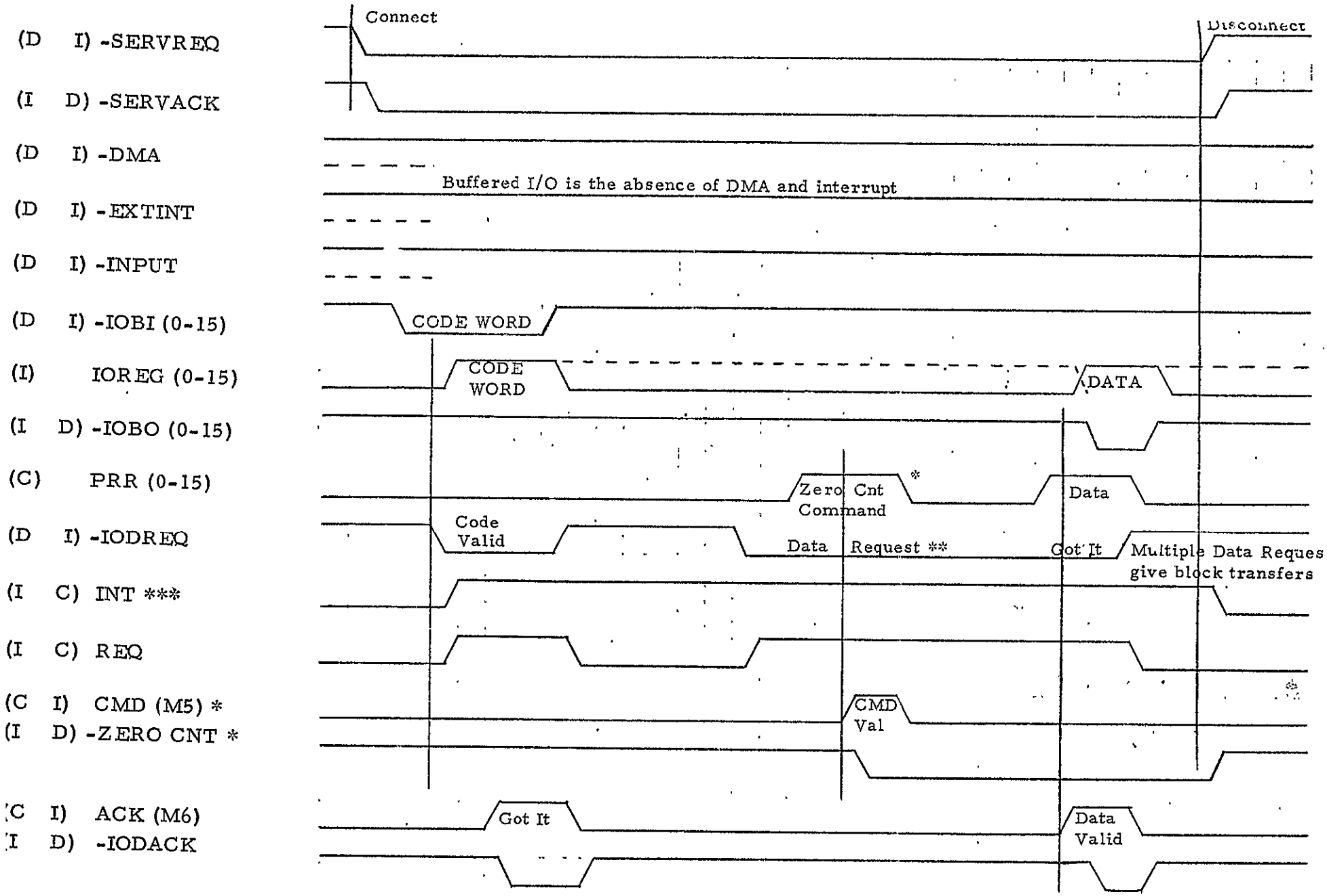
After the device connects to the channel, DMA operation is a series of independent read and write cycles. Read and write operations can be mixed but with the following stipulation. The -INPUT line must be stable from the fall of -IODREQ (indicating address valid) to completion of that memory cycle. An input (store) cycle is complete when the second -IODACK pulse rises (ends). And an output



SIGNAL SOURCE: (D) = DEVICE
(I) = I/O PART OF SUMC-IIB
(C) = CPU

FIGURE 4.11-3

BUFFERED INPUT SEQUENCE



Signal Source: (C) = CPU
(I) = I/O part of SUMC-IIB
(D) = Device

* Zero count is generated by the CPU before transfer of the word which will fill/empty the memory buffer area.

** Rise of -IODREQ indicates receipt of data by the device.

Figure 4.11-4. BUFFERED OUTPUT SEQUENCE

TABLE 4.11-1. I/O CHANNEL CODE WORD

	T	I/O	EX-INT	UNUSED	CPU USE	DEVICE ADDRESS	FUNCT. CODE
BIT	0	1	2	3 5	6 7	8 11	12 15

IOTSE BITS	USE	
0	LOGIC 0 - I/O SERVICE LOGIC 1 - TSE INTERRUPT	NORMAL CONVENTION
1	LOGIC 1 - BUFFERED INPUT LOGIC 0 - BUFFERED OUTPUT	USED ONLY FOR BIT 0 = 0
2	LOGIC 1 - EXTERNAL INTERRUPT LOGIC 0 - BUFFERED I/O	
3-4	DMA ERROR	
5	UNUSED	
6-7	CPU USE	
8-15	TAG (DEVICE ADDRESS) OR INTERRUPT CODE	

Note: The device or tester is responsible for generating the code word in the format shown above, but inverted bit-by-bit at the cable.

(read) cycle is complete when the memory drops DMACK (about 100 ns after the device raises - IODREQ).

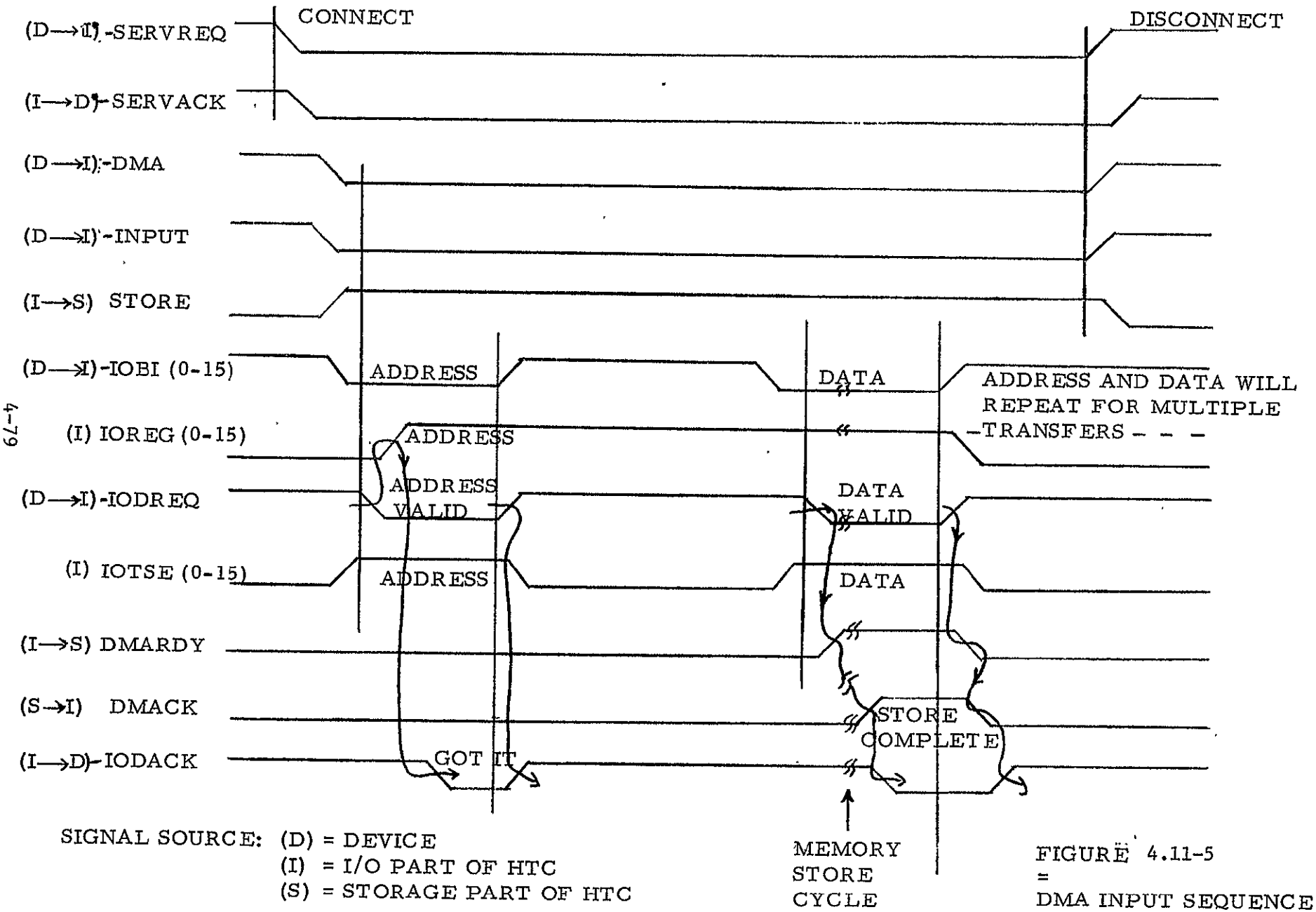
On a store operation the device first provides the address and signals its validity by dropping - IODREQ. The I/O stores the address in the IOREG (see Figure 4.11-1), signals receipt of the address with -IODACK and completes the "handshaking" with the device. When ready, the device places the data on the bus and again signals with -IODREQ. The I/O issues DMA RDY to the memory subsystem. When the store operation is complete the memory raises DMACK to signal that the data has been stored, and the handshaking is completed (see Figure 4.11-5).

The direct memory read is very similar to write from the device's point of view but is somewhat simplified at the I/O to memory interface (see Figure 4.11-6). The device provides the address as before and the I/O stores it and generates a DMA RDY signal to the memory which (in conjunction with STORE = 0) initiates the memory read. If the memory is idle the read will begin immediately. If busy the read will not be started until the current memory operation is complete. When the memory has performed the read and the data are on the MEMOUT lines, the DMACK line is raised to the I/O. Meanwhile the I/O uses -IODACK to acknowledge receipt of the address. When the device is ready it requests the data by again dropping -IODREQ. If the memory has read the data it will be waiting and the I/O will respond immediately with -IODACK. If memory is not ready the -IODACK signal will be held up until DMACK signals the memory read is complete. The device acknowledges receipt of the data by raising -IODREQ and the I/O responds by raising -IODACK to the device and dropping DMA RDY to the memory. Since the I/O has a register for DMA data, device delays in requesting data do not hold up the CPU but the shared I/O bus is tied up.

By switching the -INPUT line from high (read) to low (store) between memory cycles, a device can perform a read/modify/store operation but the CPU is not kept out of memory between the read and store cycles no matter how fast the modify operation is. (The memory will alternate memory cycles between the CPU and I/O when there are conflicting requests.)

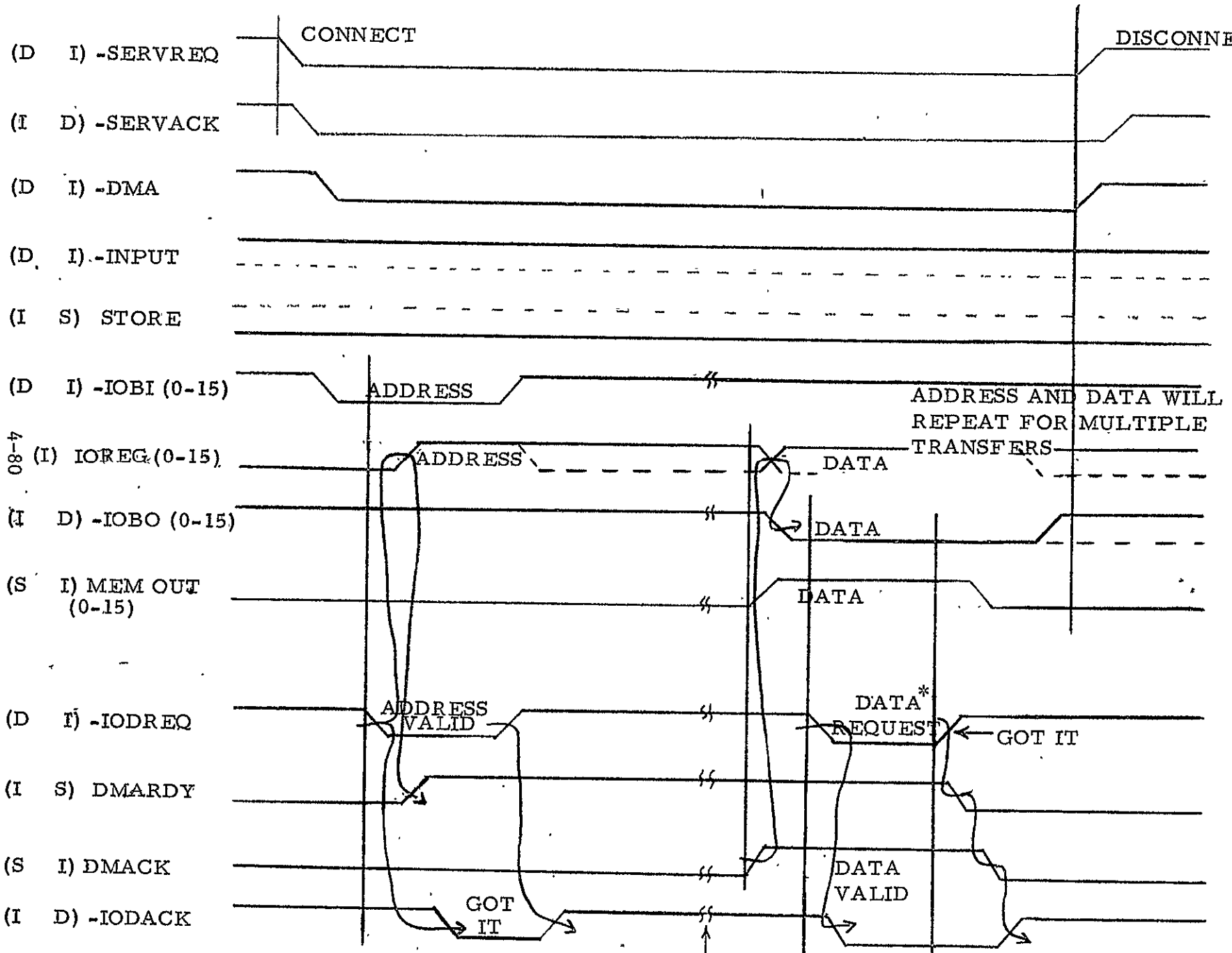
A line named -LOCKOUT is used to keep the CPU from using memory when it might interfere with DMA operation. The device must drop -LOCKOUT before starting the memory cycle. +LOCKOUT is sent to the memory. This signal has no effect on the memory until a DMA operation is requested by the device and granted by the memory. Thereafter, the CPU will be locked out of memory (memory busy is sent to the CPU) as long as the +LOCKOUT is held up.

All DMA operations are 16 bits wide. Any device can, of course, read a 16-bit word and only use part of it. Storing partial words, however, imposes greater problems. A device can read a half-word, modify the word, then store it in the same location effecting a part-word store. If, however, the CPU stores something in that location after the read but before the restore it will be destroyed.



SIGNAL SOURCE: (D) = DEVICE
(I) = I/O PART OF HTC
(S) = STORAGE PART OF HTC

FIGURE 4.11-5
=
DMA INPUT SEQUENCE



SIGNAL SOURCE: (D) = DEVICE
(I) = I/O PART OF SUMC-IIB
(S) = STORAGE PART OF SUMC-IIB

FIGURE 4.11-6

DMA OUTPUT SEQUENCE

* If data request comes before Data Valid -IODACK will be delayed

Storage assignments (table size) should be made to avoid this problem, however, LOCKOUT can be used to prevent this. Due to the sensitive nature of the LOCKOUT signal it will not generally be made available to I/O devices. Interface units having access to the lockout features will be able to keep the CPU out of memory during a read/modify/store operation and for high speed bursts of I/O data.

During a DMA operation, the I/O provides DMA error processing. The I/O provides two gated signals to the CPU. One signal is the DMA parity error and is gated by MROM M6. The second is the logical OR of three DMA error signals (Store Protect, Address, or Specification) also gated by MROM M6. Any DMA error signal causes the following:

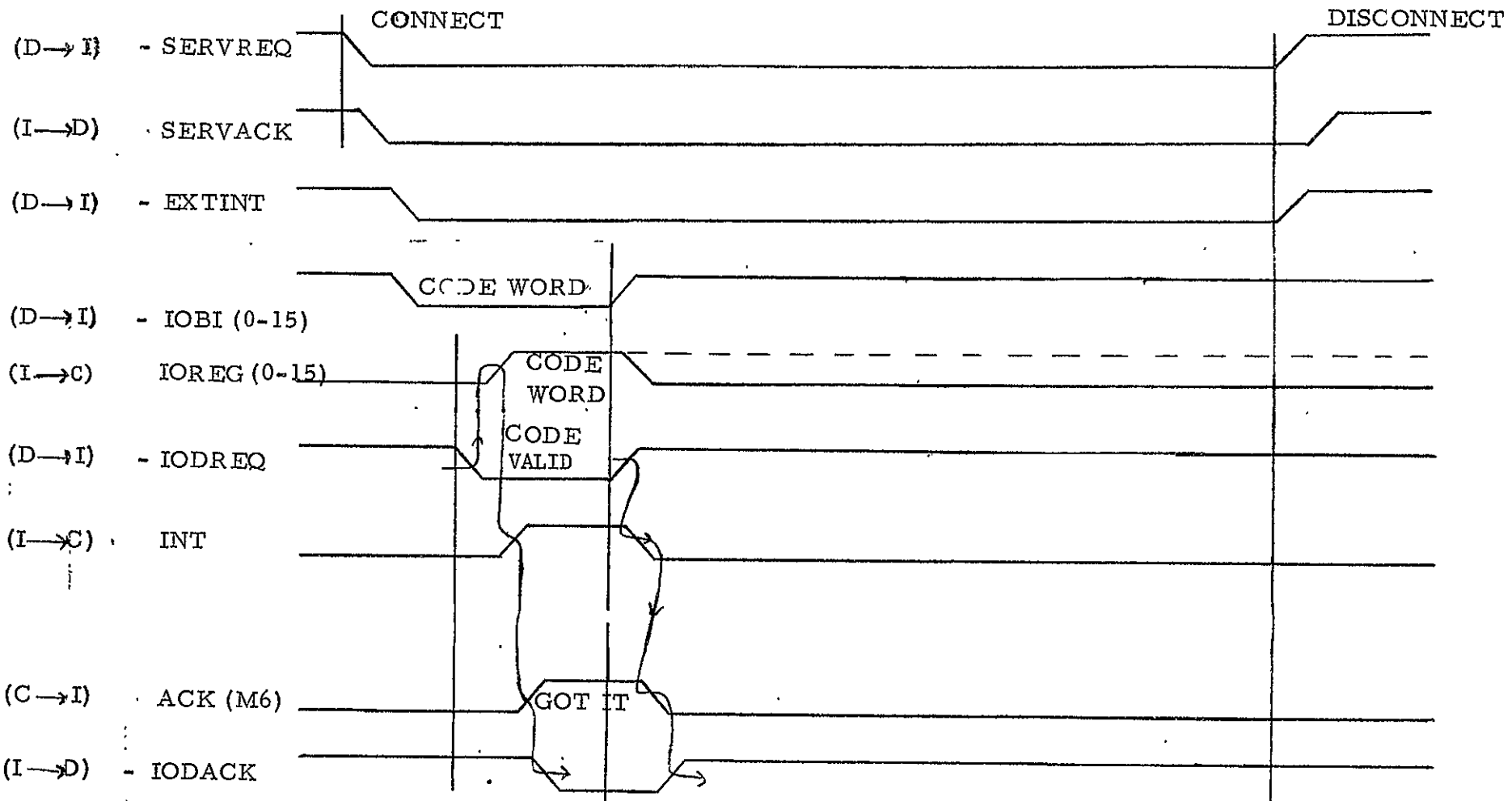
- CPU Interrupt
- Reset I/O Register
- Terminate DMA Operation

4.11.1.3 External Interrupt permits a device to interrupt the normal program sequence. Interrupt handling is done by microcode so it is an architectural function. The SUMC-IIB transfers control to the supervisor by storing the old PSW and loading a new one (as done by S/360). A single hardware level of I/O interrupt is provided. Programmed, or microprogrammed priorities can be implemented. All I/O interrupts can be enabled/inhibited by the set system mask instruction or similar microcode (described in Direct I/O subsystem).

In the interrupt sequence an I/O Channel Code word is sent from the device to the computer for use by the interrupt handling microprogram and the subsequent interrupt program associated with the interrupting device. The interrupt sequence is shown in Figure 4.11-7 and the I/O Channel Code Word format is shown in Table 4.11-1.

4.11.1.4 Direct I/O provides two types of service: (1) command and data transfers with I/O devices and (2) CPU to I/O control operations. All of the operations involve the CPU generating a command word in the PRR and identifying it by microprogram bit M5. The command words are shown in Figure 4.11-8. The first four and system reset are programmer initiated by use of the Start I/O instruction. Zero Count is embedded in the buffered I/O microcode sequence and cannot be programmer initiated. Branch Out and Set Interrupt Mask are part of the microcode for non-I/O instructions. Branch Out is a special branch instruction used to end an interrupt program and is not supported in System/360. Set Interrupt Mask is part of Set System Mask and Load PSW instructions.

Only the Direct Input and Output instructions place command words on the output but for I/O devices. Direct Input and Output follow the sequences of Figures 4.11-9 and 4.11-10, respectively. Test and Reset Interface follow the sequences of Figures 4.11-11 and the remaining four command types use the non hand shaking sequence of Figure 4.11-12.



SIGNAL SOURCE: (D) = DEVICE
(I) = I/O PART OF SUMC-IIB
(C) = CPU

FIGURE 4.11-7

EXTERNAL INTERRI
SEQUENCE

PRR WORD

0 4	5	6	7	8 15	
1-1	1	1	1	COMMAND[1]	DIRECT OUTPUT[2]
0-0	0	0	0	COMMAND[1]	DIRECT INPUT[2]
0-0	0	1	1	X --------- X	RESET INTERFACE (Halt I/O)
0-0	1	1	0	X --------- X	TEST INTERFACE[2]
0-0	1	0	0	X --------- X	ZERO COUNT
0-0	1	0	1	X --------- X	BRANCH OUT
0-0	0	0	1	X --------X M	SET INTERRUPT MASK[3]
0-0	0	1	0	X --------- X	SYSTEM RESET

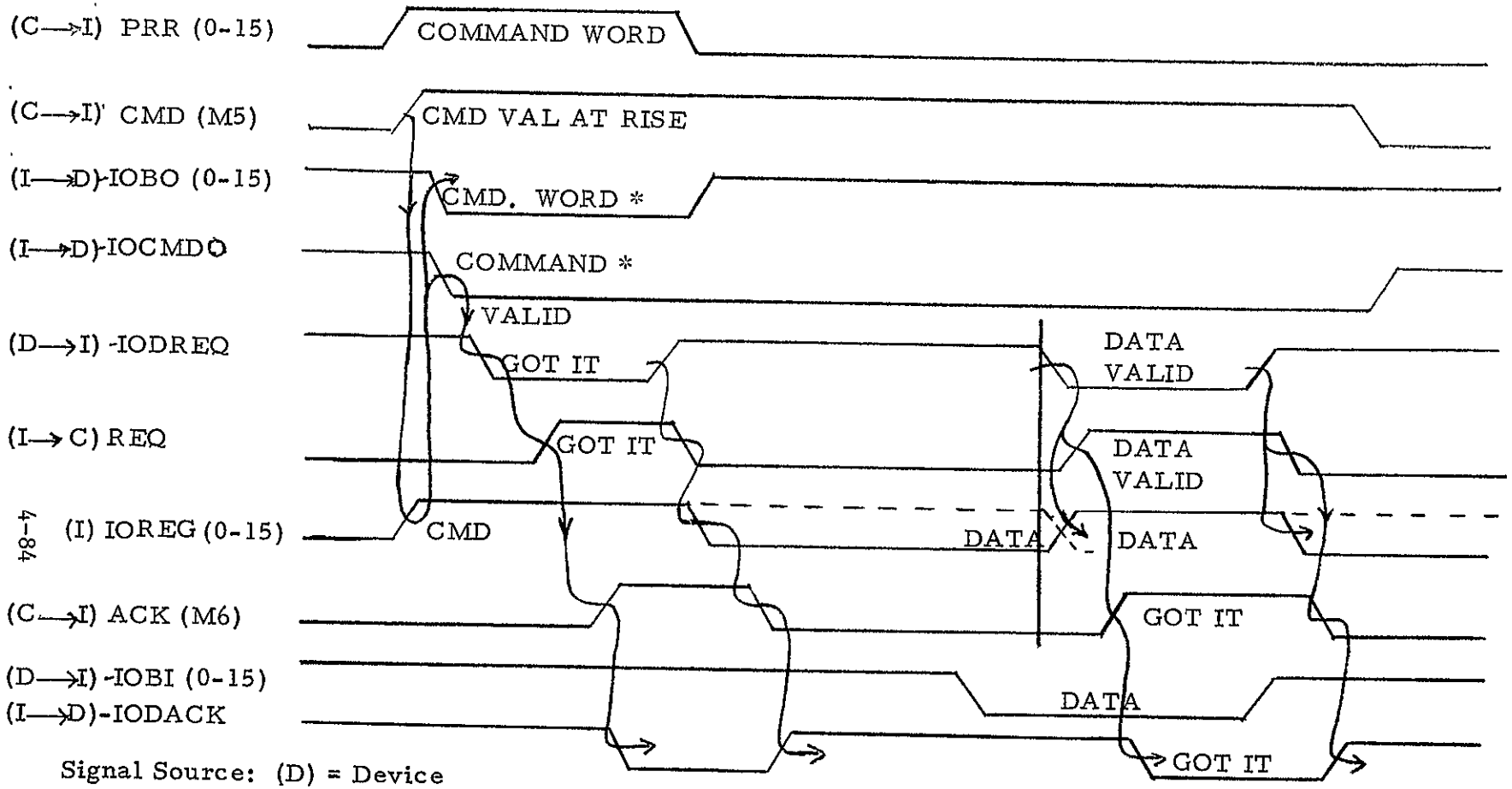
NOTES:

1. The least significant 4 bits of the command are the device address. The most significant bit of the 8 bit command shall be a 1 for an I/O command and a 0 for a TSE command. The signals on the I/O bus out will be the logical complement of that in the CPU due to the inverting drivers.

2. For direct in, direct out, or test interface, the I/O logic will place a 1 in ALU transfer field bit location 0 before raising the ACK line if the channel is busy. Direct input or output will be terminated if the channel is busy, and will place the 16 bit command word on the Bus Out if not busy.

3. If M equals 0 all I/O and TSE interrupts will be inhibited. Inhibited interrupts will remain pending and the I/O channel will be hung.

Figure 4.11-8. Direct I/O Command Word

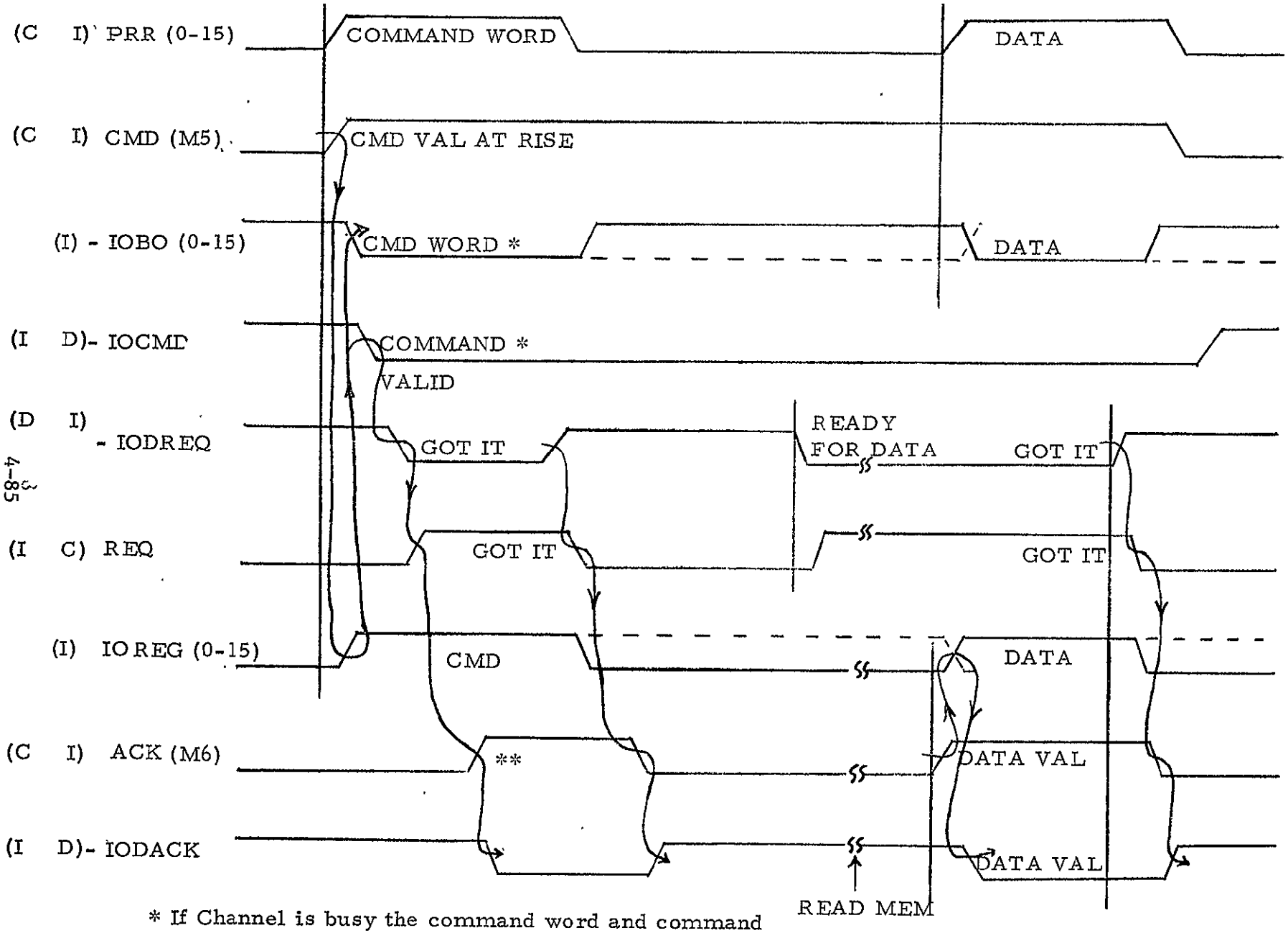


Signal Source: (D) = Device
(I) = I/O Part of SUMC-IIB
(C) = CPU

FIGURE 4.11-9

DIRECT IN SEQUENCE

* If Channel is busy the command word and command valid signals will be suppressed and the busy bit (IOTSE 0) set to one before the REQ line is raised to the CPU.

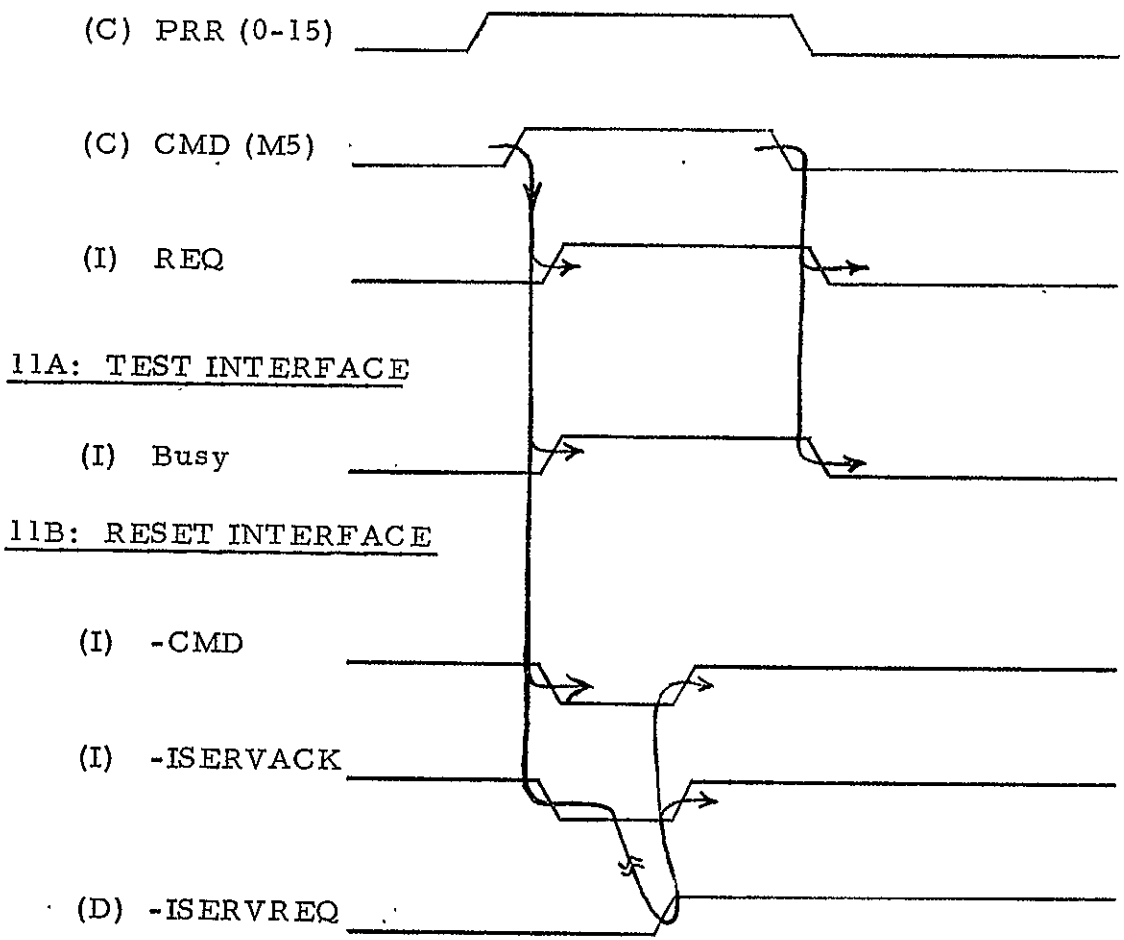


* If Channel is busy the command word and command valid signals will be suppressed and the BUSY signal set to one before the REQ line is raised to the CPU.

** If BUSY the CPU will drop CMD and terminate the sequence.

FIGURE 4.11-10

DIRECT OUT SEQUENCE



Signal Source:

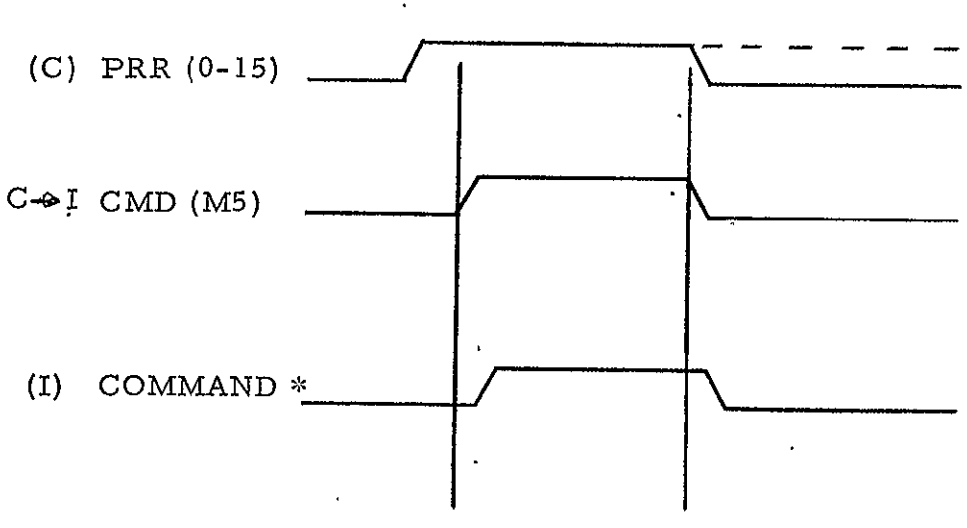
(C) = CPU

(I) = I/O Logic

(D) = Device

FIGURES 4.11-11

Test and Reset Command



* The "Command" line shows the timing relationship of the decoded PRR data for: Zero Count, Branch Out, System Reset or Set Interrupt Mask. Duration of the pulse is one microinstruction cycle.

Source:

(C) = CPU

(I) = I/O Logic

FIGURE 4.11-12

Commands Without Handshaking

Each time a Direct Input or Output instruction is executed, the I/O hardware will place a logic one or zero in bit location zero of the transfer field input to the ALU MUX depending upon whether the I/O channel is busy. A one indicates busy and the I/O sequence is terminated. The busy signal is also used for test interface.

The Direct I/O command word is also used for CPU to I/O commands. These six commands were shown in Figure 4.11-8 and are explained below.

- Reset Interface immediately halts any I/O operation and clears the I/O channel by lowering -IDCMDO while working -ISERVACK low. Both lines will be held down until all devices have reset as evidenced by no active service requests (either I/O or tester).

- Zero Count sends a zero on the -ZEROCNT line until the device currently performing I/O raises the -ISERVREQ line (DISCONNECT).

- Branch Out sends a zero on the -BRANCH OUT line for a duration determined by the microcode. The pulse is used by an IOP with hardware interrupt priority to facilitate unstacking interrupts. The command is generated by the CPU at completion of an external interrupt program. This command is not used in support of S/360.

- Test Interface tests for channel busy. A 1 in the Ø bit location of the transfer field at rise of ACK indicated busy.

- Set Interrupt Mask sets or clears a one bit interrupt mask with bit 15 of the Command Word. A one enables interrupts. This mask allows more freedom in writing the microprogram for interrupt handling (especially register save). Interrupts should be re-enabled as soon as possible because a request for another interrupt with the interrupts masked OFF will lock up the channel.

- System Reset command and MROM M5 equal a logic one activates the system reset line at the I/O interface. The reset is a negative polarity pulse with a minimum duration of 4.5 u sec presented at the I/O interface.

Electromechanical devices such as typewriters, perforated tape readers, and punches will have a special operation under Direct I/O, as shown below.

- The SUMC-IIB I/O places the command word and data word on the line normally as shown in Figure 4.11-10.

- The addressed device takes the command and data word and starts to perform the indicated operation (type a character, etc.)

- The DO sequence is terminated and the channel freed up. (All standard so far.)

(a) Programmer option: Normally during system operation, the program would go do useful work while the device is executing the command.

- When the device has completed its task and is ready for the next task (such as type another character) it will generate a standard I/O interrupt to indicate device ready.

- If the program had more tasks another DO would be generated and the sequence repeats.

- If there were no more tasks the program would not respond with another DO unless it were to turn the device off or otherwise change its mode of operation.

(b) Alternate option: Under abnormal conditions, when there is neither useful program work nor other I/O operations going on, the programmer may immediately generate another DO instruction.

- The I/O will generate the command word but since the device is not ready it will not acknowledge the command and both the channel and CPU will be hung-up until the device has finished the first task and is ready for the next.

- Then the device acknowledges the command (with -IODREQ) and the sequence proceeds.

- Rule: If the next command is waiting when the device is ready the interrupt sequence is suppressed.

The Direct IN (DI) sequence is similar to DO except that the normal sequence would be to use a DO to "start reading". When the device had the first word ready, and held in a buffer, it would generate a ready interrupt and the computer would respond with a direct in. Each time the device has a new character (word) ready a ready interrupt is generated and the computer responds with a DI instruction. The input operation is terminated either by the program not sending the DI or by sending a DO to change mode, etc.

4.11.2 Direct Memory Access (Separate Interface)

This paragraph describes the DMA as a separate interface which is considered standard. The integrated form is available as an option. This direct access to main store provides the maximum data rate transfer between the computer and device. The CPU hardware does not participate in the DMA operations, and, therefore, DMA operations can take place in the middle of an instruction.

If the CPU is not requesting memory service at the time a DMA operation takes place, there will be no impact on CPU performance. In case of conflicts between CPU and DMA, memory service is alternated so that neither can lock the other out (except for the DMA use of LOCKOUT). DMA service is device initiated by the -DMA READY signal. (See Figure 3.11-13)

On store operations the device places the address and data on their associated input buses, drops INPUT Signal and signals its validity by dropping the -DMA Ready signal. When the store operation is complete the memory control logic raises the DMACK line to signal that the data has been stored and the handshaking is completed. Reference Figure 4.11-14 for this DMA input sequence.

The DMA read is accomplished by the device placing the address of the location to be read on the address bus and signaling it validity by dropping DMA Ready signal. The memory support logic causes a memory read with the contents of the addressed location placed on the DMA data bus out. Valid data on this bus is signaled by the +DMA acknowledge line. Reference Figure 4.11-15 for this DMA output sequence.

A line named -LOCKOUT is used to keep the CPU from using memory when it might interfere with DMA operation. The device must drop -LOCKOUT before starting the memory cycle. This signal has no effect on the memory until a DMA operation is requested by the device and granted by the memory. Thereafter, the CPU will be locked out of memory (memory busy is sent to the CPU) as long as the LOCKOUT is held down. The DMA lines are identified in Table 4.11-2.

During a DMA operation the following error conditions are checked and the results indicated to the device.

- DMA Parity Error
- DMA Store Protect Error
- DMA Specification Error
- DMA Address or Specification Error

The Parity and Store Protect errors are latched in the CPU once they are detected. The Specification and Address errors are present as long as the DMA address is present on the interface.

4.11.2 I/O Loading

The following identifies the loading characteristic of the input output signals for the I/O, DMA and TSE signals.

- I/O and DMA Outputs utilize a S/N 54128W line driver.
- I/O, DMA, and TSE Data and Address inputs present an equivalent of one TTL load.
- I/O, DMA, and TSE presents an equivalent of no more than 3 TTL loads.
- TSE Outputs provide an equivalent output of a S/N 5404W.

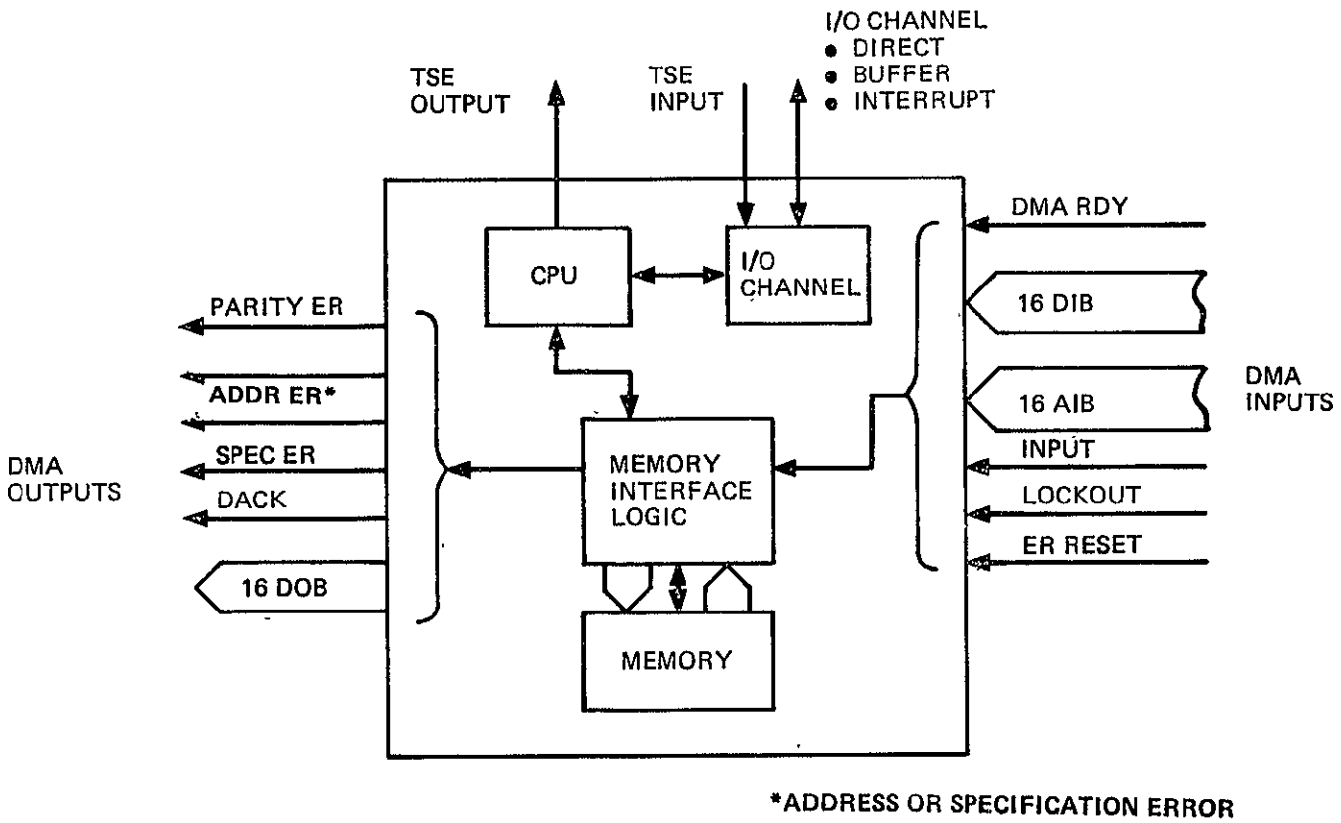


FIGURE 4.11-13. DMA BLOCK DIAGRAM

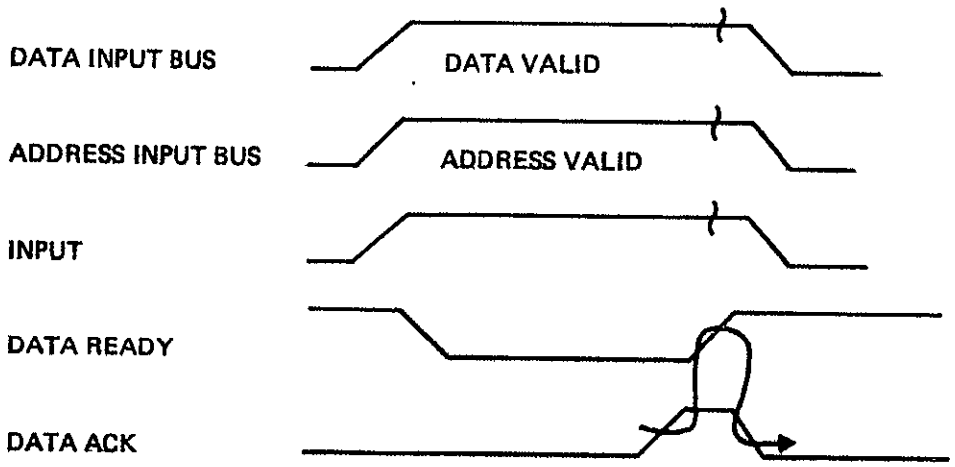


FIGURE 4.11-14. DMA STORE SEQUENCE

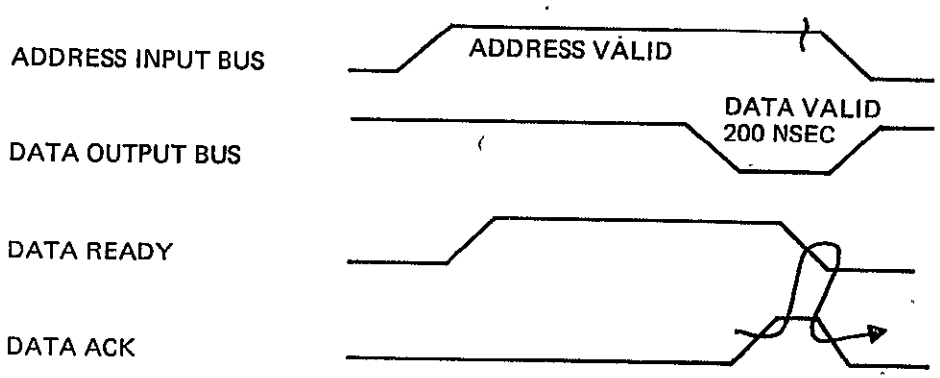


FIGURE 4.11-15. DMA READ SEQUENCE

Table 4.11-2. DMA Signal Definition

SIGNAL	DEFINITION
+ADDR (0-15)	The 16 address lines associated with a memory operation. These lines must be stable 100ns before the fall of -DMA READY.
+DATA IN (0-15) (SUMC-IIB Input)	The 16 data lines used by the controller to provide data to the SUMC-IIB for storage operations. These lines must be stable at the fall of -DMA READY.
-DATA OUT (0-15) (SUMC-IIB Output)	The 16 data lines from the SUMC-IIB to the controller for read operations. These lines will be valid at the rise of +DMACK until DMA READY is removed.
-DMA READY (SUMC-IIB Input)	The one-to-zero (high-togground) transition of this line initiates a DMA operation at completion of the current CPU memory operation. Individual line definitions define the timing relationships of data and control lines to this line.
-DMA Input (SUMC-IIB Input)	A zero on this line (100ns before fall of -DMA READY) indicates a store operation is to be performed.
-LOCKOUT (SUMC-IIB Input)	If this signal is held at ground, the SUMC-IIB will not grand CPU memory cycles at completion of the current DMA cycle; but will hold the memory for DMA use. The controller can change back and forth between read and write operations under lockout conditions.
+DMA ERROR RESET (SUMC-IIB Input)	When this line is at ground the parity and store protect error latches will be reset. This signal must have a ground condition for at least 100 ns.
+DMACK (SUMC-IIB Output)	A one-to-zero transition on this line indicates completion of the DMA cycle.
-DMA Parity Error	Indicates bad parity occurred on Memory Read.
-DMA Store Protect	Indicates memory location is protected against write operation.
-DMA Specification Error	Indicates a Byte memory operation is requested (least significant address bit = 1).
-DMA Address or Specification Error	Indicates a memory location was addressed with an address greater than memory implemented or the least significant address bit was a ONE.

Table 4.11-3. I/O Interface Line Definition

A minus (-) sign in front of the signal name indicates that a logic 1 is represented by a ground level.

SIGNAL NAME	DEFINITION
-ISERVACK	A unique line which indicates that an I/O device has control of the channel for an externally controlled I/O operation.
-TSERVACK	Like -ISERVACK except dedicated to the tester.
-IODACK	For input from an I/O device, indicates reception of data; for output to an IO device, indicates valid data on bus (used by both I/O devices and tester).
-IOCMDO	Informs device that a command is on the out bus (used by devices and tester).
-BRANCH OUT	A pulse on this line indicates that the CPU has branched from an interrupt routine (not used to support S/360 architecture).
-ZERO COUNT	A pulse on this line indicates that the current word being transferred to (from) an I/O buffer will fill (or empty) the buffer.
-IOBO (0-15)	Data Out bus. For data output I/O devices, this bus is valid during the time -IODACK is active. For command word output, this bus is valid between the activation of -IOCMDO and -IODACK.
-IOBI (0-15)	This is a time multiplexed bus used to input data, direct memory access storage address, device identity, and interrupt codes.
-INPUT	For externally controlled I/O (DMA* and Buffered I/O), activation causes an input sequence.
-DMA*	For externally controlled I/O, specifies a direct memory access operation.
-EXTINT	Specifies that the attachment is for an external interrupt
-ITINT	Like -EXTINT but dedicated to the tester.
-IODREQ	For input from a device, indicates good data on the input bus; for output to a device, requests the channel to place data on the output bus.

*When the DMA is integrated into the channel.

Table 4.11-3. I/O Interface Line Definition (Continued)

SIGNAL NAME	DEFINITION
-ISERVREQ	Unique line with which a device requests control of the channel for an externally controlled I/O operation.
-TSERVREQ	Like -ISERVREQ but dedicated to the tester.
-LOCKOUT *	When DMA service is granted the CPU is inhibited from memory service if this line is down.
-TBI (0-15)	This is an input bus used to send data and interrupt codes from the tester.
-TDREQ	For input from the tester, indicates good data on the input bus; for output to the tester, requests the channel to output data to the tester.
-TSEL	This signal switches the IO/TESTER MUX to the tester use. This line will not interrupt another I/O operation without manual intervention by the operator.
IOREQ	All I/O data, addresses and command words go through the I/O Reg except data to be stored by DMA*.
REQ	For input, this line will inform the CPU that there is good data on the input bus; for output, this line requests the CPU to place data on the output bus.
-INT	Informs the CPU that I/O service is required either for external interrupt or Buffered I/O.
CMD	Informs the tester and I/O portion that a command is to be output.
-ACK	For input from the tester or an I/O device, indicates reception of data; for output to the tester or an I/O device, indicates valid data.

* Used when the DMA is integrated in the I/O channel.

4.12 SUMC-IIB/TESTER INTERFACE

The SUMC-IIB/TSE Interface provides computer control, the capability to display various registers, the capability to utilize read/write memories to simulate the processor MROM and IROM, and a communications path between the computer and select external devices (TSE, typewriter, tape reader). This interface is depicted in Figure 4.12-1; the line descriptions are given in Table 4.12-1.

4.12.1 Computer Control

The start/stop line provides the capability to start or stop CPU operation at any given time. The TSE utilizes this start/stop line and the pressure of bit R15 to provide the single instruction mode of operation. The TSE utilizes the start/stop line and the MROM Address lines to provide the single micro instruction mode of operation.

The reset line provides the TSE with the capability of resetting the processor. A pulse on this line causes the CPU to reset the MROM address to zero. The oscillator control line provides the TSE with the capability to select either the internal or external oscillator.

4.12.2 Displays

The SUMC-IIB/TSE interface utilizes two 6-input multiplexers to multiplex out the contents of the different registers. Table 4.12-2 describes the data selected through each multiplexer and the control inputs required. These multiplexers are paired and controlled by three control lines from the TSE. The contents of the Program Counter is "trapped" from the MAR at a particular point in each instruction. The point at which the MAR contains the valid program count is identified by the Load IR left signal from the computer.

4.12.3 IROM/MROM Simulator Memory Interface

The SUMC-IIB/TSE interface provides the capability to interface with two external read/write memories. The interface will consist of the IROM and MROM addresses, the IROM and MROM data, and the MROM/IROM select lines. The sequence required for an IROM read is given in Figure 4.12-2; the sequence required for an MROM read is given in Figure 4.12-3.

4.12.4 Communications

The SUMC-IIB/TSE interface provides a Direct I/O capability for communication from the processor to the TSE, and an external interrupt capability for communication from the TSE to the processor. Three different data transfer types will be utilized for communications between the program and the TSE (including typewriter and tape reader); they are:

Table 4.12-1 Tester Interface Lines

Line Name	Description	Number	Logic
System Reset	Forces the sequencer to zero	1	+
Oscillator Control	Selects F or an external source	1	+
Clock CKX		1	-
Clock CKY		1	-
Clock CKZ		1	-
MROM Select	1 selects CPU MROM, 0 selects CSE MROM	1	
IROM Select	1 selects CPU IROM, 0 CSE IROM	1	
IROM Address		8	-
MROM Address		12	-
IROM Data		18	+
MROM Data		96	+
Mux Control	Selects TMA and TMB outputs.	3	+
TMA (0-15)	Used to gate MAR, SAR, IR, MQR, SPM, and IC	16	+
TMB (0-15)	Used to gate PRR, SDR, IR, PSW, CC, MQR16, and SPM data	16	+
MROM R15	Used to trap program counter	1	-
MROM M5 (CMD)	I/O control line	1	+
MROM M6 (ACK)	I/O control line	1	+
TBI (0-15)	TSE data out	16	
TDREQ	I/O control line	1	-
TINT	I/O control line	1	-
TSEL	I/O control line	1	-

Table 4.12-1. Tester Interface Lines (Cont'd)

Line Name	Description	Number	Logic
Connect	A negative level will indicate the Tester is connected.	1	
Start/Stop	A positive level indicates run, a negative level indicates stop.	1	
SCU Load Pulse	Used by the Tester to read data out of the MROM.		

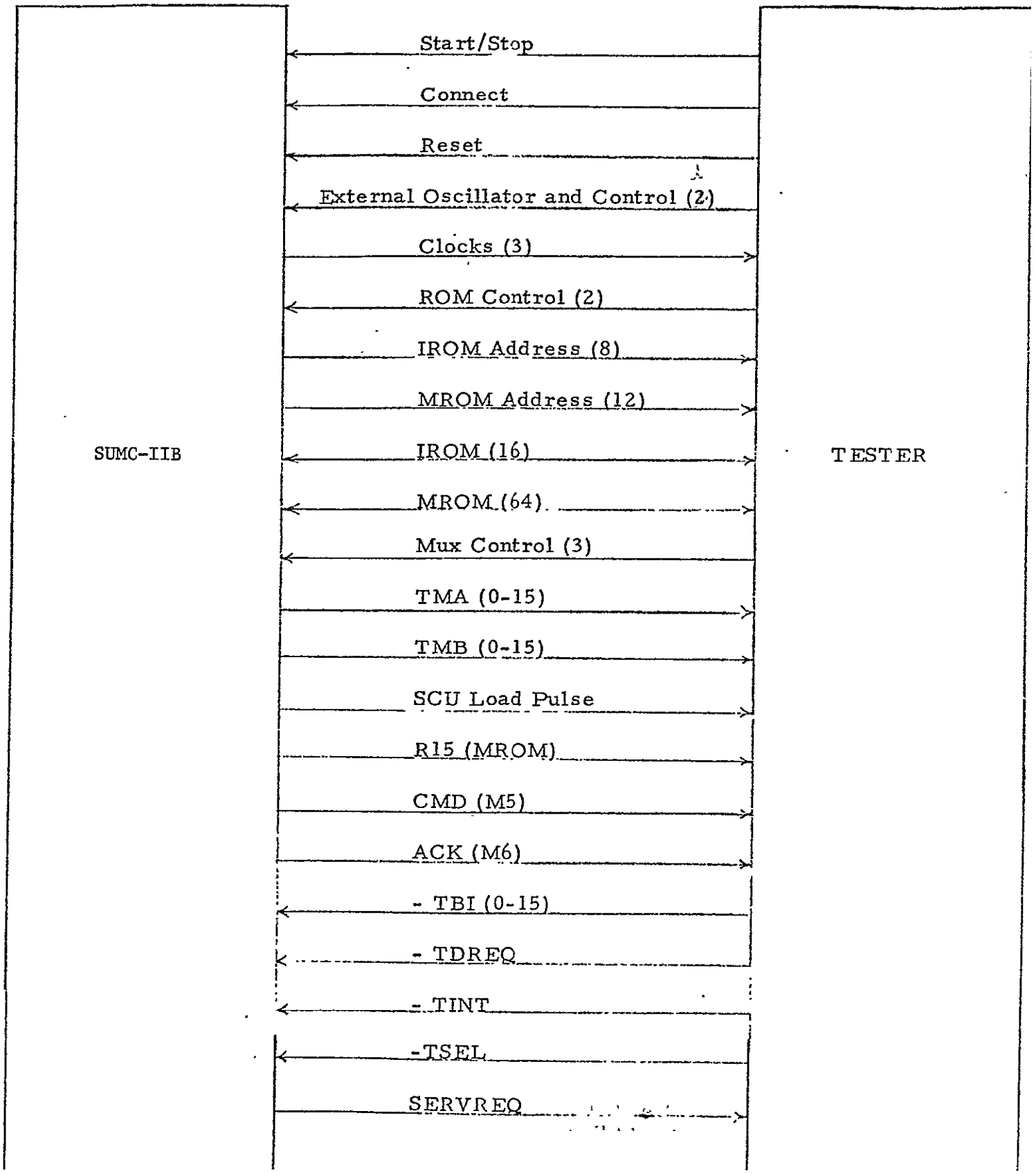
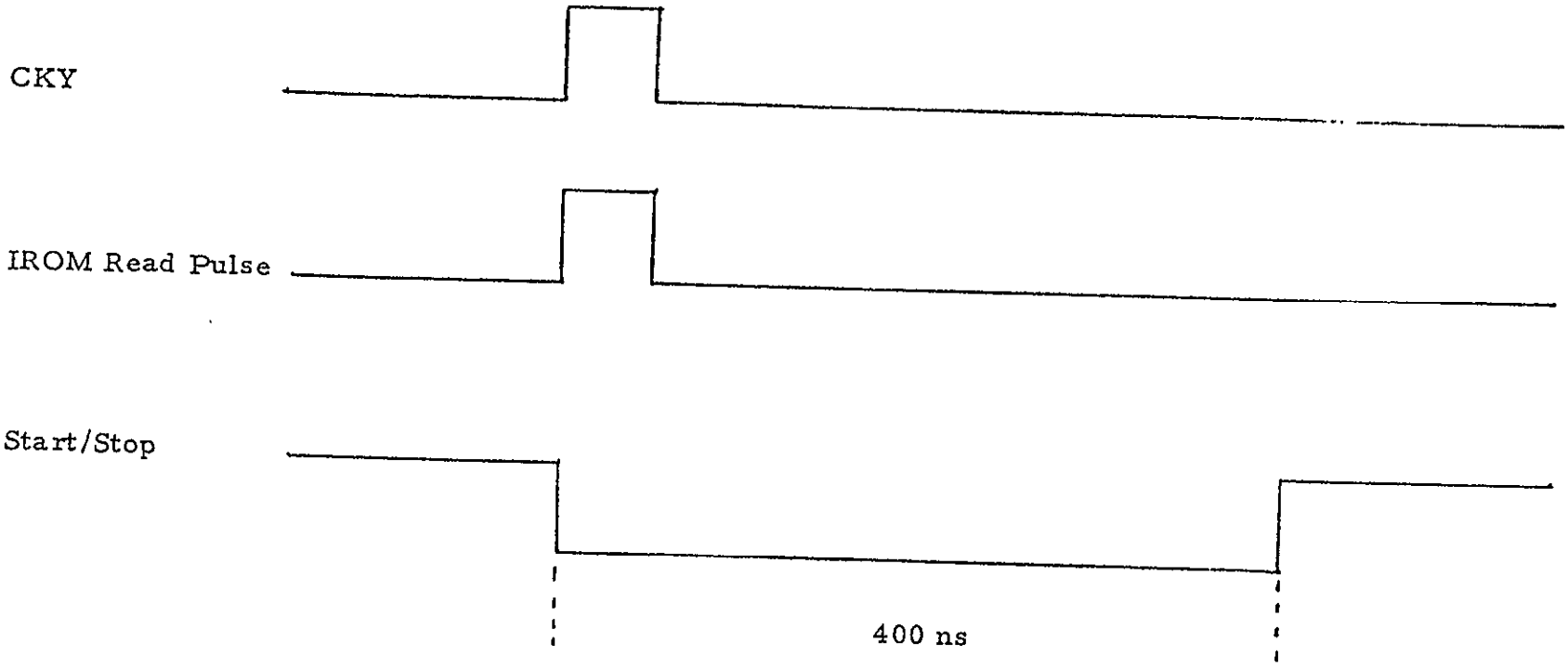


Figure 4.12-1. SUMC-IIB/TSE Interface

Table 4.12-2. Multiplexer A and B Controls

Control	Output	
	Mux A (TSE 1)	Mux B (TSE 2)
000	None	None
001	SAR	SDR
010	SPMA/IC	SPMD
011	MQR	PSW
100	MAR	PRR (CMD/DATA)
101	None	None
110	SA	PC
111	IR (LSB)	IR (MSB)



* This assumes the memory looks for a falling edge

Figure 4.12-2. TSE IROM Read

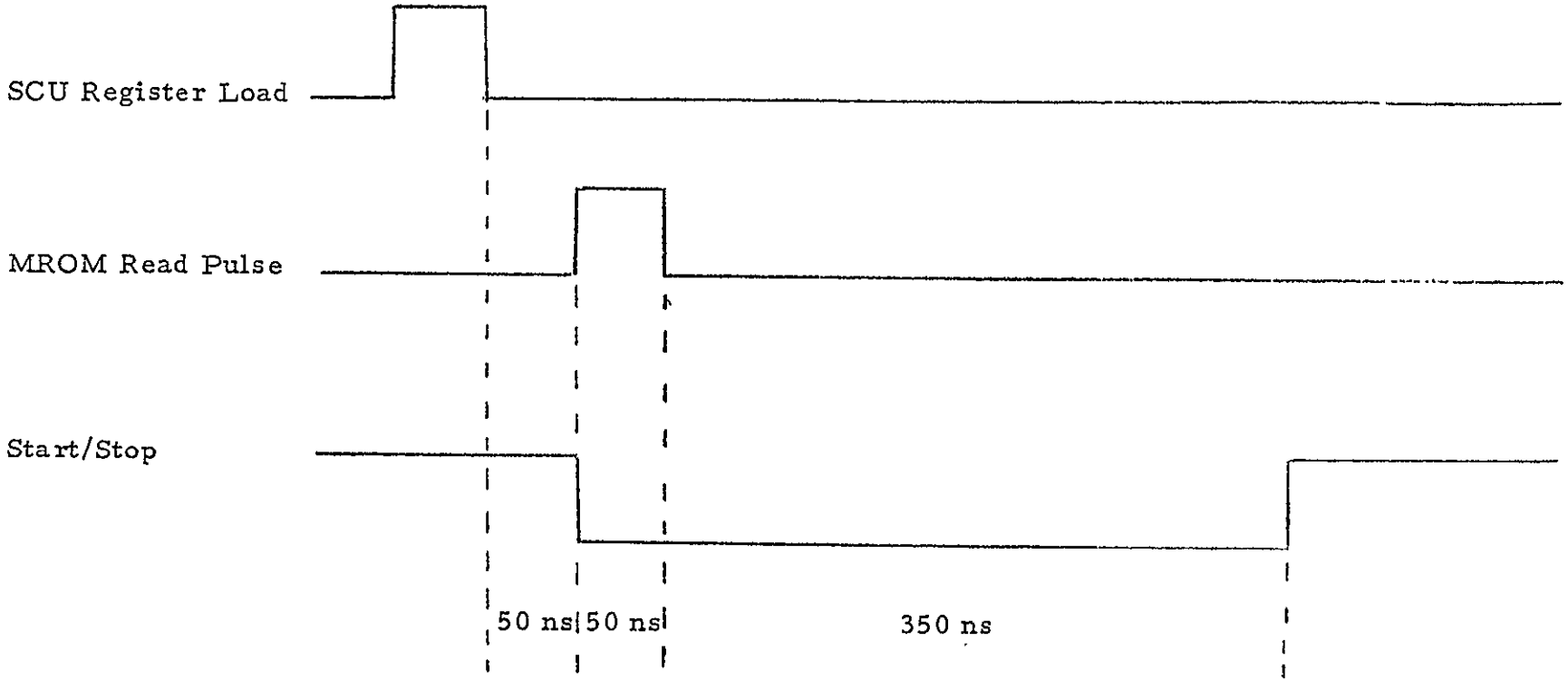


Figure 4.12-3. TSE MROM Read

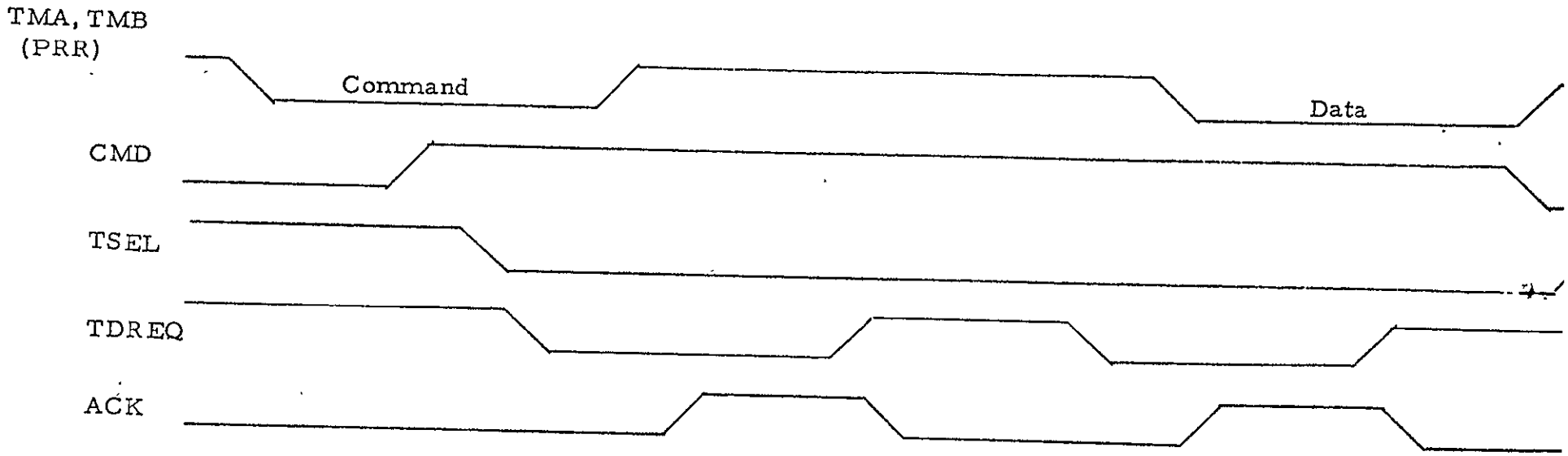


Figure 4.12-4. CPU Command/Data Out

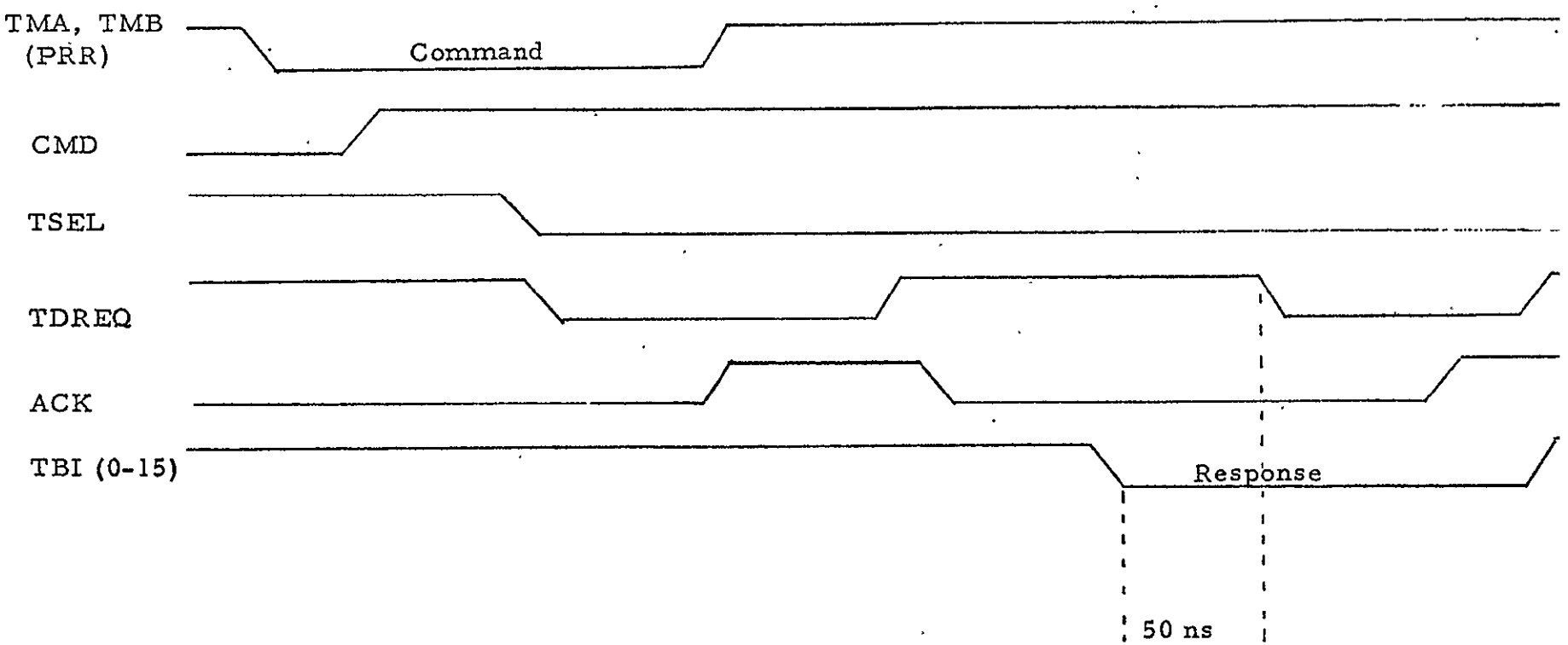
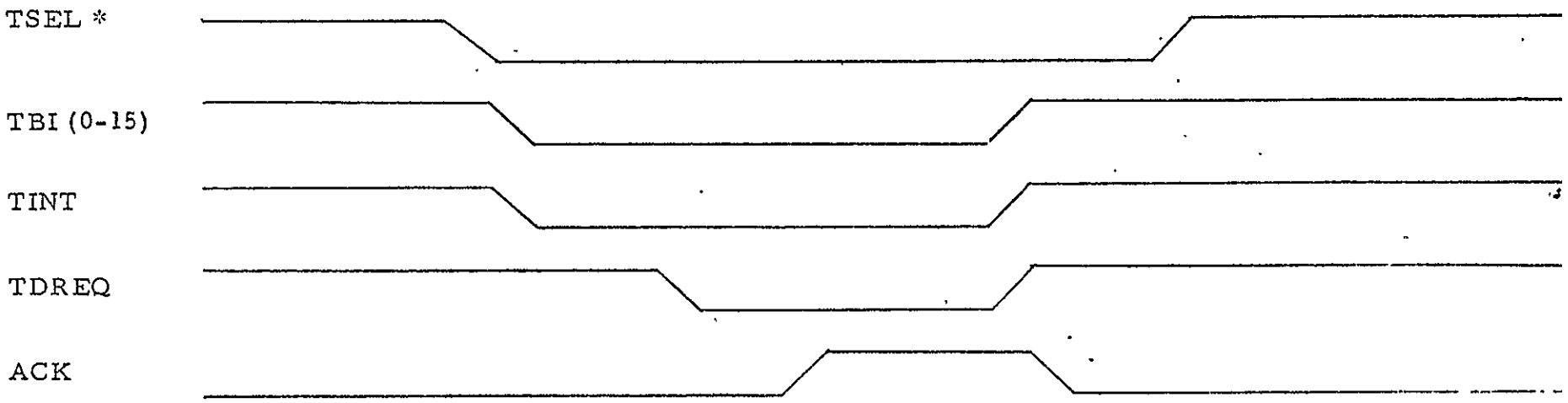


Figure 4.12-5. CPU Command Out/CSE Data Response



* TSEL must not fall unless servreq is down

Figure 4.12-6. CSE Interrupt to CPU

(1) CPU Command/Data Out

The CPU Command/Data Out transfer results in the transfer of a command word and a data word to the TSE. Both the command word and the data word will be contained in the PRR. The command data will be read out through the SUMC-IIB/TSE multiplexer under control of the TSE. The command and data will both be transferred to the TSE and the sequence completed prior to completing the command response. A sequence diagram for this transfer is shown in Figure 4.12-4.

(2) CPU Command Out/TSE Data Response

The CPU Command Out/Tester Data Response results in the transfer of a command word to the TSE and a data response word to the CPU. The command word will be contained in the PRR and will be read out through the SUMC-IIB/TSE multiplexer under control of the TSE. A sequence diagram for this transfer is shown in Figure 4.12-5.

(3) TSE Interrupt to CPU

The TSE Interrupt to CPU transfer results in the transfer of a 16 bit interrupt code to the CPU. A sequence diagram for this transfer is shown in Figure 4.12-6.

SECTION 5

SUMC-II B POWER SUPPLY FUNCTIONAL DESCRIPTION

5.1 General Description: The SUMC-II B Power Supply consists of six modules along with several transformers, discrete resistors, and capacitors packaged on a flexible multi-layer printed circuit board. The unit is capable of supplying the voltages and power levels listed below in Table 5.1-1 while operating from a 28 Vdc source in accordance with MIL-STD-704A, Category A.

Table 5.1-1. Power Supply Outputs

Type Output	Voltage	Current (Rated)	Current (Max)
Main Loop	+5 Vdc	25 Amps	28 Amps
SDR*	+8.5 Vdc	3 Amps	4 Amps
SDR*	-5 Vdc**	0.75 Amps	1.0 Amps
SDR*	-3.15 Vdc		0.06 Amps

The Power Supply will operate at efficiencies of 60% or better when delivering the above rated currents.

5.2 Functional Description: The electronics of the Power Supply are described in three parts; start circuitry, internal voltages, and main loop. These three functional areas are readily described and understood as entities. A summary following the three main functional descriptions discusses how the three elements interact. (See Figure 5.2-1)

1. Start Circuitry: Application of prime power (+28 Vdc) causes activation of the Start Regulator. The +12 Vdc output of this regulator is supplied to the Start Oscillator and transformer (T4). The ac waveform across T4 is rectified and filtered to provide + and - 16 Vdc to the Internal Voltage Circuitry. Safeguards in the Start Regulator prevent operation if the prime power input is below 19 Vdc or over 36 Vdc. An input to the Start Regulator inhibits Start Oscillator operation once the Main Loop is up and stable. Both the Start Regulator and Start Oscillator are referenced to prime power return. The + and -16 Vdc outputs are referenced to output ground.

2. Internal Voltage Circuitry: The analog and logic circuitry in the Power Supply require voltages of +12, +5, and -6 Vdc. The +12 Vdc and -6 Vdc power is developed by SDRs that are supplied their head voltages from the Start Circuitry during start and by rectified and filtered windings of the Magnetic Assembly once the Main Loop is up and stable. The internally used +5 Vdc is developed by an SDR that uses the +12 Vdc internal voltage as its head.

* - Series Dissipative Regulator
** - Not Implemented in Present Configuration

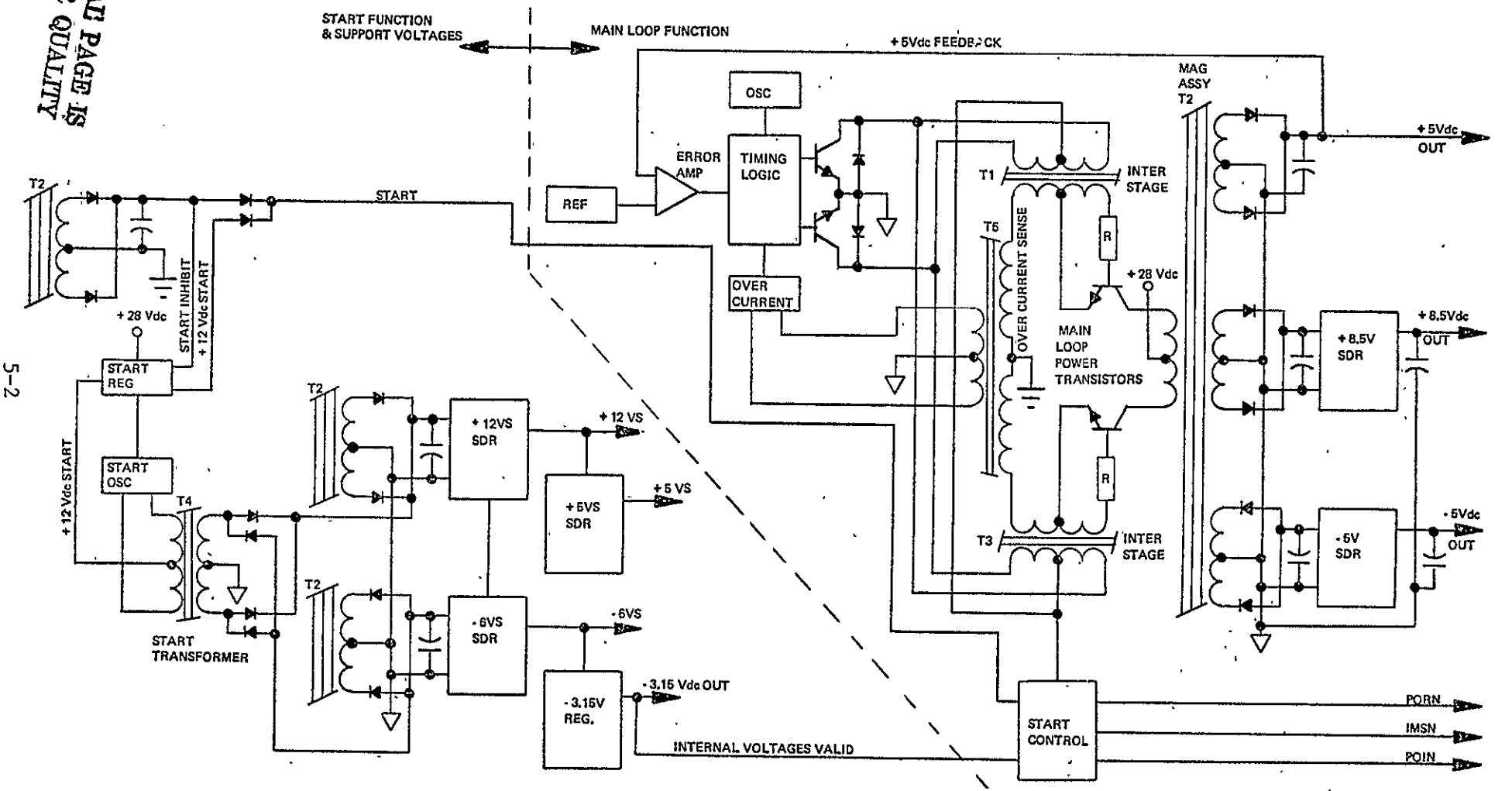


Figure 5.2-1. SUMC-IIB Power Supply Functional Diagram

One output voltage, -3.15 Vdc, because of the low current requirement (60 mA), is developed in the Internal Voltage Circuitry. A precision voltage divider is provided (between the internal -6 Vdc and output ground) with an output of -3.15 Vdc. The voltage divider output is then buffered by a unity gain operational amplifier.

The Internal Voltage Circuitry is configured such that the Main Loop is not allowed to start until the internal voltages are up and running. An inhibit from the +12 VSDR to the -6 VSDR keeps the -6 VSDR from firing until the +12 V is up. Because of the voltage divider action described above, the -3.15 Vdc will not be in spec until the -6 Vdc is up. Thus, monitoring the -3.15 Vdc output for over/under voltage insures that the +12 Vdc and -6 Vdc internal voltages are up and correct. Interestingly, the +5 Vdc need not be monitored as a safeguard since the logic in the Main Loop will not run if it is not present.

All of the elements in the Internal Voltages Circuitry are referenced to output ground.

3. Main Loop: The Main Loop of the Power Supply can best be envisioned in servo terms. There is an energy source that dumps varying amounts of energy into the Main Loop output capacitor (+5 Vdc output filter) under control of the Error Amplifier. The Error Amplifier samples the output voltage and an internal reference voltage, driving the energy source at a level that keeps the reference and output equal.

   The energy source in the Main Loop consists of the following elements:

   1) Oscillator
   2) Timing Logic (Pulse Width Modulator)
   3) Interstage Drive Transistors
   4) Interstage Transformers
   5) Main Loop Power Transistors
   6) Magnetic Assembly

   The Oscillator free runs continuously at a 25 KHz rate. Its output is developed into base drive waveforms for the two Interstage Drive Transistors. The amplitude of these base drive waveforms is

Interstage Drive Transistor Base Voltage Waveforms

constant, driving the two Interstage Drive Transistors between saturation and cut off. The resultant current flow in the Interstage Transformers causes base drive to develop in the Main Loop Power Transistors, driving them alternately from saturation to cut-off. The Power Transistors and Interstage Transformers are configured such that a bootstrap effect in the transformers forces a beta (gain) of five in the Power Transistors, rendering the circuit essentially insensitive to Power Transistor gain variations.

Each time one of the Power Transistors fires, the resultant energy in the Magnetic Assembly secondaries is rectified and stored in the output capacitors. As the output load is varied, greater or lesser amounts of energy must be dumped into the output capacitors to maintain the required +5 Vdc Main Loop output voltage. The error amplifier, responsible for controlling the output level, initiates changes in the pulse width of the base drive to the Interstage Drive Transistors. As the load increases (more current output) the pulse width is increased, transferring more energy to the output. Conversely, as less current is required, the pulses become more narrow.

Two additional secondaries on the Magnetic Assembly develop head voltages for a +8.5 Vdc and a -5 Vdc SDR.

4. Interaction: The interaction of the three functional groups discussed above is described in the following power-on sequence.

   a) +28 Vdc Prime Power is applied
   b) If the Prime Power is in spec, the Start Regulator powers the Start Oscillator
   c) The Start Oscillator provides ±16 Vdc head voltages for the Internal Voltages
   d) +12 Vdc Internal Voltage regulates and releases -6V SDR
   e) -6 Vdc SDR Regulates
   f) -3.15 Vdc Output comes in spec
   g) -3.15 Vdc valid and start from the Start Regulator cause the Start Control to release the Main Loop and begin outputting the system Timing Signals; Power On Reset Not (PORN), Inhibit Main Store Not (IMSN), and Power Off Inhibit Not (POIN).
   h) The Main Loop, +8.5 Vdc SDR, and -5 Vdc SDR regulate
   i) Secondaries on the Magnetic Assembly are rectified and filtered to provide ±16 Vdc to the Internal Voltages once the Main Loop is running.
   j) An additional secondary on the Magnetic Assembly is rectified and filtered to keep the Start line high. This level is also fed to the Start Regulator Inhibit input to shut off the Start Oscillator.

5.3 Partitioning: The descriptions above are strictly functional and give no consideration to hybrid package partitioning. Table 5.3-1 identifies each hybrid package by the functional elements it contains.

Table 5.3-1. Power Supply Partitioning

HYBRID	FUNCTIONS
Internal Voltage	Start Regulator Start Osc +12 VS SDR -6 VS SDR -3.15 Vdc Reg Part of Start Control
Pulse Width Modulator	Reference Error Amp Timing Logic Interstage Drive Transistors Over Current Part of Start Control
Series Dissipative Regulator	+8.5 Vdc SDR -5 Vdc SDR
Power Transistor	Main Loop Power Transistors
Power Diode	High Current Rectifier Diodes
24 Diode	All remaining diodes

SECTION 6

ABBREVIATIONS

A	Amperes
ACK	Acknowledge - handshaking signal
ADDR	Address
ALU	Arithmetic Logic Unit
A-MUX	Address MUX
B	Base -field of instruction
CC	Condition Code - part of PSW
CL	Carry Latch -bit in MROM word
CPU	Central Processor Unit
CTL	Control signal
D	Displacement -part of instruction
DF	Data Flow
DI	Direct Input
DIO	Direct Input or Output
DMA	Direct Memory Access
D-MUX	Data MUX
DO	Direct Output
EA	Effective Address of storage operand
EALU	Extended ALU -used for MDS operations
ES	Emulator System -Part of support software
FC	Force Carry -bit in MROM word
FCU	Function Control Unit -Controls ALU
HTC	Hybrid Technology Computer = SUMC-IIB
IC	Iteration Counter
IL	Instruction Length -Part of PSW
INT	Interrupt signal
I/O	Input or Output
IR	Instruction Register
IROM	Instruction decoding ROM
K	Thousands (K=1024 in memory address)
LSB	Least Significant Bit
LSI	Large Scale Integration
MAM	Memory Address MUX -part of DF
MAR	Memory Address REG -part of DF
MDS	Multiply, Divide or Square Root operation in ALU
MEM	Memory = Main Store
MOS	Metal Oxide Semiconductor - technology
MQM	Multiplier Quotient MUX -part of DF
MQR	Multiplier Quotient REG -part of DF
MROM	Microprogram ROM -holds control program
MS	Main Store (same as memory)
MSB	Most Significant Bit
MUX	Multiplexer

NS	Nanoseconds = 10^{-9} seconds
OFL	Overflow
OS	Operating System - supervisor program
PC	Program Counter
PCIO	Program Controlled I/O
PRM	Product Remainder MUX - part of DF
PROM	Programmable (in the field) ROM
PRR	Product Remainder REG -part of DF
PS	Power Supply
PSW	Program Status Word - current machine status
Q	Quotient
REG	Register
ROM	Read-Only-Memory
RR	Register-to-Register-Instruction Format
RS	Register-to-Storage-Instruction Format
R/W	Read Write
RX	Register Indexed - Instruction format
SAR	Storage Address Register
SDM	Storage Data Multiplexer
SDR	Storage Data Register
SEQ	Sequencer
SI	Storage Immediate - Instruction Format
SIO	Start I/O - Instruction
SIL	Storage Interface Logic
SPM	Scratch Pad Memory - Arithmetic REGs
SS	Storage-to-Storage-Instruction Format
SUMC	Space Ultrareliable Modular Computer
SUMC-IIA	SUMC technology model = HTDU
SUMC-IIB	SUMC Flight model = HTC
S/360	System/360 - IBM computer system
TMRS	Timers - Instruction
TSE	Test Support Equipment
TTL	Transistor-Transistor-Logic Technology
VDC	Volts Direct Current
W	Watts
X	Index field of instruction
XFER	Transfer
us	Microseconds = 10^{-6} seconds

APPENDIX A

APPENDIX A

SUMC-HTC MODULE AND CHIP DESCRIPTION

APPENDIX A
TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (continued)

SUMC-HTC MODULE AND CHIP DESCRIPTION

Section 1

INTRODUCTION

This appendix describes the modules and chips in the SUMC-HTC. The building blocks for HTC consist of six module types and 15 chip types.

Five of the modules are unique 148 pin multi-chip carriers utilized for specific functions. The sixth module is a two chip 100 pin carrier which brings all chip IOs to module pins. These modules are defined in subsections 2.1 through 2.5. The 15 chip types utilized by HTC are defined in subsections 3.1 through 3.15.

HTC utilizes five multi-chip carries plus eight of the two chip carries. Table 1 identifies the thirteen modules and their associated chips. Also included is module and chip quantity plus the associated IBM part numbers.

Table 1. Module and Chip Usage on SUMC-HTC

MODULE/CHIP	QUANTITY		PART No.
	MODULE/MACHINE	CHIP/MODULE	
1. DATA FLOW (148)	4		7929343-1
(a) MUX ALU		2	7928744
(b) MUX REG.		3	7928751
2. SCU (148)	1		7927334-1
(a) SEQ. MUX		3	7928747
(b) SEQ. CONTROL		1	7928748
3. TSE/SDR (148)	3		7927333-1
(a) MUX REG.		4	7928751
4. SIL (148)	1		7927338-1
(a) MUX CONTROL 1		1	7929702
(b) MUX CONTROL 2		1	7929709
(c) MEM TIMING		1	7929704
(d) MEM CONTROL		1	7929703
(e) REG.		1	7928752
5. I/O INTERFACE (148)	1		7929342-1
(a) SEQ. MUX		3	7928747
(b) AOTC		2	7929708
6. REGISTER (100)	3		7927369-1
(a) REG		2	7928752
7. FCU/EALU (100)	1		7927368-1
(a) MUX ALU		1	7928744
(b) FCU		1	7928746
8. SPM ADDRESS MUX/TIMING (100)	1		7927367-1
(a) TIMING		1	7929707
(b) MUX REG		1	7928751
9. ARCH (100)	1		7927336-1
(a) ARCH.		1	7928753
(b) REG		1	7928752
10. TIMER (100)	2		7927335-1
(a)		2	7929710
11. I/O (100)	1		7927339-1
(a) I/O		1	7929706
12. AOTC (100)	3		7927341-1
(a) AOTC		2	7929708
13. MEM SUPPORT (100)			7927337-1
(a) MEM TIMING		1	7929704

Section 2

MODULE DESCRIPTIONS

2.1 DATA FLOW MODULE

The HTC Data Flow Module consists of five logic chips packaged in a 148 pin multi-layer module. This module contains a four bit ALU, MAR, PRR and MQR. The four bit ALU is implemented with two Mux/ALU chips. The remaining three chips are Mux/Reg chips, one chip each for the MAR, PRR and MQR. Figure 1 functionally identifies the Data Flow Module.

2.1.1 ALU Mux

The Data Flow module provides a 4-bit ALU whose inputs A and B are provided from two multiplexers. Multiplexer A is a four-input mux whose control is derived from MROM bits A1, A2 and A3. This three-bit subfield selects the data which are applied to ALU input A. The combination of data which may be selected for input A are shown in Table 2.

Table 2. ALU Mux A Operation

MROM A1	MROM A2	MROM A3	ALU MXA OPERATION
0	0	0	PRR + SPMA (LOGICAL OR)
0	1	1	PRR
0	0	1	PRR + MAR (LOGICAL OR)
0	1	0	PRR + IR20-31 (LOGICAL OR)
1	0	0	SPMA
1	1	1	ZERO
1	0	1	MAR
1	1	0	IR

Multiplexer B is a three input mux with one input having the capability to be shifted right one and the MSB replaced with an external signal (SPMSG). A three bit control field MROMA5, SEL3N and SEL4N selects the ALU input B as specified by Table 3.

* ½ SPM = SPM input shifted right one with SPMSG input shifted into the MSB position.

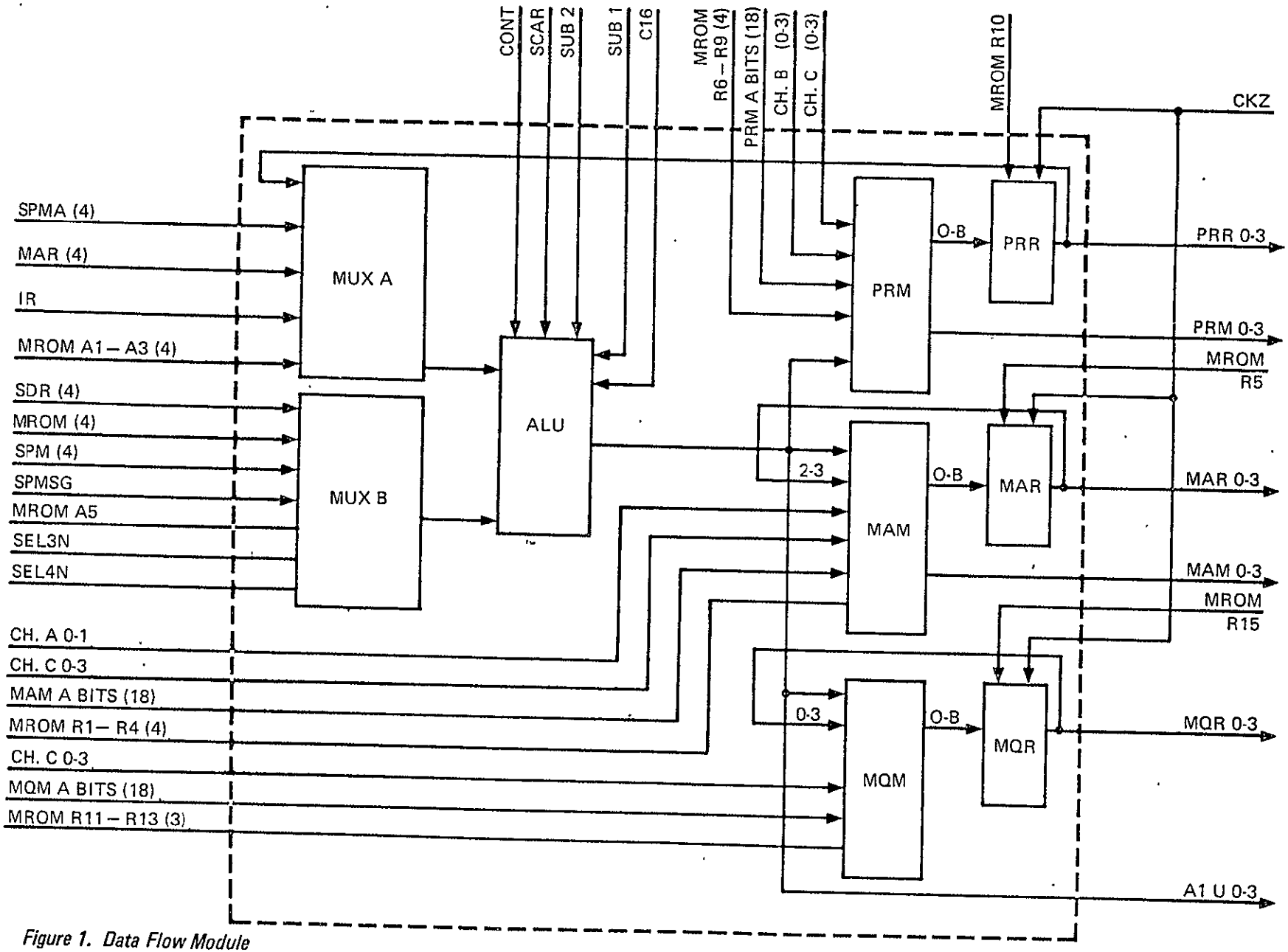


Figure 1. Data Flow Module

Table 3. ALU Mux B Operation

A5	MROM SEL3N	SEL4N	ALU MXB OPERATION
0	0	0	SDR + MROM
0	1	1	SDR
0	0	1	SDR + SPM (LOGICAL OR)
0	1	0	SDR + ½ SPM*(LOGICAL OR)
1	0	0	MROM C7-C17
1	1	1	ZERO
1	0	1	SPM
1	1	0	½ SPM*

* ½ SPM = SPM Input Shifted Right one with SPMSG input shifted into the MSB position.

2.1.2 ALU

The ALU performs the arithmetic operations specified by the five control lines as shown in Table 4.

Table 4. ALU Functions

ALU FUNCTIONS	CONT	SUB 2	SCAR	SUB1	C16*
AND	0	1	1	1	0
OR	1	0	1	0	0
XOR	0	0	1	0	0
ADD	0	0	0	0	0
SUB (A-B)	0	0	0	1	1
REV. SUB (B-A)	0	1	0	0	1

* C16 Signal is the carry into ALU 15.

2.1.3 MUX/REG

The Data Flow Module provides three 4-bit registers (PRR, MAR and MQR). Each register input is supplied by a 3 input multiplexer, with one of these inputs providing shifting capability. The MROM subfield identified below for each mux is used to control the inputs to the three respective multiplexers (Reference Tables 5,6, and 7).

Table 5. MAR Operation

MROM BITS				MAM				
R1	R2	R3	R4	0	1	2	3	OPERATION
0	0	0	0	0	0	0	0	NO SELECT
0	1	0	0	A16	A17	M2	M3	MAR
1	0	1	0	A17	M2	M3	MAMA1	LEFT 1 LOGICAL
1	1	1	0	MAMA(S)	M2	M3	MAMA1	LEFT 1 ARITHMETIC
1	0	0	1	M2	M3	MAMA1	MAMA2	LEFT 2 LOGICAL
1	1	0	1	MAMA1	MAMA2	MAMA3	MAMA4	LEFT 4 LOGICAL
1	1	1	1	MAMA(S)X	MAMA2	MAMA3	MAMA4	LEFT 4 ARITHMETIC
1	0	1	1	MAMA5	A16	A17	M2	RIGHT 1 DOUBLE
0	0	1	1	MAMA6	A16	A17	M2	RIGHT 1 LOGICAL
0	1	0	1	MAMA10	MAMA9	MAMA8	MAMA5	RIGHT 4 DOUBLE
0	0	0	1	MAMA14	MAMA13	MAMA12	MAMA6	RIGHT 4 LOGICAL
0	1	1	1	MAMA15	A16	A17	M2	RIGHT 1 LOGICAL
0	0	1	0	A17	M2	M3	MAMA16	LEFT ROTATE DOUBLE
0	1	1	0	MAMA17	MAMA18	A16	A17	RIGHT 2 DOUBLE
1	0	0	0	A0	A1	A2	A3	ALU (CH. B)
1	1	0	0	C0	C1	C2	C3	(CH. C)

A = ALU

M = MAR

MAMA = SPECIAL INPUTS FOR SHIFTING CAPABILITY

Table 6. PRR Operation

MROM BITS				PRM				OPERATION
R6	R7	R8	R9	0	1	2	3	
0	0	0	0	0	0	0	0	NO SELECT
0	1	0	0	A0	A1	A2	A3	ALU
1	0	1	0	A1	A2	A3	PRMA1	LEFT 1 LOGICAL LONG
1	1	1	0	PRMA(S)	A2	A3	PRMA1	LEFT 1 ARITHMETIC LONG
1	0	0	1	A2	A3	PRMA1	PRMA2	LEFT 2 LOGICAL LONG
1	1	0	1	PRMA1	PRMA2	PRMA3	PRMA4	LEFT 4 LOGICAL LONG
1	1	1	1	PRMA(S)X	PRMA2	PRMA3	PRMA4	LEFT 4 ARITHMETIC LONG
1	0	1	1	PRMA5	A0	A1	A2	RIGHT 1 LOGICAL
0	0	1	1	PRMA6	A0	A1	A2	RIGHT 1 ARITHMETIC
0	1	0	1	PRMA10	PRMA9	PRMA8	PRMA5	RIGHT 4 LOGICAL
0	0	0	1	PRMA14	PRMA13	PRMA12	PRMA6	RIGHT 4 ARITHMETIC
0	1	1	1	PRMA15	A0	A1	A2	RIGHT 1 ROTATE
0	0	1	0	A1	A2	A3	PRMA16	LEFT 1 LOGICAL
0	1	1	0	PRMA17	PRMA18	A0	A1	RIGHT 2 ARITHMETIC
1	0	0	0	B0	B1	B2	B3	CH. B
1	1	0	0	C0	C1	C2	C3	CH. C

A = ALU

PRMA = SPECIAL INPUTS FOR SHIFTING CAPABILITY

Table 7. MQR Operation

MROM BITS R11	R12	R13	0	1	2	3	OPERATION
0	0	0	0	0	0	0	NO SELECT
0	1	0	R0	R1	R2	R3	MQR
1	0	1	R1	R2	R3	MQMA1	LEFT 1
0	1	1	MQMA17	MAMA18	R0	R1	RIGHT 2 LOGICAL
1	0	0	A0	A1	A2	A3	ALU (CH. B)
1	1	0	C0	C1	C2	C3	CH. C

A = MLU

R = MQR

MQMA = SPECIAL INPUTS FOR SHIFTING CAPABILITY

The following MROM register subfields are used to enter data into the three registers. The MROM bit identified by Table 8 and the trailing edge of CKZ causes the respective register to be loaded.

Table 8. Register Load Controls

PRR	MROMR10 = 1
MAR	MROMR5 = 1
MQR	MROMR15 = 1

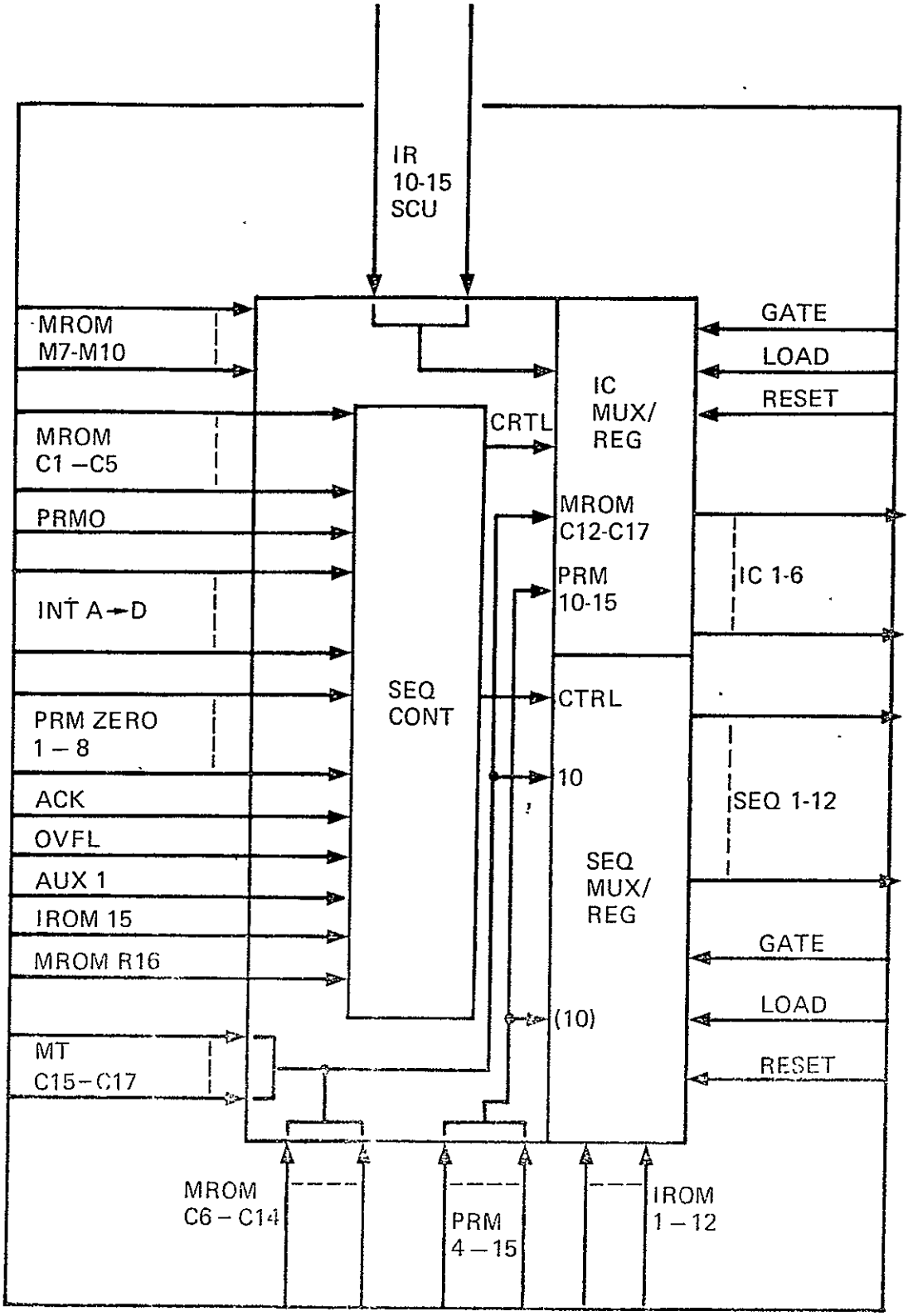
2.2 SCU MODULE

The SCU module contains one Sequence Control chip and three Sequence Mux/Reg chips interconnected on a 148 pin multi-layer substrate. This module provides for sequencing and iteration control. Figure 2 is a functional block diagram for this module.

2.2.1 SEQUENCE CONTROL

The control chip decodes five MROM bits in order to determine what action is required by the sequencer and iteration multiplexers. Table 9 specifies the MROM control bits and their associated actions. The fifth control bit MROMC5 is used to reverse the branching conditions for sequencer only i.e. for 1010 and C5=0 the sequencer will transfer the MT inputs to the sequence register if the PRM output is zero and will increment the contents of the sequence register if PRM is not equal to zero (Reference Table 9). If, however, C5=1 the transfer will be taken for PRM outputs not equal to zero and the register will increment if PRM=0.

The control chip provides the capability to determine when a MROM prefetch can occur as specified by Table 9. When the MCROM C1 thru C4 bits specify a condition in which a prefetch can occur, the appropriate action for the sequencer and iteration multipliers are taken and the SCU Register load pulse from the Timing chip loads the SCU register. This then causes the MROM to fetch the word specified by SCU register (MROM Address), thus making the MROM word available to the input of the MROM register. This completes the prefetch of MROM thus making the new word available for the next MROM cycle. For conditions which require a branch decision or a wait for data (i.e., PRM), the sequencer action is plus one anticipating the final condition. The final decision is made at the end of the microcycle prior to the loading of all data path registers. The time for this decision occurs by providing a signal from the sequence control to the timing source for stopping the clock prior to loading the data path registers. The use of prefetch is an option and is controlled by external module wiring.



C-3

Figure 2. Sequence Control Module

Table 9. Sequencer and Iteration Counter Actions

MROM C1-C5	TEST CONDITION	SEQUENCER ACTION	ITERATION COUNTER	PREFETCH ?	COMMENTS
00000	INT = 0	+1	HOLD		INT is an 'OR' of the following conditions: (1) I/O interrupt, (2) Fixed Point Overflow Error Latch, (3) Interval Timer Interrupt Latch
	INT = 1	MT → SEQ	HOLD	YES	
00001	INT = 0	MT → SEQ	HOLD		
	INT = 1	+1	HOLD		
00010	–	MT → SEQ	HOLD	YES	Unconditional Branch
00011	–	+1	HOLD		Unconditional +1
00100	REQ = 0	+1	HOLD		REQ = I/O data request line
	REQ = 1	MT → SEQ	HOLD	YES	
00101	REQ = 0	MT → SEQ	HOLD		
	REQ = 1	+1	HOLD		
00110	IC ≥ 4	HOLD	-4		
	IC < 4	MT → SEQ	HOLD	YES	
00111	IC ≥ 4	MT → SEQ	-4		
	IC < 4	HOLD	HOLD		This condition is not useful
01000	IC ≥ 4	+1	-4		
	IC < 4	MT → SEQ	HOLD	YES	
01001	IC ≥ 4	MT → SEQ	-4		
	IC < 4	-1	HOLD		
01010	–	-1	IR → IC	YES	IR = Instruction Register bits 10-15
01011	–	+1	IR → IC		
01100	IC = 0	HOLD	-1		
	IC ≠ 0	MT → SEQ	HOLD	YES	
01101	IC = 0	MT → SEQ	-1		
	IC ≠ 0	HOLD	HOLD		This condition is not useful
01110	IC = 0	+1	-1		
	IC ≠ 0	MT → SEQ	HOLD	YES	
01111	IC = 0	MT → SEQ	-1		
	IC ≠ 0	+1	HOLD		
10000	INT • IRR = 1 **	+1	HOLD		IROM bit 15 = 0 indicates RR instruction format. IROM output must be stable at the beginning of the cycle, i.e., Instruction Register Left must have been set at least two cycles earlier
	I15 • IRR = 1	MT → SEQ	HOLD	YES	
	OTHERWISE	IROM → SEQ*	HOLD		
10001	INT • IRR = 1 **	+1	HOLD		IROM output need not be stable at the beginning of the cycle, i.e., Instruction Register Left may have been loaded by the previous microword.
	I15 • IRR = 1	MT → SEQ	HOLD	NO	
	OTHERWISE	IROM → SEQ*	HOLD		* Substitutes IROM bits 13-14 for MROM bits M1-M2 for memory control. IROM bits 2-12 are gated into the sequence register.
					** INT • IRR = 1 is dominant over I15 • IRR = 1
10010	–	PRM → SEQ	HOLD		PRM bits 4-15 are gated into the sequence register
10011	–	PRM → SEQ	HOLD	NO	
10100	PRM Bit 0 = 0	MT → SEQ	HOLD		PRM positive
	PRM Bit 0 = 1	+1	HOLD	NO	PRM negative
10101	PRM Bit 0 = 0	-1	HOLD		
	PRM Bit 0 = 1	MT → SEQ	HOLD		
10110	PRM = 0	+1	HOLD		PRM bits 0-15 all 0
	PRM ≠ 0	MT → SEQ	HOLD	NO	
10111	PRM = 0	MT → SEQ	HOLD		
	PRM ≠ 0	+1	HOLD		
11000	–	+1	MT → IC	NO	MROM bits C12-C17 are gated into the iteration counter.
11001	–	+1	MT → IC		
11010	–	+1	PRM → IC	YES	PRM bits 10-15 are gated into the iteration counter. This function may also be accomplished by the MROM miscellaneous field, M7-M10 = 0111.
11011	–	+1	PRM → IC		
11100	NOT HCINT	MT → SEQ	HOLD		HCINT = hardware interval timer interrupt latch. If this latch is set, the INT input to the sequencer is also activated.
	HCINT	+1	HOLD	NO	
11101	NOT HCINT	+1	HOLD		
	HCINT	MT → SEQ	HOLD		
11110	ALU Overflow = 0	+1	HOLD		An ALU overflow is defined to have occurred if the carry-into-ALU bit 0 differs from the carry-out-of-ALU bit 0
	ALU Overflow = 1	MT → SEQ	HOLD	NO	
11111	ALU Overflow = 0	MT → SEQ	HOLD		
	ALU Overflow = 1	+1	HOLD		

I = IROM Output
MT = Modified Transfer Field - The field is modified for effective address (EA) branches (C1-CY = 0000). If an EA is not specified the MT = the C7-C17 of the MROM.

2.2.2 SEQUENCE MULTIPLEXERS/REGISTER

Two of the sequence Mux/Reg chips are used for sequence control. Each chip provides a 6 bit path. The 2 bit output of the sequencer mux/reg is utilized to provide MROM addressing capability of 4,096 words. The Sequence register load is controlled by external Gate and Load signals.

The sequence register provides a reset capability which forces the register to an all zero state.

The remaining sequence Mux/Reg chip is used for the iteration counter (IC) providing a 6 bit wide path. Like the sequence register the IC register load is controlled by external Gate and Load Signals.

The PRM inputs to the IC mux will be force selected under the conditions specified in Table 10. These conditions override those specified in Table 9 for IC action.

Table 10. IC Load from PRM

MROM				
M7	M8	M9	M10	IC-INPUTS
0	1	1	1	PRM

2.3 TSE/SDR MODULE

The TSE/SDR Module contains four Mux/Reg chips interconnected on a 148 pin multi-layer substrate. This module provides two, eight bit, six input Multiplexer/Register combinations. Figure 3 is a functional block diagram for the above module.

2.3.1 Mux/Reg

Two Mux/Reg chips are utilized for each of the 8 bit Multiplexer/Register combinations. Outputs from both the multiplexer and the register are made available from the module. Control for both multiplexers is independently derived from select Hi and Lo signals A, B, and D respectively. This three-bit field for each multiplexer selects the data which is to be transferred to the mux output and register input as specified by Table 11. The transfer of data to each register is independently controlled by input signals SEL reg and SEL Clk.

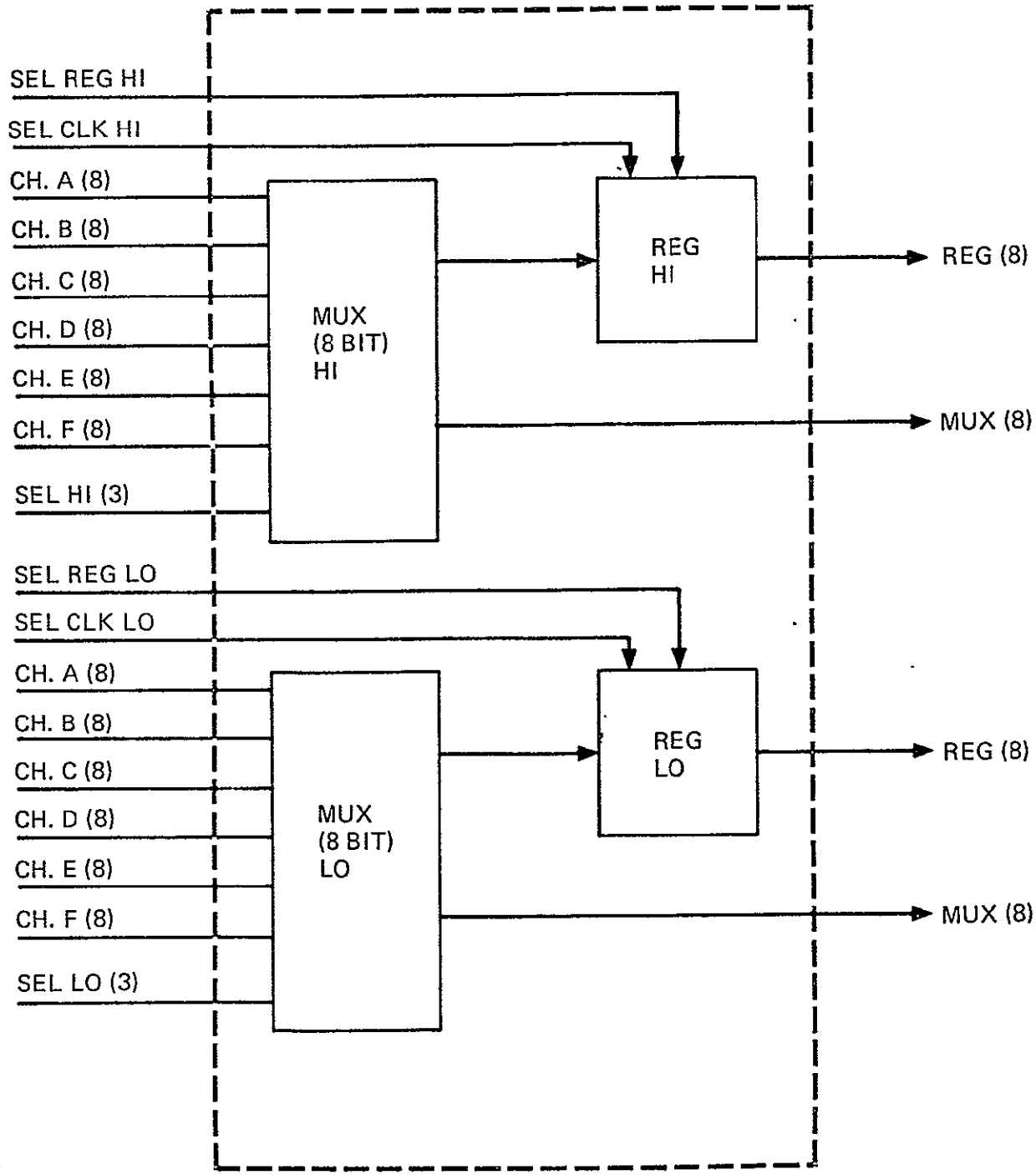


Figure 3. TSE/SDR Module

Table 11. TSE/SDR Module

A	B	D	SELECTION
0	1	0	CHANNEL A
1	0	0	CHANNEL B
1	1	0	CHANNEL C
1	1	1	CHANNEL D
0	1	1	CHANNEL E
0	0	1	CHANNEL F
0	0	0	ZERO'S

2.4 INPUT/OUTPUT INTERFACE MODULE

The Input/Output Interface (IOI) Module contains five logic chips interconnected on a 148 pin multi-layer substrate. Two AOTC chips are connected to provide a two input 18 bit multiplexer. Each AOTC chip also provides one 3 to 8 decoder. Three Sequence Mux/Reg chips provide a three input 18 bit multiplexer register combination. Figure 4 shows a functional block diagram of the IOI Module.

2.4.1 AOTC MUX

Multiplexer control of the AOTC mux is provided by SELTSE and SELIO inputs. Table 12 defines output selection with respect to the control lines.

Table 12. AOTC Mux

SELTSE	SELIO	MUX OUTPUT
0	0	ZERO
0	1	SELECT IO
1	0	SELECT TSE
1	1	IO + TSE

The two 3 to 8 decoders provide the capability as defined by Table 13.

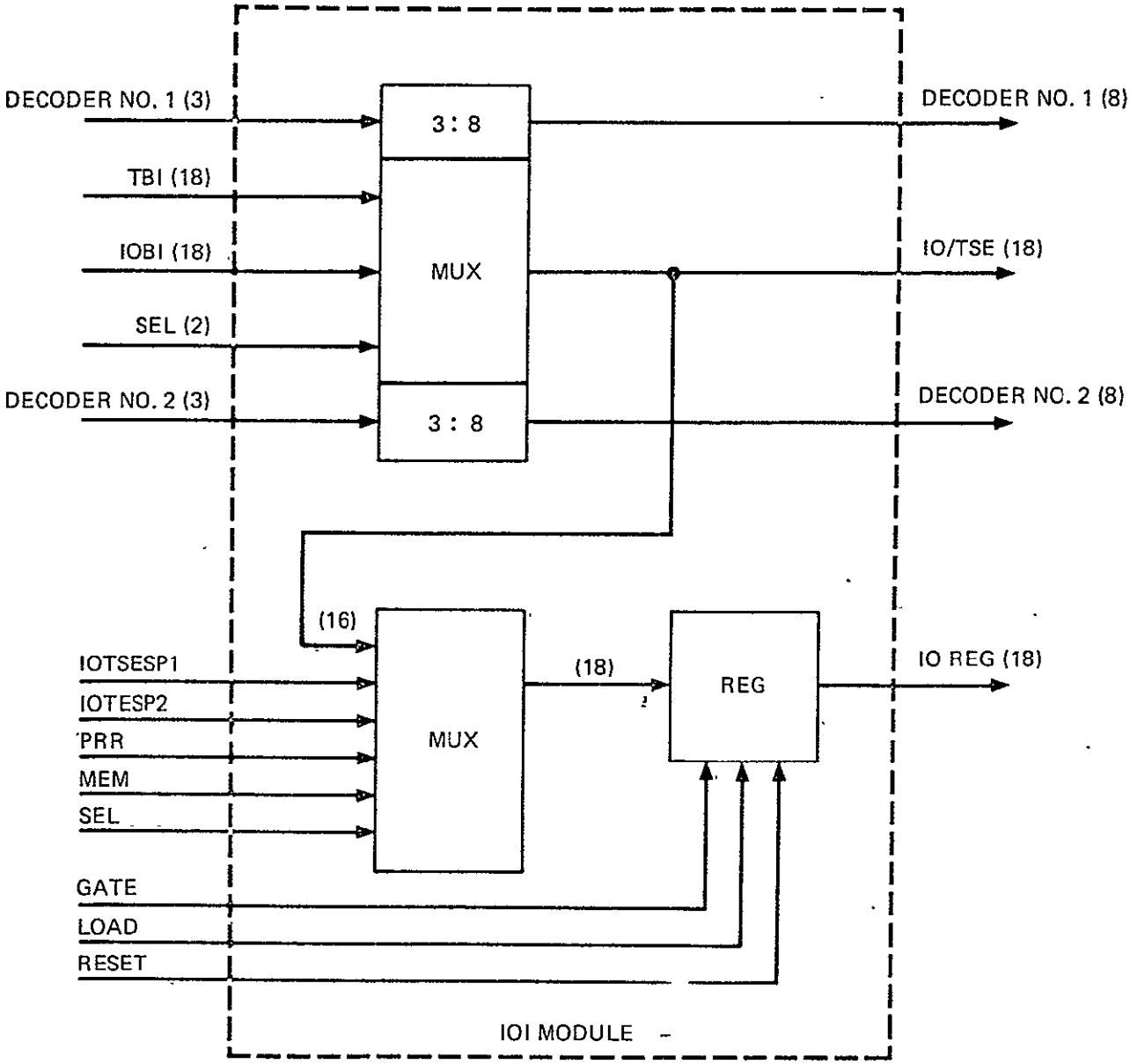


Figure 4. IOI Module

Table 13. AOTC 3 : 8 Decoder

CONTROL ENABLE	INPUT			DECODER OUTPUT							
				2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
1	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
1	0	1	0	1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	0	1	1	1	1
1	1	0	0	1	1	1	1	0	1	1	1
1	1	0	1	1	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X*	1	1	1	1	1	1	1	1

* X = Irrelevant

2.4.2 SEQUENCE MUX/REG

Multiplexer control for the sequence mux is provided by inputs SELA, SELB and SELC. These inputs select the data to be transferred to mux output as specified.

Register control is provided by inputs Gate, Load, and Reset. Table 14 specifies the conditions for which the register will load or reset.

Table 14. 101 Register Load

GATE	LOAD	RESET	OUTPUT
1	↓*	0	EQUAL INPUT
0	X†	0	NO CHANGE IN OUTPUT
X	X	1	LOGICAL ZERO

* Requires Negative Transition to Load Register

† X = Irrelevant

2.5 STORAGE INTERFACE LOGIC (SIL)

The SIL module contains five logic chips interconnected on a 148 pin multi-layer substrate. This count consists of a mux control 1, mux control 2, memory control, memory timing and Register. This module provides an interface for direct memory access (DMA) and for CPU access. The CPU interface provides full word, half word and byte operation, with the DMA utilizing only half words.

2.5.1 MUX CONTROL 1

The Mux Control 1 chip provides byte decode and enable for the memory subsystem. The chip also provides control signals required for multiplexer selection, which controls the data flow for read and write operations.

2.5.2 MUX CONTROL 2

The Mux Control 2 chip provides command decode of CPU signals, storage protect checks of CPU and DMA write operations, and control for loading the SAR and SDR. The chip also process a CPU request for a memory operation.

2.5.3 MEMORY CONTROL

The Memory Control chip provides controls for CPU or DMA use of the memory subsystem, memory start signals to the memory subsystem, DMA request/acknowledge sequence and write operation control for the SDR multiplexer. Various timed reset pulses are provided for Mux Control 1 and 2.

2.5.4 MEMORY TIMING

The Memory Timing chip as utilized by the SIL Module provides timing for the above mentioned chips.

2.5.5 REGISTER

The Register chip provides a 16 bit register which is utilized for the CPU storage address register. Register control is provided by the above mentioned chips.

Section 3

CHIP DESCRIPTIONS

3.1 MULTIPLEXER - ARITHMETIC LOGIC UNIT (MUX-ALU)

The MUX-ALU chip, part number 7928744, is a 2-bit wide functional unit consisting of two multiplexers (MUX-A and MUX-B) and an arithmetic logic unit (ALU). (Refer to Figure 1)

It has been designed to support a modular concept for which four MUX-ALU chips are interconnected to form an 8-bit MUX-ALU with look-ahead carry. The ALU has been designed to logically combine two operands A and B, producing the functions shown in Table 15.

Table 15. ALU Functions

ALU CONTROL SIGNAL VALUES					ALU FUNCTION
C	1	2	3	4	
0	0	0	0	0	A "PLUS" B
1	0	1	0	0	A "MINUS" B
1	1	0	0	0	B "MINUS" A
0	1	1	0	1	A "AND" B
0	0	0	1	1	A "OR" B
0	0	0	0	1	A "EXCLUSIVE OR" B

Table 15 also defines control signal levels used to specify each ALU function.

The multiplexers, MUX-A and MUX-B have been designed to select the source of operands A and B, respectively. Each multiplexer accepts four data input channels and three channel selection signals. Channel HE of MUX-B provides for shifting Channel E one bit position to the right. This capability facilitates multiply and divide algorithms. Table 16 defines the multiplexer selection codes.

The two bit positions of the MUX-ALU can be designated Bit 0 and Bit 1, with Bit 1 being the least significant. Two output signals provide the final ALU result for bits 0 and 1. These signals can be used as the source of the four corresponding input signals to the next (higher order) MUX-ALU chip. Table 17 defines the partial sum and partial carry signals. An ALU OVERFLOW signal indicates a change in the sign bit due to a carry. One output is the look-ahead carry which is implemented using the signals defined by Table 18.

Table 16. Multiplexer Selections

(A) MULTIPLEXER – A	
A–SELECT CODE	OPERAND A RESULT
0	(CHANNEL A) OR (CHANNEL B)*
1	(CHANNEL A) OR (CHANNEL F)
2	(CHANNEL A) OR (CHANNEL G)
3	CHANNEL A
4	CHANNEL B
5	CHANNEL F
6	CHANNEL G
7	ZERO
(B) MULTIPLEXER – B	
B–SELECT CODE	OPERAND B RESULT
0	(CHANNEL C) OR (CHANNEL D)
1	(CHANNEL C) OR (CHANNEL E)
2	(CHANNEL C) OR (CHANNEL HE)
3	CHANNEL C
4	CHANNEL D
5	CHANNEL E
6	CHANNEL HE**
7	ZERO

* The "OR" indicates that corresponding bits of the two channels are logically combined to produce an "OR" function.

**Channel HE is Channel E shifted right one bit position, i.e., one-half of Channel E.

Table 17. Partial Sum and Partial Carry Definition

(A) INPUT SIGNALS FROM LOWER ORDER CHIP
• PARTIAL SUM FROM BIT 2
• PARTIAL CARRY FROM BIT 2
• PARTIAL SUMS FROM BITS 2 AND 3 COMBINED AS A LOGICAL "AND" FUNCTION
• PARTIAL CARRY FROM BIT 3
(B) OUTPUT SIGNALS TO HIGHER ORDER CHIP
• PARTIAL SUM FROM BIT 0
• PARTIAL CARRY FROM BIT 0
• PARTIAL SUMS FROM BITS 0 AND 1 COMBINED AS A LOGICAL "AND" FUNCTION
• PARTIAL CARRY FROM BIT 1

Table 18. Look-Ahead Carry Signals

(A) INPUT SIGNALS FROM LOWER ORDER QUAD*
• X1 IS USED TO GATE THE GENERATION OF X6
• X2 IS CONNECTED TO X6 OF LOWER ORDER QUAD
• X3 IS CONNECTED TO X7 OF LOWER ORDER QUAD
• CARRY IN IS CONNECTED TO CARRY OUT OF LOWER ORDER QUAD
(B) INPUT SIGNALS TO HIGHER ORDER QUAD
• X6 IS A CARRY INTO THE NEXT HIGHER ORDER QUAD FOR THE CONDITION THAT ALL PARTIAL SUMS WITHIN THE QUAD ARE LOGIC "ONES" AND THERE IS A CARRY INTO THE QUAD
• X7 IS A CARRY INTO THE NEXT HIGHER ORDER QUAD FOR ANY OF THE REMAINING CONDITIONS WHICH SHOULD PRODUCE A CARRY FROM THE QUAD

* A QUAD is defined to mean a pair of MUX-ALU chips.

3.2 MULTIPLEXER - REGISTER (MUX-REG)

The MUX-REG chip, part number 7928751, is a 4-bit wide functional unit consisting of a multiplexer (MUX) and a register (REG). It has been designed to support a modular concept for which a larger MUX-REG, a multiple of 4-bits wide, can be implemented simply by using multiple MUX-REG chips (See Figure 1).

The MUX has been designed to select the source of data to be loaded into the REG. The MUX accepts three data input channels (A, B, and C), four channel selection signals, and eighteen special shift signals to provide the functions defined by Table 19. Channel A can be selected unaltered or with a number of shift operations performed on it. Channels B and C can only be selected unaltered.

Loading of the REG is controlled by two input signals -- a GATE signal and a CLOCK signal. If the GATE signal is a logical "zero", loading of the REG is inhibited. If the GATE signal is a logical "one", the REG will be loaded on the falling edge of the CLOCK signal.

In addition to four MUX output signals and four REG output signals, a special output signal is provided which indicates an all zero MUX output condition. When the four MUX output bits are all equal to a logical "zero" the special signal will be equal to a logical "one".

3.3 FUNCTION CONTROL UNIT (FCU)

The FCU chip, part number 7928746, provides for control of the MUX-ALU and the Extended ALU. In particular, the FCU facilitates the multiply, divide, and square root algorithms.

Table 19. Multiplexer Selections

MUX SELECT CODE	FUNCTION	MUX OUTPUT DATA BITS*			
		0	1	2	3
0	ZERO	0	0	0	0
1	ARITHMETIC SHIFT RIGHT 4	S14	S13	S12	S6
2	LOGICAL SHIFT LEFT 1	A1	A2	A3	S16
3	ARITHMETIC SHIFT RIGHT 1	S6	A0	A1	A2
4	CHANNEL A	A0	A1	A2	A3
5	LOGICAL SHIFT RIGHT 4	S10	S9	S8	S5
6	ARITHMETIC SHIFT RIGHT 2	S17	S18	A0	A1
7	LOGICAL SHIFT RIGHT 1	S15	A0	A1	A2
8	CHANNEL B	B0	B1	B2	B3
9	LOGICAL SHIFT LEFT 2	A2	A3	S1	S2
10	LOGICAL SHIFT LEFT 1	A1	A2	A3	S1
11	LOGICAL SHIFT RIGHT 1	S5	A0	A1	A2
12	CHANNEL C	C0	C1	C2	C3
13	LOGICAL SHIFT LEFT 4	S1	S2	S3	S4
14	ARITHMETIC SHIFT LEFT 1	S7	A2	A3	S1
15	ARITHMETIC SHIFT LEFT 4	S11	S2	S3	S4

* KEY: A2 = Channel A Bit 2
B1 = Channel B Bit 1
C3 = Channel C Bit 3
S9 = Special Shift Signal 9

The FCU examines MROM A6 through MROM A12 to determine the commanded ALU operation. If the commanded operation is not a Multiply, Divide or Square root (MDS), the control signals to the MUX-ALU and the Extended ALU are simple functions of the MROM signals. For MDS operations, these control signals are functions of certain parameters in addition to the MROM signals. Table 20 lists the MUX-ALU control signals which are generated by the FCU. Table 21 lists the Extended ALU control signals which are generated by the FCU. Table 22 lists the remaining FCU output signals.

3.4 ARCHITECTURE (ARCH) CHIP

The ARCH chip, part number 7928753, provides certain features which are needed for System/360 emulation. These features are Condition Code, Effective Address Branch, Exception Monitoring, and Miscellaneous Decodes.

The ARCH functions are selected by MROM M7 through MROM M10 as defined by Table 23. Figure 5 is a block diagram showing the ARCH features.

Table 20. FCU Output Signals to the MUX-ALU

CONTROL SIGNAL NAME	DESCRIPTION
C16	CARRY SIGNAL INTO ALU
SUB2	ALU CONTROL SIGNAL 1
SUB1	ALU CONTROL SIGNAL 2
CONT	ALU CONTROL SIGNAL 3
SCAR	ALU CONTROL SIGNAL 4
SEL3*	MUX-B SELECT SIGNAL 2
SEL4*	MUX-B SELECT SIGNAL 3

* Signals used by both the MUX-ALU and the Extended ALU.

Table 21. FCU Output Signals to the Extended ALU

CONTROL SIGNAL NAME	DESCRIPTION
SUB1L	EXTENDED ALU CONTROL SIGNAL 2
C(N+4)	CARRY SIGNAL INTO EXTENDED ALU
SQR	EXTENDED ALU CHANNEL E BIT 2
F25A	EXTENDED ALU CHANNEL E BIT 1
SEL3*	MUX-B SELECT SIGNAL 2
SEL4*	MUX-B SELECT SIGNAL 3

* Signals used by both the MUX-ALU and the Extended ALU.

Table 22. Miscellaneous FCU Output Signals

CONTROL SIGNAL NAME	DESCRIPTION
MQR BIT 16	EXTENSION OF MQR GENERATED BY STORING MQR BIT 14 TO SUPPORT MULTIPLY OPERATION
ALU 0 STO	ALU BIT 0 STORED
SPM 0 STO	SPM BIT 0 STORED
Q	QUOTIENT GENERATED DURING DIVIDE OPERATION
ALU BIT 18	PROVIDED TO SUPPORT SQUARE ROOT OPERATION
MQRL	MQR LOAD ENABLE SIGNAL
SPM SG	GENERATED SCRATCH PAD MEMORY SIGN SIGNAL PROVIDED TO SUPPORT SPECIAL MDS OPERATION

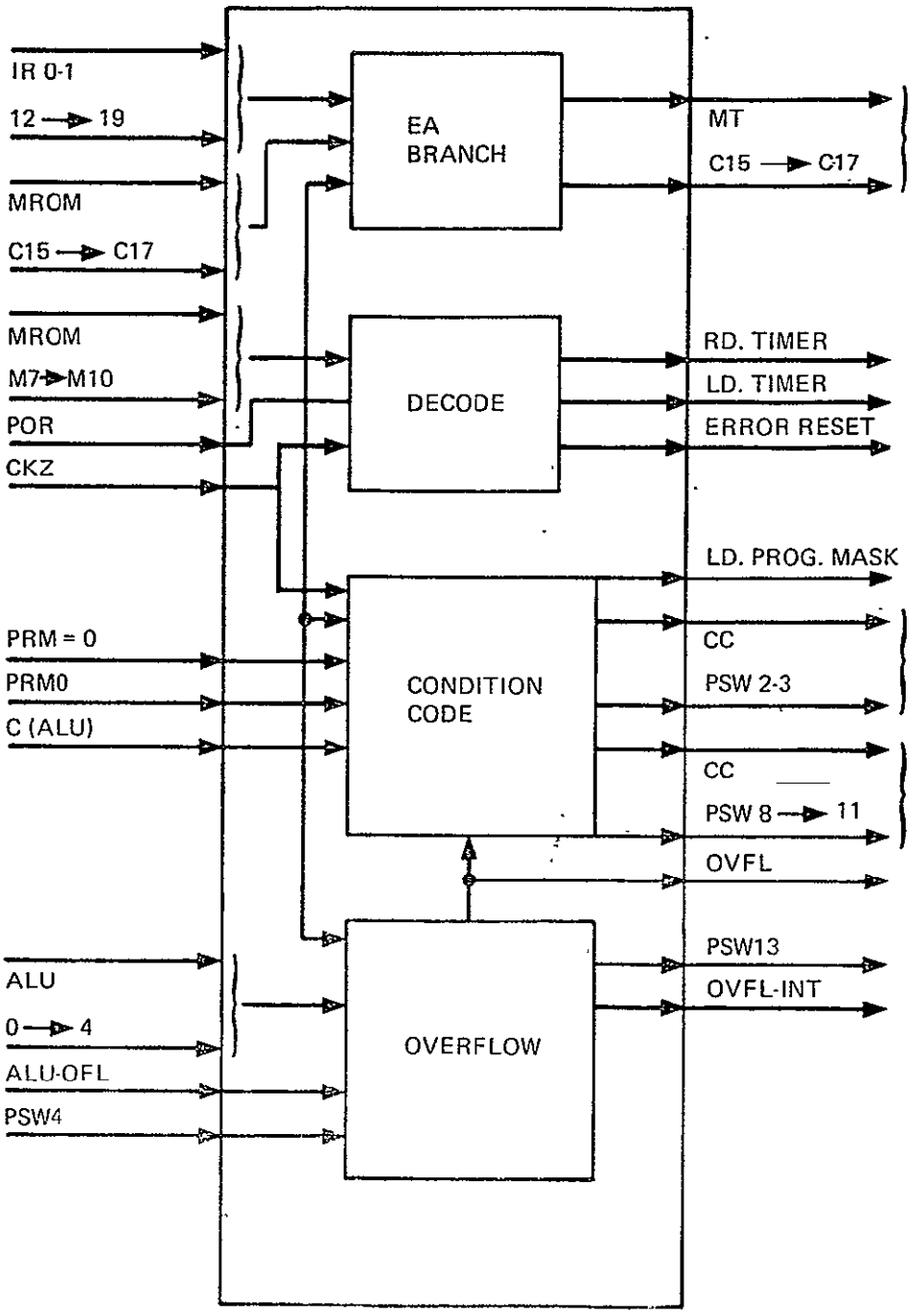


Figure 5. Architecture Chip

The Exception Monitoring feature provided by the ARCH chip consists of an Overflow Error latch (PSW BIT 13) and interrupt (PSW ERR INT). The latch is set at the fall of CLOCK Z by an overflow condition (either ALU or PRM Shift Overflow as defined for condition code generation) and the presence of PSW BIT 4. PSW BIT 4 set to a logical "zero" inhibits setting the latch. The interrupt signal is generated directly from the latch. A signal called RESET ERROR STORAGE is generated by the Reset Error function (Code 0011 from Table 23) and also by a Power On Reset Signal. This RESET ERROR STORAGE signal is used to reset the Overflow Error latch and interrupt. It is also used elsewhere in the machine to reset other error latches.

Table 23. ARCH Code Definition

MROM BITS M7	M8	M9	M10	ARCH FUNCTION
0	0	0	0	NO OPERATION
0	0	0	1	LOAD TIMER
0	0	1	0	LOAD CONDITION CODE WITH ALU BITS 2 AND 3
0	0	1	1	RESET ERRORS
0	1	0	0	EFFECTIVE ADDRESS BRANCH
0	1	0	1	READ TIMER
0	1	1	0	LOAD CONDITION CODE WITH ALU BITS 2 AND 3 AND LOAD PROGRAM MASK
0	1	1	1	NOT AN ARCH FUNCTION
1	X	X	X*	LOAD CONDITION CODE AS SPECIFIED BY TABLE 25.

* 1XXX indicates all binary code values from 1000 through 1111.

The Condition Code feature consists of a 4-valued code which is generated in the two forms shown in Table 24. The conditions which generate the code are specified in Table 25.

Table 24. Forms of Condition Code

CODE VALUE	2-BIT FORM PSW BIT		4-BIT FORM PSW BIT			
	2	3	8	9	10	11
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

Table 25. Condition Code Definition

M7	M8	M9	M10	0	1	2	3
1	0	0	0	PRM = 0, CC = 0	PRM = -	PRM = + OR CC ≠ 0,PRM ≠ -	
1	0	0	1	PRM = 0, CC = 0	PRM = -	PRM = + OR CC ≠ 0, PRM ≠ -	ALU OVFLO OR CC = 3
1	0	1	0	PRM = 0 CC = 0, NC	(PRM ≠ 0,OR CC ≠ 0) NC	PRM = 0 CC = 0, C	(PRM ≠ 0 OR CC ≠ 0), C
1	0	1	1	PRM = 0, CC = 0	PRM = -	PRM = + OR CC ≠ 0, PRM ≠ -	ALU OVFLO OR PRM SHF L4 OVFLO OR CC = 3
1	1	0	0	PRM = 0	PRM = -	PRM = +	
1	1	0	1	PRM = 0	PRM = -	PRM = +	ALU OVFLO
1	1	1	0	PRM = 0, NC	PRM ≠ 0, NC	PRM = 0, C	PRM ≠ 0, C
1	1	1	1	PRM = 0, CC = 0	PRM = -	PRM = + OR CC ≠ 0, PRM ≠ -	ALU OVFLO OR PRM SHF L1 OVFLO OR CC = 3

KEY:
C = CARRY out of ALU
NC = NO CARRY out of ALU
CC = Previous Condition Code Value
, = Comma indicates logical "AND"

Selection of the Effective Address Branch function (Code 0100 from Table 23) causes the three low-order bits of the MROM Transfer Field to be replaced by a modifier. This modifier is dependent upon the type of System/360 instruction being emulated as specified by Table 26. The ARCH utilizes IR Bits 0, 1, 2, 12, 13, 14, 15, 16, 17, 18 and 19 to generate the modifier.

Table 26. Effective Address Branch Modifier Definition

S/360 INSTRUCTION TYPE	MODIFIER
RR	110
RX WITH NO BASE AND NO INDEX	000
RS WITH NO BASE	000
SI WITH NO BASE	000
RX WITH BASE AND NO INDEX	001
RS WITH BASE	001
SI WITH BASE	001
RX WITH INDEX AND NO BASE	010
RX WITH BASE AND INDEX	011
SS WITH NO BASE	100
SS WITH BASE	101

3.5 INPUT/OUTPUT CONTROL (IOC)

The IOC chip, part number 7929706, is that integral part of the Input/Output (IO) function which generates the sequence of signals to control I/O data flow. Therefore, it must interface with a number of component functions within the computer and outside of the computer. These functions include the following:

1. Central Processing Unit (CPU)
2. Storage Interface Logic (SIL)
3. I/O Device
4. Test Support Equipment (TSE)
5. I/O Interface Module
6. Architecture Module
7. Output Drivers

Figure 6 is a block diagram of the I/O function.

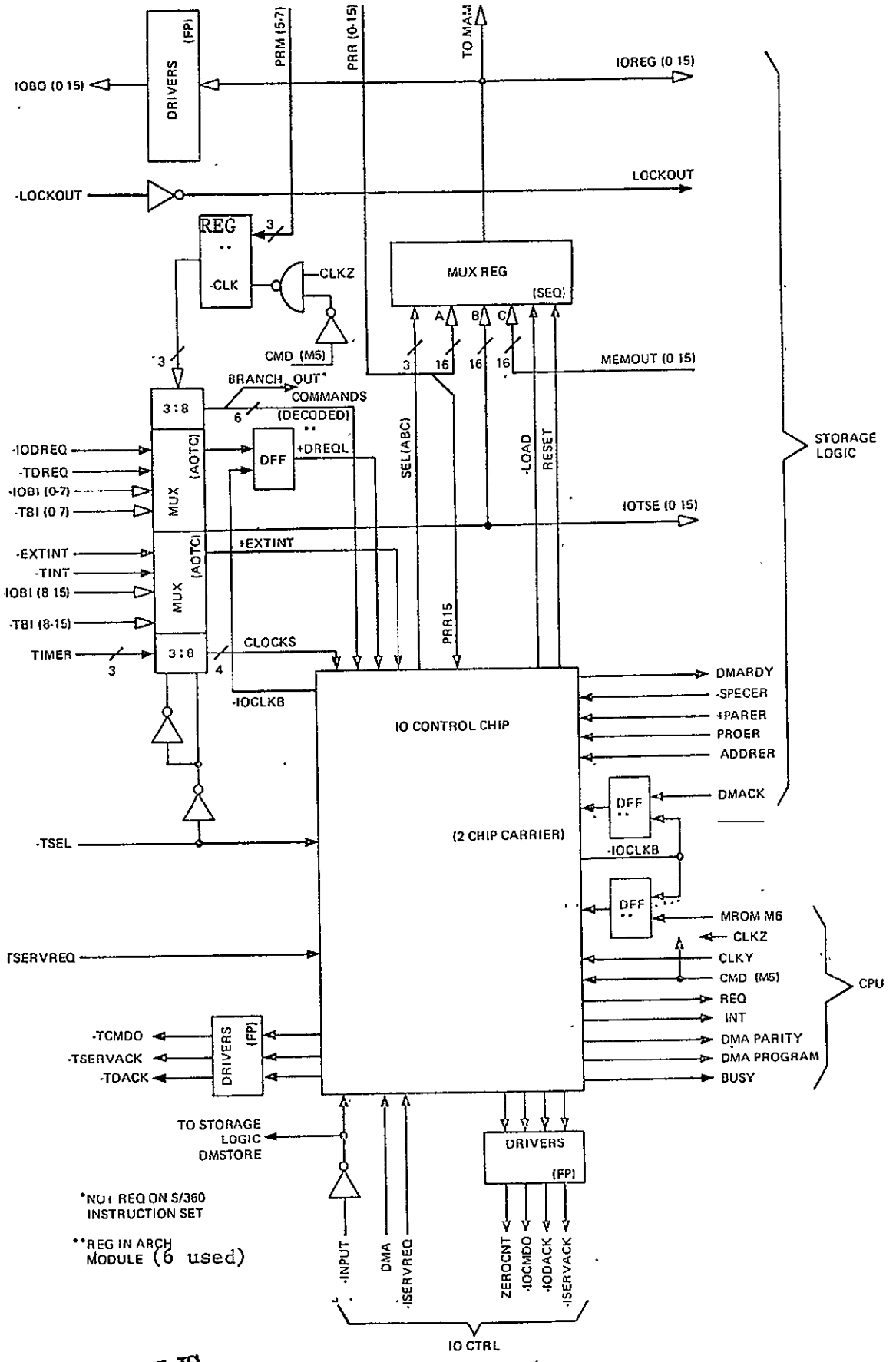


Figure 6. SUMC-IIB I/O Block Diagram (with Integrated DMA)

Four clocks generated by the I/O Interface Module are used to synchronize the I/O operations. An I/O operation is initiated by a request from one of three sources -- CPU, I/O Device, or TSE. The IOC responds to a request, if the channel is not in use, with an acknowledge signal to the requesting source. The IOC determines which I/O operation has been requested and provides the appropriate sequence of signals needed to control that operation.

Only one request can be serviced at a given time. If more than one request is received at the same time, service will arbitrarily be given to one source and withheld from the other (by the absence of its acknowledge signal) until the first has completed its operation.

As a part of the Direct Memory Access (DMA) operation, the IOC provides for DMA Error processing. Any DMA Error shall cause the following events to occur:

1. CPU Interrupt
2. Reset of I/O Register
3. Release of SIL for CPU operations
4. Suspending Release I/O Device until a reset interface command is sent by the CPU.

The IOC provides two gated output signals. One of the signals is the DMA PARITY ERROR signal gated by MROM M6. The other is a logical "OR" of the remaining three DMA Error signals (STORE PROTECT, ADDRESS, SPECIFICATION) also gated by MROM M6.

3.6 SEQUENCE CONTROL CHIP

The Sequence Control Chip is a specialized decoder chip which controls the Sequence and Iteration counters in the HTC computer.

Inputs to the chip include the C & M bits from the MROM, the contents of the Iteration Counter, and some signals controlling conditional branching.

Outputs from the chip consist of select lines to the Sequence and Iteration counter to select either HOLD, LOAD, or STEP.

3.7 SEQUENCE MUX CHIP

The Sequence Multiplexer chip is a six bit 3 input mux reg with basic arithmetic capability. It will select and store any of the 3 inputs and can increment or decrement by 1 or 4. The chip part number is 7928747.

The inputs to the chip are:

a. Three sets of six-bit data

b. Four select lines to select one of the data inputs or to place the register in an incrementing/decrementing mode.

c. Two lines to select increment/decrement by one or by four

d. Two lines to place the multiplexer in either increment or decrement mode

e. A load pulse to load the register with the new multiplexer contents.

The outputs from the chip are:

a. Six bits of data from the storage register.

b. Six bits of data from the multiplexer.

c. Carry/borrow out.

3.8 REGISTER CHIP

The Register Chip consists of four independent four bit registers and is part number 7928752 (See Figure 7).

Each register has four data inputs, four data outputs, a DC reset line, and two load lines.

There is no inversion between data in and data out. The register loads data in on the negative transition of either load line. The reset input is positive.

3.9 TIMING CHIP

The Timing Chip is basically a programmable, synchronous, four-bit Grey code counter. Each state of the counter is triggered off alternate edges of the input oscillator providing a frequency doubling effect for part of the clock cycle. Figure 8 shows the clock operation.

Inputs to the chip consist of:

a. Two oscillator sources and a select line to provide square wave frequency standard for the clock.

b. Two reset lines which reset the clock to the wait 2 state.

c. Four speed select lines which program the basic clock cycle length to either 300, 400, 500 or 600 ns with a 10 MHz frequency source.

d. Two hold lines to hold the clock in either the wait 1 or wait 2 states.

e. A conditional line which can be used to extend the normal cycle length by 300 ns.

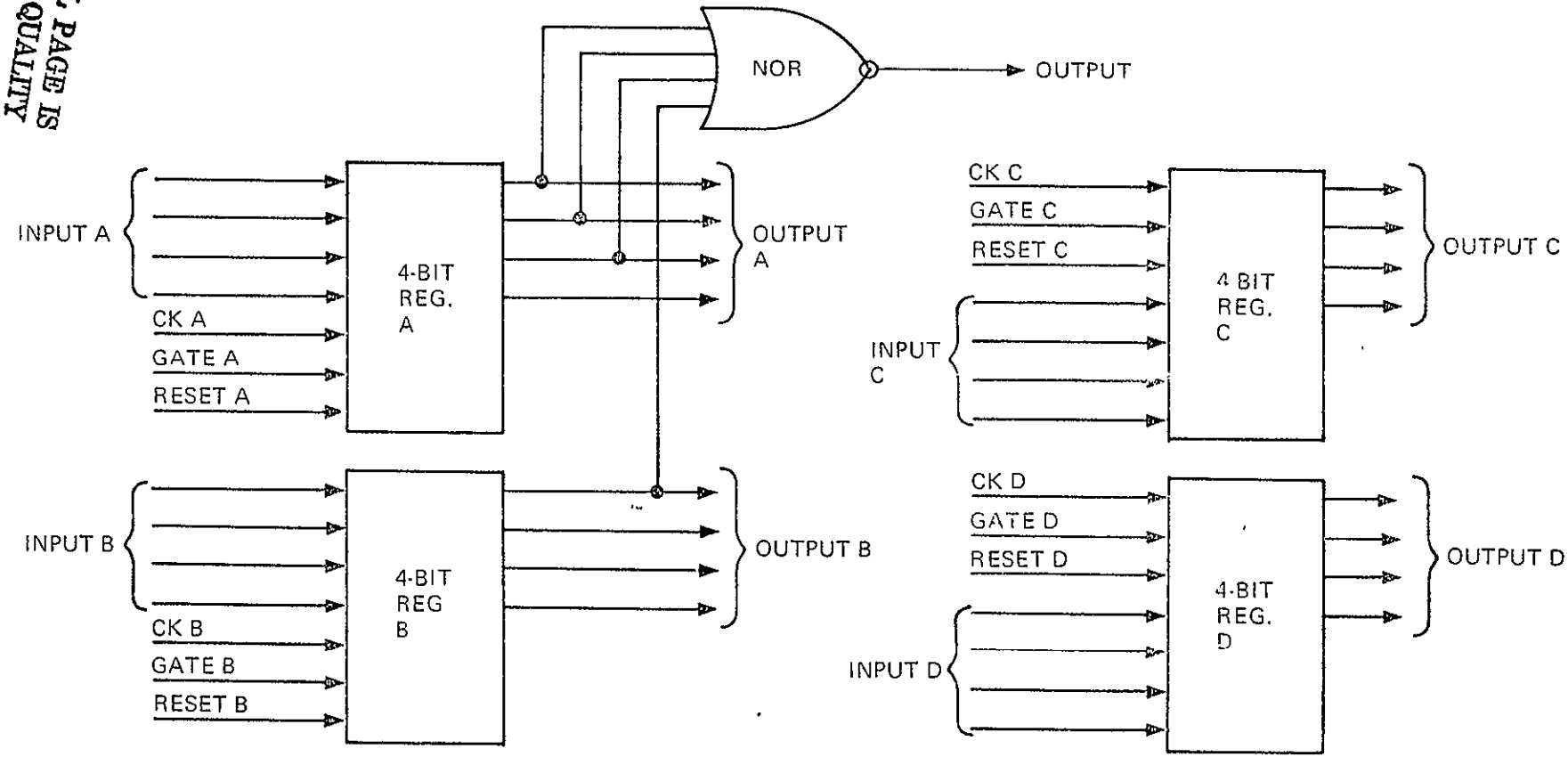
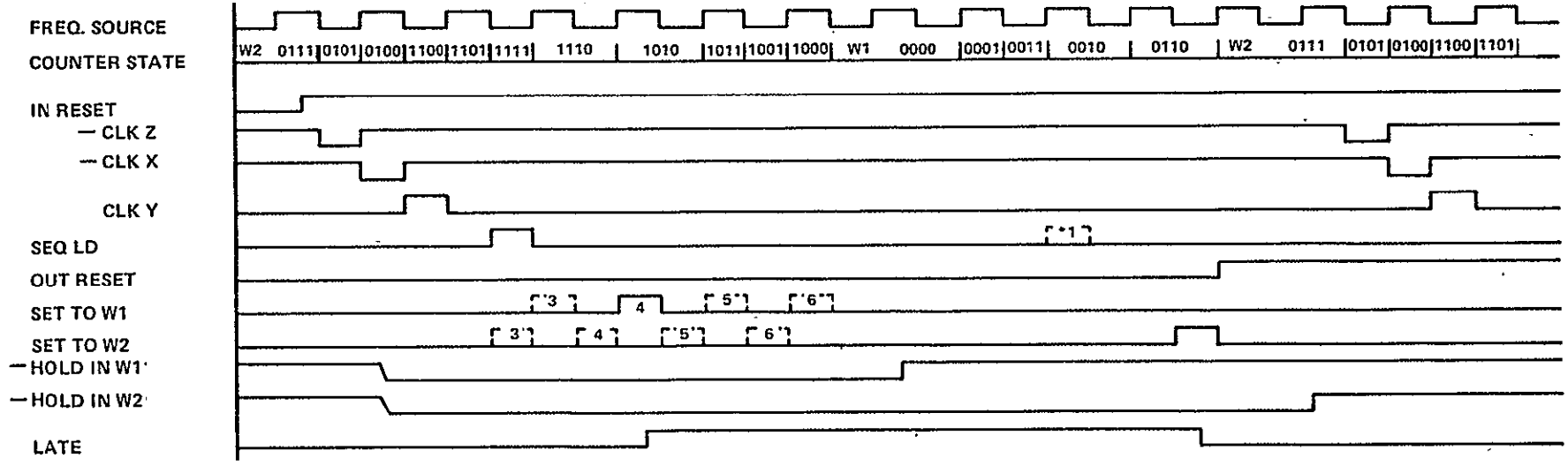


Figure 7. Register Chip



*1 IF LATE LOAD IS REQUIRED

3, 4, 5, 6 LENGTH BASIC CYCLE IN NS X 100
BASED ON 10 MHZ FREQUENCY SOURCE

Figure 8. Timing Sequence

Outputs from the chip consist of:

a. The four bits of the counter

b. Three clock lines Z, X, Y.

c. A delayed reset line triggered from the reset inputs.

d. Reset inhibit lines which are active when the clock is being held in wait 1 or wait 2.

e. A load pulse and extended cycle line used for loading the sequence counter

3.10 TIMER CHIP

The timer chip is an 8 bit ripple counter which is loadable and provides a gated output. Timer Chip part number is 7929710. Figure 9 is a functional diagram of the timer chip.

Inputs to the chip consist of:

a. Eight bits of input data

b. Two load select lines to select a six or eight bit load.

c. A two-bit code to select either load, read, or reset interrupt.

d. A clock input to gate the load and reset functions.

Output from the chip consist of:

a. Eight bits of data

b. An interrupt line which is set when the counter overflows from all 1's to zero's.

3.11 MUX CONTROL 16/32

This chip provides the following functions:

- Byte Decode
- Byte Enable to memory
- Data read and write routing
- CPU specification check
- Special one-pulse circuit to load CP SDR on write functions.

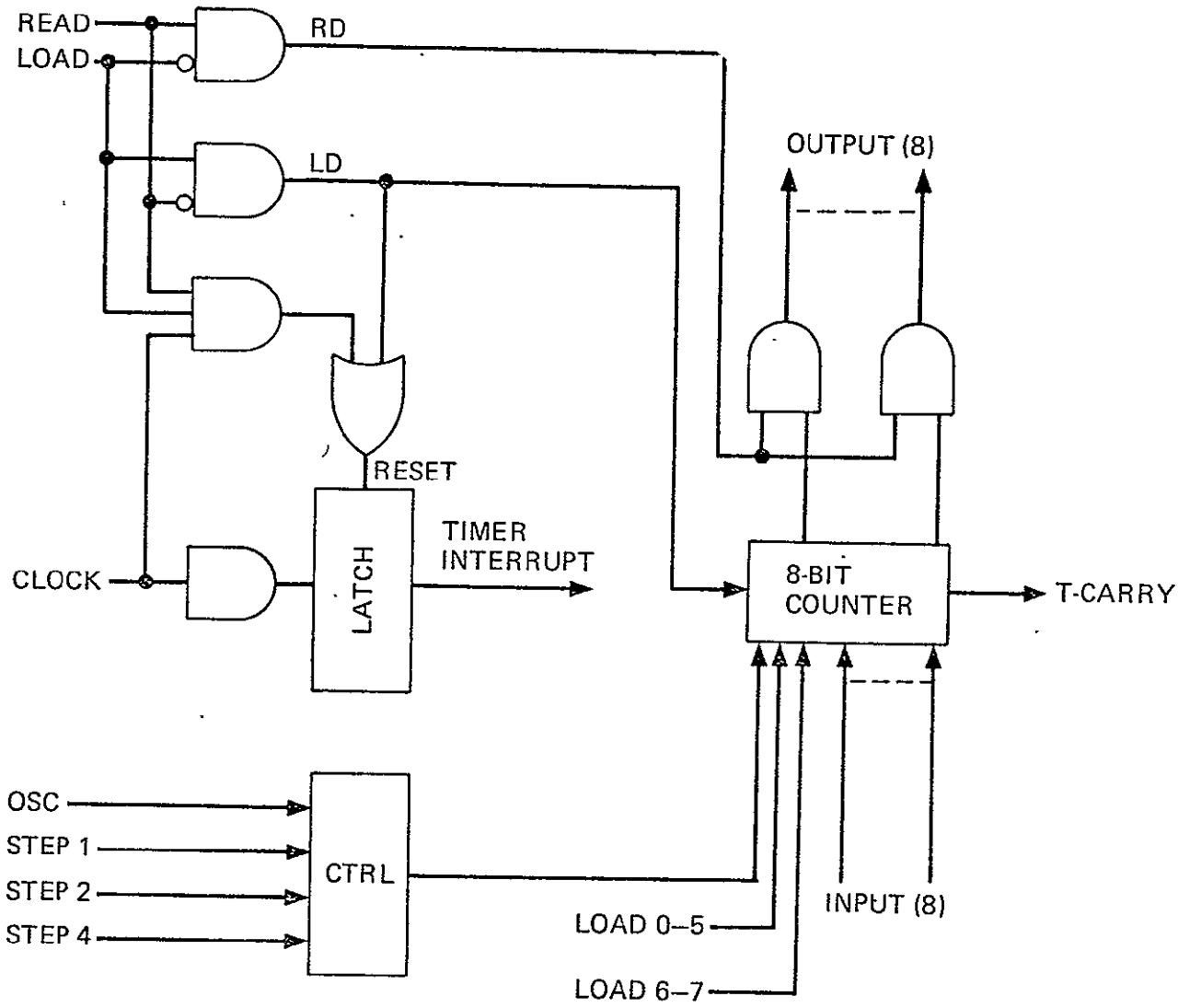


Figure 9. Timer Chip

Chip operation is explained in the following paragraph in conjunction with Figure 10.

The specification check circuitry checks that a CPU memory address is on the correct boundary for halfword and fullword operations. Table 27 represents conditions of address, command and the state of the disabled output.

Table 27. Address Functions

14 SAR	15 SAR	FULL	HALF	16	32	DISABLE (0=OK)	SAR 14
0	0	1	0	1	0	0	1
0	1	1	0	1	0	1	1
1	0	1	0	1	0	0	1
1	1	1	0	1	1	1	1
0	0	0	1	1	0	0	0
0	1	0	1	1	0	1	0
1	0	0	1	1	0	0	1
1	1	0	1	1	0	1	1
0	0	1	0	0	1	0	0
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	1
1	1	1	0	0	1	1	1

Note that when FULLWORD and 16 BIT lines are both 1, 14 SAR (Storage Address Register 14) is changed to a 1 if the line was a zero. This operation does not exist for the 32 bit configuration. This configuration of 14 SAR to 1 on full and 16 bit is required since the memory module is 32 bits wide and only 16 bits can be handled by the CPU.

Signals 14 SAR and 15 SAR are now decoded to provide four mutually exclusive outputs (Byte 1 to 4). The decoder is inhibited to provide no output if there is a specification error.

The outputs of the byte decoder are ended with the CPU commands (FULL WORD, HALF WORD, BYTE) and two control lines CPU GOT MEM and DMA GOT MEM to provide the Byte enable signals which to to the memory access 1, 2, or 4 bytes of storage. Table 28 shows the valid combination of inputs which provide the enable byte signals. Note in Table 28 that the DMA accesses Halfwords in the 16 bit CPU and only full words in the 32 bit CPU. With Byte 0 in (spec error) no enable outputs are generated.

The multiplexer data path latches are set to select various mux inputs depending on CPU operations. This control logic has been specifically designed to provide correct data positioning for byte read and write with

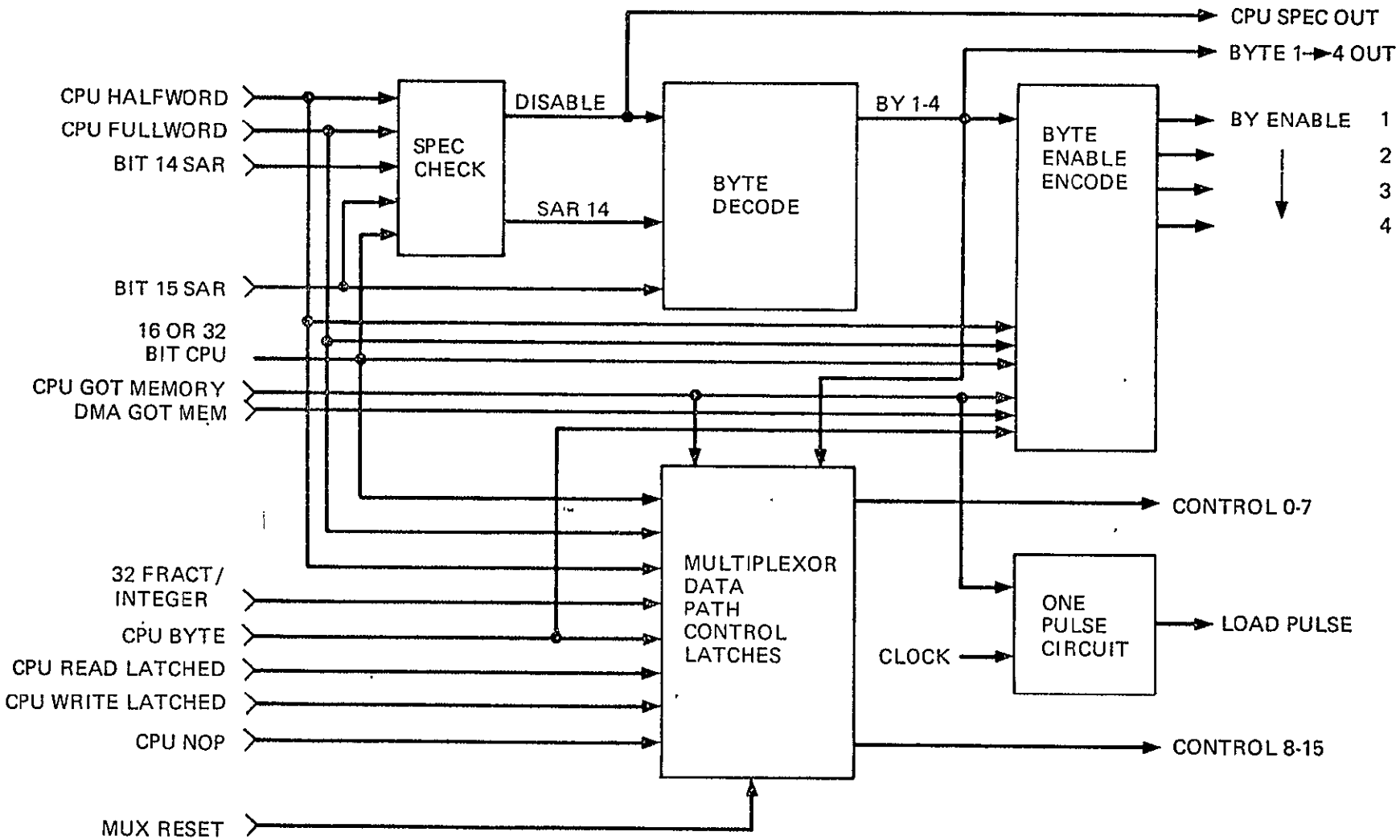


Figure 10. Mux Control Chip

Table 28. Byte Signal Generation

16	32	BYTE	FULL	HALF	CPU Got	DMA Got	BYTE 1, 2, 3, 4	ENABLE by 1, 2, 3, 4
1	0	1	0	0	1	0	1 or 2 or 3 or 4	1 or 2 or 3 or 4
0	1	1	0	0	1	0	1 or 2 or 3 or 4	1 or 2 or 3 or 4
1	0	0	1	0	1	0	3	3, 4
1	0	0	0	1	1	0	1	1, 2
1	0	0	0	1	1	0	3	3, 4
0	1	0	0	1	1	0	1	1, 2
0	1	0	0	1	1	0	3	3, 4
0	1	0	1	0	1	0	1	1, 2, 3, 4
1	0	0	0	0	0	1	1	1, 2
1	0	0	0	0	0	1	3	3, 4
0	1	0	0	0	0	1	1	1, 2, 3, 4
X	X	X	X	X	X	X	0	0

a 16 or 32 bit CPU and halfword read and write with a 32 bit fractional or integer CPU. The mux control signals are designed to position data properly when used with microcode instructions for an IBM/360 type computer. The 32 bit fractional CPU capability is added to allow correct operation with CP-2 type microcode instructions (or any other 32 fractional CPU). For S/360 type (integer) CPUs the data path always right justifies bytes or halfwords on read operations no matter where data are fetched from storage. On write operation data are presented to the mux right justified and the mux control positions the data for byte or halfword store operations to the correct position for the main storage system.

3.12 MUX CONTROL NO. 2

This chip performs the following functions:

- Command decode of CPU signals
- Storage Protect Checks of CPU write and read modify write
- Storage Protect Check of DMA write
- Loading CP SAR
- Loading CP SDR
- Special CPU Operations
- CPU request for memory generation

The chip is shown in block form in Figure 11.

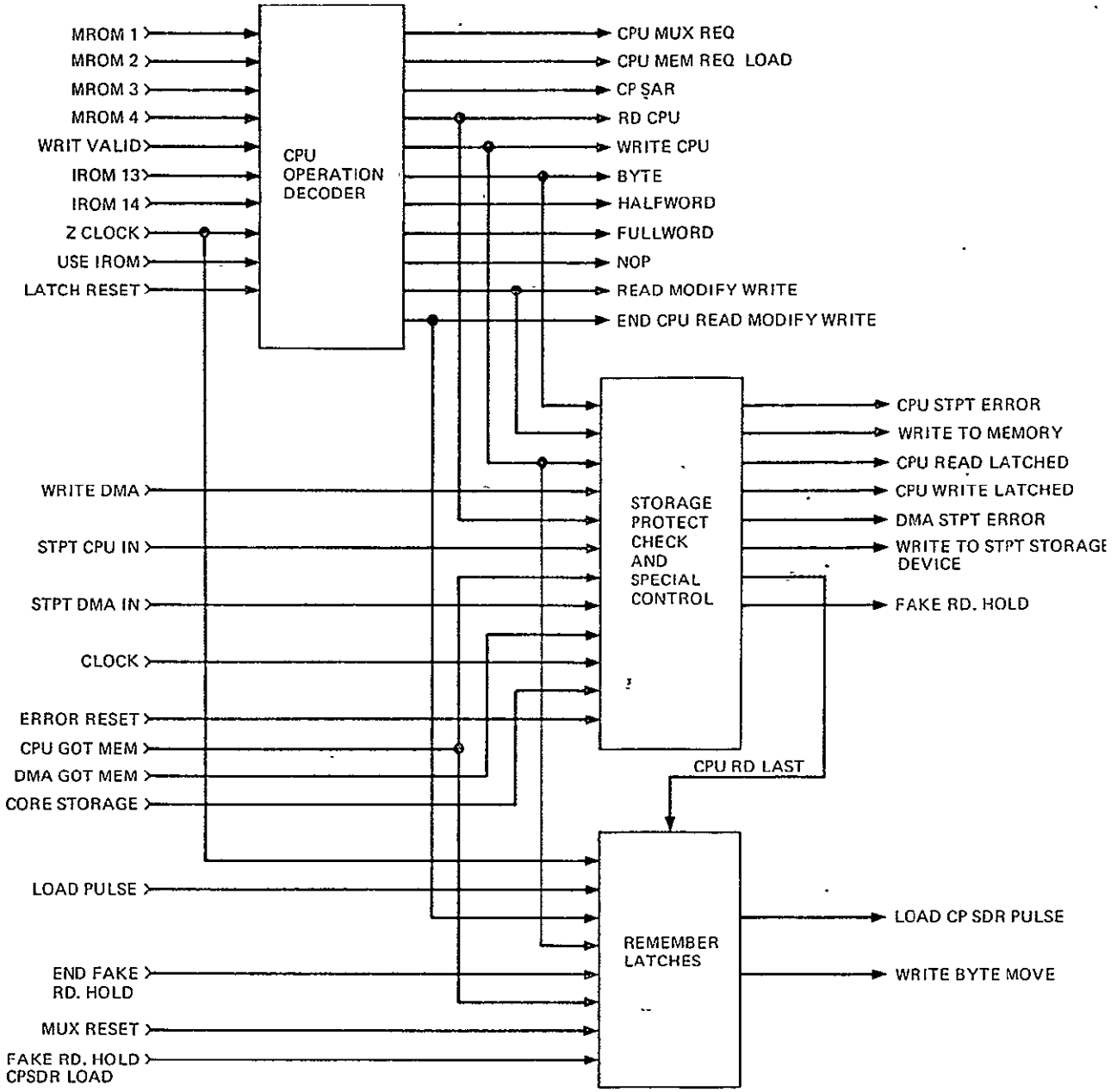


Figure 11. Mux Control Chip No. 2

All CPU operations are the result of storing MROM 1 to 4 or IROM 13 and 14 MROM.1 and 2 in latches at "Z" time and then decoding the output of the latches. The CPU MEM REQ line is activated by and CPU operation requiring access to main storage (Read, Write, Read and Hold). The CPU MUX REQ is the OR of CPU MEM REQ and two special CPU operations; Write NOP and Write More NOP. The WRITENOP function loads the CPSDR with 16 or 32 bits of data from the PRR and requires no main store cycle. The Write None NOP is used to load storage protect information into a 16 x 2 bit memory from the PRR bus.

The storage protect feature allows different protect "keys" for the CPU and DMA. The presence of a storage protect bit is checked on all Write operations and on Read and Hold and Write Move (NOP) operations. A storage protect violation results in setting the appropriate error latch (CPU or DMA) and changing the Write to Memory line to the read polarity.

All CPU operations are started at approximately the fall of Z clock and are terminated before the next Z clock. Since the MROM or IROM bits are valid until the fall of Z, the CPU operation latches are reset before the rise of Z. On a CPU read operation the memory data is present at the rise of Z and the CPSDR load pulse is generated by "anding" CPU Rd Last and Z.

To provide the ability to support core memory or memory which is not accessible in 16 or 32 bit data widths, the special control section contains provision for generation of a "FAKE" Read and Hold followed by Write when the CPU requests a Write-Byte or Write-Move-Byte or Read•Hold-Byte.

The Remember Latches keep track of the "FAKE" operations to provide load CP SDR pulses at the proper time to insure that all data of interest is retained for storing in the memory.

The CPU Read•Hold is checked for storage protect violation because a core memory Read-Modify-Write cycle must be terminated with Write.

3.13 MEMORY CONTROL (MEM CTRL)

This chip performs the following functions:

- Select CPU or DMA operation
- Check Various CPU and DMA errors
- Provide special reset pulses for CPU operation
- Start Memory and CPU memory busy
- DMA request/acknowledge sequence
- Priviledged operation lockout of other Port
- Write DATA SDR Mux control

Refer to Figure 12 for chip block diagram.

This chip provides two basic functions, which are:

1. Selection of who (CPU or DMA) is to use main storage and

2. Generate various timed reset pulses for MUX CTRL 16/32 MUX CTRL2.

The who got and priority logic decide which port (CPU or DMA) is to get the memory for a cycle and performs a lockout function for the opposite port if either DMA LOCKOUT is active or Rd ⊕ HOLD is active. These two signals result in priviledged operations of memory by one port, excluding the other port until the operation is completed. Included in this logic is the capability for the Test Equipment to halt all memory operations at the end of the present cycle.

The error latch logic detects abnormal addressing (requesting memory locations outside of the installed capacity) and sends error signals to the CPU for recovery. When an address outside of installed capacity is requested, the start memory signal is terminated. DMA operation on an odd (15 SAR = 1) boundary results in no start memory, sets DMA SPEC latch and sends an immediate ACK. Also any DMA ADDRESS or SPEC error inhibits the DMA LOCKOUT signal.

The start memory latch is set only for Valid DMA read or write and valid CPU read or write cycles. The CPU can request use of the memory circuits for loading the SDR (WRITE NOP) or loading the Storage Protect register (Write Move NOP). These two operations inhibit starting of main store and lockout the DMA port for a short time until the CPU operation has been completed.

The special reset and pulse timing section develops eight output signals. Three signals are used to reset CPU latches and mux control latches. Two signals are used to provide Data Load Clock (DLC) and Split Cycle Store (SCS) for use with core memory. The remaining 3 signals are used for initiating the end of a DMA SEQ, holding off start of memory until the DMA ACK and DMA RDY signals are both zero and delaying starting memory when using core storage until 150 NS after the fall of memory busy.

The CPU LRST signal resets CPUMEMBUSY and the latches on MUX CTRL 2 which store MROM 1 to 4, so that the logic can accept a new command from the CPU.

3.14 MEMORY TIMING

The memory timing chip provides the following functions:

- Timing Pulses for standard storage
- Timing Pulses for fast storage
- Read and Write byte width I/O bus enable

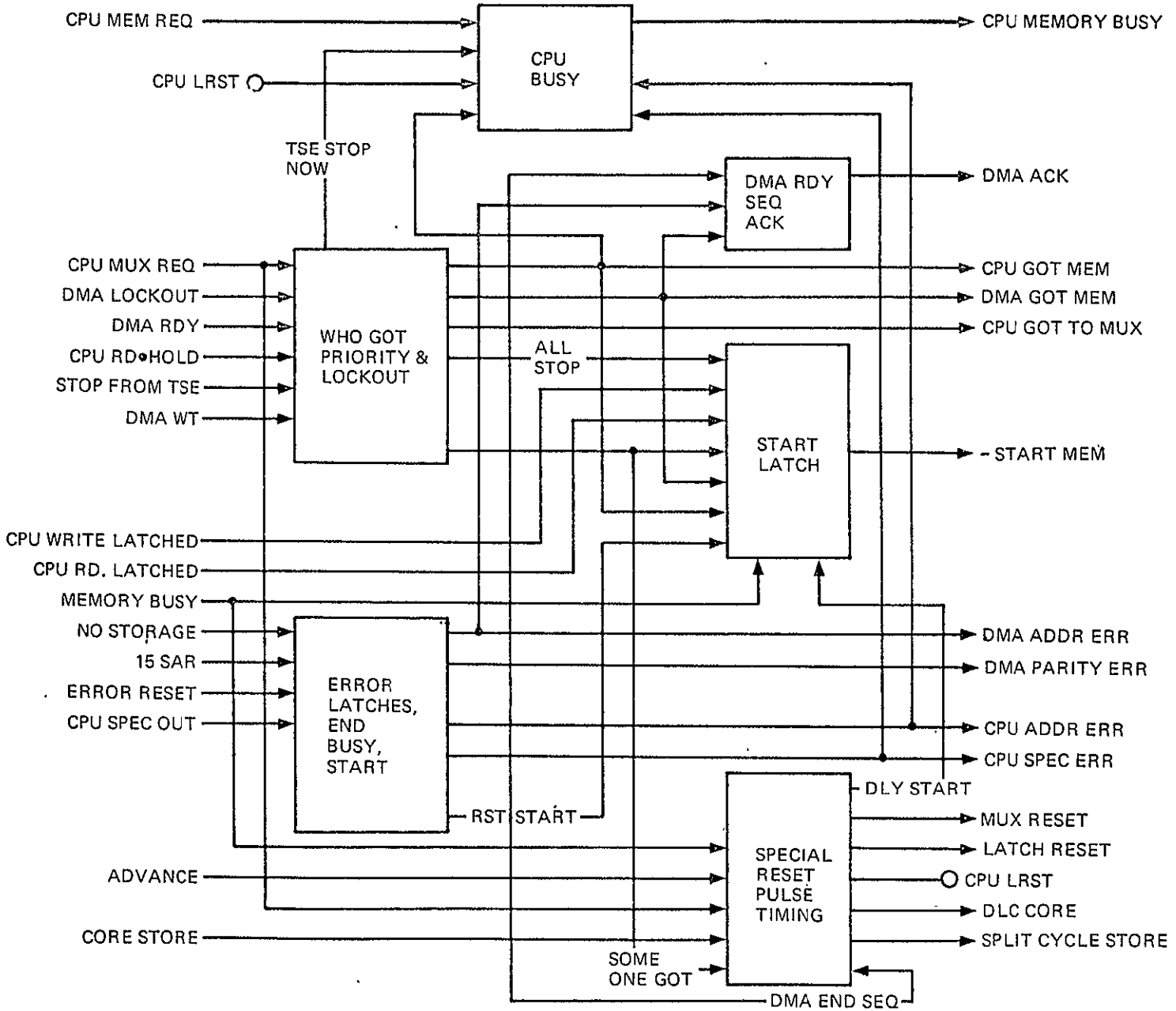


Figure 12. Memory Control Chip

- Special timing circuit for use in the Storage Interface Module
- General purpose OR circuits
- Dotable control outputs for multiple storage assemblies

The chip is shown in block diagram form in Figure 13.

The ring counter is started by decoding two select signals and the start signal. The timer is a 6 stage "switch-tail" ring counter which counts from 000000 to 111111 and starts counting toward 000000. The two control inputs, standard, and fast determine at what point the counter stops counting and force the count to 000000 and reset the start latch.

The time decoder supplies 6 output signals used for controlling a standard or fast memory and 2 lines for feedback to the CPU. The width of the various pulses is determined by worst case access time for the memory technologies and a clock rage of 9 to 10 MHZ. The memory busy and advance signals are replicas of signals which exist for a core technology memory and indicate when the memory is in operation and when data has been read or written. These two signals are dottable so that more than one memory module can be accessed (selected by select inputs).

The bus enable logic takes the read/write line and ANDS the "START ME" and ENABLE BYTE signals to develop input (write) and output (read) enable signals for on module data routing. The output enable signals are latched to provide read data to the CPU until the next memory cycle is started.

The special circuit is used to develop signals used by the MUX CTRL 2 chip to simulate a Read-Modify-Write cycle when writing a byte and using a core type memory.

3.15 AND/OR-TRUE/COMPLEMENT (AOTC) CHIP

The AOTC chip provides the following functions:

- True or complement outputs
- Two input, nine bit multiplexers
- Parity check/generation
- 3 to 8 decoder

Figure 14 is a block diagram of the AOTC chip. The multiplexer output is available in either true or complement form and is controlled by a True-Complement signal input. This signal set to a logic one will provide the outputs equal to the selected input and with this signal set to a logic zero the output will be equal to the complement of the input.

The input to the chip is provided from two sources, A and B. The input

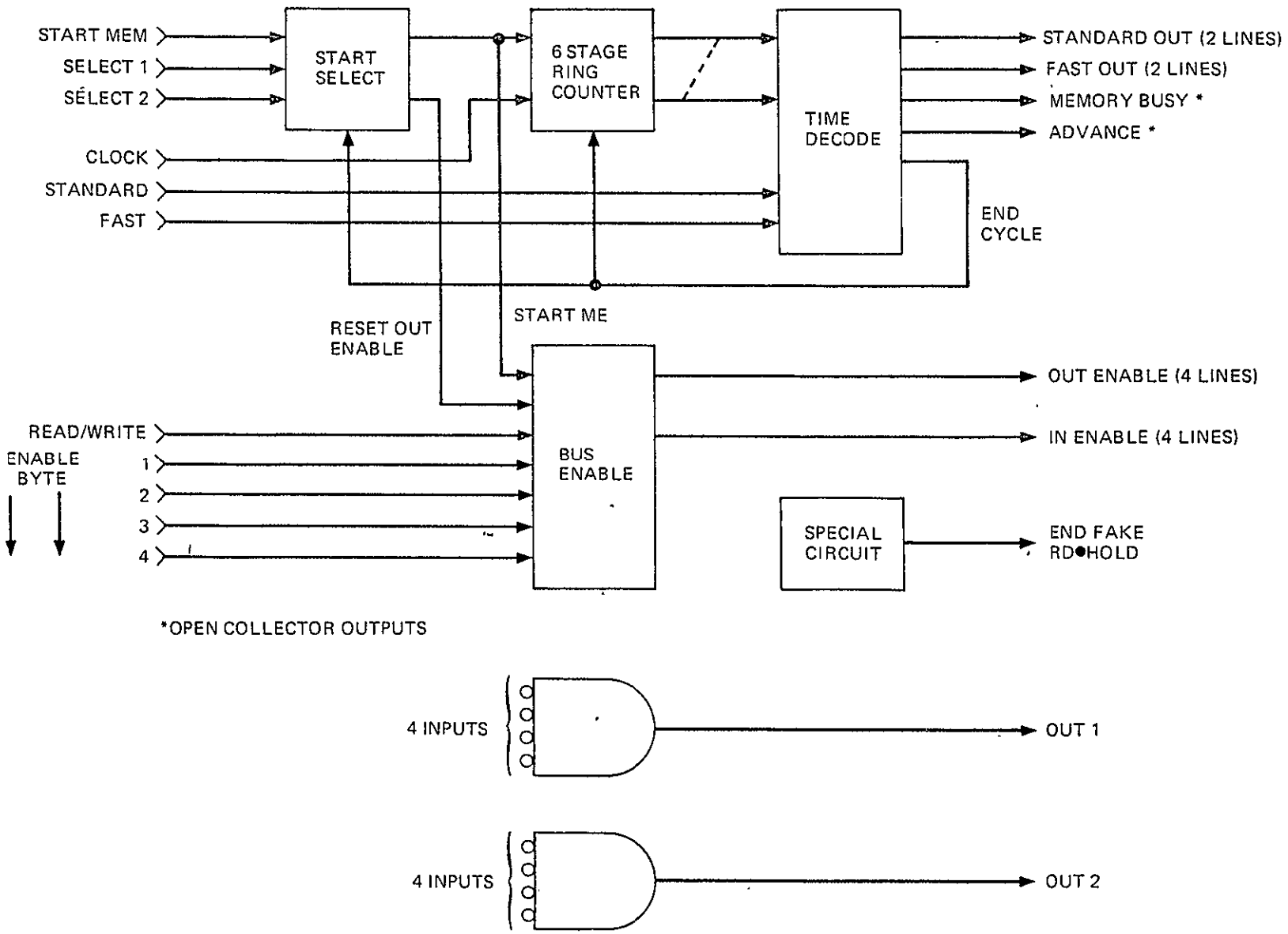


Figure 13. Memory Timing Chip

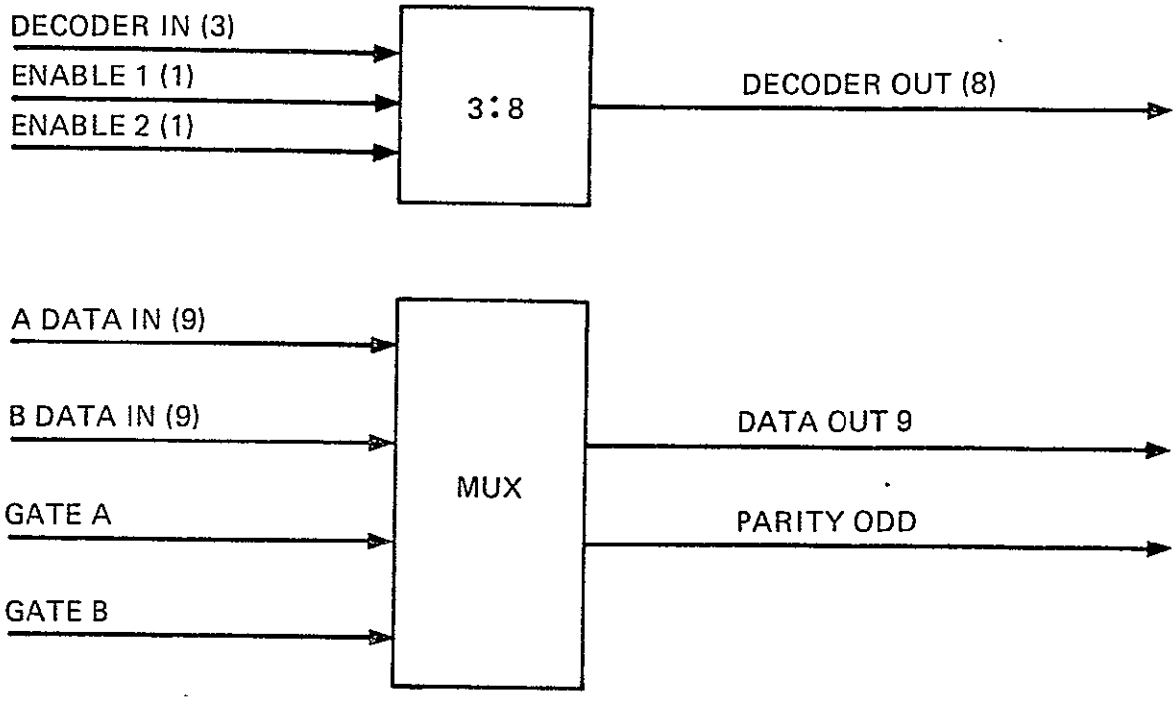


Figure 14. AOTC Chip

selected for transfer to output is controlled by Gate A and Gate B input signals. Table 29 defines output selection with respect to these control lines.

Table 29. Multiplexer Control

GATE A	GATE B	OUTPUT
0	0	ZERO
0	1	SELECT B
1	0	SELECT A
1	1	A + B

The AOTC chip contains an odd parity tree which can be used either to generate or check parity for 8 data bits. If the input to the ninth bit is set to a logical one the output of the parity network will generate odd parity. Parity is check on all data as it is selected to be transfered to the output and the parity signal made available external to the chip. The AOTC 3 to 8 decoder is defined by Table 30. Control of the decoder is accomplished by Enable 1 and Enable 2 signals.

Table 30. AOTC 3 : 8 Decoder

CONTROL		INPUT			OUTPUT							
ENABLE2	ENABLE 1				2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
0	1	0	0	0	0	1	1	1	1	1	1	1
0	1	0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	0	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	0	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0
X	0	X	X	X*	1	1	1	1	1	1	1	1
1	X	X	X	X	1	1	1	1	1	1	1	1

* X = Irrelevant

APPENDIX B

APPENDIX B

COMPLETE LISTING OF SUMC-IIB INSTRUCTIONS

1. SUMC-IIB STANDARD INSTRUCTION SET

NAME	MNEMONIC	TYPE	OPERAND	CODE	USEC
ADD	AR	RR	R1,R2	1A	2.0
ADD	A	RX	R1,D2(X2,B2)	5A	2.8
ADD HALFWORD	AH	RX	R1,D2(X2,B2)	4A	3.0
ADD LOGICAL	ALR	RR	R1,R2	1E	2.0
ADD LOGICAL	AL	RX	R1,D2(X2,B2)	5E	2.8
AND	NR	RR	R1,R2	14	2.0
AND	N	RX	R1,D2(X2,B2)	54	2.8
AND	NI	SI	D1(B1),I2	94	3.1
AND	NC	SS	D1(L,B1),D2(B2)	D4	4.8+2.0L
BRANCH AND LINK	BALR	RR	R1,R2	05	3.3+.3B
BRANCH AND LINK	BAL	RX	R1,D2(X2,B2)	45	3.4
BRANCH ON CONDITION	BCR	RR	M1,R2	07	2.0+1.1B
BRANCH ON CONDITION	BC	RX	M1,D2(X2,B2)	47	2.1+1.1B
BRANCH ON COUNT	BCTR	RR	R1,R2	06	2.5+1.1B
BRANCH ON COUNT	BCT	RX	R1,D2(X2,B2)	46	2.6+1.1B
BRANCH ON INDEX HIGH	BXH	RS	R1,R3,D2(B2)	86	7.2
BRANCH ON INDEX LOW OR EQUAL	BXLE	RS	R1,R3,D2(B2)	87	7.2
COMPARE	CR	RR	R1,R2	19	2.2
COMPARE	C	RX	R1,D2(X2,B2)	59	3.0
COMPARE HALFWORD	CH	RX	R1,D2(X2,B2)	49	3.3
COMPARE LOGICAL	CLR	RR	R1,R2	15	2.4
COMPARE LOGICAL	CL	RX	R1,D2(X2,B2)	55	3.1
COMPARE LOGICAL	CLC	SS	D1(L,B1),D2(B2)	D5	4.5+2.1L
COMPARE LOGICAL	CLI	SI	D1(B1),I2	95	2.2
CONVERT TO BINARY	CVB	RX	R1,D2(X2,B2)	4F	20.2+10.8Z
CONVERT TO DECIMAL	CVD	RX	R1,D2(X2,B2)	4E	40.7+1.5S
DIAGNOSE		SI		83	3.1+Diagnostic
DIVIDE	DR	RR	R1,R2	1D	51.0
DIVIDE	D	RX	R1,D2(X2,B2)	5D	51.8
EXCLUSIVE OR	XR	RR	R1,R2	17	2.0
EXCLUSIVE OR	X	RX	R1,D2(X2,B2)	57	2.8
EXCLUSIVE OR	XI	SI	D1(B1),I2	97	3.1
EXCLUSIVE OR	XC	SS	D1(L,B1)D2(B2)	D7	4.8+2.0L
EXECUTE	EX	RX	R1,D2(X2,B2)	44	5.4+Target
INSERT CHARACTER	IC	RX	R1,D2(X2,B2)	43	2.5
LOAD	LR	RR	R1,R2	18	2.0
LOAD	L	RX	R1,D2(X2,B2)	58	2.8
LOAD ADDRESS	LA	RX	R1,D2(X2,B2)	41	2.3
LOAD AND TEST	LTR	RR	R1,R2	12	1.9
LOAD COMPLEMENT	LCR	RR	R1,R2	13	2.0
LOAD HALFWORD	LH	RX	R1,D2(X2,B2)	48	2.9
LOAD MULTIPLE	LM	RS	R1,R3,D2(B2)	98	3.2+1.9N
LOAD NEGATIVE	LNR	RR	R1,R2	11	2.3
LOAD POSITIVE	LPR	RR	R1,R2	10	2.3
OAD PSW	LPSW	SI	D1(B1)	82	8.1

NAME	MNEMONIC	TYPE	OPERAND	CODE	USEC
MOVE	MVI	SI	D1(B1),I2	92	2.3
MOVE	MVC	SS	D1(L,B1),D2(B2)	D2	4.6+1.1L
MOVE NUMERICS	MVN	SS	D1(L,B1),D2(B2)	D1	5.1+2.0L
MOVE WITH OFFSET	MVO	SS	D1(L1,B1),D2(L2,B2)	F1	4.4+3.3L
MOVE ZONES	MVZ	SS	D1(L,B1),D2(B2)	D3	5.4+2.0L
MULTIPLY	MR	RR	R1,R2	1C	30.2
MULTIPLY	M	RX	R1,D2(X2,B2)	5C	30.4
MULTIPLY HALFWORD	MH	RX	R1,D2(X2,B2)	4C	13.6
OR	OR	RR	R1,R2	16	2.0
OR	O	RX	R1,D2(X2,B2)	56	2.8
OR	OI	SI	D1(B1),I2	96	3.1
OR	OC	SS	D1(L,B1),D2(B2)	D6	4.8+2.0L
PACK	PACK	SS	D1(L1,B1),D2(L2,B2)	F2	3.1+4.1L
SET STORAGE KEY	SSK	RR	R1,R2	08	3.5
SET PROGRAM MASK	SPM	RR	R1	04	1.5
SET SYSTEM MASK	SSM	SI	D1(B1)	80	6.3
SHIFT LEFT DOUBLE	SLDA	RS	R1,D2(B2)	8F	4.1+2.6Q+1.3R
SHIFT LEFT SINGLE	SLA	RS	R1,D2(B2)	8B	3.0+.3Q+.3R
SHIFT LEFT DOUBLE LOGICAL	SLDL	RS	R1,D2(B2)	8D	3.6+1.3q+1.3r
SHIFT LEFT SINGLE LOGICAL	SLL	RS	R1,D2(B2)	89	3.0+.3Q+.3R
SHIFT RIGHT DOUBLE	SRDA	RS	R1,D2(B2)	8E	4.1+3Q+3R
SHIFT RIGHT SINGLE	SRA	RS	R1,D2(B2)	8A	2.4+.3Q+.3R
SHIFT RIGHT DOUBLE LOGICAL	SRDL	RS	R1,D2(B2)	8C	3.6+3Q+3R
SHIFT RIGHT SINGLE LOGICAL	SRL	RS	R1,D2(B2)	88	2.3+.3Q+.3R
START I/O	SIO	SI	D1(B1)	A5	8.9+I/O Delay
STORE	ST	RX	R1,D2(X2,B2)	50	3.1
STORE CHARACTER	STC	RX	R1,D2(X2,B2)	42	2.5
STORE HALFWORD	STH	RX	R1,D2(X2,B2)	40	2.5
STORE MULTIPLE	STM	RS	R1,R3,D2(B2)	90	3.6+1.9N
SUBTRACT	SR	RR	R1,R2	1B	2.0
SUBTRACT	S	RX	R1,D2(X2,B2)	5B	2.8
SUBTRACT HALFWORD	SH	RX	R1,D2(X2,B2)	4B	3.0
SUBTRACT LOGICAL	SLR	RR	R1,R2	1F	2.0
SUBTRACT LOGICAL	SL	RX	R1,D2(X2,B2)	5F	2.8
SUPERVISOR CALL	SVC	RR	I	0A	12.3
TEST AND SET	TS	SI	D1(B1)	93	3.4
TEST UNDER MASK	TM	SI	D1(B1),I2	91	3.5
TIMER READ/SET	TMRS	RS	R1,R3,D2(B2)	A4	7.4+4.6T
TRANSLATE	TR	SS	D1(L,B1),D2(B2)	DC	4.6+2.1L
TRANSLATE AND TEST	TRT	SS	D1(L,B1),D2(B2)	DD	4.6+2.2L
UNPACK	UNPK	SS	D1(L1,B1),D2(L2,B2)	F3	6.8+1.5L

NOTES:

B - 1 if branch is successful, otherwise 0

L - number of first operand bytes processed

N - number of registers processed

q - shift count divided by 16

Q - shift count divided by 4

r - shift count modulo 15

R - shift count modulo 3

S - sum of the decimal digits, except last digit

T - 1 if timer is being set, otherwise 0

Z - 8 minus number of leading zero bytes

Add .5 if instruction type is RX and an index register is specified.

2. SUMC-IIB SHORT OPERAND (16-BIT) OPTION

NAME	MNEMONIC	TYPE	OPERANDS	OP CODE	μSEC
ADD HALFWORD IMMEDIATE	AHI	RI	R1,I2	BA	2.8
ADD SHORT	AS	RX	R1,D2(X2,B2)	53	2.2
ADD SHORT IMMEDIATE	ASI	RI	R1,I2	AA	2.0
ADD SHORT REGISTER	ASR	RR	R1,R2	CA	1.7
BRANCH UNCONDITIONAL	BU	RX	D2(X2,B2)	73	2.1
BRANCH UNCONDITIONAL REGISTER	BUR	RR	R2	CE	1.7
COMPARE HALFWORD IMMEDIATE	CHI	RI	R1,I2	B9	3.1
COMPARE LOGICAL SHORT	CLS	RX	R1,D2(X2,B2)	65	2.6
COMPARE LOGICAL SHORT IMMEDIATE	CLSI	RI	R1,I2	B5	2.1
COMPARE LOGICAL SHORT REGISTER	CLSR	RR	R1,R2	C5	1.7
COMPARE SHORT	CS	RX	R1,D2(X2,B2)	61	2.4
COMPARE SHORT IMMEDIATE	CSI	RI	R1,I2	A9	2.2
COMPARE SHORT REGISTER	CSR	RR	R1,R2	C9	2.0
DIVIDE SHORT	DS	RX	R1,D2(X2,B2)	4D	15.1
DIVIDE SHORT IMMEDIATE	DSI	RI	R1,I2	BØ	14.9
DIVIDE SHORT REGISTER	DSR	RR	R1,R2	CD	14.9
LOAD ADDRESS SHORT	LAS	RX	R1,D2(X2,B2)	51	2.0
LOAD COMPLEMENT SHORT REGISTER	LCSR	RR	R1,R2	C3	1.7
LOAD FULL TO SHORT REGISTER	LFSR	RR	R1,R2	ØB	2.1
LOAD HALFWORD IMMEDIATE	LHI	RI	R1,I2	B8	2.6
LOAD HALFWORD REGISTER	LHR	RR	R1,R2	DØ	1.9
LOAD NEGATIVE SHORT REGISTER	LNSR	RR	R1,R2	C1	1.9
LOAD POSITIVE SHORT REGISTER	LPSR	RR	R1,R2	CØ	1.9
LOAD SHORT	LS	RX	R1,D2(X2,B2)	74	2.2
LOAD SHORT IMMEDIATE	LSI	RI	R1,I2	A8	2.0
LOAD SHORT REGISTER	LSR	RR	R1,R2	C8	1.5
LOAD AND TEST	LT	RX	R1,D2(X2,B2)	62	2.8
LOAD AND TEST SHORT	LTS	RX	R1,D2(X2,B2)	52	2.2
LOAD AND TEST SHORT REGISTER	LTSR	RR	R1,R2	C2	1.5
MULTIPLY HALFWORD IMMEDIATE	MHI	RI	R1,I2	BC	13.4
MULTIPLY SHORT	MS	RX	R1,D2(X2,B2)	71	7.8
MULTIPLY SHORT IMMEDIATE	MSI	RI	R1,I2	B3	7.8
MULTIPLY SHORT REGISTER	MSR	RR	R1,R2	CC	7.6
NORMALIZE	NRM	RR	R1,R2	CF	3.9+1.1Q+1
AND SHORT	NS	RX	R1,D2(X2,B2)	64	2.2
AND SHORT IMMEDIATE	NSI	RI	R1,I2	B4	2.0
AND SHORT REGISTER	NSR	RR	R1,R2	C4	1.5
OR SHORT	OS	RX	R1,D2(X2,B2)	66	2.5
OR SHORT IMMEDIATE	OSI	RI	R1,I2	A6	2.0
OR SHORT REGISTER	OSR	RR	R1,R2	C6	1.5

NAME	MNEMONIC	TYPE	OPERANDS	OP CODE	μSEC
SUBTRACT HALFWORD IMMEDIATE	SHI	RI	R1,I2	BB	2.8
SHIFT LEFT ARITHMETIC SHORT	SLAS	RS	R1,D2(B2)	A3	2.6+.3Q+.3R
SHIFT LEFT LOGICAL SHORT	SLLS	RS	R1,D2(B2)	A1	2.6+.3Q+.3R
SHIFT RIGHT ARITHMETIC SHORT	SRAS	RS	R1,D2(B2)	A2	2.6+.3Q+.3R
SHIFT RIGHT LOGICAL SHORT	SRLS	RS	R1,D2(B2)	A0	2.6+.3Q+.3R
SUBTRACT SHORT	SS	RX	R1,D2(X2,B2)	72	2.2
SUBTRACT SHORT IMMEDIATE	SSI	RI	R1,I2	AB	2.0
SUBTRACT SHORT REGISTER	SSR	RR	R1,R2	CB	1.7
EXCLUSIVE OR SHORT	XS	RX	R1,D2(X2,B2)	63	2.2
EXCLUSIVE OR SHORT IMMEDIATE	XSI	RI	R1,I2	A7	2.0
EXCLUSIVE OR SHORT REGISTER	XSR	RR	R1,R2	C7	1.5
TEST BITS	TB	RX	R1,D2(X2,B2)	75	3.5
TEST BITS IMMEDIATE	TBI	RI	R1,I2	AE	3.0

NOTE: Q = Shift Count divided by 4
R = Remainder from division by 4

3. SUMC-IIB DOUBLE PRECISION (64-BIT) OPTION

NAME	MNEMONIC	TYPE	OPERANDS	OP CODE	μSEC
ADD DOUBLE	AD	RX	R1,D2(X2,B2)	6A	6.1
ADD DOUBLE REGISTER	ADR	RR	R1,R2	2A	6.1
COMPARE DOUBLE	CD	RX	R1,D2(X2,B2)	69	6.4
COMPARE DOUBLE REGISTER	CDR	RR	R1,R2	29	6.4
LOAD COMPLEMENT DOUBLE REGISTER	LCDR	RR	R1,R2	23	5.3
LOAD DOUBLE	LD	RX	R1,D2(X2,B2)	68	4.4
LOAD DOUBLE REGISTER	LDR	RR	R1,R2	28	5.0
SUBTRACT DOUBLE	SD	RX	R1,D2(X2,B2)	6B	6.1
SUBTRACT DOUBLE REGISTER	SDR	RR	R1,R2	2B	6.1
STORE DOUBLE	STD	RX	R1,D2(X2,B2)	60	5.2

4. SUMC-IIB FLOATING POINT OPTION

NAME	MNEMONIC	TYPE	OPERAND	OP CODE	μSEC
ADD NORMALIZED	AER	RR	R1,R2	3A	20.8+A+N
ADD NORMALIZED	AE	RX	R1,D2(X2,B2)	7A	21.2+A+N
ADD UNNORMALIZED	AUR	RR	R1,R2	3E	19.8+A
ADD UNNORMALIZED	AU	RX	R1,D2(X2,B2)	7E	20.4+A
COMPARE	CER	RR	R1,R2	39	16.0+A
COMPARE	CE	RX	R1,D2(X2,B2)	79	16.2+A
DIVIDE	DER	RR	R1,R2	3D	48.5+N
DIVIDE	DE	RX	R1,D2(X2,B2)	7D	48.6+N
HALVE	HER	RR	R1,R2	34	9.5+N
LOAD AND TEST	LTER	RR	R1,R2	32	4.2
LOAD COMPLEMENT	LCER	RR	R1,R2	33	4.6
LOAD NEGATIVE	LNER	RR	R1,R2	31	4.6
LOAD POSITIVE	LPER	RR	R1,R2	3Ø	4.4
LOAD	LER	RR	R1,R2	38	3.1
LOAD	LE	RX	R1,D2(X2,B2)	78	2.9
MULTIPLY	MER	RR	R1,R2	3C	33.7+N
MULTIPLY	ME	RX	R1,D2(X2,B2)	7C	33.8+N
STORE	STE	RX	R1,D2(X2,B2)	7Ø	3.7
SUBTRACT NORMALIZED	SER	RR	R1,R2	3B	21.7+A+N
SUBTRACT NORMALIZED	SE	RX	R1,D2(X2,B2)	7B	21.8+A+N
SUBTRACT UNNORMALIZED	SUR	RR	R1,R2	3F	21.0+A
SUBTRACT UNNORMALIZED	SU	RX	R1,D2(X2,B2)	7F	20.8+A

NOTES: A = .3 μsec per digit for exponent alignment
N =1.4 μsec per digit for normalization