952-5497-2

SYSTEM/4 PI

ENGINEERING

DESCRIPTION

MODEL EP

IBM NO 66-584-002

ORIGINATING GROUP Department 584

CONTENT APPROVED BY

CONTRACT NO

DATE November 7, 1966

IBM FEDERAL SYSTEMS DIVISION
ELECTRONICS SYSTEMS CENTER
OWEGO NEW YORK

PREFACE

This Engineering Description has been prepared for applications engineers and Model EP users. It provides a broad based introduction to the Model EP architecture, support software and to the concepts of microprogramming. The technologies used in Model EP and the logical design of the CPU and Multiplex Channel are described. Much of the information in this document is merely descriptive of design and implementation techniques and does not constitute a commitment to deliver equipment or services in exact conformance to that described.

A companion document, Model EP Function Specification, #66-584-001, describes the functions and features of System/4 Pi Model EP and particularizes them in relation to System/360.

These documents supersede and replace the previous Model EP, Product Description (3-001-56). In addition to an expansion of the information provided, these documents include numerous revisions. Most of these are changes in detail rather than in concept but they are too numerous to identify individually.

This document has been issued in preliminary form. It is incomplete and may contain some errors of detail. It will be completely revised and reissued early in 1967.

Table of Contents

1.0 INTRODUCTION

Model EP provides the power and facilities of a large scale general purpose processor for use in the aerospace and mobile environment. Three basic characteristics contribute toward the unique capabilities which it offers:

- o Architectural compatibility with System/360 reduces the effort and expense required to write and verify operational programs.
- o Microprogramming capability permits the incorporation of problem oriented instructions and procedures thereby providing the throughput of a special purpose processor within the context of a general purpose machine.
- o A number of special features are offered which enhance the system operation in a real-time environment and which provide for flexible multisystem operation.

The compatibility with System/360, the computational power obtained through microprogramming and the versatility offered through the many features, have produced a processor system which introduces a previously unavailable level of performance to the hardened environment.

1.1 System/360 Compatibility

System/4 Pi Model EP is based upon System/360 and its primary instruction sets, status sequencing, interruption handling and input/output system are fully compatible with System/360 thereby permitting bi-directional compatibility with ground-based computers. Provided that equivalent facilities are available and that programs are not time-dependent, programs written for Model EP may be run on any System/360 processor and identical results will be obtained. Conversely, programs available for System/360 (including certain software aids) may be used in Model EP.

Bi-directional compatibility offers the opportunity for efficient program development and for exact program verification in a host machine. It provides the ability to make use of a large repertory of existing programs and to call upon a large pool of experienced programmers.

Many features and options such as storage sharing, fall within the architectural concepts of System/360. Some of the features and options such as the use of special macroinstructions, are not ordinarily available in System/360 and require that specialized software be utilized. Such software will be offered in a standard, modular package which will provide effective support for the Model EP processor.

1.2 Microprogramming

The Model EP is a microprogrammed processor. The exact details of performance of each instruction are stored in a read-only memory (ROS) and new or modified instructions may be added merely by changing the contents of this memory. The concepts of microprogramming and an introduction of its techniques are provided in section 4.

In real-time applications, the microprogramming facility permits the processor to be "tuned" to the application such that problem solution times can be drastically reduced for specific requirements. One class of special instructions that can be added is mathematical routines such as square root, tangent, arc tangent, matrix multiply, etc. For example, if many trigometric operations are required, such as in Euler angle conversion, the use of sine and cosine instructions simplifies programming and will decrease problem solution time.

In some applications, it is necessary to perform complex and unusual routines repetitively. Examples are searching for a specific field in a bank of words to find if there is a match between any of the words searched and a specific key. Another example would be computation of radar steering angles: It is often possible to create highly specialized instructions to perform such applications dependent operations. Through microprogramming, the entire response of Model EP to a specific problem may be modified such that the processor will operate like a special purpose machine for unique parts of a problem and will continue to be a powerful, general purpose processor for the remainder of the application requirements.

1.3 Features

Model EP offers a large number of options and features in order that each installation may be systematically optimized. The Functional Specification, #66-584-001, provides detail concerning the functions which are available and their application. Briefly, the CPU may be equipped with a variety of standard instruction sets as well as with special instructions as mentioned above. The computational speed of the CPU may be varied by means of choosing general registers stored in mainstore or a hardware implementation of those registers.

The storage system may be modularly expanded in units of 8K words to a maximum of 131K words. Memory systems from several processors may be interconnected to form a "storage pool" with each processor having full access to the entire pool or with each processor having its isolated part and each processor having access to a limited pool. The storage protection feature may be utilized to protect specific blocks of information for prevention of illegal storage in the block or for both store and fetch protection.

The input/output system is designed to allow a wide span of performance for System/360 compatible channels and to permit the effective design of special purpose channel facilities. In particular, the ability of the storage system to accommodate multiple buses permits simultaneous and asynchronous loading of mainstore through stand-alone channels.

Processors are equipped with a System/360 compatible interruption system. In addition, real-time operations may be enhanced by use of a 32 level priority interrupt system which provides programmed masking, automatic priority resolution and automatic nesting of interrupts. Special power supply designs are offered to accommodate widely varying power sources and to provide automatic shutdown and restart in the event of transient power losses.

Multisystem operations may be achieved through the use of shared mainstore systems, shared input/output devices and direct communication between processors. Multisystem operation may be controlled by means of special instructions (Test and Set, Direct Read and Direct Write), by special features (Multisystem Signalling, and Direct Control) and by exchanging control signals through shared I/O devices or channel-to-channel adapters.

2.0 ARCHITECTURE

2.1 General Concepts

Model EP introduces the broad base capability of large scale processors into the aerospace and mobile environment. Model EP is a general purpose system that can readily accommodate scientific, control and communications applications. A Basic instruction set provides a fixed point and logical capability. A Standard instruction set provides the addition of logical manipulations of variable field length. A Floating Point feature may be added to either the Basic or Standard set. Other instructions are related to a variety of special features which are used to enhance real-time operations or provide for multisystem operations. Special purpose instructions may be incorporated in Model EP to optimize performance for specific applications.

Interplay of equipment and program is an essential consideration in System/4 Pi. The system is designed to operate with a supervisory program that coordinates and executes all I/O instructions, handles exceptional conditions, and supervises scheduling and execution of multiple programs. System/4 Pi provides for efficient switching from one program to another, as well as for the relocation of programs in storage. To the problem programmer, the supervisory program and the equipment are indistinguishable.

IBM can provide System/4 Pi programs that control and schedule the use of CPU facilities, main storage, input/output devices, etc. These programs are designed to control all system resources, including programs supplied by the user.

2.2 System/360 Compatibility

The Model EP is based on System/360 and the architecture is substantially identical to System/360 with the exception that:

(1) Model EP may be supplied with an instruction set which is smaller than the System/360 Standard set. The System/4 Pi Basic Set is the System/360 Standard Set less the 12 Variable Field Length (VFL) instructions and the 2 Convert instructions.

(2) Model EP may be supplied with instructions not available in System/360. Such instructions may be associated with:

a. features not generally available in System/360 such as Priority Interrupt.

b. special mathematical microinstructions such as Square Root, Sine/Cosine, Arc Tangent, etc.

c. special application-oriented microinstructions such as Compute Radar Angle, etc.

Based on the compatibility with System/360, the documents describing that system may be used as valid descriptions of the Model EP. The Model EP Functional Specification, #66-584-001, particularizes Model EP within the context of System/360 and provides additional information about system application, feature availability and processor operation under exceptional conditions.

System/4 Pi is compatible with System/360; that is, any program gives identical results on Model EP or on any System/360 model. This rule is subject to the following limitations:

1. The systems facilities used by a program should be the same in each case. For example, the optional CPU features and the storage capacity, as well as the quantity, type, and priority of I/O equipment, should be equivalent. System facilities are understood to include instruction set availability such that each machine has at least the set of instructions used in the program.

2. The program should be independent of the relation of instruction execution times and of I/O data rates, access times, and command execution times.

3. The compatibility rule does not apply to detail functions for which neither frequency of occurrence nor usefulness of result warrants identical action in all models. These functions, all explicitly identified in 360 PO, are concerned with the handling of invalid programs and machine malfunctions.

4. Certain special features such as Priority Interrupt and Power Loss Protection/Recovery are transparent to 360 architecture but their operation usually may not be replicated on a System/360 processor.

2.3 System Structure

The material in this section has been prepared as an introduction to those aspects of system structure which are common to both System/360 and System/4 Pi. This material is largely extracted from System/360 Principles of Operation. Changes have been incorporated to delete unavailable features, to shift emphasis, to enhance clarity and to mention priority interrupt and special purpose input/output channels. The material in this section is introductory in nature. The reader is referred to System/360 Principles of Operation for an extensive description of the architecture and to System/4 Pi Functional Specification for a detailed description of features available in Model EP.

The basic structure of a System/4 Pi consists of main storage, a central processing unit (CPU), the input/output channels, and the input/output devices attached to the channels through control units. It is possible for systems to communicate with each other by means of shared I/O devices, a channel, or shared storage. Figure 1 shows the basic organization of a single system.

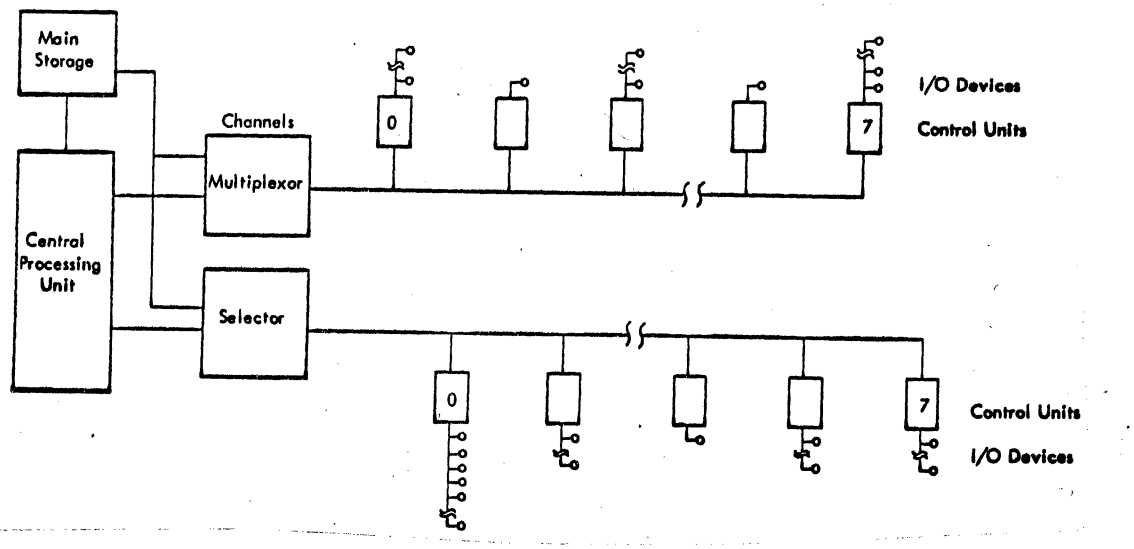


Figure 1. IBM System/4 Pi Basic Logical Structure

2.3.1 Main Storage

Storage units may be either physically integrated with the CPU or constructed as stand-alone units. The storage cycle speed is not directly related to the internal cycling of the CPU.

Main storage may be shared by CPU's. Fetching and storing of data by the CPU are not affected by any concurrent I/O data transfer or by reference to the same storage location by another CPU. If a CPU and a channel concurrently refer to the same storage location, the accesses normally are granted in a sequence that assigns higher priority to references by channels. If the first reference changes the contents of the location, any subsequent storage fetches obtain the new contents.

Information Formats - The system transmits information between main storage and the CPU in four units, of eight bits at a time. Each eight-bit unit of information is called a byte, the basic building block of all formats. A ninth bit, the parity or check bit, is transmitted with each byte and carries odd parity on the byte. The parity bit cannot be affected by the program; its only purpose is to cause an interruption when a parity error is detected. Reference in this manual to the size of data fields and registers exclude the mention of the associated parity bits.

Bytes may be handled separately or grouped together in fields. A halfword is a group of two consecutive bytes and is the basic building block of instructions. A word is a group of four consecutive bytes; a double word is a field consisting of two words (Figure 2). The location of any field or group of bytes is specified by the address of its leftmost byte.

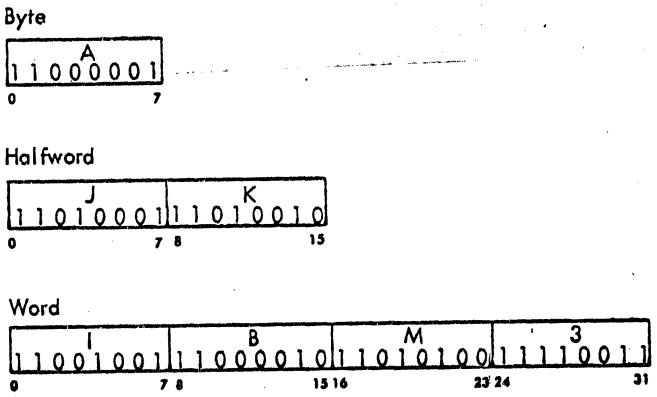


Figure 2. Sample Information Formats

The length of fields is either implied by the operation to be performed or stated explicitly as part of the instruction. When the length is implied, the information is said to have a fixed length, which can be either one, two, four, or eight bytes.

When the length of a field is not implied by the operation code, but is stated explicitly, the information is said to have variable field length. Variable-length operands are variable in length by increments of one byte.

Within any program format or any fixed-length operand format, the bits making up the format are consecutively numbered from left to right starting with the number 0.

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 24-bit binary address to accommodate a maximum of 16,777,216 byte addresses. This set of main-storage addresses includes some locations reserved for special purposes.

When only a part of the maximum storage capacity is available in a given installation, the available storage is normally contiguously addressable, starting at address 0. An addressing exception is recognized when any part of an operand is located beyond the maximum available capacity of an installation.

Information Positioning - Fixed-length fields, such as halfwords and double words, must be located in main storage on an integral boundary for that unit of information. A boundary is called integral for a unit of information when its storage address is a multiple of the length of the unit in bytes. For example, words (four bytes) must be located in storage so that their address is a multiple of the number 4. A halfword (two bytes) must have an address that is a multiple of the number 2, and double words (eight bytes) must have an address that is a multiple of the number 8.

Storage addresses are expressed in binary form. In binary, integral boundaries for halfwords, words, and double words can be specified only by the binary addresses in which one, two, or three of the low-order bits, respectively, are zero. (Figure 3). For example, the integral boundary for a word is a binary address in which the two low-order positions are zero.

Variable-length fields are not limited to integral boundaries, and may start on any byte location.

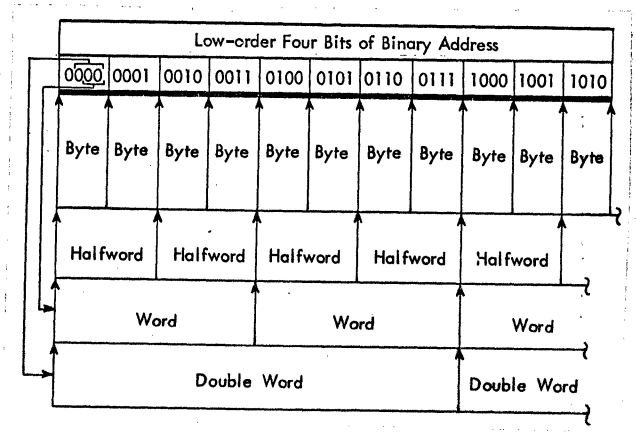


Figure 3. Integral Boundaries for Halfwords, Words and Double Words

2.3.2 Central Processing Unit

The central processing unit (Figure 4) 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.

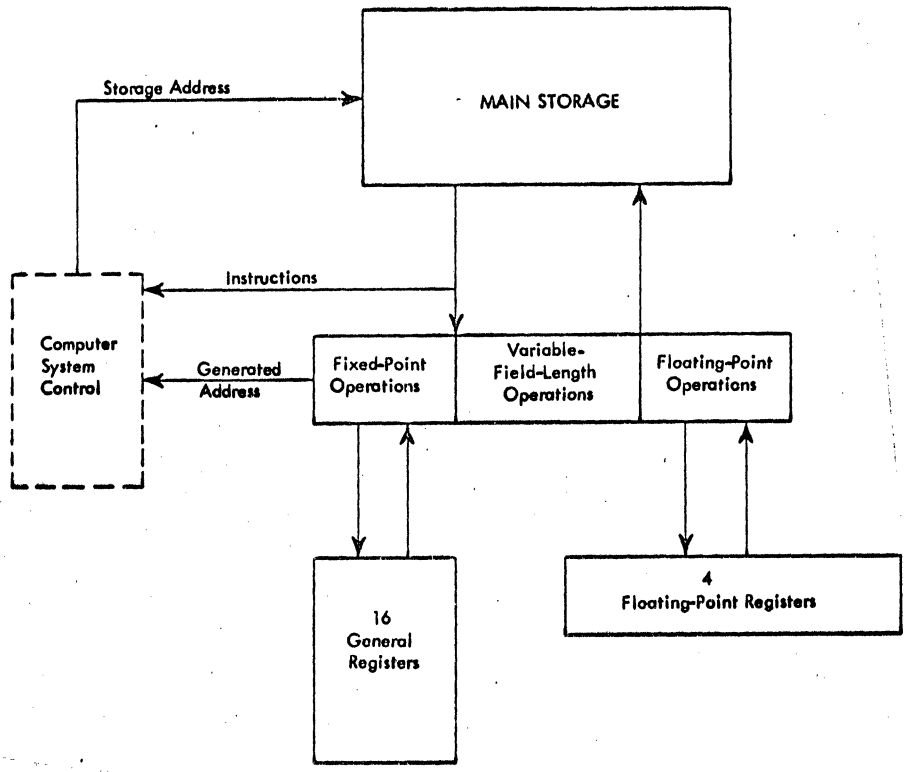


Figure 4. Basic Concept of Central Processing Unit Functions

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

The CPU provides 16 general registers for fixed-point operands and four floating-point registers for floating-point operands. Implementation of these registers may be in special circuitry or in a separate area of main storage. In either case, the address and functions of these registers are identical.

General Registers - The CPU can address information in 16 general registers. 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 (Figure 5). Some instructions provide for addressing multiple general registers by having several R fields.

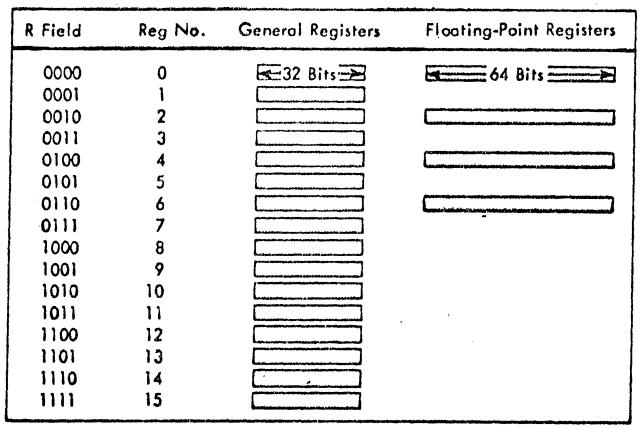


Figure 5. General and Floating-Point Registers

For some operations, two adjacent general registers are coupled together, providing a two-word capacity. In these operations, the addressed register contains the high order operand bits and must have an even address, and the implied register, containing the low-order operand bits, has the next higher address.

Floating-Point Registers - Four floating-point registers are available for floating-point operations. They are identified by the numbers 0, 2, 4, and 6 (Figure 5). These floating-point registers are two words (64 bits) in length and can contain either a short (one word) or a long (two words) floating-point operand. A short operand occupies the high-order bits of a floating-point register. The low-order portion of the register is ignored and remains unchanged in short-precision arithmetic. The instruction operand code determines which type of register (general or floating-point) is to be used in an operation.

2.3.3 Arithmetic and Logical Unit

The arithmetic and logical unit can process binary integers and floating-point fractions of fixed length, and logical information of either fixed or variable length. Processing is done in parallel using a 32 bit wide adder-shifter path and an 8 bit wide mover path which can operate simultaneously.

Arithmetic and logical operations performed by the CPU fall into three classes: fixed-point arithmetic, floating-point arithmetic, and logical operations. These classes differ in the data formats used, the registers involved, the operations provided, and the way the field length is stated.

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. See Figure 6. To preserve precision, some products and all dividends are 64 bits long.

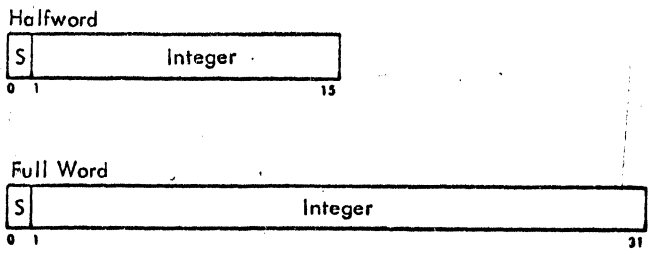


Figure 6. Fixed-Point Number Formats

Because the 32-bit word size readily accommodates a 24-bit address, fixed-point arithmetic can be used both for integer operand arithmetic and for address arithmetic. This combined usage provides economy and permits the entire fixed-point instruction set and several logical operations to be used in address computation. Thus, multiplication, shifting, and logical manipulation of address components are possible.

The absence of the need for recomplementation and the ease of extension and truncation make two's-complement notation desirable for address components and fixed-point operands. Since integer and addressing algorisms often require repeated reference to operands or intermediate results, the use of multiple registers is advantageous in arithmetic sequences and address calculations.

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.

Floating- Point Arithmetic - Floating-point numbers occur in either of two fixed-length formats -- short or long. These formats differ only in the length of the fractions (Figure 7.)

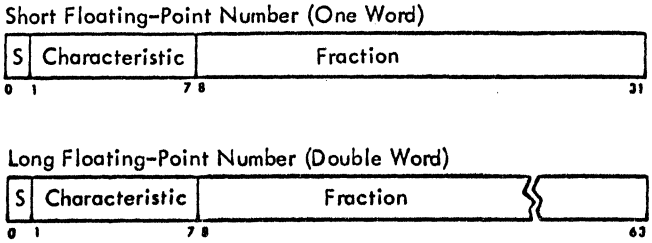


Figure 7. Short and Long Floating-Point Number Formats

Floating-point operands are either 32 or 64 bits long. The short length, equivalent to seven decimal places of precision, permits a maximum number of operands to be placed in storage and gives the shortest execution times. The long length, used when higher precision is desired, gives up to 17 decimal places of precision, thus eliminating most requirements for double-precision arithmetic.

The operand lengths, being powers of two, permit maximum efficiency in the use of binary addressing and in matching the physical word sizes of the different models. Floating-point arithmetic is designed to allow easy transition between the two formats.

The fraction of a floating-point number is expressed in hexadecimal (base 16) digits, each consisting of four binary bits and having the values 0-15. In the short format, the fraction consists of six hexadecimal digits occupying bits 8-31. In the long format the fraction has 14 hexadecimal digits occupying bits 8-63.

The radix point of the fraction is assumed to be immediately to the left of the high-order fraction digit. To provide the proper magnitude for the floating-point number, the fraction is considered to be multiplied by a power of 16. The characteristic portion, bits 1-7 of both formats, is used to indicate this power. The characteristic is treated as an excess 64 number with a range from -64 through +63, and permits representation of decimal numbers with magnitudes in the range of 10^{-78} to 10^{75}.

Bit position 0 in either format is the sign (S) of the fraction. The fraction of negative numbers is carried in true form.

Four 64-bit floating-point registers are provided. Arithmetic operations are performed with one operand in a register and another either in a register or from storage. The result, developed in a register, is generally of the same length as the operands. The availability of several floating-point registers eliminates much storing and loading of intermediate results.

Logical Operations - Logical information is handled as fixed- or variable-length data. It is subject to such operations as comparison, translation, editing, 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 (Figure 8).

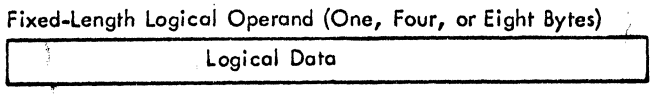


Figure 8. Fixed-Length Logical Information

A large portion of logical information consists of alphabetic or numeric character codes, called alphametic data, and is used for communication with character-set sensitive I/O devices. This information has the variable-field length format and can consist of up to 256 bytes (Figure 9). It is processed storage to storage, left to right, an eight-bit byte at a time.

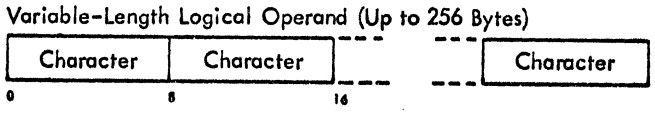


Figure 9. Variable-Length Logical Information

2.3.4 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 10 shows five basic instruction formats.

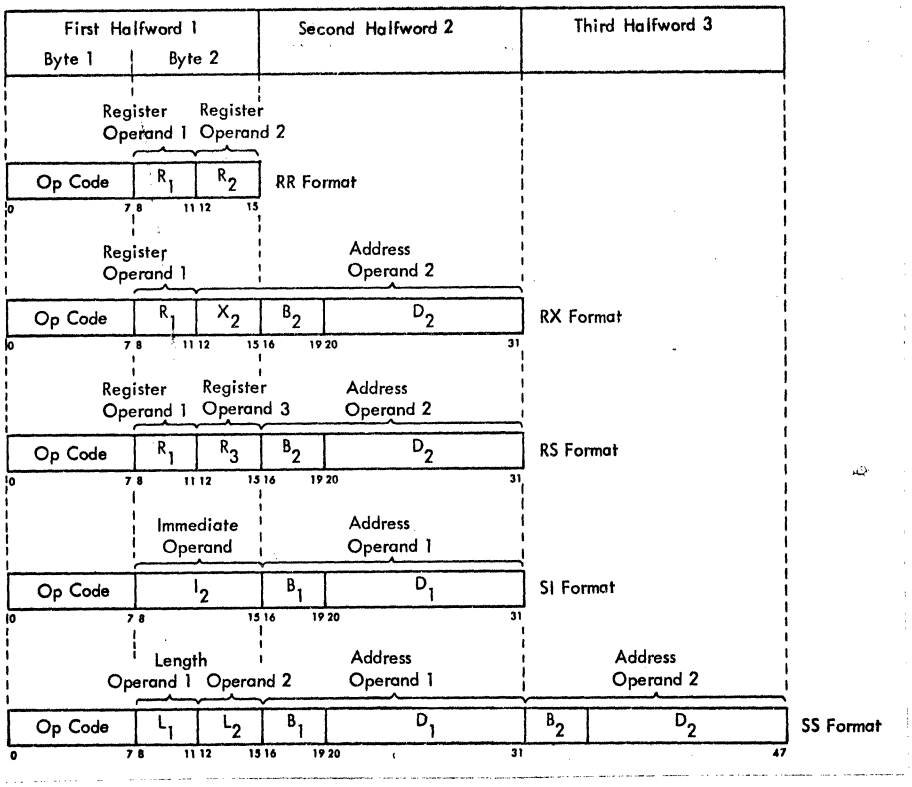


Figure 10. Five Basic Instruction Formats

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.

Instruction Length Recording

Bit Positions (0-1)	Instruction Length	Instruction Format
00	One halfword	RR
01	Two halfwords	RX
10	Two halfwords	RX or SI
11	Three halfwords	SS

The second byte is used either as two 4-bit fields or as a single eight-bit field. This byte can contain the following information:

Four-bit operand register specification (R1, R2, or R3)
Four-bit index register specification (X_2)
Four-bit mask (M_1)
Four-bit operand length specification (L_1 or L_2)
Eight-bit operand length specification (L)
Eight-bit byte of immediate data (I_2)

In some instructions a four-bit field or the whole second byte of the first halfword is ignored.

The second and third halfwords always have the same format:

Four-bit base register designator (B_1 or B_2), followed by a 12-bit displacement (D_1 or D_2).

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 or floating-point 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:

Base Address (B) is a 24-bit number contained in a general register specified by the program in the B 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 any 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) is a 24-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 within an array.

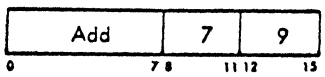
Displacement (D) is a 12-bit number contained in the instruction format. It is included in every address computation. The displacement provides for relative addressing up to 4095 bytes beyond the element or base address. In array-type calculations the displacement can be used to specify one of many items 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 24-bit positive binary integers. The displacement is similarly treated as a 12-bit positive binary integer. The three are added as 24-bit binary numbers, ignoring overflow. Since every address includes a base, the sum is always 24 bits long. The address bits are numbered 8-31 corresponding to the numbering of the base address and index bits in the general register.

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 of 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.

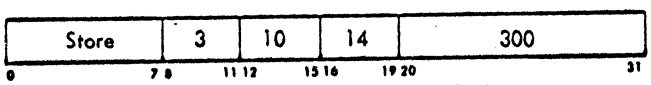
As an aid in describing the logic of the instruction format, examples of two instructions and their related instruction formats follow.

RR FORMAT



Execution of the ADD instruction adds the contents of general register 9 to the contents of general register 7 and the sum of the addition is placed in general register 7.

RX FORMAT



Execution of the STORE instruction stores the contents of general register 3 at a main-storage location addressed by the sum of 300 and the low-order 24 bits of general register 14 and 10.

Sequantial Instruction Execution - Normally, the operation of the CPU is controlled by the instructions taken in sequence. An instruction is fetched from a location specified by the instruction address in the current PSW. The instruction address is then 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.

Conceptually, all halfwords of an instruction are fetched from storage after the preceding operation is completed and before execution of the current operation, even though physical storage word size and overlap of instruction execution with storage access may cause actual instruction fetching to be different. Thus, it is possible to modify an instruction in storage by the immediately preceding instruction.

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

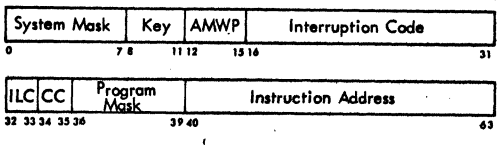
Branching - The normal sequential execution of instructions is changed when reference is made to a subroutine, when a two-way choice is encountered, or when a segment of coding, such as a loop, is to be repeated. All these tasks can be accomplished with branching instructions. Provision is made for subroutine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

Decision-making is generally and symmetrically provided by the BRANCH ON CONDITION instruction. This instruction inspects a two-bit condition code that reflects the result of a majority of the arithmetic, logical, and I/O operations. Each of these operations can set the code in any one of four states, and the conditional branch can specify any selection of these four states as the criterion for branching. For example, the condition code reflects such conditions as nonzero, first operand high, equal, overflow, channel busy, zero, etc. Once set, the condition code remains unchanged until modified by an instruction that reflects a different condition code.

The two bits of the condition code provide for four possible condition code settings: 0, 1, 2, and 3. The specific meaning of any setting is significant only to the operation setting the condition code.

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 branches, being specialized, provide increased performance for these tasks.

Program Status Word - A double word, the program status (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 11 shows the PSW format.



0-7	System mask	14	Wait state (W)
0	Channel 0 mask	15	Problem state (P)
1	Channel 1 mask	16-31	Interruption code
2	Channel 2 mask	32-33	Instruction Length code (ILC)
3	Channel 3 mask	34-35	Condition code (CC)
4	Channel 4 mask	36-39	Program mask
5	Channel 5 mask	36	Fixed-point overflow mask
6	Channel 6 mask	37	Decimal overflow mask
7	External mask	38	Exponent underflow mask
8-11	Protection key	39	Significance mask
12	ASCII-8 mode (A)	40-63	Instruction address
13	Machine check mask (M)		

Figure 11. Program Status Word Format

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

Each class has two related PSW's called "old" and "new" in unique main-storage locations (Figure 12). 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.

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 keep I/O and external interruptions pending, the program mask may cause four of the 15 program interruptions to be ignored, and the machine-check mask may cause machine-check interruptions to be ignored. Other interruptions cannot be masked off.

	Address	Length	Purpose
0	0000 0000	double word	Initial program loading PSW
8	0000 1000	double word	Initial program loading CCW1
16	0001 0000	double word	Initial program loading CCW2
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
64	0100 0000	double word	Channel status word
72	0100 1000	word	Channel address word
76	0100 1100	word	Unused
80	0101 0000	word	Timer
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
128	1000 0000		Diagnostic scan-out area*

* The size of the diagnostic scan-out area depends upon the particular system's CPU and I/O channels.

Figure 12. Permanent Storage Assignments

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.

Input/Output Interruption - An I/O interruption provides a means by which the CPU responds to conditions in the channels and I/O units.

An I/O interruption can occur only when the mask bit associated with the channel is set to one. The address of the channel and I/O unit involved are recorded in bits 16-31 of the old PSW. Further information concerning the I/O action is preserved in the channel status word (CSW) that is stored during the interruption.

Program Interruption - Unusual conditions encountered in a program create program interruptions. These conditions include incorrect operands and operand specifications, as well as exceptional results. The interruption code identifies the interruption cause. Figure 13 shows the different causes that may occur.

Interruption Code	Program Interruption Cause
1 00000001	Operation
2 00000010	Privileged operation
3 00000011	Execute
4 00000100	Protection
5 00000101	Addressing
6 00000110	Specification
7 00000111	Data
8 00001000	Fixed-point overflow
9 00001001	Fixed-point divide
10 00001010	Decimal overflow
11 00001011	Decimal divide
12 00001100	Exponent overflow
13 00001101	Exponent underflow
14 00001110	Significance
15 00001111	Floating-point divide

Figure 13. Interruption Code for Program Interruption

Supervisor-Call Interruption - This interruption occurs as a result of execution of the instruction SUPERVISOR CALL. Eight bits from the instruction format are placed in the interruption code of the old PSW, permitting an identification to be associated with the interruptions. A major use for the instruction SUPERVISOR CALL is to switch from the problem-state to the supervisor state. This interruption may also be used for other modes of status-switching.

External Interruption - The external interruption provides the means by which the CPU responds to signals from the interruption key on the system control panel, the timer, and the external signals of the direct control feature.

An external interruption can occur only when system mask bit 7 in the PSW is one.

The source of the interruption is identified by the interruption code in bits 24-31 of the PSW (Figure 14). Bits 16-23 of the interruption code are made zero.

Interruption Code Bit	External Interruption Cause	Mask Bit
24	Timer	7
25	Interrupt key	7
26	External signal 2	7
27	External signal 3	7
28	External signal 4	7
29	External signal 5	7
30	External signal 6	7
31	External signal 7	7

Figure 14. Interruption Code for External Interruption

Priority Interruption - In real time applications, the computing power of the CPU needs to be coupled with the ability to respond quickly to external events, and to distinguish between many different external situations with a minimum of programming. These abilities can be provided with a special feature called Priority Interrupt. The 32 levels of priority interrupt feature are supplemental to the single level, 6 line external interrupt feature described above. The two systems are logically and physically independent.

The priority interrupt feature permits individual masking and hardware nesting of up to 32 levels of priority. Information transmitted from the interrupting source may be expanded to allow as many as 256 external conditions at each level of priority.

When the priority interrupt is requested and enabled at the same time that one of the basic System/4 Pi interrupts is requested and enabled, resolution of priority may be controlled by the programmer to permit the priority interruption to be serviced first.

Machine-Check Interruption - The occurrence of a machine check (if not masked off) terminates the current instruction, initiates a diagnostic procedure, and subsequently causes the machine-check interruption. A machine check cannot be caused by invalid data or instructions. The diagnostic scan is performed into the scan area starting at location 128. Proper execution of these steps depends on the nature of the machine check.

Priority of Interruptions - During execution of an instruction, several interruption requests may occur. Simultaneous interruption requests are honored in the following predetermined order:

Machine Check
Program or Supervisor Call
External
Input/Output

The program and supervisor-call interruptions are mutually exclusive and cannot occur at the same time.

When more than one interruption cause requests service, the action consists of storing the old PSW and fetching the new PSW belonging to the interruption which is taken first. This new PSW subsequently is stored without any instruction execution and the next interruption PSW is fetched. This process continues until no more interruptions are to be serviced. When the last interruption request has been serviced, instruction execution is resumed using the PSW last fetched. The order of execution of the interruption subroutines is, therefore, the reverse of the order in which the PSW's are fetched.

Thus, the most important interruptions - I/O, external, program or supervisor call - are actually serviced first. Machine check, when it occurs, does not allow any other interruptions to be taken.

Program States - Over-all CPU status is determined by four types of program-state alternatives, each of which can be changed independently to its opposite and most of which are indicated by a bit or bits in the PSW. The program-state alternatives are named stopped or operating, running or waiting, masked or interruptible, and supervisor or problem state. These states differ in the way they affect the CPU functions and the manner in which their status is indicated and switched. All program states are independent of each other in their functions, indication, and status-switching.

Stopped or Operating States: The stopped state is entered and left by manual procedure. Instructions are not executed, interruptions are not accepted, and the timer is not updated. In the operating state, the CPU is capable of executing instructions and being interrupted.

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 or operator intervention from the console. In the wait state, no instructions are processed, the timer is updated, and 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 the system, program, and machine interruptions. When the CPU is interruptible for a class of interruptions, these interruptions are accepted. When the CPU is masked, the system interruptions remain pending, while the program and machine-check 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, all I/O instructions 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.

2.3.5 Protection Features

Two protection features are available. These features make it possible to protect the contents of main storage from destruction or misuse. When the store-protection feature is installed, attempts to modify storage are monitored. The addition of the fetch-protection feature to the store-protection feature provide for monitoring of all accesses to storage.

Protection is achieved by dividing main storage into blocks of 2,048 bytes, and by associating a five-bit key with each block. Two instructions - SET STORAGE KEY and INSERT STORAGE KEY - are provided for assigning and inspecting the code in a key. The same code may be used in many keys.

A user's right of access to storage is identified by a four-bit protection key. For references caused by the CPU, the protection key in the current PSW is used; accesses by channels are controlled by the protection key assigned to the associated I/O operation.

When protection applies to a main-storage reference, the key in storage is compared with the protection key associated with the reference. Access to the location, for both operands and instructions, is granted only when the two keys match. The keys are said to match when the four high-order bits of the key in storage are equal to the protection key or when the protection key is zero. When store-and-fetch protection is installed, the low-order bit of the key in storage is used to specify whether or not fetching is to be monitored.

When a protection mismatch is detected, the content of the protected main-storage location remains unaltered. A protection violation due to a CPU reference causes the instruction to be suppressed or terminated and program execution to be altered by an interruption. A violation due to an I/O operation causes the I/O operation to be terminated, with the protection mismatch indicated in the channel status word stored at the end of an I/O operation.

2.3.6 Timer Feature

The timer consists of a full word in main-storage location 80. The timer word is counted down at a rate determined by an external pulse source. The timer word is treated as a signal integer following the rules of fixed-point arithmetic. An external interruption condition is signalled when the value of the timer word goes from positive to negative.

An updated timer value is available at the end of each instruction execution but is not updated in the stopped state. The timer is changed by addressing storage location 80. As an interval timer, the timer is used to measure elapsed time over relatively short intervals. It can be set to any value at any time.

2.3.7 Direct Control Feature

The direct control feature provides two instructions, READ DIRECT and WRITE DIRECT, and six external interruption lines. The read and write instructions provide for the transfer of a single byte of information between an external device and the main storage of the system. It is usually most desirable to use the data channels of the system to handle the transfer of any volume of information and use the direct data control feature to pass controlling and synchronizing information between the CPU and special external devices.

Each of the six external signal lines, when pulsed, sets up the conditions for an external interruption.

2.3.8 Multisystem Operation

The design of System/360 permits communication between individual CPU's at several transmission rates. The communication is possible through shared control units, through a channel-to-channel adapter, and through shared storage. Interconnection of CPU's is further enhanced by the direct control feature (described in the previous section), which can be used to signal from one CPU to another, and by facilities for direct address relocation, malfunction indication, and external CPU initialization.

The relocation procedure applies to the first 4,096 bytes of storage. This area contains all permanent storage assignments and, generally, has special significance to supervisory programs. The relocation is accomplished by inserting a 12-bit prefix in each address which has the high-order 12 bits set to zero and hence, pertains to location 0-4095. Two manually set prefixes are available to permit the use of an alternative area when storage malfunction occurs. The choice between the prefixes is determined by a prefix trigger set during initial program loading.

To alert one CPU to the possible malfunction of another CPU, a machine check-out signal is provided, which can serve as an external interruption to another CPU.

Finally, provision is made for starting one CPU by a signal from another CPU.

2.3.9 Input and Output

The following information is introductory in nature. For thorough definition of the input/output system, see System/360 Principles of Operation.

Input/Output Devices and Control Units - Input/output operations involve the transfer of information to or from main storage and an I/O device. Input/output devices include such equipment as card read punches, magnetic tape units, disk storage, drum storage, typewriter-keyboard devices, printers, teleprocessing devices, and process control equipment.

Many I/O devices function with an external document, such as a punched card or a reel of magnetic tape. Some I/O devices handle only electrical signals, such as those found in process-control networks. In either case, I/O device operation is regulated by a control unit. The control-unit function may be housed with the I/O device, as is the case with a printer, or a separate control unit may be used. In all cases, the control-unit function provides the logical and buffering capabilities necessary to operate the associated I/O device. From the programming point of view, most control-unit functions merge with I/O device functions.

Each control unit functions only with the I/O device for which it is designed, but each control unit has standard-signal connections with regard to the channel to which it is attached.

Input/Output Interface - So that the CPU may control a wide variety of I/O devices, all control units are designed to respond to a standard set of signals from the channel. This control-unit-to-channel connection is called I/O interface. It enables the CPU to handle all I/O operations with only four instructions.

Special Purpose Channels - In order to effectively integrate Model EP into existing systems and to accommodate unusual data transfer situations such as those arising from radar inputs, special purpose channels may be designed and attached to a Model EP. These channels may operate as special purpose transfer paths under control of a standard channel or they may be entities with unique control instructions and architecture. Such channels may be used in addition to or in place of standard channels described below.

Channels - Standard channels connect with the CPU and main storage and, via the I/O interface, with control units. Channels control transfer of data between I/O devices and main storage. The CPU program is free to resume processing data after initiating channel control of I/O operations, including concurrent operation of several I/O devices on a single channel, as well as concurrent operation of several other channels.

A channel may be an independent unit, complete with necessary logical and storage capabilities, or it may time-share CPU facilities and be physically integrated with the CPU. In either case, channel functions are identical. Channels may be implemented, however, to have different maximum data transfer capabilities.

The System/360 has two types of channels: multiplexor and selector. The channel facility necessary to sustain an operation with an I/O device is called a subchannel. The selector channel has one subchannel; the multiplexor channel has multiple subchannels.

Channels have two modes of operation: burst and multiplex.

In the burst mode, all channel facilities are monopolized for the duration of data transfer to or from a particular I/O device. The selector channel functions only in the burst mode.

The multiplexor channel functions in either the burst mode or in the multiplex mode. In the multiplex mode, the multiplexor channel can sustain concurrent I/O operations on several subchannels. Bytes of data are interleaved together and routed to or from the selected I/O devices and to or from the desired locations in the main storage. In the multiplex mode, the multiplexor channel's single data path is time-shared by the concurrently operating I/O devices, each of which uses a subchannel.

In burst mode, the multiplexor channel's data path is pre-empted by a burst-mode device; other devices attached to the channel cannot transfer data or communicate with the CPU until the burst-mode device releases the data path.

Some I/O devices can operate only in burst mode. Other I/O devices have a manual switch in the control unit that may be set to a burst-mode or to a multiplex-mode position, when attached to a multiplexor channel. When attached to a selector channel, an I/O device can operate only in burst mode.

Input/Output Instructions - The System/360 uses only four I/O instructions:

START I/O
TEST I/O
HALT I/O
TEST CHANNEL

Input/output instructions can be executed only while the CPU is in the supervisor state.

Start I/O - The START I/O instruction is used to initiate an I/O operation. The address part of the instruction specifies the channel and I/O device.

Test I/O - Execution of the TEST I/O instruction sets the condition code in the PSW to indicate the state of the addressed channel, subchannel, and I/O device, and may cause a CSW to be stored. The instruction may be used to clear I/O interruption conditions, selectively by device.

Halt I/O - The HALT I/O instruction terminates a channel operation.

Test Channel - Execution of the TEST CHANNEL instruction sets the condition code in the PSW to indicate the state of the channel addressed by the instruction. The resulting condition code indicates one of the following: channel available, interruption condition in channel, channel working, or channel not operational.

Input/Output Operation Initiation - An I/O operation is initiated by a START I/O instruction. If the necessary channel and device facilities are available, START I/O is accepted and the CPU continues its program. The channel independently governs the I/O device specified by the instruction.

Channel Address Word - Successful execution of START I/O causes the channel to fetch a channel address word (CAW) from the main-storage location 72. The CAW specifies the byte location in main storage where the channel program begins.

Figure 15 shows the format for the CAW. Bits 0-3 specify the storage-protection key that will govern the I/O operation. Bits 4-7 must contain zeros. Bits 8-31 specify the location of the first channel command word (CCW).

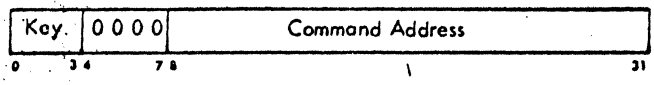


Figure 15. Channel Address Word Format

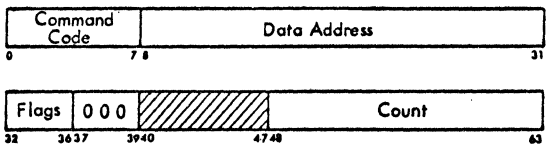
Channel Command Word - The byte location specified by the CAW is the first of eight bytes of information that the channel fetches from main storage. These 64 bits of information are called a channel command word (CCW). Only the START I/O instruction may cause the channel to fetch CCW's.

One or more CCW's make up the channel program that directs channel operations.

A channel command word can specify one of six commands:

Read
Write
Read Backward
Control
Sense
Transfer In Channel

If more than one CCW is to be fetched, the CCW's are to be fetched sequentially, except when transfer in channel is encountered. Figure 16 shows the format for CCW's.



The command code specifies the operation to be performed (read, write, rewind, etc.).

The data address specifies the first byte location in main storage for a data transfer type of operation.

The flag bits may specify chaining to another CCW, suppression of a possible incorrect-length indication, etc.

The count specifies the number of bytes for a data transfer operation.

Figure 16. Channel Command Word Format

Input/Output Commands

Read - The read command causes data to be read from the selected I/O device and defines the area of main storage to be used.

Write - The write command causes a write operation on the selected I/O device and defines the data in main storage to be written.

Read Backward - The read-backward command causes a read operation in which the characters are read from the external document in reverse order by the I/O device. Bytes read backward are placed in descending main storage locations.

Control - The control command contains information used to control the selected I/O device. This control information is called an order. Order information may be entirely contained in the command code, or the control command may provide a data address and byte count for additional order information in main storage such as the address of a particular disk storage track.

Orders are peculiar to the particular I/O device in use; orders can specify such functions as rewinding a tape unit, loading a tape cartridge, or line skipping on a printer. A control command may cause mechanical motion by an I/O device, or it may specify a function altogether electronic in nature, such as setting the recording density for a tape unit operation.

The general relationship of I/O instructions, commands, and orders is shown in Figure 17.

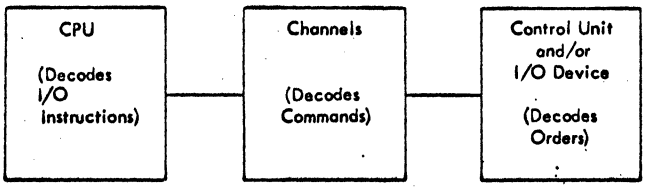


Figure 17. Relationship of I/O Instructions, Commands, and Orders

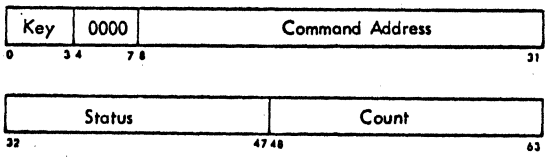
Sense - The sense command specifies the beginning main storage location to which sense information is transferred from the selected control unit. One or more bytes of sense data may be specified, depending upon the type of I/O device. The sense data provides detailed information concerning the selected I/O device, such as a stacker-full condition of a card reader or a file-protected condition of a reel of magnetic tape on a tape unit. Sense data have significance peculiar to the type of I/O device involved.

Transfer in Channel - The transfer-in-channel (TIC) command specifies the location of the next CCW to be fetched and used by the channel. The TIC command is used whenever the programmer wants to specify a CCW that is not located at the next higher double word location in main storage. The TIC command permits a programmer to cause execution of any CCW, including a CCW immediately preceding a TIC command, except that the channel will not permit a TIC command to specify execution of another TIC command. Also, the CAW may not address a TIC command.

Input/Output Termination - Input/output operations terminate with the device and channel signalling end of operation and a request for an I/O interruption.

A command can be rejected during an attempt to execute a START I/O, however, by a busy condition, by a channel programming error, etc. The condition code set in the PSW by an unsuccessful START I/O instruction will indicate one of the following: that a channel status word (CSW) has been stored to detail the conditions that precluded initiation of the I/O operation, that the equipment is busy, or that the addressed equipment is not operational.

Channel Status Word - The channel status word (CSW) provides information about the termination of an I/O operation. It can be formed or reformed by START I/O, TEST I/O, HALT I/O or by an I/O interruption. The instruction TEST CHANNEL does not affect the CSW. Figure 18 shows the CSW format.



The key field contains the protection key used in the last operation.
The command address specifies the location plus 8 of the last CCW used.
The status field contains a unit status byte and a channel status byte. The unit status byte may indicate one or more conditions, such as control unit end, device end, busy, etc. The channel status byte may indicate a channel programming error, a channel data check, etc.
The count field specifies the residual count of the last CCW used.

Figure 18. Channel Status Word Format

Input/Output Interruptions - Input/output interruptions are caused by termination of an I/O operation or by operator intervention at the I/O device. An I/O interruption stores the current PSW in the I/O old PSW location, and places the I/O new PSW in control of the system. The I/O new PSW, when made current by an I/O interruption, may cause CPU interrogation of the channel status word, or take whatever action is considered appropriate by the programmer.

An I/O interruption request may be initiated by an I/O interruption condition in a device, a control unit, or a channel. When a channel has multiple I/O interruption requests pending, it establishes a priority sequence for them before initiating an I/O interruption request to the CPU. Conditions responsible for I/O interruption requests remain pending in the I/O devices or channels until they are accepted by the CPU.

2.3.10 System Control Panel

The system control panel provides the switches, keys, and lights necessary to operate and control the system. The need for operator manipulation of manual controls is held to a minimum by the system design and the governing supervisory program. The result is fewer and less serious operator errors.

System Control Panel Functions - The main functions provided by the system control panel are the ability to: reset the system; store and display information in main storage, in registers, and in the PSW; and load initial program information.

System Reset - The system-reset function resets the CPU, the channels, and on-line control units and I/O devices. In general, the system is placed in such a state that processing can be initiated without the occurrence of machine checks, except those caused by subsequent machine malfunction.

Store and Display - The store-and-display function permits manual intervention in the progress of a program. The function may be provided by a supervisory program in conjunction with proper I/O equipment and the interrupt key. Or, the system-control-panel facilities may be used to place the CPU in the stopped state, and then to store and display information in main storage, in general and floating-point registers, and in the instruction-address portion of the PSW.

Initial Program Loading - The initial-program-loading (IPL) procedure is used to begin or renew system operation. The load key is pressed after an input device is selected with the load-unit switches. This causes a read operation at the selected input device. Six words of information are read into main storage and may be used for reading more information into any part of main storage. Upon completion of the IPL read operation, the double word from location 0 is made the current PSW for subsequent control of the system.

The system controls are divided into three sections: operator control, operator intervention, and customer engineering control.

Operator Control Section - This section of the system control panel contains the operator controls required when the CPU is operating under supervisory program control.

The main functions provided are the control and indication of power, the indication of system status, and operator-to-machine communication. These include:

Power-on key
Power-off key
Interrupt key
Wait light
Manual light
System light
Test light
Load light
Load key

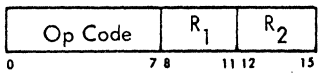
Operator Intervention Section - This section of the system control panel provides controls required for operator intervention into normal programmed operation. These include:

System reset key
Stop key
Start key
Rate switch (single cycle or normal processing)
Storage-select switches
Address switches
Data switches
Store key
Display key
Set IC key
Address compare switches.

2.4 Instruction Sets

Detailed descriptions of each instruction set and of instruction execution, status switching, interruptions and input/output operations are contained in System/360 Principles of Operation. The chart below summarizes the instructions available.

RR Format

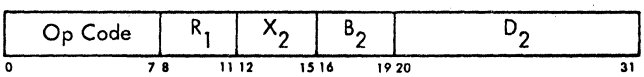


Fixed Point		*Floating Point*	
Load		Load S/L	
Load and Test		Load and Test S/L	
Load Complement		Load Complement S/L	
Load Positive		Load Positive S/L	
Load Negative		Load Negative S/L	
Add		Add Normalized S/L	
Add Logical		Add Unnormalized S/L	
Subtract		Subtract Normalized S/L	
Subtract Logical		Subtract Unnormalized S/L	
Compare		Compare S/L	
Multiply	E	Halve S/L	
Divide	E	Multiply S/L	
		Divide S/L	

Logical		*Status Switching*	
Compare		Set Program Mask	2
AND		Supervisor Call	3
OR		Set Storage Key	Z
Exclusive OR		Insert Storage Key	Z

Branching	
Branch on Condition	1
Branch and Link	
Branch on Count	

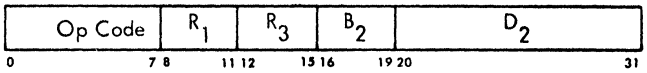
RX Format



Fixed Point		*Floating Point*	
Load H/F		Load S/L	
Add H/F		Add Normalized S/L	
Add Logical		Add Unnormalized S/L	
Subtract H/F		Subtract Normalized S/L	
Subtract Logical		Subtract Unnormalized S/L	
Compare H/F		Compare S/L	
Multiply H		Multiply S/L	
Multiply F	E	Store S/L	
Divide F	E	Divide S/L	
Convert to Binary			
Convert to Decimal			
Store H/F			

Logical		*Branching*	
Compare		Branch on Condition	1
Load Address		Branch and Link	
Insert Character		Branch on Count	
Store Character		Execute	
AND			
OR			
Exclusive OR			

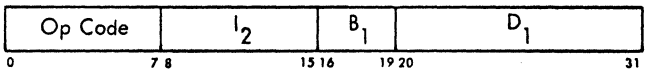
RS Format



Fixed Point		*Logical*	
Load Multiple		Shift Left Single	2
Store Multiple		Shift Right Single	2
Shift Left Single	2	Shift Left Double	E,2
Shift Right Single	2	Shift Right Double	E,2
Shift Left Double	E,2		
Shift Right Double	E,2		

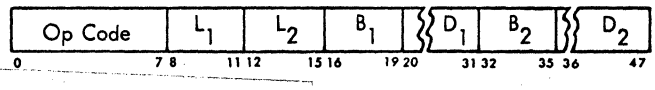
Branching
Branch on High
Branch on Low-Eq

SI Format



Input/Output		*Status Switching*	
Start I/O	4	Load PSW	4
Test I/O	4	Set System Mask	4
Halt I/O	4	Write Direct	Y
Test Channel	4	Read Direct	Y
		Diagnose	
Logical		Test and Set	4
Move			
Compare			
AND			
OR			
Exclusive OR			
Test Under Mask			

SS Format



Logical	
Move	5
Move Numeric	5
Move Zone	5
Compare	5
AND	5
OR	5
Exclusive OR	5
Translate	5
Translate and Test	5
Edit	T,5
Edit and Mark	T,5

FORMAT NOTES

E	R_1 must be even
F	Fullword
H	Halfword
L	Long
S	Short
T	Decimal feature
Y	Direct control feature
Z	Protection feature
1	R_1 used as mask M_1
2	R_2 or R_3 ignored
3	R_1 and R_2 used as immediate information
4	I_2 ignored
5	L_1 and L_2 used as eight-bit L field

All floating-point instructions are part of the floating-point feature.

3.0 SOFTWARE

The system programming provided for 4 Pi/EP support includes the assembler and compiler for the language adopted, the program check-out system (simulator)and serviceprograms. These packages execute on a host S/360 and interface with OS/360. Code is produced for execution on EP. In addition, a control system is being developed for execution on EP in a real time environment.

3.1 Language Translators

Assembler

The assembler for EP is similar to the OS/360 assembler. Additions and modifications are as follows:

1. The assembler will allow for easy definition of operation codes since it may frequently be called upon to assemble instructions which are not found in the normal instruction set for the 4 Pi/EP.

2. Dynamic Relocation

   The assembler will provide many of the tables required for operation of the dynamic relocation procedure in the Control System.

Compiler

The language of the compiler designed for 4 Pi support provides the following features:

1. Real Time Control

   Scheduling of program segments or procedures are accomplished by the control system based on information provided by the user. The language contains statements that allow the user to dynamically request and free data storage areas and to allow the loading and overlaying of program and data blocks under program control. Statements which facilitate the scheduling and execution of program segments are included. In addition the language provides a flexible method of allowing user control of interrupt conditions.

2. Optimization Control

Optimization of object code is a primary design criteria, with the user retaining control over specification of the type of optimization required for his application. The user is allowed to specify Time, Space, Reentrant or Recursive code generation as required. The user control, in conjunction with the compiler objectives of producing optimized code through (a) Redistribution, (b) Elimination, (c) Simplification, and (d) Code selection, assists in controlling the use of hardware resources.

3. Computational Capability

The language provides a powerful computational capability in order to meet the complex data manipulation and calculation requirements that arise in a real time environment.

The ability to describe and manipulate complex data arrangements includes such features as floating, fixed, complex and integer arithmetic on both binary and decimal data with multiple and variable precision; relational and logical operations; loop control; array operations; subscripting; complex data structures and strings with bit and character referencing; movement of data with optional editing; and an ability to use machine language code.

4. Input/Output Capabilities

The language provides a notation for the control of Input/Output devices at a level of device independence consistent with the requirements of a particular application. The language allows symbolic referencing of devices consistent with the philosophy of reducing system overhead and maximizing the flexibility of device usage.

3.2 Program Checkout System (Simulator)

The problems encountered in checkout of time dependent programs require facilities which allow the programmer to control the real time environment. This is provided with a simulator program which executes on a S/360 host computer and provides the necessary real time environment for 4 Pi EP programs. The simulator, in addition to providing checkout facilities, can provide valuable information through flow analysis of the executing programs which can then be used for further program optimization. Facilities are designed for interfacing with a user supplied FORTRAN IV program which provides the real time inputs and interrupts for specific applications and allows the user to simulate EP with full interrupt handling capabilities. The simulation is handled through a set of interfacing sub-

routines which may be called by the user. A comprehensive list of user calling sequences are made available including time snaps, dumps and traces. Other provisions include input/output, interrupt, and real time control over the simulation.

3.3 Service Programs

The main items considered here are Program Preparation Processor facilities, and the system generation and maintenance facilities.

A. Program Preparation Processor

The Program Preparation Processor will operate on the S/360 host computer and provide the following facilities:

1. Combine and link program modules which were separately compiled or assembled into one executable program for execution on 4 Pi EP.

2. Construct user's library files.

3. Retrieve modules from the user's library files to be incorporated into the program.

4. Construct tables and directories required for control of the program and modules.

5. Provide error messages and program cross reference tables.

6. Generate a program test library for use by the simulator.

The Program Preparation Processor provides functions which are similar to those provided in OS/360 with the extensions required for interface with the 4 Pi Control System.

B. System Generation and Maintenance

An essential feature which frequently receives insufficient attention is system generation and maintenance. System maintenance facilities provide for updating symbolic files as well as generating and incorporating modifications to the system. In a modular system, the generation facilities must allow for easy combination of modules to provide facilities which can be tailored to the application.

3.4 Control System

An important design goal of the control system is to provide support to a wide range of real time applications and hardware configurations without penalizing users with excessive and unnecessary overhead. To provide this flexibility, a basic control system is being designed and a number of options and suboptions will be provided. For a given application, the basic system, together with the optional modules selected can be assembled to produce a specific system.

The following functions are included in the control system:

1. Scheduling

   The scheduling function manages the sequence in which programs are executed. By associating a priority with a specific program, the user determines the sequence of application program execution.

   Optionally, the user can select one of two scheduling algorithms at system generation time. The first algorithm provides for the processing of non-cyclic type programs, the second provides for cyclic programs (those programs which require periodic repeated entry).

   The final user option is the specification that an interval timer is to be associated with the scheduling routine. The interval timer provides a means, other than program segmentation, of returning control to the scheduler for the possible processing of higher priority programs.

2. Data Management

   The data management function controls all operations associated with input/output devices. These operations include channel scheduling, cataloging and storage of data files, data transfer between executable and auxiliary storage, program and data files control, and detection of errors occurring during input/output operations. As with the other functions of the control system the extent of control is specified by the user of the system.

3. Resources Management

   The resources management function controls, allocates and accounts for the following system resources:

   Core storage

   Auxiliary storage
   Peripheral devices

There is a basic resources management function which performs a configuration check on programs requested for loading, preallocating, at load time, core and peripheral devices. Resources and management also allocates core, within the preallocated amount, at program load time and when modules are loaded dynamically; allocates core bins dynamically; and allocates peripheral devices, within the preallocated amount, when requested during program execution. Upon request and when a program is cancelled or terminated, resources are returned to the system.

An option to resources management provides buckets management. Direct Access Storage devices designated as "bucket" devices are managed by this option. Buckets are preallocated to a program at load time; allocated and released to it, within the preallocated amount, when requested during execution and normally released at program cancellation time.

Core coalescing and rollout are offered as suboptions of Basic Resources Management to make space in core as required for loading programs of higher priority. In order to optimize core use, all programs can be relocated dynamically in core with the exception of a program currently being executed by another CPU and a program which is active in an I/O operation.

4. System Communication

The function of the control program which provides for system interface with the user is called system communication. Such communications provides for messages to and from operator stations and between application program and system. Standard system functions such as load, execute, terminate, change priority, etc., are provided.

The message set is minimal and is available in all levels of the system. The construction of the communication section is such as to allow the easy addition of application dependent messages to the system.

5. Interrupt Supervision

The interrupt routines provides facilities for handling I/O, machine, program, SVC, external, and priority interrupts. Options provide for the batching of I/O interrupts for periodic processing and also allow the user to specify whether external interrupts are to be processed immediately or the processing routine driven by priority from the program request queue.

4.0 MICROPROGRAMMING CONCEPTS AND TECHNIQUES

This section was not available for inclusion in this preliminary edition of Model EP, Engineering Design. There are several excellent descriptions of microprogramming as it is applied to System/360. The information in these articles is pertinent and applicable to Model EP in concept if not in detail.

1. The Design of Processor Controls Using a Read-Only Store

   Dorothy L. Schnable — IBM Report TR00.1318, Aug. 1965

2. Control Storage Systems

   S. M. Pitkowsky — IBM Report TR00.1360 (IBM Conf.) Nov. 1965

3. Microprogramming Manual for the IBM System/360 Model 50

   S. S. Husson — IBM Report TR00.1479 (IBM Conf.) July 1966

5.0 DETAILED DESCRIPTION

5.1 CPU Data Flow

The CPU data flow consists primarily of two parallel paths which may be active simultaneously. One is a thirty-two bit wide adder/shifter path fed by several thirty-two bit registers. The other is an 8-bit wide mover path which performs logical operations on bytes of data selected from various CPU registers. The CPU operates on a 0.416 microsecond cycle time which is the time required for a register-to-register transfer through the adder/shifter path or through the mover path. Figure 5-1 is a block diagram of the CPU data flow.

The data flow shown on this figure may be segregated into three different groups. The registers to the left and right of the figure (AB, PSW, JA, JB, Q, R, E, etc.) are largely concerned with control, instruction fetching and storage accessing. The local storage at the top of the diagram provides the 16 registers which may serve as accumulators, index registers or as temporary storage registers. The working registers WA - WF and registers BR, MR, etc., make up the general purpose data flow which performs the detailed execution of each instruction. This general purpose data flow is also shared by the instruction fetch system and integrated I/O system.

Register and Data Transfer Path Descriptions

- AB Register (address buffer) - This is a 16-bit register used to hold all addresses associated with CPU storage requests.

- MK Register (mark register) - This is a 4-bit register which is normally set simultaneously with the AB register. It is used to control store or regenerate for each of the four bytes of the 36-bit storage word.

- PSW Register (Program Status Word Register - Status information concerning the current operating program is contained in several groups of registers from where it controls system operations essential to that program mode. These groupings of registers, though not physically adjacent, are collectively referred to as the PSW Register. The logical register is 64 bits in length and is shown as an adjacent grouping on the flow diagram. The first 16 bits contain the system mask, the key and the AMWP bits. The remaining 16 bits of the first word is the interruption code which does not require a physical register. This code is generated by the CPU at the time of the interruption and is not retained when a previously stored status word is reloaded. The second PSW word contains the Instruction Length Code, the Condition Code, and the Program Mask and a 24 bit instruction address. The four high order address bits are not physically implemented and the remaining 20 bit physical register is labeled JA on the flow diagram.

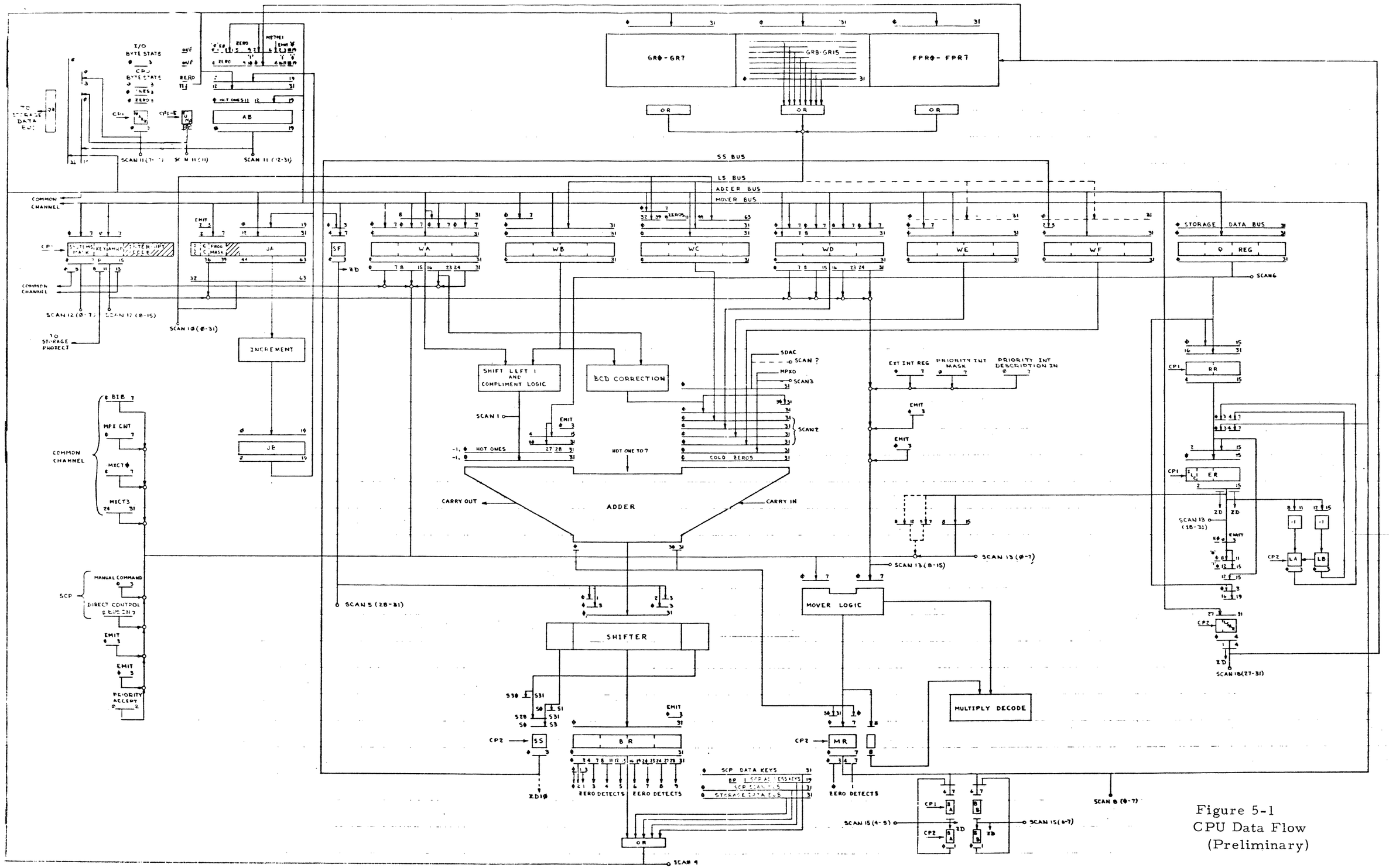


Figure 5-1
CPU Data Flow
(Preliminary)

- JA/JB Registers (Instruction Address Registers) - 20 bits of the JA register together with the instruction address incrementer and the 20 bit JB register are used to keep track of the address in main storage of the next instruction. Because all instructions are either 2, 4, or 6 bytes long, and main storage delivers data in one word (4 byte) increments, the instruction address is updated by either two or four each time it is used.

- Q Register (instruction buffer) - This is a 32-bit register which is used to buffer an instruction word. It is fed directly from the main storage data registers and has outgates going to the R Register, parallel adder and local storage address register. After the last halfword of Q is gated to R, the next two halfwords in the instruction stream are fetched from main storage and placed in Q. This is overlapped with execute time when possible.

- R Register (intermediate buffer register) - This is a 16-bit (one halfword) transitory register between Q and E. It extends Q register to a three halfword instruction buffer to allow overlapping the reloading of Q for RX, RS, and SI instructions that are not on word boundaries.

- E Register (execution register) - This is a 16-bit register containing the first halfword of the instruction which is being executed. Bits 0 and 1 of instruction specify the instruction length. These two bits also form the instruction length code (ILC) of the program status word (PSW). The high-order 8 bits (bits 0-7) are used as function branch bits for operation decoding. The low-order 8 bits (8-15) feed the local store address register and the two length counters.

- WA Register (working register A) - This is a 32-bit register which is a primary source of data for the adder left input. It can be gated direct true, direct complement, left 1 true, and left 1 complement to adder left. In addition, any single byte of data from WA can be gated to the mover left input. Ingating to WA is from the mover bus or the main bus.

- WB Register (working register B) - This is a 32-bit register which is a primary source of data for the adder left input. It can be gated direct true, direct complement, left 1 true and left 1 complement to adder left. Ingating to WB is from local store or the main bus.

- WC Register (working register C) - This is a 32-bit register which is a primary source of data for the adder right input. It is gated direct true to adder right. Ingating to WC is from the main bus, Local Store or the instruction address register JA. This latter path is used for formatting program status words and for address store compares.

- WD Register (working register D) - This is a 32-bit register which is a primary source of data for the adder right input. It is gated direct true to adder right. In addition, any byte of data from WD can be gated to the mover right input. Ingating to WA is from the mover bus or the main bus.

- WE Register (working register E) - This is a 32-bit register which is gated direct true to the adder right input. It is used for VFL and floating-point instructions. Ingating to WD is from the main bus.

- WF Register (working register F) - This is a 32-bit register which is gated direct true to the adder right input. It is used for VFL and floating-point instructions. Ingating to WF is from the main bus.

- Adder - The main adder is 32 bits wide; the extra high-order bit being necessary to implement the 2 bit-at-a-time multiply algorithm (left inputs to the adder can be gated direct or left 1). The adder employs carry-look ahead techniques and requires eight delays for a full add operation.

- Mover - The mover is a functional unit used to manipulate one byte blocks of data in half-byte increments. In addition, the mover can perform the logical operation "and", "or", and "exclusive or" on two bytes of data. During multiply the mover path is used to strip off bits of the multiplier and assemble bytes of partial product. During divide the mover path is used to couple bits of low-order dividend into the main shifter and accumulate bytes of quotient.

- BR (Bus Register) - This is a 32-bit register which temporarily holds the shifted sum so that it may be returned to a first rank working register or local storage. Data to be written into main storage is also gated from BR.

- MR (mover register) - This is an 8-bit register which temporarily holds a byte of data so that it may be returned to a first rank working register.

- SS Register - This is a 4-bit register which buffers the high-order or low-order 4 bits of the adder output during shift operations. The SS register together with the SF register provides coupling for double length shift operations.

- SF Register - This is a 4-bit register used in the shifter data path. When adder outputs are shifted, the bits entering the 4-bit SS register are temporarily stored in SF. Bits from SF can then be entered into the shifter to fill vacated positions on succeeding shift micro-orders. This provides coupling for double length shift operations.

- Length Counters LA and LB - These two 4-bit counters can be either incremented or decremented by 1. In addition, they can be coupled into one 8-bit counter which can be decremented by 1. These counters are used for instructions having iterative type algorithms which require counting.

- Byte Counters BA and BB - These two 2-bit counters are used to keep track of which WA and/or WB register byte is being handled through the mover path. They are used to indirectly control both byte ingating and byte outgating from these registers.

5.2 Microprogramming Control

The CPU is controlled by a permanently recorded microprogram which is stored in a read-only store (ROS) and is supplemented by conventional control logic. Each time the ROS is addressed, a microinstruction is read. When decoded, the microinstruction controls the routing of data in the CPU and provides means of selecting the next microinstruction.

Figure 5-2 is a block diagram of ROS. To understand the operation of ROS, it is helpful to note its relationship to conventional controls. Conventional controls may be characterized by sequence triggers, and by the control lines activated by the sequence triggers as a function of the operation to be performed and data conditions. Each cycle that the CPU may take represents a state of the CPU as defined by the control circuitry. Each state, in turn, specifies which control lines are to be activated during that cycle and which state is to follow next. The defined state will cause the next sequence trigger to be set in the following cycle. In some cases, the next state may be contingent upon a branch condition in which one of two or more sequence triggers must be selected.

In ROS-controlled CPU's, the sequence triggers are replaced by microinstructions or ROS words. Each ROS word consists of a predetermined bit pattern and represents a state of the CPU. The addressed ROS word controls the CPU during a particular machine cycle it is in use. When decoded, the ROS word defines all control lines that are to be activated during the machine cycle. Also contained in the ROS word is the address of the next ROS word to be used. If the address of the next ROS word is dependent on data conditions (for example, branch if overflow occurs), a base address and the conditions to be tested (branch tests) are specified in the ROS word. In this case, one ROS word is associated with each of the possible data conditions; the ROS word whose associated conditions are satisfied is the next to be addressed. Thus, ROS eliminates the need for most of the complex sequencing networks.

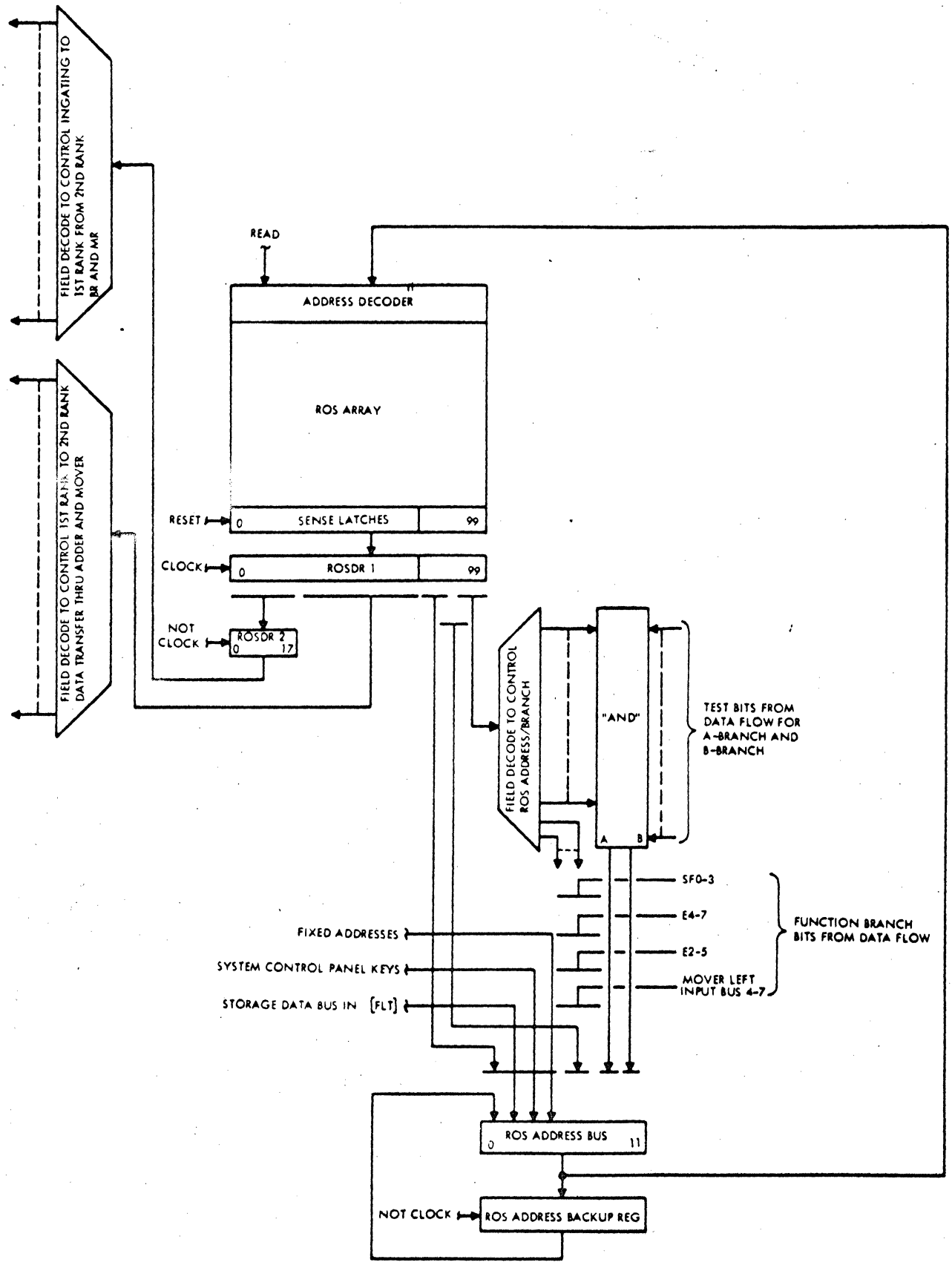


Figure 5-2 ROS Block Diagram
(Preliminary)

ROS Microinstruction (ROS) Word

Each ROS word consists of unique predetermined bit configuration grouped into 25 control fields (Table 5-1). The number of bits within a field determines the number of unique control signals (micro-orders) available within that field. (In a 4-bit field, for example, 16 distinct micro-orders can be defined, only one of which can be activated at any one time). The micro-orders are grouped functionally within the fields according to two rules.

1. All micro-orders grouped in a field must be mutually exclusive since only one micro-order within that field may be specified at a time.

2. Micro-orders that are functionally similar (such as micro-orders that control outgating to the adder right input bus) are grouped in one field for ease of decoding.

Usually, rule 1 results in rule 2.

When decoded, each micro-order activates one or more control lines that condition logic gates to perform the function specified by the micro-order. Each micro-order is assigned a mnemonic code that defines the control function performed. As an example, Table 5-2 lists the micro-orders and associated micro-commands pertaining to the adder function control field of the ROS word. The bit configurations of that field and their function are also specified.

ROS Addressing and Branching

Microinstructions are not executed from sequential read-only storage addresses under control of an instruction counter, but rather each microinstruction explicitly states its possible successors. A given microinstruction may have as many as sixty-four different successors, but most do not have more than four. The choice of a successor is made on the basis of branching tests specified by the microinstructions. This powerful branching ability of the microprogram contributes significantly to reducing the number of words of read-only storage required.

A read-only storage address consists of twelve bits. Bits 41-46 of a microinstruction furnish directly the first six address bits (0-5) of its successor. The next four bits of address of the succeeding microinstruction may come from any of several sources. If bits 47-49 of a microinstruction are not all zero, then the normal addressing mode is used, in which bits 50-53 of the microinstruction are used directly as bits 6-9 of the address of the successor. If bits 47-49 of the microinstruction are all zero, then a special addressing mode is used and bits 50-53 of the microinstruction are decoded to determine the type of address generation used. Certain of these codes cause four bits to be selected from some register of the CPU and used as bits 6-9 of the address of the successor microinstruction. This latter technique is termed a function branch.

Table 5-1

ROS FIELD ASSIGNMENT
(Preliminary)

Function	Bits	ROSDR	FLD
Parity	1	0 (for 0-29)	
Mover Left Input	4	1-4	LM
Mover Rt Input	3	5-7	RM
Spare	1	8	
Mover Action	4	9-12	MA
Mover Destination	4	13-16	MD
Adder Left Input/Action	4	17-20	AL
Adder Right Input	3	21-23	AR
Spare	1	24	LA
Adder Latch Ingating and Shift Control	5	25-29	AS
Parity	1	30 (for 30-64	
Adder Function	3	31-33	AF
Spare	1	34	
Adder Destination	6	35-40	AD
ROS Base Addr.	6	41-46	RB
ROS Addr. Cont'd Fld	3	47-49	RC
Functional Branch	4	50-53	FB
A-Branch Condition	6	54-59	AB
B-Branch Condition	5	60-64	BB
Parity	1	65 (for 65-99)	
Emit Field	4	66-69	EF
Memory Request	3	70-78	MR
Instruction/Operand Request Control	3	73-75	IC
Byte Ctr Control	2	76-77	CF
Byte Ctr BA	1	78	CA
Byte Ctr BB	1	79	CB
Length Ctr Control	4	80-83	LC
Stat-Setting	6	84-89	SS
Local Store Address	3	90-92	LS
Local Store Outgate	2	93-94	LO
Spare	3	95-97	
Main Bus Ingate Control	2	98-99	BC

Table 5-2

ADDER FUNCTION CONTROL FIELD AF (BITS 36-38)
(Preliminary)

Micro Order MNEMONIC	Bit Configurations			Function
	36	37	38	
NOP	0	0	0	Left and right inputs are added. Carries are not entered or saved.
DPDVC	0	0	1	Left and right inputs are added. Carry from position 0 is saved in carry out stat. A carry in is forced also if SF = 0.
DPADC	0	1	0	Left and right inputs are added. Carry out stat is entered as a carry in.
DVI	0	1	1	Left and right inputs are added. A carry in is forced if DCS is "0" and no carry in if DCS is "1". Carry out of adder position 0 is gated to carry stat.
C8 S	1	0	0	Left and right are added. Carry from bit 8 to bit 7 is blocked and carry out of position 8 is entered into carry stat.
O1 S	1	0	1	Left and right inputs are added. Carry out of position 0 and 1 are EXOR into C stat.
CAR S	1	1	0	Left and right inputs are added. Carry stat on if there is carry out of position 0, off if there is no carry out.
C1 S	1	1	1	Left and right inputs are added. Carry out of position 1 is set into C stat. Carry from position 8 to 7 is blocked.

The final two bits of the successor microinstruction address are referred to as bit A and bit B. They are each determined on the basis of tests specified in the microinstruction in the condition branch test fields. Typical tests are of the various stats, zero tests of the adder latch, of various counters, etc. The A and B condition branches allow a four-way branch from any microinstruction. If only a two-way branch or no branch is required, bit A or bit B may be set to a constant value.

The normal microprogram sequencing described above can be overridden by the occurrence of either of two types of events. The first type is the detection of an exceptional condition indicative of a program or machine check. Invalid addresses or address specification and protected addresses are examples of conditions detected in this manner. When such an event occurs, the normal address of the successor microinstruction derived as described above is ignored. Instead, a special wired-in fixed address, characteristic of the type of event, is forced into the read-only storage address to determine the successor microinstruction. This technique is referred to as a microprogram trap, and is analogous to a program trap in machine-language programming, except that no record is kept of the address that would normally have been used.

The second type of event that can cause extraordinary microprogram sequencing is a request by an I/O channel for service. This request indicates that a channel requires the use of the microprogram control and certain CPU facilities to handle the data or control words for an I/O unit. The request is honored by transferring the address that would normally have been used for the successor microinstruction to a backup register and forcing a fixed read-only storage address instead. This process is termed an I/O break-in. When the channel has completed use of the CPU facilities, control is returned to the original CPU microprogram by using the contents of the backup register as the next microinstruction address.

5.3 Main Storage

The basic operating main storage is packaged in 8,192 word modules. In addition, 256 words of "BUMP" storage are available in each module, all main storage words are 36 bits long. A storage block diagram and timing diagram are shown in Figures 5-3 and 5-4. The cycle time, measured from "select" to the succeeding "select" pulse is 2.5 microseconds. The access time is 900 nanoseconds, maximum, measured from the "select" pulse.

There are two operating modes for storage, "Read-Write" and "Split-Cycle".

Read-Write - This mode is the normal read-restore mode.

Split-Cycle - In this mode the storage cycle consists of three parts: read, dead time (compute time), and store. The "Select" signal initiates the read portion. The dead-time (or compute time) is under the control of the CPU and no limit is placed on this time. The store portion of the split-cycle operation is initiated by the "split-cycle store command."

The inputs to Main Storage (MS) are controlled by the Bus Control Unit (BCU). Using a busy signal generated by the storage module the BCU determines the availability of storage. Once a request for storage is honored all other requests are held until the BCU is free to handle the request and storage is not busy.

Once a request has been granted access to storage, the BCU accepts the address, data, and control bits for processing and transmission to storage. The low order 13 bits of the address is multiplexed to the Storage Address Register (SAR) of each 8K storage module. These 13 bits (2^{13}) are sufficient to specify each of the 8,192 word locations. The SAR provides these address bits to the decode circuits. The three high order bits are decoded to specify the 8K storage module required.

The "Select" signal is sent to the seclected storage to initiate the storage cycle. The storage timing initiates the address decoding in the storage module. At this time, the storage module also provides the BCU with a storage busy signal.

Another signal may be sent with the storage address called "bump". This signal is multiplexed to each storage module the same as the normal address. The bump storage address is not decoded until the "select" pulse initiates the storage cycle. The bump storage area is added to the main core storage arrays by adding two additional drive wires per plane in the Y dimension. Although bump storage is physically part of the core array for the main storage it does not use any of the main storage addresses. Bump storage has its own address lines in the Y dimension. The data in bump storage is used to store the control words for the multiplexor channel. When the Soft Local Store option is selected, the general and/or floating point registers are also stored in bump storage. Bump storage is not directly addressable by the programmer.

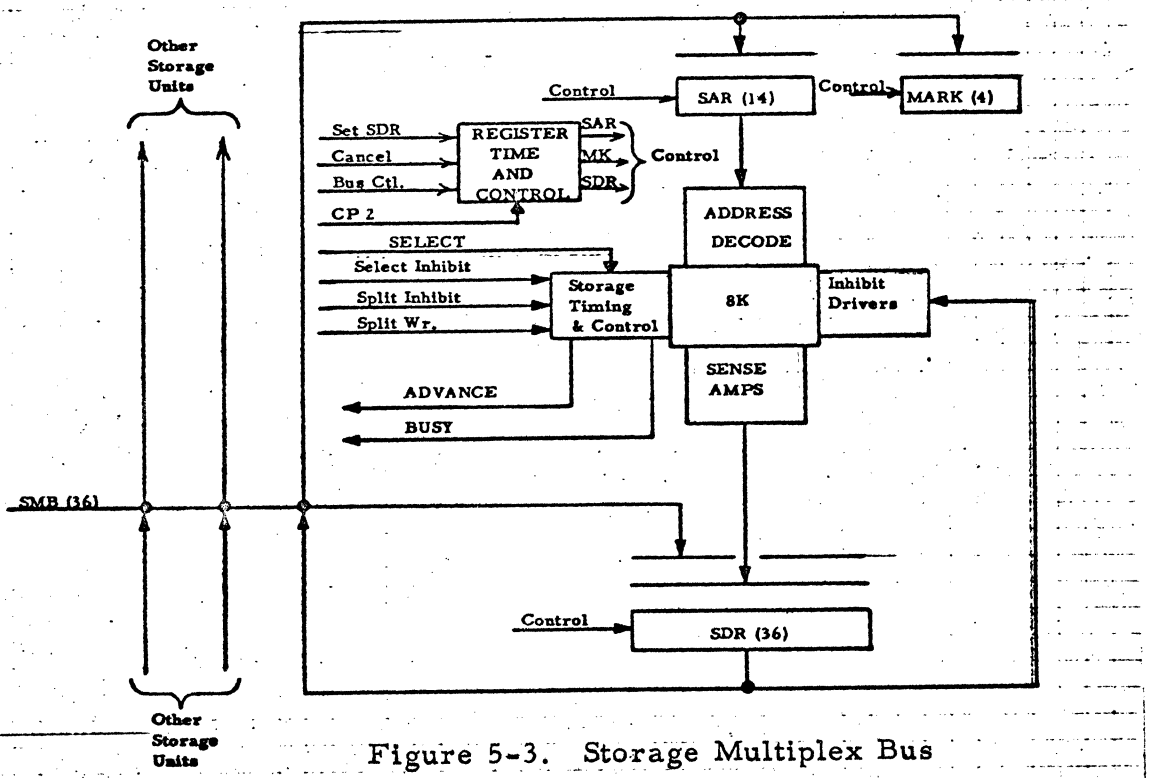


Figure 5-3. Storage Multiplex Bus

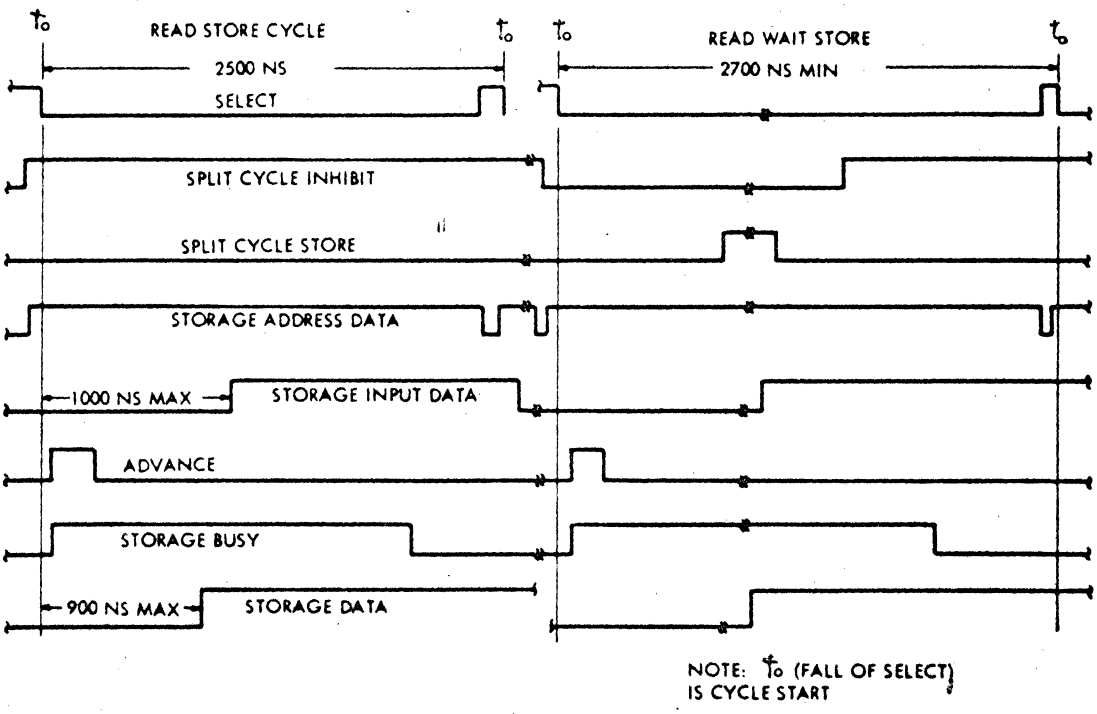


Figure 5-4 Storage Timing Diagram

After 900 nanoseconds the data from the addressed storage location is available at the storage sense amplifiers and is strobed into the Storage Data Register (SDR). Prior to inserting the output data in the SDR, storage generates an "advance" pulse to the BCU. The BCU notifies the requesting area that data from storage is available.

The BCU also supplies 4 mark bits which are used to control the information which will be stored in the addressed location. Each of the 4 bits is used to gate a byte (8 bits plus parity) into the SDR from the sense amplifiers or the Storage Multiplex Bus (SMB). A 1-bit in a mark located will cause its associated byte of data on the SMB to be placed in the SDR. The contents of the SDR are then restored in the addressed storage location. Mark bits of 0000 result in no data from the CPU being placed in the SDR and the unchanged word read from storage will be restored in the location that it was read from resulting in a read operation. If the mark bits are 1111, the data on the storage multiplex bus will replace the data read from storage in the SDR and this new data will be restored resulting in a store operation. Use of mark bits make possible the modification of words in storage on a byte basis.

To perform a split-cycle operation the BCU transmits a "split cycle store inhibit" signal to the memory timing circuits. At the end of the read cycle the memory timing will be stopped until a "split cycle store" signal is received. The data in the SDR is then stored in the addressed location.

5.4 Timing System

5.4.1 Data Flow Timing - Figure 5-5 is a simplified illustration of Data Flow Timing.

Conceptually, the 0.416 us CPU cycle is symmetrically divided into two intervals. . . a 0.208 us CLOCK time and a 0.208 us NOT-CLOCK time. Registers which are outgated to the adder or mover are called first rank registers and are strobed at CLOCK time. Data is good at the output of a first rank register early in clock time and stays good throughout the remainder of CLOCK time and the following NOT-CLOCK time. Registers which receive data from the adder/shifter and the mover shifter are called second rank registers and are strobed at NOT-CLOCK time. Data is good at the output of a second rank register late in NOT-CLOCK time and remains good throughout the following CLOCK time.

5.4.2 Local Storage Timing - Figure 5-6 is a simplified illustration of Local Store. Local store consists of a group of registers which are strobed at CLOCK time of a cycle and are denoted as first rank registers. They are driven by second rank registers. However, local store registers are not outgated to second rank registers. Instead, they are outgated to one of two first rank working registers in the data flow.

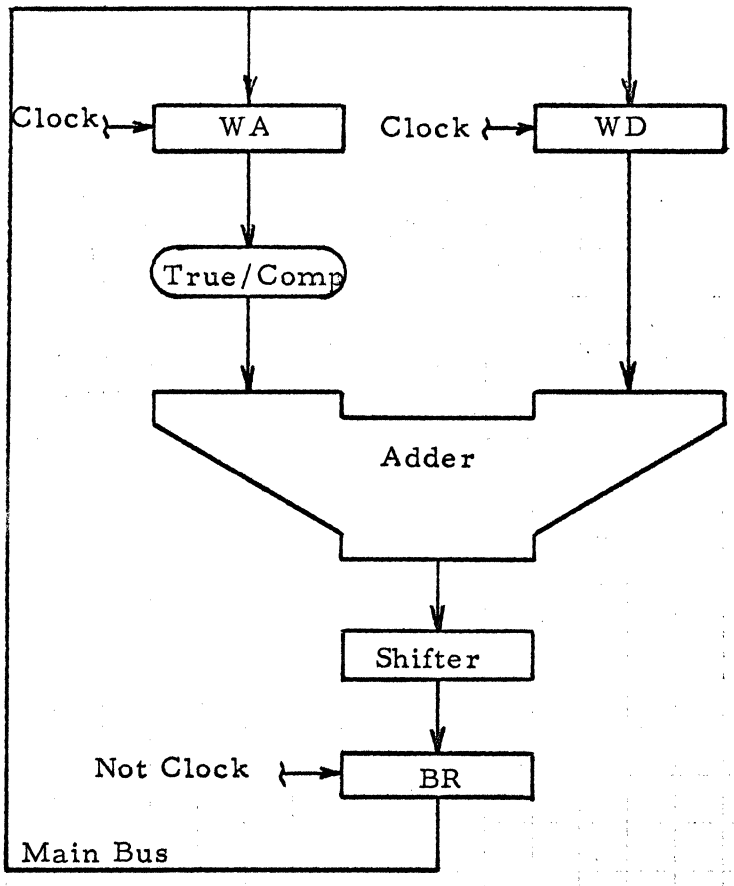
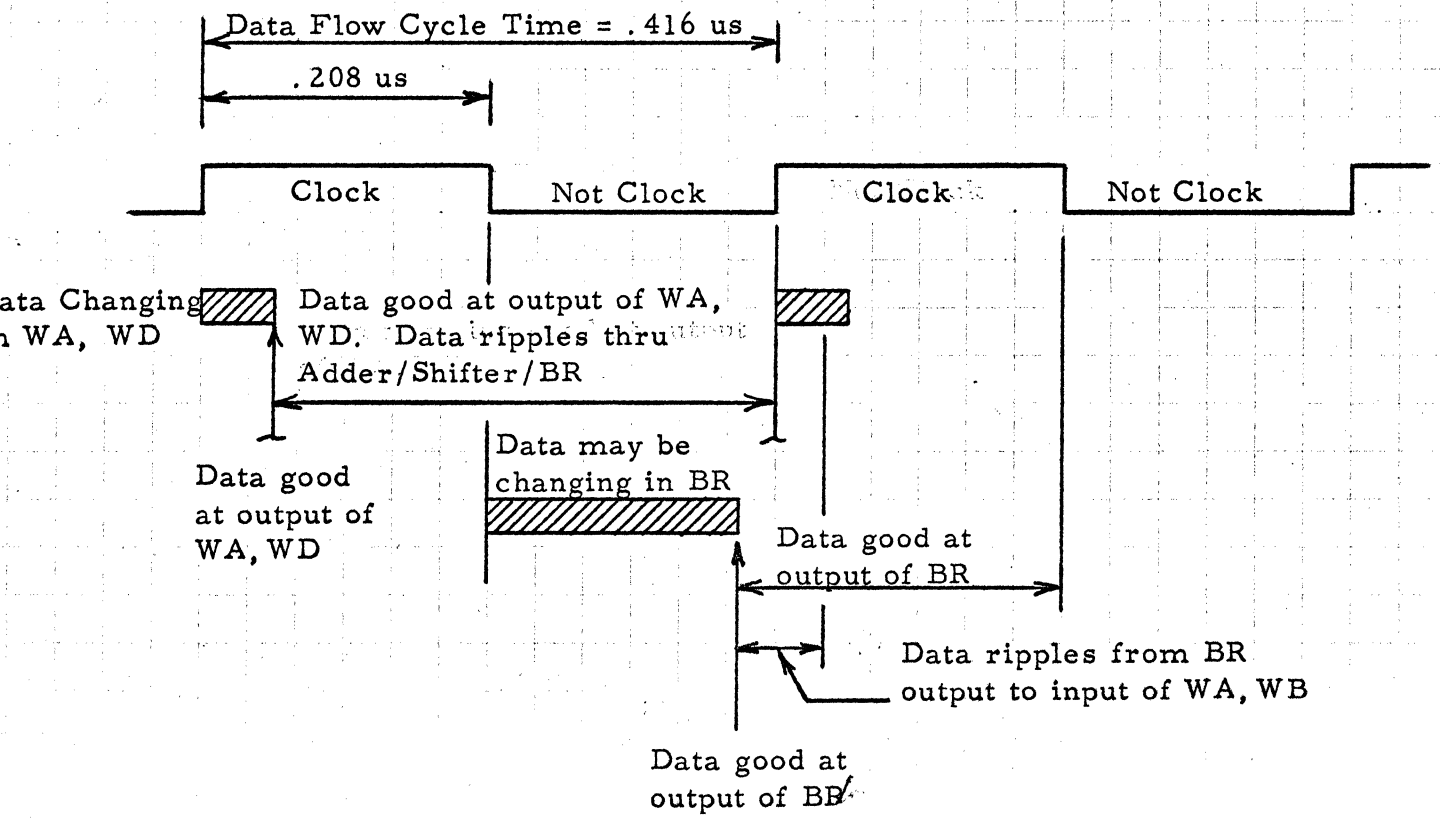


Figure 5-5
Data Flow Timing



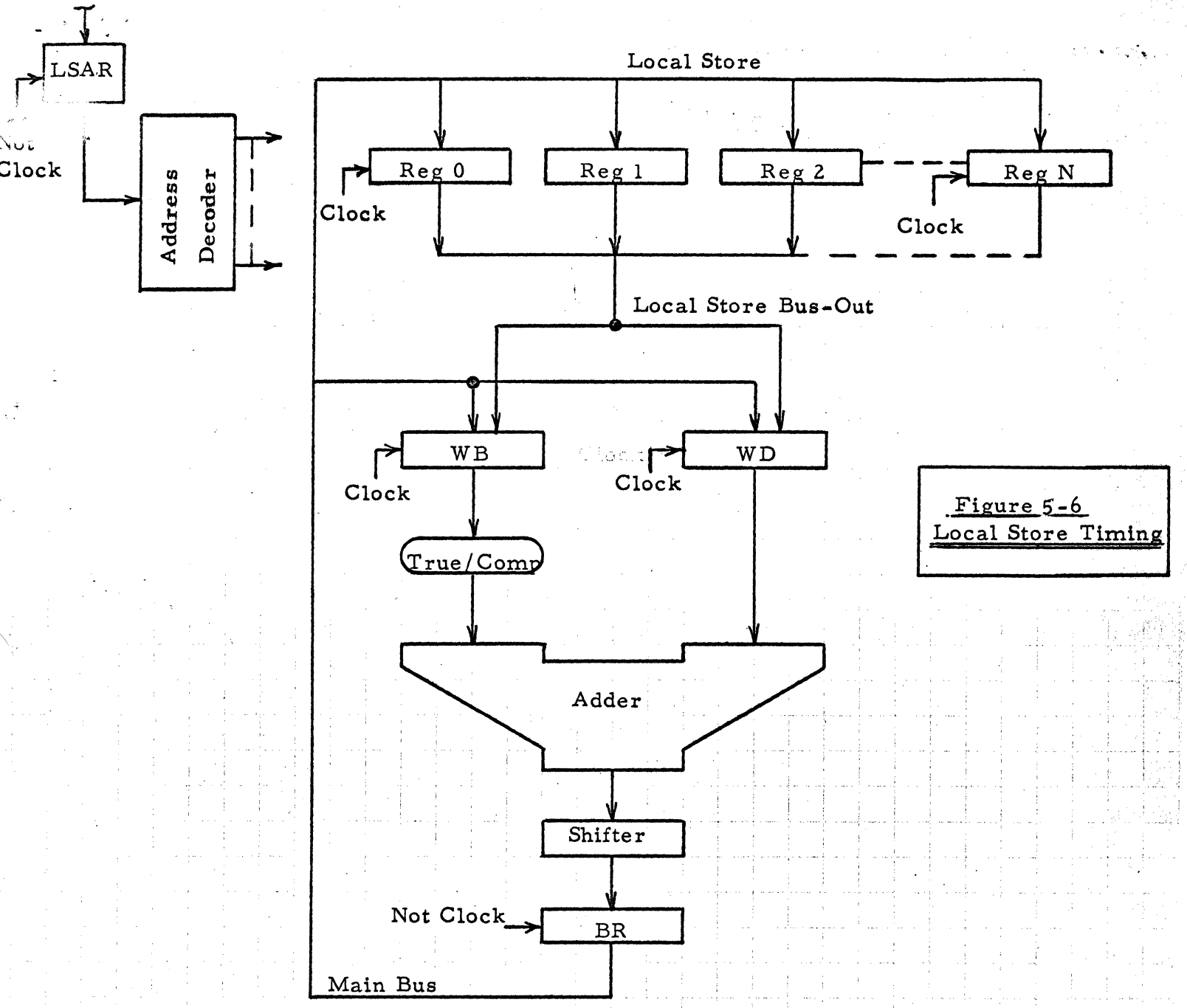
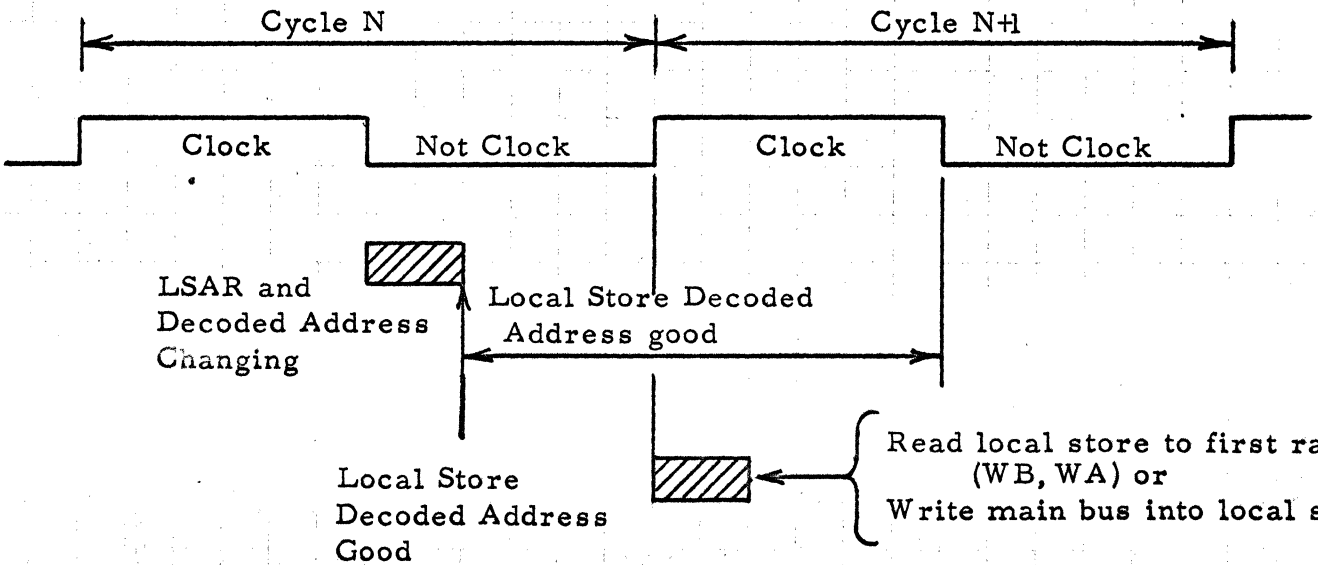


Figure 5-6
Local Store Timing

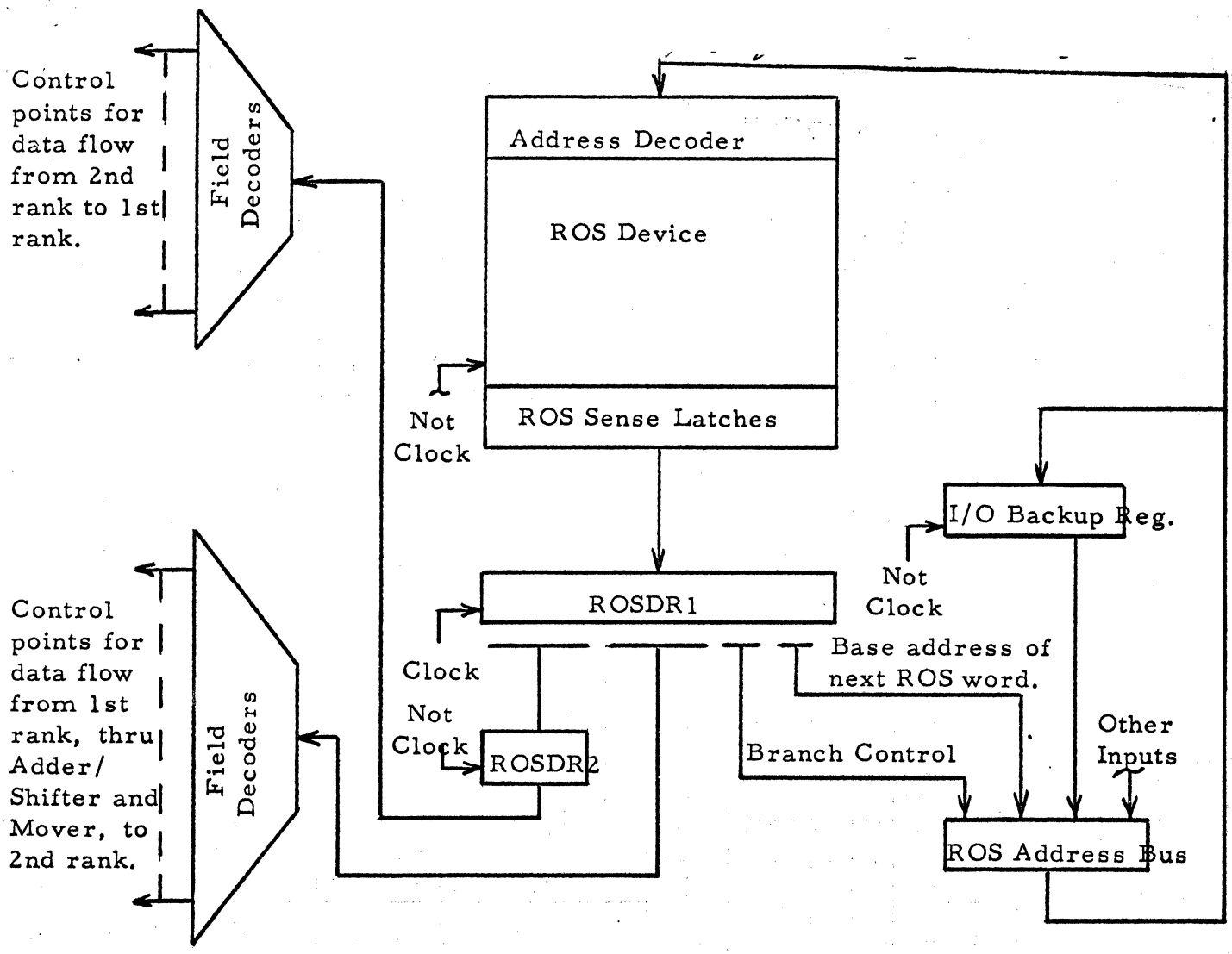


For this reason, local store appears like a memory having a 0.416 us cycle time. The local store address register is strobed at NOT-CLOCK time of cycle N and the address is decoded. If the ROS word controlling cycle N specifies local store read then the selected local store register will be read a first rank during CLOCK time of cycle N+1. A local store read can occur simultaneously with other data transfer through the main adder. If the ROS word controlling cycle N specifies Local Store Write, then the information on the Bus will be set into the selected local storage register during CLOCK time of cycle N+1.

5.4.3 ROS Timing - Figure 5-7 is a simplified illustration of ROS.Read-Only Storage has a 0.185 us access and is cycled at 0.416 us. During NOT-CLOCK time of CPU cycle N, a 100 bit word is being read from ROS to control CPU cycle N+1. At clock time the 100 bit word is transferred from the ROS sense latches to a first rank ROS data register (ROSDR1). Note this is the same time that the Data Flow first ranks are being strobed. Groups of bits (fields) from ROSDR1 are decoded to activate the data flow path from first rank to second rank. These control point enabling signals will be good early in CLOCK time and stay good throughout the remainder of CLOCK time and the following NOT-CLOCK time.

Certain fields of the 100 bit ROS word specify ingating to first ranks from second ranks. Bits corresponding to these fields are transferred from ROSDR1 to a second rank ROS data register (ROSDR2) at NOT-CLOCK time. These fields are then decoded from ROSDR2 to provide control point enabling signals which will control the data flow from second rank to first rank during the next CLOCK time.

5.4.4 Main Storage-ROS - Data Flow Timing Interaction - Figure 5-8 illustrates the relationship between CPU cycles and the main storage cycle. These are 6 CPU cycles under one memory cycle. For convenience, the cycles are denoted R1, R2, R3, W1, W2 and W3. Requests for main storage are issued early in clock time of cycle R1. The data from storage is available for setting into a CPU first rank working register at the beginning of CLOCK time of cycle W1. Storage requests issued under ROS control specify the number of CPU cycles which are to elapse before the data is expected from memory. If the requested data will not be available on the specified cycle, the CPU clock is effectively stopped until such time as the data becomes available. The clock is then started and the microprogram continues. This method of interfacing ROS sequencing with storage timing allows using storage systems of longer access times and eliminates the need for excessive No-Op commands in the ROS.



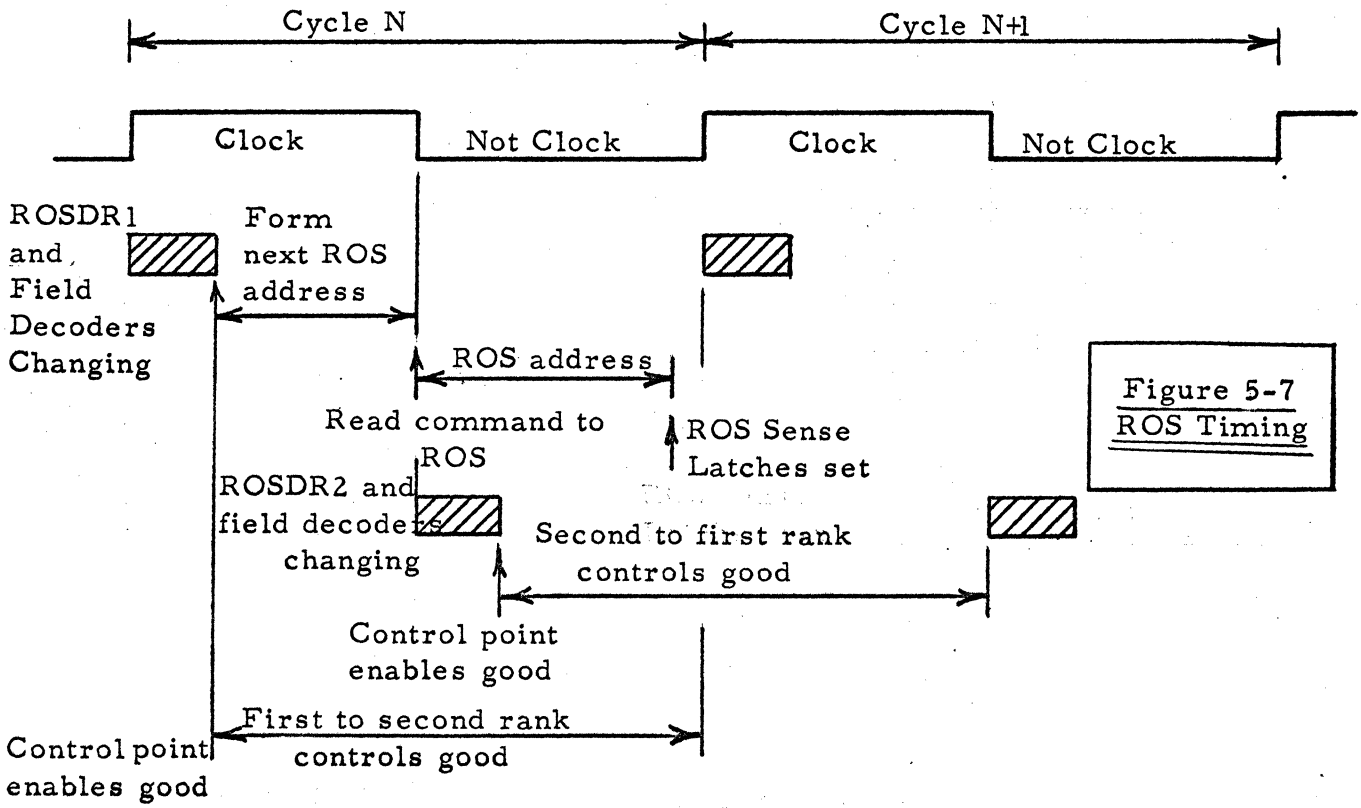
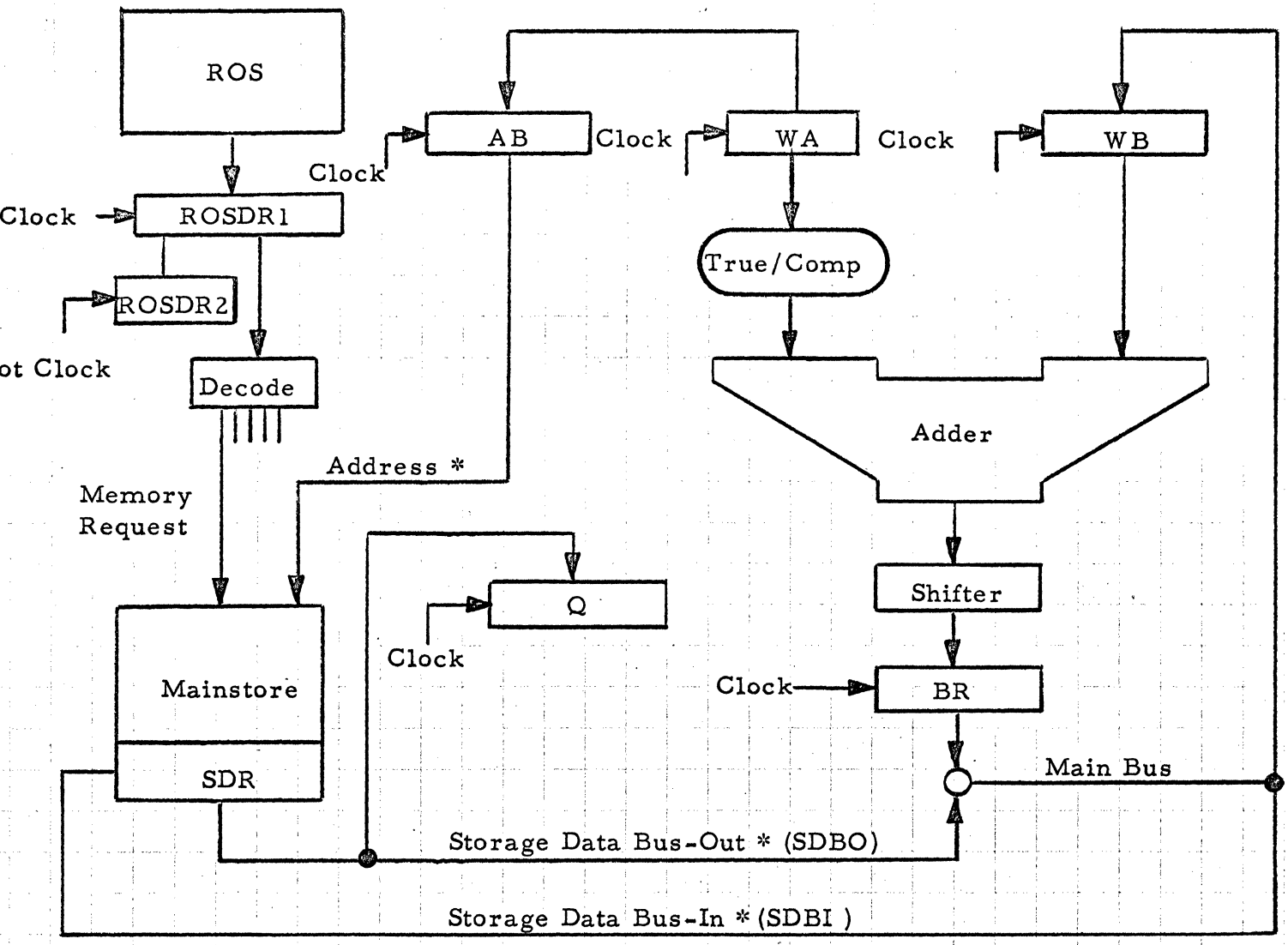


Figure 5-7
ROS Timing

The clocking system is generated from a 4.8 MHz crystal controlled oscillator. The CLOCK and NOT-CLOCK signals are repowered on each page to minimize clock-SKEW conditions. Certain critical timing signals used to synchronize main store operations are enabled by adjustable delay units in order to assure accurate storage timing and to minimize "jitter" between memory modules.



*These three signals are actually transmitted over a single, time-shared Storage Multiplex Bus (SMB)

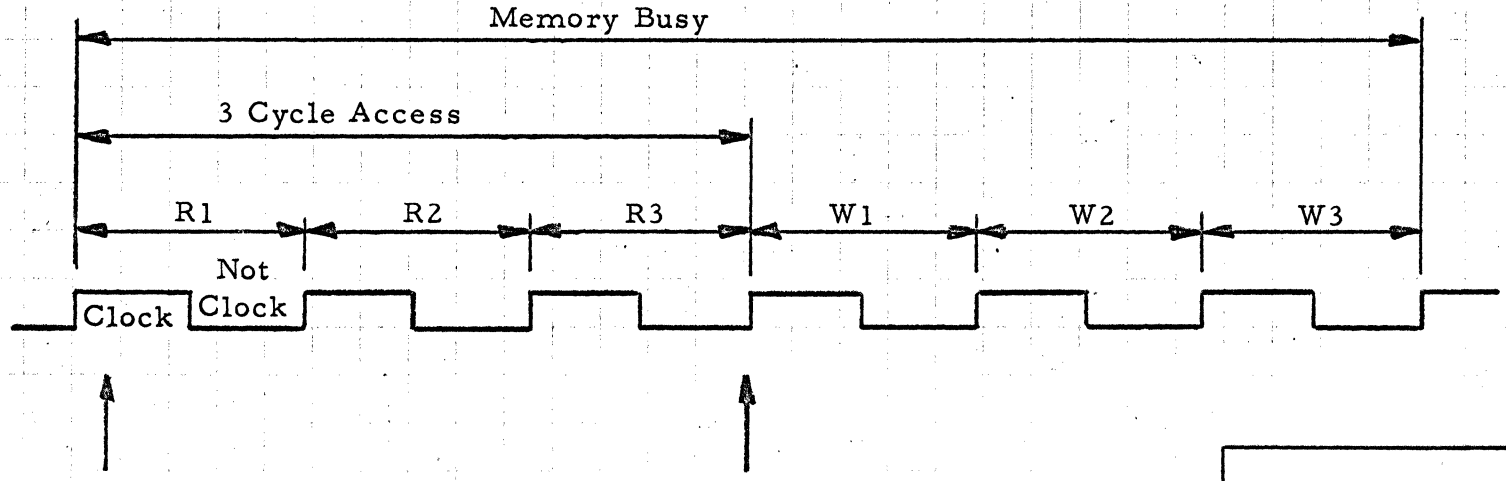


Figure 5-8
Mainstore Timing

6.0 MULTIPLEX CHANNEL DESIGN

6.1 Introduction

The EP Multiplexor Channel is an IBM System/360 compatible channel with several additional 4 Pi-EP modes of operation. The channel can accommodate up to 256 input/output sources and, depending on the data rates of the devices, the channel may time-share its facilities in order to simultaneously service a large number of these sources.

The channel provides a standard I/O Interface to control units. The control units transfer and interpret standard control signals, control the input/output devices and transfer data between the channel and the device. All data is transferred over the I/O Interface in a uniform format of one byte (eight bits) and parity (one bit) for each transfer.

The Multiplexor Channel is fully buffered. After the initial Start I/O instruction is given, the channel will perform a variety of control and message transfer functions between the device and the storage system without further attention from the CPU program. Only when the designated functions are complete, or when exceptional conditions occur, is the program called on for further participation.

The EP Multiplexor Channel is physically integrated with the EP Central Processing Unit (CPU). Many of the CPU data paths are used by the channel and a major portion of the channel controls are under microprogram control. However, the channel does contain physical data paths and control logic to the extent required to achieve efficient performance. The channel is capable of relatively high performance but, at the same time, the amount of hardware required is minimized by sharing CPU hardware.

The EP Multiplexor Channel can operate in one of four modes: Multiplex, Burst, Lockout and Direct Input/Output. In the Multiplex mode, many devices may operate simultaneously by interleaving their operations and time-sharing the channel facilities. While only one device may actually be using the channel at a time, the channel operation is fast enough that many slow speed devices may be simultaneously serviced without conflict.

In the Burst mode, only one device at a time may use the channel facilities. Burst mode data rates may be higher than multiplex due to the fact that the device or control unit stays tied to the channel for the duration of a burst transfer of data and less time is required for interface sequences. Burst mode and Multiplex mode operations use CPU facilities when a byte transfer is taking place. These operations are transparent to the program being performed but they do cause "interference" which results in an apparent increase in program performance time. In between data transfers, the CPU continues to operate at normal speed.

Lockout mode is the same as Burst mode except that the CPU facilities are dedicated to servicing the channel for the duration of the transfer. Very high data rates are achieved in this mode at the expense of 100% CPU interference.

Direct Input/Output is a separate type of I/O transfer operation that utilizes the Multiplexor Channel facilities in order to minimize the number of I/O interface and the hardware required. Direct I/O is a means of transferring two or four bytes of data between a device and the CPU directly under the control of a CPU instruction. The CPU is tied up for the duration of the operation. The Direct I/O transfers usually operate on the non-privileged mode thereby permitting direct interaction between the problem program and specialized real-time devices.

6.2 Channel - Control Unit Interface

The connection between the channel and the control units is called the I/O Interface. This interface provides an information format and a signal sequence which is used in common by all control units. The I/O Interface consists of 34 lines as listed in Table 1. The Bus Out and Bus In provide data paths for all byte transfer whether the information is data, status or address. The data paths are eight bits wide plus an odd parity bit. The Tags provide a means of indicating when a byte of information is on one of the Busses and what the byte is. The Selection controls are used for scanning or selecting attached control units. The rise and fall of the Tags and Selection Controls transmitted over the interface are controlled by interlocked responses. This removes the dependence of the interface on circuit speed and makes it applicable to a wide variety of circuits and data rates.

The EP I/O interface is designed to handle up to eight control units and a maximum of 256 devices. The EP channel contains two sets of lines for the Bus-Out, Out Tags and Out Selection Controls. One is a differential set; the other à single ended set. Each line is driven by a single NAND gate. Incoming lines may be equipped to receive either single-ended or differential signals.

Table 1

I/O Interface Lines

BUS-OUT Position P, 0-7	BUS OUT For transmitting data from the channel to the Control Units
BUS-IN, Position P, 0-7	BUS IN For transmitting data from Control Units to the channel
Address-Out Address-In Command-Out Status-In Service-Out Service-In	Tags
Operational-Out Operational-In Hold-Out Select-Out Select-In Suppress-Out Request-In	Selection Controls
Timing-Out Inhibit-In	Special Control Lines

Timing-Out is a 2.4 MC clock pulse. Inhibit-In is raised by a control unit when it wishes to inhibit the parity check by the channel on data provided by the control unit.

6.3 The Multiplexor Channel Data Flow

The multiplexor channel provides the path for data transfer between the storage system and the control units. The Model EP multiplexor channel is integrated; it utilizes CPU data paths and ROS control in effecting its function. Figure 6-1 is a block diagram of the multiplexor channel data flow. It illustrates the data paths in the channel itself as well as the CPU data paths which are used by the channel.

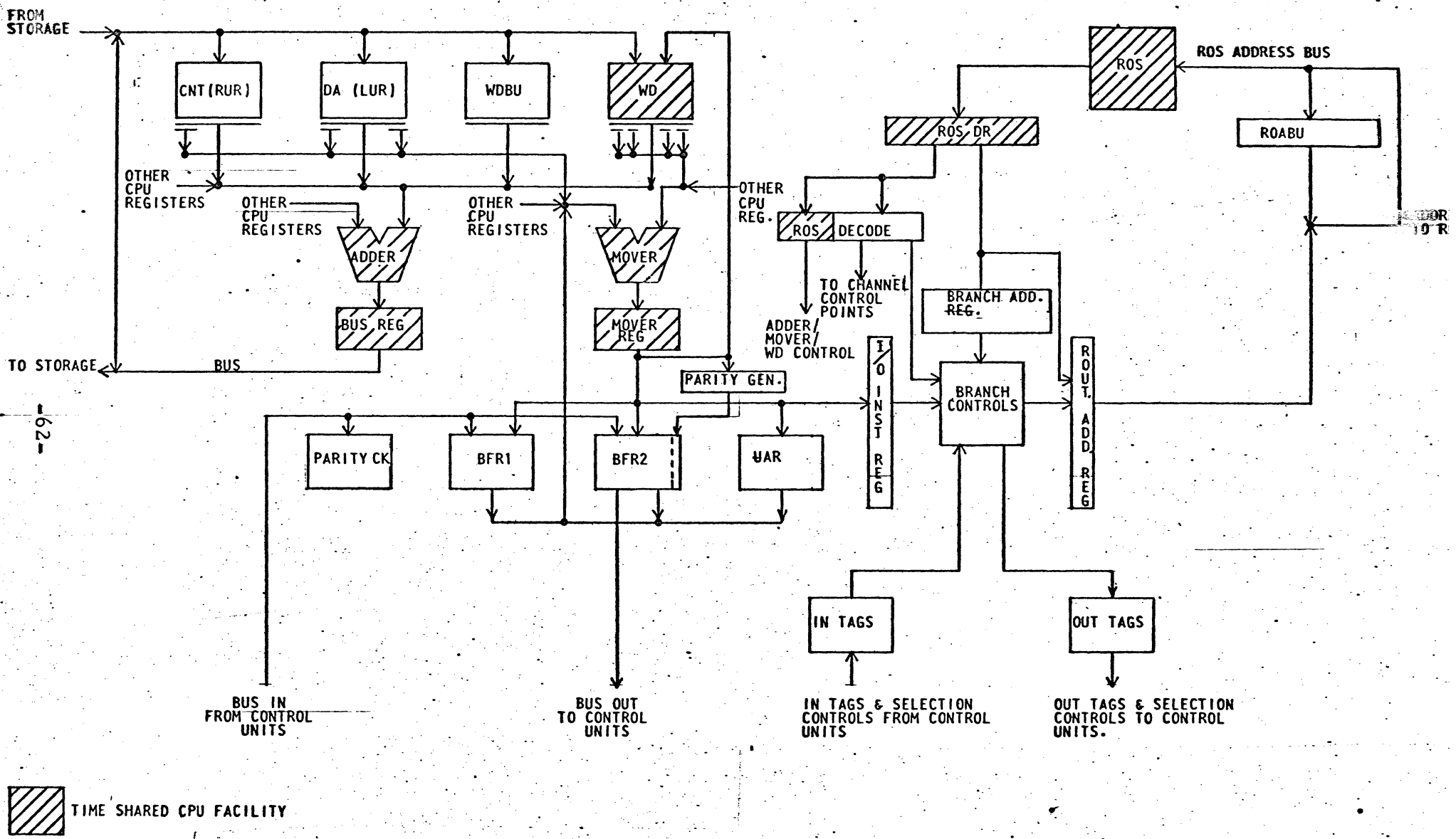
There are two 8 bit (BFR1, UAR) and one 9 bit buffer (BFR 2) in the multiplexor channel. The 9 bit buffer contains a parity position along with the 8 data bit positions. It is the only buffer which feeds the Bus-Out. All outgoing data must pass through it. The Bus-In feeds both the 8 bit Buffer 1 and Buffer 2. Normally data is placed in Buffer 2 while Buffer 1 is used for unit addresses or status bytes. The other 8 bit buffer, the Unit Address Register (UAR) is fed by the mover and can only be gated to the mover left input. It is used to store a unit address between channel microroutines.

The WD backup register (WDBU) is a 32 bit register which is used to store the contents of the WD register while the channel is using WD. At the time the channel breaks into a CPU, the contents of WD are fed through the Adder to the Bus Register. The Bus is gated into WDBU. This saves any data the CPU had in WD at the time the channel breaks in on the CPU operation. The WD register is used for channel data transfers because its full width can be gated into the Adder or each byte can be individually gated into the mover.

The uses for RUR and LUR registers will be explained after channel control words have been considered. These two 32 bit registers can be loaded from the main bus and can be gated into Adder Right. Some byte gating into the left mover input has also been provided for special purposes.

Since the Bus Register and the Mover Register are both clocked on CP2, the three channel buffers, as well as RUR, LUR, WDBU and WD registers are all clocked with CP1. These registers are used when the multiplexor channel is involved in a data transfer operation. In addition to these register, CPU registers WA, WB and WC are used in performing input/output instructions.

To transfer data from main storage to a control unit, a word is read from storage and placed in WD. The proper byte of the word in WD is gated to the mover and out to BFR 2 in the multiplexor channel. The contents of BFR 2 are placed on Bus-Out along with a correct parity bit. To transfer data from a device to main storage, a control unit places the data on the Bus-In. The channel gates the data on Bus-In into Buffer 1 or 2 and performs a parity check on the transferred byte. The contents of the proper buffer are then gated into the left mover input, through the mover and into the correct position of the WD register.



Multiplexor Channel
Block Diagram
Figure 6-1

The position of the WD register in which the byte is placed is normally determined by the lower two bits of the data address associated with the data transfer. The storage address is set up and the byte in WD is inserted in the word read from storage and the revised word is stored.

6.4 Multiplexor Channel Control

The controls for the channel are integrated with the CPU. The channel is controlled by a combination of micro controls and hard wired controls. When a data transfer operation is taking place or an I/O instruction is being initiated, the channel is mostly micro-controlled with a small amount of hard wired controls involved. When the channel is not using the CPU for iether of these functions, the channel is entirely hardware controlled.

The control philosophy in the EP Multiplexor Channel design is to use the CPU only at the instant there is data to be transferred and to use hardwired controls for the long periods required for interface sequences or waiting for an operation to begin. This minimizes hardware costs and, at the same time, keeps the interference to the CPU relatively low.

6.4.1 Micro-Control - When the channel breaks into the CPU operation, a latch is set to indicate that the CPU is now operating in the "I/O mode." This signal is used to turn off the decoding of certain micro-orders which are not used by the multiplexor channel and to turn on the decoders for other micro-orders used by the channel and not by the CPU. Thus, there are two modes of decoding the bits from the Read-Only Storage (ROS); CPU and I/O mode. The creation of these modes reduces the number of bits required in each micro-word.

The data flow in the channel is controlled by the following ROS fields:

Mover Left Input
Mover Destination
Adder Right Input
Bus Destination
In Bus
Out Bus

Some of these fields are interpreted exactly the same in both CPU and I/O modes of operation. Several fields have some micro-orders interpreted differently in each mode. Other CPU mode fields are completely eliminated in the I/O mode to create new fields for the multiplexor channel control. The use of the first four fields in the above list is similar to CPU usage. The In-Bus field is a redefinition of CPU field LO. The In-Bus (IB) field is used to select the channel buffer or stats to be gated to the mover left input. The Mover Left Input determines whether this or some other mover left input will be gated into the mover. In other words, there are two levels of gating

on the mover left input. The Out Bus (OB) field is a redefinition of the CPU field CF. It is used to control the data flow from the CPU or from the Bus-In into the channel buffers. Many of the channel control functions are also under micro-order control. Special micro-orders have been added to the A and B branch fields to provide the capability for performing branches in the I/O microroutines. Such branching is controlled by the condition of special purpose latches or stats. The setting or resetting of these stats is controlled partially by micro-orders which have been added to the SS and LSC fields and by the Multiplexor Stat field (MS) which is a redefinition of the CPU LS field. Some of the stats are also set as a result of various conditions detected by hardware controls.

The ROS output signals for controlling points within the CPU are decoded in the CPU. The decoders which are used to set channel controls are located on the Multiplexor Channel pages.

6.4.2 Hardwired Control - When the channel is not sharing CPU facilities, it is completely under hardwired control. At such times, the multiplexor channel is either waiting for a data transfer request from a control unit, waiting for an instruction from the CPU, in the beginning stages of either of these operations, or completing an interface sequence with a control unit after a data transfer has taken place. At these times, the channel controls are making logic decisions based on the I/O Interface sequences or signals from the CPU.

Some of the signals sent to the channel by the CPU are those which indicate that the channel is to perform an I/O instruction. These signals are sent to the I/O instruction Register in the channel. This register remains set until the portion of the instruction which ties up the CPU is completed.

There are two address registers associated with the channel control. The first is a 4 bit register called the Branch Address Register (BAR). The contents of this register are decoded to define a unique branch point in the hardwired logic. For each branch point there are sets of unique conditions which determine what the channel is to do next. In a given branch point one set of conditions on the I/O Interface may cause the channel to create the starting address of a data handling routine in the 6 bit Routine Address Register (RAR) and request the services of the CPU. The RAR can be set by all six bits of the A branch field enabling the microprogram to go directly from one channel routine to another. The RAR can also be set as the result of a hardware branch. The address in each case points to the first word in a microroutine.

6.4.3 Break-In - As a result of an interface sequence or an instruction from the CPU, the channel reaches a point where it requires the use of the CPU. The channel sets up the address of the first word in the microroutine which will be used and raises a line to signal the CPU that its services are required. The starting ROS Address remains in the RAR until the CPU

honors the request for service. When the request is accepted, the ROSDR is forced to all zeros. Thus, there will be no "break-in cycle" with no micro-control exerted. The address in RAR is gated onto the ROS Address Bus to fetch the first I/O mode micro-control word. Under channel control the contents of WD are gated through the Adder into WDBU where they will be saved for the duration of the I/O microroutine. The CPU mode address being sent to the ROS is available in the ROS Address Backup Register. At the time the CPU honors the I/O break-in request, it freezes the contents of this register. This saves the address of the ROS word which the CPU would have performed next. After one full ROS cycle, the first word in the I/O routine is read into the ROSDR and the channel is now under micro-control. During this break-in cycle, an "I/O mode" latch is set so that the micro-orders in this routine will be decoded as required by the channel.

6.4.4 Break-Out - During the last cycle or Break-Out cycle of the I/O processing, the micro-orders are used to put the contents of the ROS Address Backup Register onto the ROS Address Bus. The contents of WDBU are sent through the Adder into WD and a signal is sent to the channel to reset the I/O mode latch. During the next cycle, the channel is under hardware control and the CPU is performing the micro-orders it would have done had the channel not broken in. Thus, the CPU has been returned to its original state.

6.5 Start Input/Output

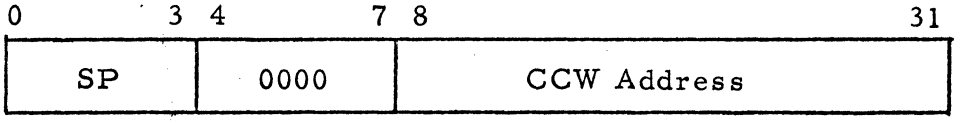
The following is a brief description of the SIO operation. The reader is referred to the System/360 Principles of Operation for more complete details on all of the I/O instructions other than Direct I/O.

The Start I/O instruction generates the unit address and accesses a Channel Address Word (CAW) from a reserved location in main store. The CAW provides the storage protect key to be used, and points to the Channel Command Word (CCW) which is a double word in length. The computation of the unit address obtained from the Start I/O instruction, the storage protect key obtained by the CAW and various pieces of control information obtained from the CCW are used to formulate a new set of control words called Unit Control Words (UCW's). The UCW's provide the personality for the operation which is to be performed. In the Model EP, these words are stored in bump storage at an address which is directly correlated with the unit address. The process of forming and storing these control words is called "forming a subchannel." The following writeup will provide a sequential description of this process.

6.5.1 Forming a Subchannel

The SIO instruction has an SI format. The contents of the register addressed by B_1 are added to D_1 to form the channel and device address (unit address). The unit address is also the address of the subchannel and it is used to address the locations in bump memory that are associated with the particular device. These bump locations become the subchannel. A subchannel is formed as follows:

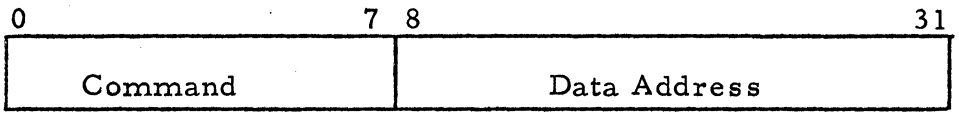
The Start I/O instruction causes a Command Address Word (CAW) to be fetched from location 72 in main storage. The format of this word is given below.



CAW and UCW 0 Format

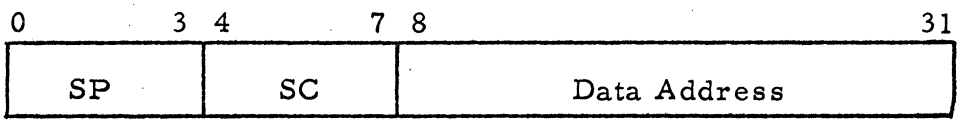
The CAW contains the storage protection key, four zeros, and the Channel Command Word (CCW) address. Eight (8) is added to the CCW address and it, along with the protect key and zeros, are stored in bump storage as UCW 0 (Unit Control Word). The unit address formed from SIO instruction designates the address in bump storage for UCW 0 as well as the other UCW's. The Command Address (CA) in UCW 0 now points to a CCW (Channel Control Word) eight bytes away from the one we are now going to fetch.

The unaltered CCW address (before eight was added) is used to address main storage to fetch CCW 1.



CCW 1 Format

CCW 1 contains the data address which is the address from which the first word of data will be fetched or in which the first word of data will be stored. An eight bit command which is to be sent to the device is also contained in CCW 1. The data address (DA), the storage protect key from the CAW and four bits called sequence controls are stored in Bump next to UCW 0. This new word is UCW 1.



UCW 1 Format

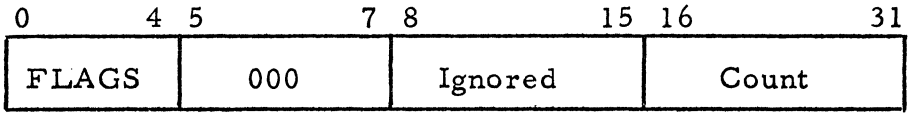
The codes and meaning of the sequence controls (SC) are given below. Their use will become more apparent later in the discussion of the channel operation. Basically, they define whether the subchannel is idle, busy or has an interrupt pending and what type of interrupt it is.

Sequence Controls

STATE	CODE
IDLE	0000
BUSY	0001
CMD CMD Chaining End Received	0011
Channel End in IB	0101
Channel End Que'd	0111
Device End In IB	0110

IB is the interrupt buffer located in bump storage

Four (4) is now added to the original CCW address. This new address is used to fetch CCW 2.



CCW 2 Format

CCW 2 contains five flag bits and a count which defines how many bytes of data are to be transferred. The Flags are listed below.

0.	CDA	Data Chaining
1.	CC	Command Chaining
2.	SILI	Suppress Incorrect Length Indication
3.	SKIP	SKIP
4.	PCI	Programmed Controlled Interruption

From the Command in CCW 1 three operation (OP) bits are formed for use by the channel. The OP codes and functions are given below.

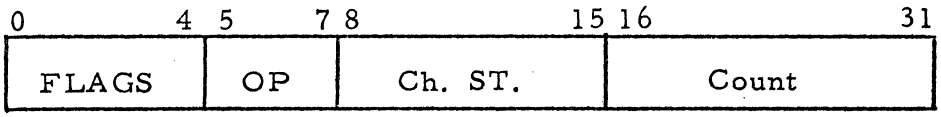
OP Code

Code	Function
000	Input Forward
001	Input Backward
110	Output
011	Input Skip
111	STOP
100	En d STATUS - No Wrong Length Record
101	End STATUS - Wrong Length Record

The channel translates all of the commands sent to the device into the above OP codes since these are the only ones recognized by the channel. The last two codes are not really operations but status indications used by the channel.

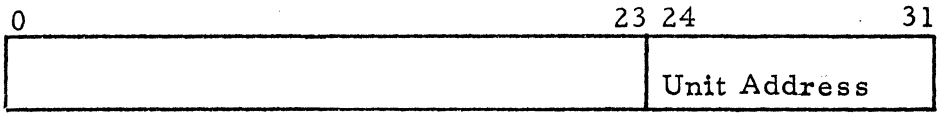
Eight bits are used to define the Channel Status. The channel status is used to indicate error conditions detected during the operation of the subchannel and to carry PCI flags, which have not yet caused an interruption, through command chaining.

The Flags, OP Channel Status, and Count are stored in Bump in the location next to UCW 1 and become UCW 2.



UCW 2 Format

UCW 0, 1 and 2 constitute a subchannel for a nonshared subchannel (that is, one which is used by only one device). A UCW 3 is required for a shared subchannel. UCW 3 contains the unit address for the device which is presently using the subchannel.



UCW 3 Format

Sixteen devices can share a subchannel. For shared subchannels, the Unit Address is shifted right four places to get the unit address prime. The unit address prime is used to address the UCW's. All devices sharing a subchannel have the same upper four bits of unit address and thus the same unit address prime. Because of this addressing, for each shared subchannel there is one less non-shared subchannel. There is a maximum of eight (8) shared subchannels. A one (1) in the high order bit of the unit address indicates to the EP multiplexor channel that the device is using a shared subchannel.

6.5.2 Device Selection - After the CPU has fetched CCW 1, it issues a Start I/O Signal (SIO) to the multiplexor channel. The CPU then enters a countdown loop. If the channel has not responded by the time the CPU has counted to zero, a time out signal is issued. The channel should then set CPU condition code 2 indicating that a burst mode operation is being performed and the SIO cannot be handled.

At the end of the first countdown, the CPU initiates a second countdown. If the channel does not set the code 2 by the time the CPU counts out of the second loop, the channel is forced to a machine check micro routine if the CPU is enabled for machine checks. If the CPU is in a disable mode, a CSW is stored and condition code 1 is set.

There are a number of things which will cause the CPU to store a Channel Status Word (CSW) and set condition code 1. For details, the reader is referred to the System/360 Principles of Operation. However, it should be noted here that one such condition is that the device is selected and it is found that for various reasons the device or control unit cannot execute the SIO.

6.5.3 Interface Signaling Sequences - For the following discussion, it will be assumed that both the control unit and the device are able to perform the required operation.

Requests for data transfer have a higher priority than SIO. When the channel gets to the polling state, it will look for data transfer requests or instructions such as SIO. If there is a request for data transfer, the channel will branch from the polling state to handle the data. It will continue to do this until there are no data transfer requests. It will then recognize the SIO, set up the proper ROS address in RAR and break into the CPU countdown loop. The CPU will fetch CCW 2 and initiate the selection of the device. The unit address is placed in BFR2 and Address Out is raised. Then Select Out and Hold Out are raised.

Select Out is wired through the control unit serially. The first control unit inspects the unit address. If it is not an address associated with that control unit, it propogates the Select Out signal to the next control unit. This continues until a control unit accepts the unit address or the Select Out comes back to the channel as Select In. In the latter case, the operation is discontinued. In the former, the control unit will raise Operational In and inhibit the propogation of the Select Out any further. Thus, the control units are wired to the Select Out Signal in the order of their priority. This fact is of significance when control units are attempting to request service. Since Select Out is raised in response to Request-In, a control unit cannot gain access to the channel until all control units of higher priority have no outstanding requests for service.

Returning to SIO, the control unit which raised Operation In places the unit address of the device on the Bus-In and raises Address-In. The channel resets Select Out and gates the unit address into one of its buffers. In the CPU, the unit address sent out and the one received are compared. If they compare, the command from CCW 1 is placed in BFR2 and Command Out is raised. The control unit inspects the command and if it can perform the command it sends a zero unit status and raises Status-In. The channel

gates the status byte into one of its buffers and resets Command Out. If the unit status is zero, the CPU sets condition code 0 indicating to the program that the operation has been initiated. The CPU ends the count-down loop and fetches the next instruction.

The channel waits for the control unit to drop Operational-In. If Operational-In drops,the channel assumes that the control unit has gone to multiplex mode of operation. The channel returns to polling to look for a request for service from this or any other device or an I/O instruction from the CPU. If the control unit keeps Operational-In up and brings up Service-In, the channel begins burst mode operation.

6.6 Data Transfer

For the purposes of illustration, the following discussion assumes an input forward operation. The main difference between input and output other than direction is that for output, the channel must fetch a data byte before the devices request it. Also, actions taken as a result of errors detected during output operations differ somewhat from those taken for input operation. The following is a brief description of multiplex, burst, and lockout operation on the multiplexor channel.

6.6.1 Multiplex Mode - Between each byte transferred in the multiplex mode the control unit drops its Operational-In Line. To initiate a data transfer, the control unit raises its Request-In line when the I/O interface is idle. The multiplexor channel being in the polling state will respond with Select Out. The Select Out is propogated from control unit to control unit. The first control unit which needs service may take over the interface and inhibits Select Out from getting to all units of lower priority. If the unit originating the Request-In receives the Select-Out, it raises Operational-In, places the unit address on Bus-In and raises Address-In.

The channel now forces the address of a routine called Count Fetch and Update in the RAR and requests a break-in. When the routine request is honored by the CPU, the break-in procedure previously described takes place and the Count Fetch and Update Routine is entered. The unit address is gated from the Bus-In through a channel buffer into the CPU. Command-Out is raised to indicate that the address has been accepted. The unit address is used to address UCW 2 in bump storage. The count contained in this word is decreased by one and stored back in UCW 2. The fetch and update is done in one split storage cycle. UCW 1 is now fetched, one is added to the data address and it is stored back in Bump. UCW 2 is placed in the RUR while UCW 1 is gated into the LUR.

When the control unit raises Service-In, the channel enters a Data Handling Routine. The data byte is gated from Bus-In through BFR2 to the WD Register. It is placed in the proper byte position under the control of the lower two bits of the data address now in LUR. The data address is used to fetch a word from main storage. The lower two bits of the data address are used to put the byte from WD into the proper byte of the word read from storage. The word is now restored with just the one byte change.

The count in RUR is checked to see if the byte transferred is the last byte required. If it is, an ending procedure is entered. If it is not, when the control unit drops Operational-In, the channel goes to the polling state to look for another request.

6.6.2 Burst Mode - As indicated, burst mode can be forced by the control unit holding up Operation-In immediately following SIO. Burst mode can also be forced by the control unit holding Operational-In after a byte transfer. While holding up Operational-In, the control unit raises Service-In and the channel enters a data handling routine. The data byte is handled in the same manner described for multiplex mode. Again, the count is checked to determine if it is the last byte. If it is not, the channel awaits the next Service-In (or the fall of Operational-In if the control unit switches to multiplex mode). When the channel is operating in the burst mode only one control unit can be operating with it. Data rates are increased over multiplex mode due to the fact that the device does not have to go through initial selection each time it needs to transfer a byte of data. For the duration of the burst operation, UCW 1 and 2 are kept in LUR and RUR. They are stored only at the end of the burst. This reduces the number of main memory accesses required per byte transferred from three/byte in multiplex mode to one/byte in burst mode. In burst mode, CPU interference per byte is reduced and data rates are increased.

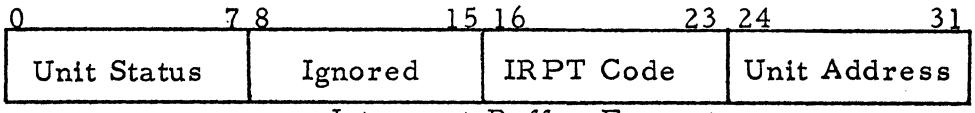
In burst mode, the channel is idle between each byte transfer awaiting the next Service-In. The CPU continues its normal program and is not interrupted until the Service-In has occurred. When I/O request arises, the break-in cycle will be held off if the CPU has initiated a storage request. The channel will be required to wait as long as 3.3 usec in the worst case with a statistical average of about 1.25 us delay in getting a request honored.

6.6.3 Lockout Mode - Lockout mode operation is similar to burst mode except there is no channel break-out between byte transfers. The data rates can be higher than burst but the CPU interference is 100 percent for the duration of the data transfer.

Lockout mode is forced on the channel and control unit by the programmer. When an address of eight is used to address the multiplexor channel instead of the usual zero, the multiplexor channel and the addressed device enter the lockout mode.

6.7 I/O Interrupts

The EP multiplexor channel contains a 32 bit interrupt buffer (IB). This buffer is actually a location in bump storage. Information pertaining to an I/O interrupt is stored in this buffer and then an interrupt request is sent to the CPU. A latch called IBFULL is set when there is interrupt information in IB and another I/O interrupt is being requested. It is reset after the interrupt has been processed. The format of the word in the interrupt buffer is shown below.



Interrupt Buffer Format

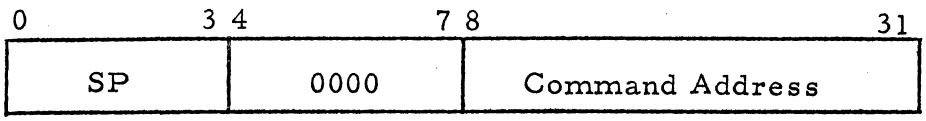
The interrupt code (IRPT Code) indicates the type of interrupt being requested. The upper six bits of this code are zero. The code for the lower two bits is given below.

Interrupt Code

PCI - Channel End	00
PCI	01
Device End	10
Channel End	11

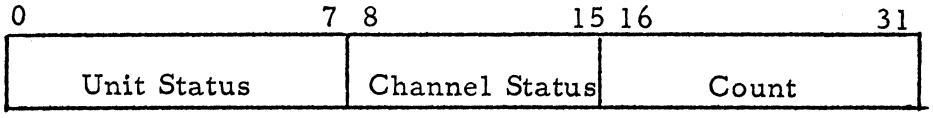
6.7.1 Interrupts From a Device - When a unit status which is to cause an I/O interrupt is presented to the channel, the channel breaks into the CPU and fetches the sequence controls from the UCW 1 associated with the device which is presenting the status. The channel examines the sequence controls to determine whether they are active or inactive and sets the interrupt code to channel end or device end respectively. Thus, the interrupt code is more of an indication to the channel what type of interrupt is to be performed rather than the exact cause of the interrupt. The latter is indicated by the unit status which is placed in the IB. The interrupt code and the unit address of the device causing the interrupt are also placed in the IB and the IBFULL latch is set. The channel also stores an up-to-date UCW 1 and 2. The IB contains device information while the UCW contain sequence controls and the channel status, which together indicate the state of the channel and any error conditions detected thus far in the operation of the subchannel.

The setting of the IBFULL latch signals the CPU that an I/O interrupt is pending. If the channel is not masked (PSW bit 0), at the end of an instruction (End OP) the CPU will accept the interrupt provided no higher priority interrupts have been requested. The CPU will use the contents of the IB and UCW 0 and 2 to store a PSW (Program Status Word) and a CSW (Channel Status Word). The IBFULL latch is reset and the channel continues normal operation. The CSW is a double word with the format given below.



CSW 1

SP = Storage Protect Key



CSW 2

Among the other items stored in the PSW is an interrupt code (bits 16-31) containing the unit address of the device causing the I/O interrupt.

When the CPU accepts a channel end interrupt, its set the sequence controls in a subchannel to idle. The subchannel is now available but the device remains active until it has provided the channel with a device end status.

If a unit status which is to cause an I/O interrupt is presented to the channel and the interrupt buffer contains a channel end for this device or an interrupt for some other device, the channel will not be able to handle the new interrupt until the old one is cleared. In such case, the channel raises Command-Out in response to Status-In causing stacking of the status at the control unit or device. When the channel is in the polling state and the interrupt buffer is full, the channel raises Suppress-Out to prevent these needless presentations of suppressable requests.

6.7.2 Program Controlled Interrupts - During SIO, the channel places the PCI Flag from the CCW in the subchannel UCW2. The channel checks for a PCI Flag each time a byte is transferred for that subchannel. If the PCI Flag is a one, the IB is loaded and the IBFULL latch is set. The channel continues normal operation and the CPU handles this interrupt as it would any other I/O interrupt. The channel can place an interrupt in the IB for a PCI and channel end for the same device at the same time. If the IB is full at the time the PCI Flag is detected, the channel will continue normal operation and catch the PCI on a subsequent byte transfer for that subchannel.

If a PCI is indicated on an immediate operation, the channel stores it in the channel status portion of the CSW and no interruption occurs.

6.8 Chaining

The EP multiplexor channel has the capability for performing both data and command chaining. For data chaining, one command is used for the entire operation along with more than one data address and byte count. Data chaining provides the capability to read from or store into different areas of main storage on a single operation. Command chaining utilizes more than one command data address and byte count. This permits more than one operation to take place using various areas of memory under the control of a single Start I/O instruction. Since either types of chaining utilize only a single CPU instruction, an increase in data handling efficiency is realized by their use.

6.8.1 Data Chaining - Data chaining is indicated by a one in the data chaining position of the flags in the CCW2 fetched by the SIO instruction or by a chaining operation. This flag is stored in the UCW2 of the addressed subchannel. The initial data transfer takes place under the control of the command, count, and data address from the CCW's fetched. When the count is reduced to zero the command address from UCW0 is used to fetch a new set of CCW's. The command address is also updated by eight and restored in UCW0. The channel ignores the command that is contained in the new CCW unless it is a Transfer in Channel. The new CCW's are used to form a set of UCW's. The OP code is not changed. No device selection is made. The next time a data transfer is requested by the device on this subchannel, the transfer is made under the control of the new flags, count and data address. This process is repeated until the count goes to zero and no chaining is indicated or until the device presents channel end status. At such time, the operation is terminated in the same manner as any other operation. To a control unit, data chaining appears as one continuous operation.

There is very little hardware in the channel for data chaining, as it is performed by microroutines controlling existing channel logic.

6.8.2 Command Chaining - Command Chaining is indicated by a one in the command chain position of the flags in the CCW2 fetched by an SIO or by a chaining operation. The channel operation from the SIO up to the point where the control unit presents an ending status is normal. When the control unit presents a channel end status, the channel raises Suppress-Out at least 250 nsec before raising Service-Out which accepts the status. This sequence indicates to the control unit that Command Chaining will take place after the device presents device end. Suppress-Out is dropped by the channel when the control unit drops Operational-In. It should be noted at this point that neither channel end or device end cause an interrupt when command chaining is to be performed. When the device presents device end, the channel again raises Suppress-Out at least 250 nsec before raising Select-Out in response to Status-In. However, this time Suppress-Out does not fall with the fall of Operational-In. It remains up until the device is reselected to present it with a new command.

After the device end the channel fetches the command address in UCW0 and uses it to fetch a new CCW1 and 2. The command address is increased by eight and restored in UCW0. New UCW's are formed. An initial selection sequence is used to reselect the device. The new command is set to the device. The operation from here is the same as for SIO. With the exception of the use of Suppress-Out Command Chaining appears like a series of SIO instructions to the control unit. When the count reaches zero and no further command chaining is indicated, the operation is ended normally with the usual I/O interruptions.

6.9 Other I/O Instructions

Again the reader is referred to the System/360 Principles of Operation for details.

A. Test Channel (TCH) - The CPU issues a TCH signal to the channel and enters a countdown loop. If the channel gets to the polling state and there is no outstanding requests from the control units before the CPU counts out, the channel will recognize the TCH. If the IBFULL Latch is reset, the channel will set condition code 0 to indicate the channel is available. If the IBFULL Latch is set, the channel will set condition code 1 to indicate that there is an interruption pending in the channel.

If the CPU counts out before a reply is received from the channel, the channel should set condition code 2 indicating a burst mode operation is in progress. If the channel does not reply to the count out before a second count out occurs, the CPU sets condition code 3 indicating that the channel is not operational. The CPU will then return to fetch the next instruction.

B. Test Input/Output (TIO) - The CPU issues a TIO signal to the channel and enters a countdown loop. If the channel recognizes the TIO before the CPU counts out,the channel fetches the UCW1 from the addressed subchannel. If the sequence controls indicate that there is an interruption in IB for the addressed device, the TIO stores a CSW, clears the IB and sets condition code. This indicates that a CSW has been stored.

If the sequence controls are set to IDLE the channel selects the addressed device to determine its state. If the device is available, the channel sets condition code 0 indicating the device is available and the subchannel is idle. If the TIO detects an interruption pending at the device, the unit status is accepted from the device and a CSW is stored. The interruption condition is thus cleared and condition code 1 is set as before.

If the device is busy or the control unit contains an interruption condition for a device other than the one addressed a busy bit is stored in the CSW and condition code 1 is set.

If the channel does not respond to the TIO before the CPU counts out, a time out signal is issued to the channel. The channel should respond by setting condition code 2 indicating that the channel is busy. If the channel fails to respond by the time the CPU counts out a second time, the same procedure is followed as for SIO.

C. Halt Input/Output (HIO) - The CPU issues a HIO signal to the channel and enters a countdown loop.

If the channel reaches the polling state with no request in from any control unit, it examines the state of the addressed subchannel. If there is a channel end type interrupt in IB for the addressed subchannel, the channel sets condition code 0 and no further action is required since the operation is obviously reaching a normal end.

If there is no interruption pending in the subchannel, the channel attempts to select the device. If the control unit responds with the control unit busy sequence (Status-In and Operation-In Not) the OP code in the UCW2 in the addressed subchannel is set to stop. A busy bit is stored in the CSW and condition code 1 is set. When the device next tries to transfer data, the channel will indicate a stop to the device. If the device is successfully selected, the channel (upon receiving Address-In), performs an interface disconnect sequence on the I/O interface. This is used to halt the operation at the device. A CSW is stored and condition code 1 is set.

If the channel is in a burst mode operation (the CPU has timed out once), an interface disconnect sequence will be presented to whatever control unit and device are on the interface at that time independent of the unit address in the HIO instruction. The channel sets condition code 2 indicating that a burst mode operation has been terminated.

If the channel fails to respond to the HIO by the time the CPU counts out a second time, the same procedure is followed as for SIO.

6.10 Direct Input/Output

There are four Direct I/O instructions.* Direct Input is used to transfer data from a device to main storage. Direct Output is used to transfer data from main storage to a device. Sense is used to transfer special device status information from the device to main storage. Control is used to send unique control commands to a device. All Direct I/O operations involve the transfer of 3 bytes, usually one status and two data bytes, between the device and the CPU main storage. There are no CAW's, CCW's, UCW's or CSW's associated with Direct I/O.

To illustrate Direct I/O, an Input operation is described. The I/O interface sequences are essentially the same for all four instructions. Direct I/O instructions utilize the CPU for the duration of the operation.

*As described in this section, DIO provides a non-privileged transfer of two bytes between mainstore and device. Special implementations allow four byte transfers, transfer between general registers and devices, etc.

The bits in the Direct I/O instruction are used to form a data address,unit address, and command. A DIO signal is issued to the channel. When the channel reaches the polling state, it breaks into the CPU, puts the unit address in BFR2 and raises Address-Out and Select-Out. The addressed devices respond with Operational-In which resets Address-Out. No address is sent to the channel by the device and thus no comparison is done. The Control Unit raises Service-In. The channel places the command byte on Bus-Out and raises Command-Out. The CPU then initiates a countdown loop and the channel waits for a hardwire branch signal.

The control unit must drop Service-In to reset Command-Out and then bring Service-In up again to get the first data byte. The channel forms the address of the proper microroutine and breaks in on the CPU countdown. The data byte is gated from the Bus-In to BFR2 to mover and into the proper byte position of WD. In this case, the byte position is either byte 1 if the upper half of a word is to be stored or byte 3 if the lower half of a word is to be stored. Service-Out is raised and the channel breaks out. The process is repeated. This time byte 0 or byte 2 is filled in WD. Service-Out is raised and the control unit drops Operational-In. The CPU stores the halfword at the location indicated by the data address. The other half of the word is not altered. The CPU now fetches the next instruction and the channel is placed in the polling state.

Each time the channel is in a branch awaiting an interface sequence from the control unit, the CPU enters a countdown loop. If the sequence is not completed by the time the CPU counts out, the channel performs an interface disconnect. The program is notified of this through the use of the condition codes. This prevents a control unit from tying up the CPU when the control unit has failed.

6.11 Resets

During the CPU system reset operation, the CPU issues a system reset signal to the channel. This signal is used by the channel to clear out its instruction latches, reset the BAR and General Branch Latch, reset various channel stats, and place the channel in the polling state. The UCW1 for each subchannel is stored as all zeros. Thus, the sequence controls indicate idle (0000). The UCW2 for each subchannel is stored with the OP code set to Stop (111) and zeros in the rest of the word. Thus, no data transfer can take place on a subchannel until after a Start I/O has been issued for that subchannel. However, an attention interrupt may take place prior to an SIO.

The system reset signal from the CPU will also cause a "System Reset" on the I/O Interface (Operational-Out and Suppress-Out down concurrently). This should cause all control units to reset. The channel can also perform a "Selective Reset" (Suppress-Out and Operational-Out up and the Operational-Out falls). This is done under microprogram control to reset just the control unit which is on the interface at that time. Other control units are not affected.

The multiplexor channel may be reset under microprogram control. The issuance of the Multiplexor Channel Reset causes the channel toreset as it would for system reset but does not cause a "System Reset" on the interface. A "System Reset" or "Selective Reset" may be microprogrammed along with the channel reset.

6.12 Multiplexor Channel Performance

Table 2 lists estimates for multiplexor channel data rates and the resultant CPU interference. It should be noted that actual data rates achievable are dependent not only on the CPU and channel but also on delays associated with the signal transition times on the I/O Interface and the delays associated with each control unit. Each system must be analyzed in this respect.

Table 2

Data Rate	CPU Interference	Mode
18KB	25%	Multiplex
36KB	50%	Maximum Multiplex
115KB	75%	Maximum Burst
400KB	100%	High Speed

In multiplex mode, the transfer of a byte of data requires about 14 us of processor time and transfers may be repeated at about 18.5 us interval. In burst mode, the first byte requires about 14 us of CPU time but all successive bytes in the burst require about 5 us of processor time and may be repeated at about an 8 us interval. In lockout mode, bytes may be transferred each 2.5 us and the processor is utilized for this entire time.

Up to eight (8) control units may be attached to the EP Multiplexor Channel. For each 8192 words of main store (256 words of Bump) there can be up to 64 subchannels up to a maximum of 128 subchannels. There may be up to eight (8) shared subchannels leaving 120 non-shared subchannels. The 8 shared subchannels, if used, are associated with the first group of 64 subchannels. The upper four bits of the shared subchannel address are zero. A maximum of 256 devices may be attached through control units to the EP Multiplexor Channel. If more than 128 devices are used, shared subchannels must be used.

7.0 TECHNOLOGIES

The term"technologies"refers to the physical components, devices and techniques which are used in building a computer. The following sections present preliminary descriptions of the techniques applicable to Model EP. In general, EP is a conservative design in that it utilizes only components and techniques which have been proven to be acceptable in stringently monitored programs.

7.1 Logic Circuits

The Model EP uses a series of circuits similar to the Texas Instrument Series 5400 TTL monolithic integrated circuits. These circuits are designed to meet military specifications and operate over the temperature range from -55°C to +125°C. Over 10,000 of these devices have been purchased for engineering model fabrication and extensive circuit testing. A minimum sample size of 100 units of each type was used for device characterization prior to circuit release. Turn-on and turn-off delays and transition times have been characterized over the temperature range from -55°C to +125°C with loadings from 10 to 300 mmf on each circuit output. Testing was performed to measure turn-on and turn-off delays at worst case conditions with a combination of Vcc 5% below nominal and temperature at -55°C with loads from 15 to 400 uufd. The data obtained from these tests has been fully utilized to guarantee proper operation even under anticipated worst-case conditions. Figure 7-1 and 7-2 shows typical rise, fall and propagation delays for NAND gates with 50 pf load.

Simultaneously, circuit application studies were performed to characterize the following:

- Noise sensitivity of all circuit modules
- Noise Sources

  a. Line-to-line cross talk
  b. Signal line reflections
  c. Voltage spikes on ground and voltages terminal at the flatpack due to circuit switching on the flatpack.
  d. Voltage spikes on ground and voltage distribution caused by circuit switching.
  e. Voltage drops on ground lines caused by signal currents.
  f. Pick-up signals caused by currents in the ground and voltage distribution system flowing to decoupling capacitors.
  g. Voltage drops on signal lines in passing through pluggable connectors.

- o Signal delay on terminated and unterminated lines
- o Effect of load clustering on termination mis-match
- o Effect of various loads typified by logic design requirements.

All testing to date shows that the design selected meets or exceeds circuit requirements. Since finalization of the specifications, two additional integrated circuit manufacturers have stated the ability to supply these devices.

The flatpack types used consist of NAND gates, AND-OR-Invert gates and Polarity Holds. A DOT-AND gate is a NAND which is modified so that a number of NAND outputs may utilize a common collector resistor thereby performing the DOT-AND function. In several instances, NAND gates are selected for greater drive capability or for better breakdown characteristics. These components are identified by special numbers as indicated. The following list identifies the flatpacks which are utilized within EP.

Description	Part Number
Quad 2 Input Nand	2839
Triple 3 Input Nand	2840
Dual 4 Input Nand	2841
Single 8 Input Nand	2845
Dual-Dual 2 Input And-Or-Invert	2966
Single-Quad 2 Input And-Or-Invert	2967
Quad 2 Input "Dot And" Gate	2968
Dual Polarity Hold	2836
Quad 2 Input Nand-Extended Drive	2969
Dual 4 Input Nand-Extended Drive	2970
Quad 2 Input Nand-High Breakdown	3875
Quad 2 Input "Dot And" Gate-High Breakdown	3874

The logic circuits have been designed to operate with a +5 Volt supply and a total variation of ±0.25 Volt at the circuit terminals. Permissible fan outs are 14 units of load except for the extended drive circuits which can drive 18 units of load. The table below indicates the average power dissipation for typical circuits.

Nand Gate	11.4 mw
A-O-I Gate	16 to 25.5 mw
Polarity Hold	38 mw

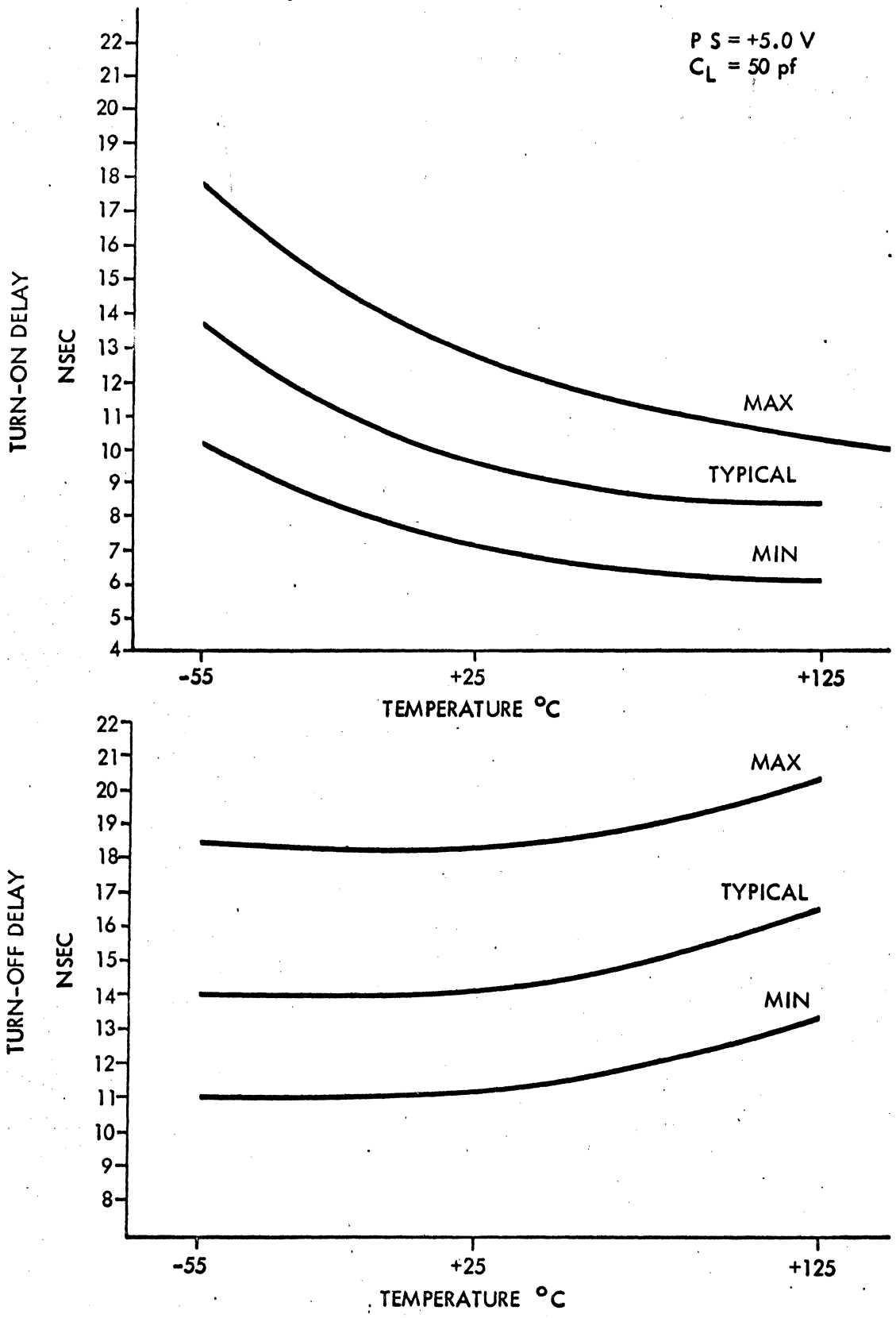


Figure 7-1. Turn-On and Turn-Off Delays Over Temperature Range

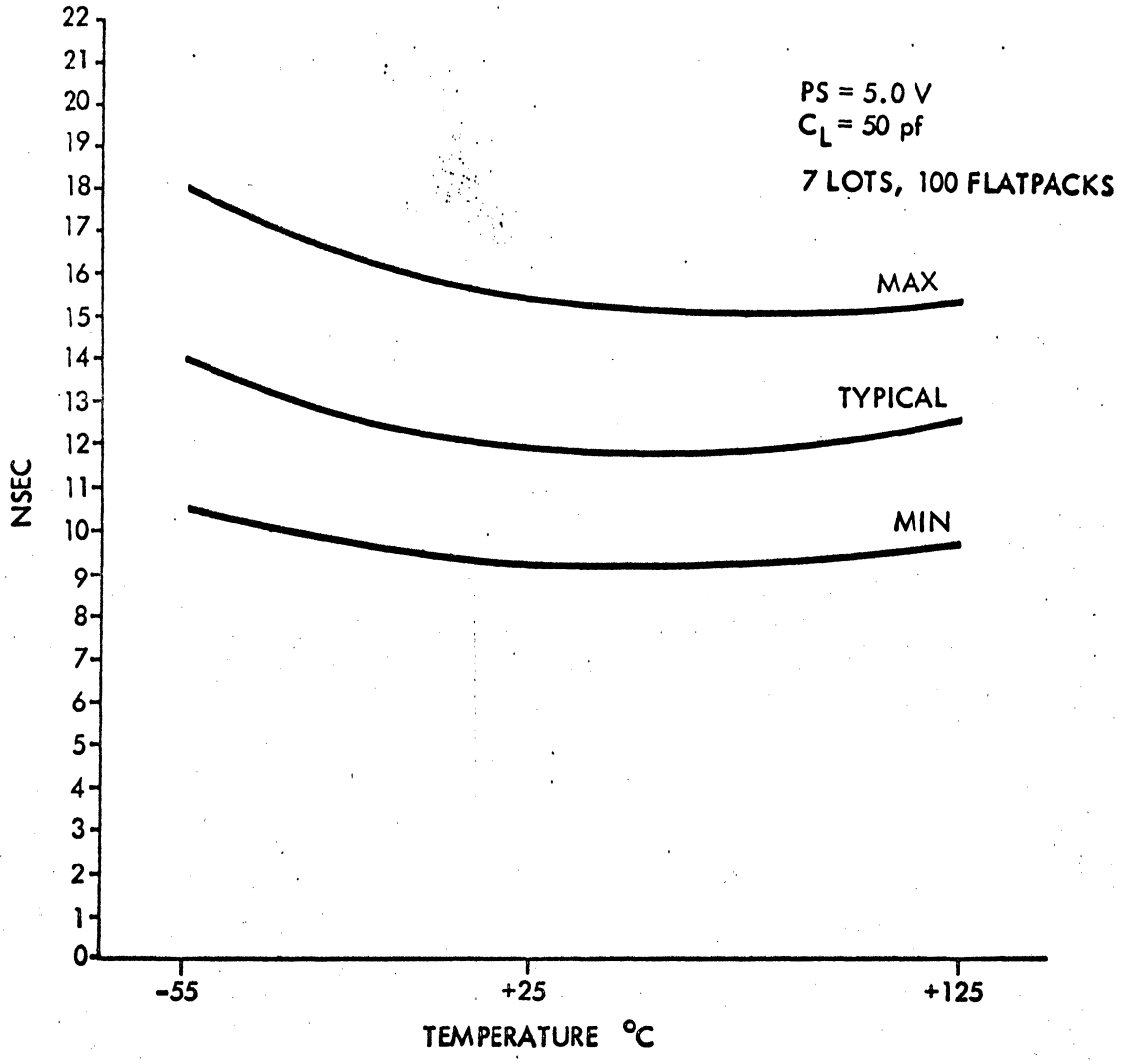


Figure 7-2. Propagation Delay Over Temperature Range

7.2 Page and Back Panel Assembly

The logic circuits are mounted on pluggable electronic subassemblies called "page," (Figure 7-3) which consists of two multilayer printed circuit boards bonded to a metal frame. An insulator separates each board from the frame. Two 98-pin connectors, similar to the ones being used for the Saturn V Guidance Computer, are fastened along the lower edge of the frame. The connector was developed for the Saturn V and has demonstrated its performance and reliability in extensive tests and field service. It has been qualified for use in man-rated space vehicles. Feed-through connections and test points are provided along the upper edge of the frame.

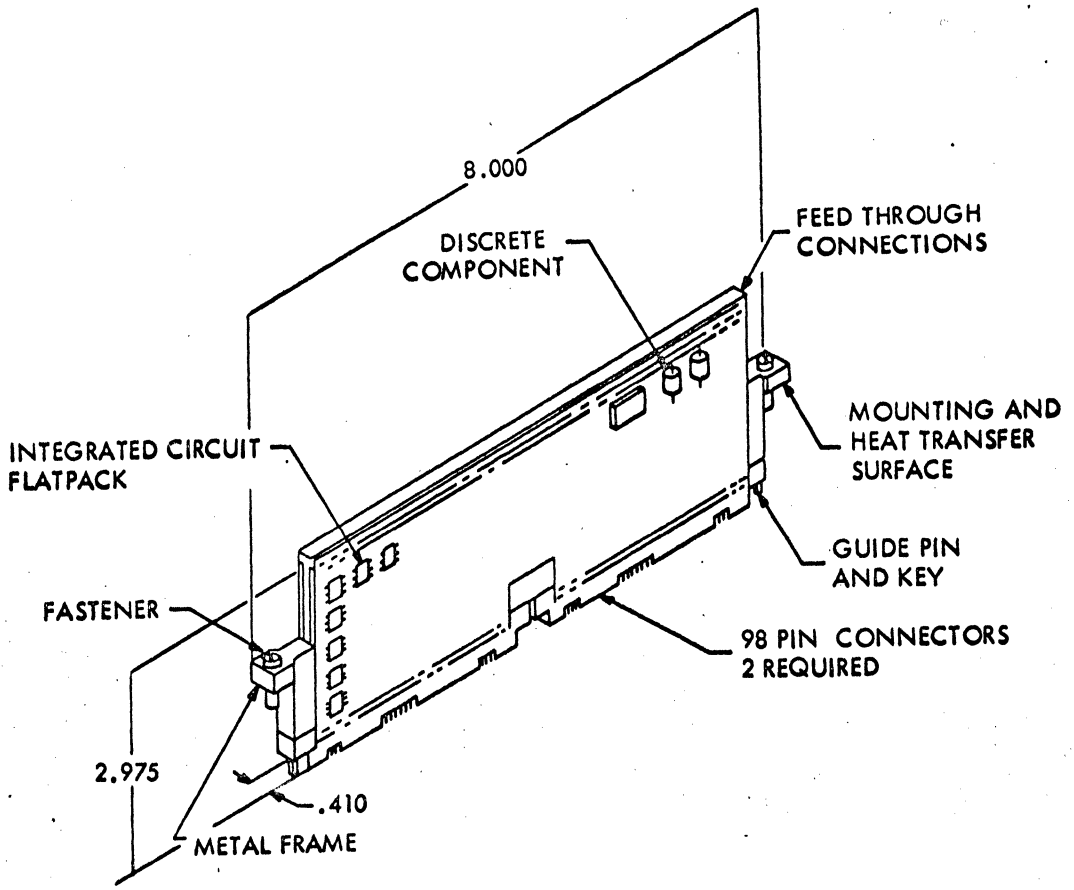


Figure 7-3. Pluggable Electronic Subassembly

The page is fastened to the supporting structure at the two mounting flanges along an axis through the page center of gravity. Additional support is provided by the connectors. The page will be cooled by thermal conduction through the frame and mounting flange. Keyed guide pins project from the lower edge of the page to prevent mis-location during installation. A tool will be furnished for page insertion and withdrawal. Guides will be provided to facilitate handling during page insertion and withdrawal.

The multilayer printed circuit boards are made of multiple layers of etched copper clad epoxy-glass laminates which are bonded together under heat and pressure. Four copper layers are used for signal interconnection and two layers are used for voltage and signal ground. Alternate copper layers are used as isolation ground planes which shield the signal layers and which establish a characteristic impedance of about 35 Ω for these lines. Connections between conductor layers are made through plated holes. These boards have been in volume production since 1962 and have been successfully used in many systems produced by IBM including the Titan II, Titan III, Gemini, Saturn I, and Saturn V Guidance Computers.

The integrated circuit flatpacks are soldered to etched patterns on the surface of the multilayer printed circuit boards. These multilayer boards allow the flatpacks to be closely spaced. Figure 7-4 shows one side of a typical page with flatpacks mounted. Each page contains two multilayer boards with up to 78 flatpacks per board. Discrete components are soldered in plated holes or on the board surface, depending on terminal configuration. A conformal coating is applied for component support and environmental protection.

Component Bonding to MIB

The improvement of component packaging density and the desirability of building large functional blocks have increased the thermal dissipation problem on the component level. Higher operating environments have also required closer thermal management of the packaging designs. Conformal coating are presently being used in existing programs such as Saturn V Launch Vehicle Computers to improve thermal dissipation and vibration resistance of components mounted to MIB's. Thermal studies being performed on the present system are in progress to determine the need for flatpack and component bonding to improve the thermal dissipation. The results of these studies will determine whether or not component bonding is imperative. However, in response to this potential requirement, an investigation is in process to provide an acceptable solution.

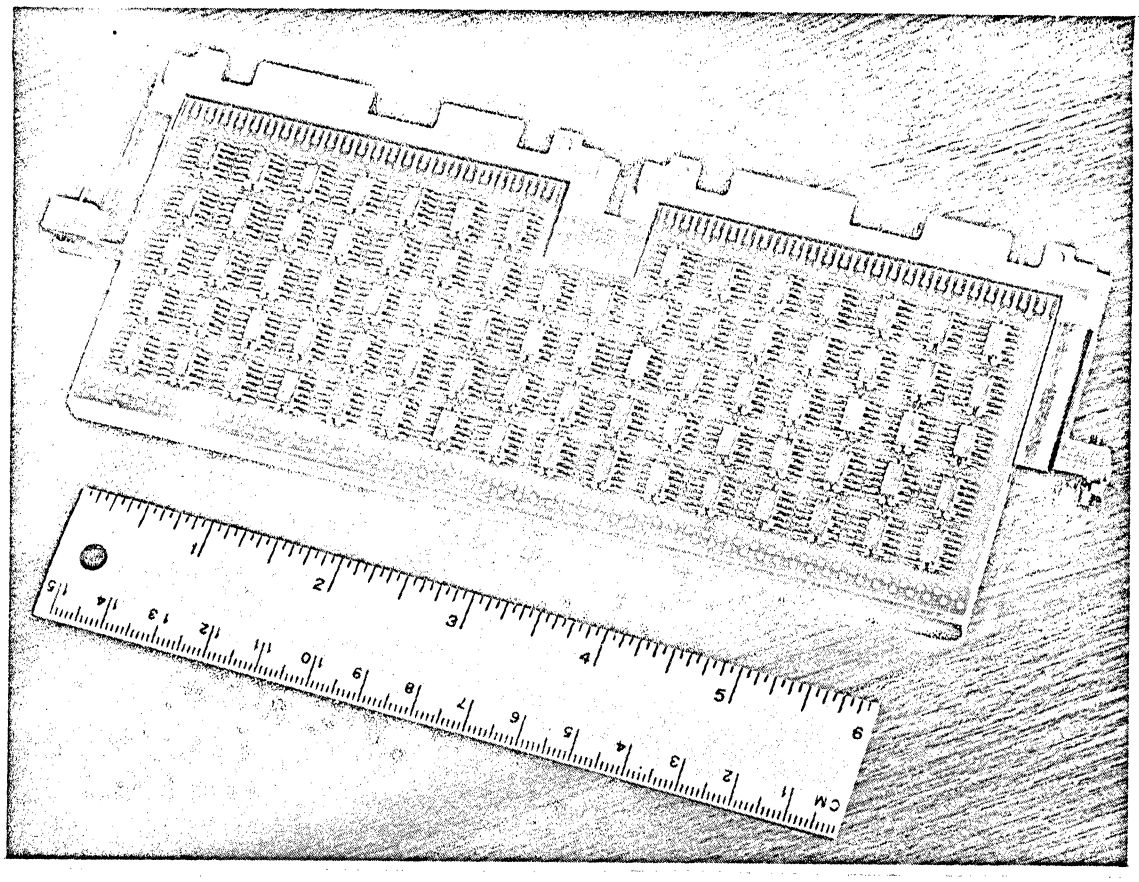


Figure 7-4. Typical Integrated Circuit Subassembly

The bonding material and process must meet the following requirements:

1. Provide an adequate thermal interface.

2. Permit simple application technique without interfering with the soldering operation.

3. Withstand soldering temperatures.

4. Be compatible with any conformal coating system.

5. Maintain adequate electrical insulation during normal operation.

6. Be easily severed to enable replacement of the flatpacks.

7. Improve the vibration resistance and minimize the stress on the solder joints.

The selection of the best bonding material must meet these conflicting requirements by analysis of the trade-offs available. Studies and vibration tests on previous programs indicate that the conformal coating improves the vibration resistance. This reduces the design bond strength of the bonding material and improves the probability of meeting the requirement for easy removal during flatpack replacement. The design of the bond must also protect the lands on the MIB by providing a weak plane at a point other than the face of the MIB. Field maintenance requirements also usually prohibit the use of oven curing adhesives, and require single techniques for flatpack replacement.

Preliminary evaluation of suitable materials has indicated that one of the flexible Polyurethane Elastomers offers the best probable compromise of the requirements discussed. A thixotropic version of the Polyurethane elastomer conformal coating is under investigation for use as a suitable bonding material. This material exhibits a reasonably good pot life and has properties that permit easy removal of the components, but still has sufficient bond strength to provide a good thermal interface. The use of a different catylist would permit room temperature curing for field use, for both bonding and recoating the component.

Interconnections between pages are made through a back panel assembly. A multilayer printed circuit board similar to that used for pages is bonded to a metal support plate and connector receptable terminals are soldered to plated through holes in the board. Interconnections between assemblies in the processor utilize flat cable which maintains the characteristic impedance.

All pluggable connections within the computer are made through one type of connector. It is a 98-pin connector which was designed for a similar application in the IBM Saturn V Guidance Computer. The contacts are a blade and form configuration providing redundant current path for each circuit. Gaskets seal the contact interface and between the receptable and back panel. Solder connections join the connector terminals to printed circuit boards or wires. External connectors on the computer are a round shell type, similar to the Bendix pygmy, using crimp connections to the wiring harness and poke-home contacts.

7.3 Interface Circuits and Interconnections

Signals which are transmitted between processor and mainstore use 30 ohm flat cable when the processor and mainstore are in the same structure. When they are in separate boxes, the chassis are carefully bonded and signals are transmitted between the boxes using standard TTL drivers and receivers. The transmission line consists of 90 ohm cable and the most distant storage unit is equipped with terminating resistors of the proper impedance to assure that no significant reflections will occur.

All other signals between EP and the outside world are transmitted either through a single ended driver-receiver pair or else by means of a differential driver-receiver pair.

The single ended system utilizes a TTL NAND driver which is selected for high breakdown and the receiver is a special Nand gate which is designed with threshold rejection of ±2 volt peak noise or ground shift. The output of the Single Ended Receiver (SER) can drive one unit load which must be located within the three-inch line length of the receiver output. Typical power dissipation of an SER is 17 milliwatts. Up to three SER's may be attached to one driver.

When greater common mode rejection is necessary, the signals may be transmitted differentially at TTL logic levels and may be received by a Differential Receiver (DR) which is designed to reject common mode noise and ground shift of up to ±5 volt peak. Typical power dissipation of a DR is 25 milliwatts. Up to five DR's may be driven from the same set of drivers.

Generally, 90 ohm twisted pair shielded lines will be carried between driver and receiver and shields will be terminated at one end only.

All shields on the twisted pairs must be insulated from each other and from ground except for the connection at one end. The shields may be commoned at the entrance to the chassis and the common shield should be returned to the power supply ground. Wherever cables must be run for any distance, an external high efficiency shield should be placed around the bundle and should be commoned to the chassis at one end of the cable.

Both the SER and DR transmission systems may encounter signal reflections which will result in delays proportional to the line length and to the number of reflections encountered.

7.4 Mainstore

The main storage is supplied in modules of 8,192 words of 36 bits/word. Each storage module is a destructive readout (DRO) memory and is designed to operate at a continuous 2.5 usec memory cycle. Each memory cycle consists of a read followed by a write operation. The access time for this system is approximately 0.9 usec. The memory system utilizes a coincident current (3-D) organization. Operation is possible over the component case temperature range of -55°C to +100°C.

The storage element for this system is an IBM 13/21 (I.D.-O.D. in mils) lithium nickel ferrite toroidal core. This core uses the same basic wide temperature material used in a larger diameter IBM core

currently in production. The core characteristics are as follows:

Operating Temperature Range:	-55°C to +100°C
Core Temperature Coefficient:	-0.00244 ma/ma/°C
Half Select Current at 26°C: (200 nsec Rise Time)	270 ma
Worst Case "one" (100°C):	8.0 mv
Worst Case "zero" (-55°C):	2.0 mv
Switch Time	500 nsec

The plane used in the memory array (Figure 7-5) is a militarized version of a commercial IBM plane. Automatic equipment is used to wire the planes. This plane is approximately 6 inches by 6 inches by 0.156 inches and contains 16,284 cores. Each plane contains eight mats, and each mat contains 2,048 cores, thus each plane contains 4 bits/word of the 4K memory. Welded connections are used throughout the memory array.

This plan is being utilized because of the significant advantages of manufacturing cost of machine wired planes and manufacturing process control. Foam pads are used between the coated core planes to satisfy the vibration requirements.

A block diagram of the 8K memory module is shown in Figure 7-6. The system is organized such that the "X" half select driver which drives the long dimension (128 x 36) is clocked first. The "X" current thus has time to reach full amplitude at a slower rate, which reduces driver supply voltage requirements.

Operating modes of the memory system are:

- o Read-Regenerate Cycle
- o Clear - Store Cycle
- o Split Cycle (Read-Compute-Store)

The systems gates, drivers, inhibit drives, and sense amplifiers utilize integrated circuits to the degree present technology permits. The monolithic circuits used in the memory system are basic "off-the-shelf" units obtained from integrated circuit manufacturers. The high voltage current address drivers, address gates, and inhibit drivers consist of monolithic pre-drivers which are transformer coupled to discrete chip output stages. The sense amplifier is basically a monolithic "off-the-shelf circuit" with additional resistors added for threshold setting, and sense line terminates.

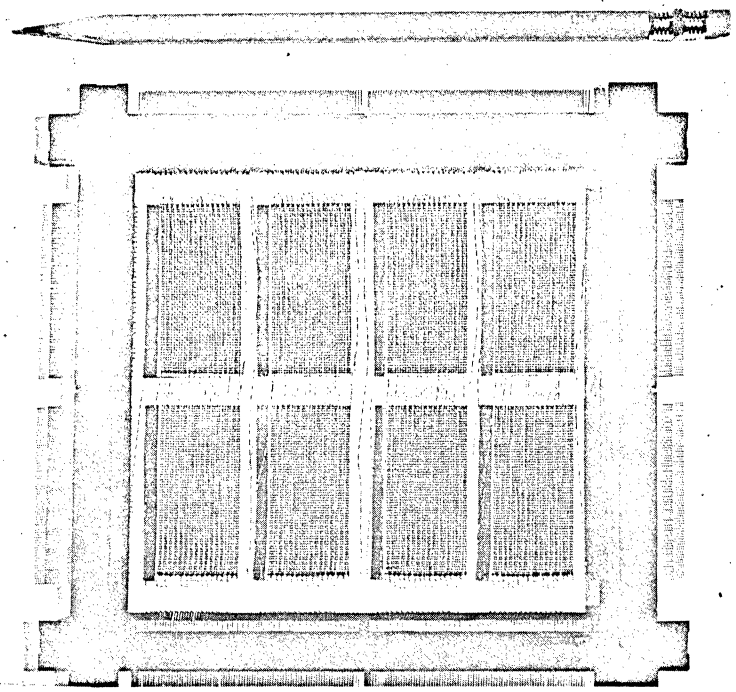


Figure 7-5. Main Storage Memory Plane

The timing generator is of hybrid design using flat-pack chip transistors and precision R-C networks. Decoding is accomplished by monolithic pre-drive stages in the address gates.

The memory power is approximately 80 watts when operating with a 50 percent mix of ones and zeros at a continuous 2.5-usec cycle time. The standby power is approximately 12 watts.

Details of the Memory Assemblies are shown in Figure 7-7. The memory array is fabricated from core planes which are militarized version of planes used on the IBM System/360. Each plane has 16,384 cores. Planes are conformally coated and foam padding is placed between planes for environmental protection of cores.

The array is mounted in a housing which serves also to mount pluggable electronic subassemblies and acts as a thermal path. Pluggable electronic subassemblies similar to those in the central computer are used to mount the memory circuits. A distribution multilayer printed circuit board is used to interconnect memory circuits and the array. The main memory assembly is removable as a module. Electrical connections to the assembly are made through pluggable input/output connectors.

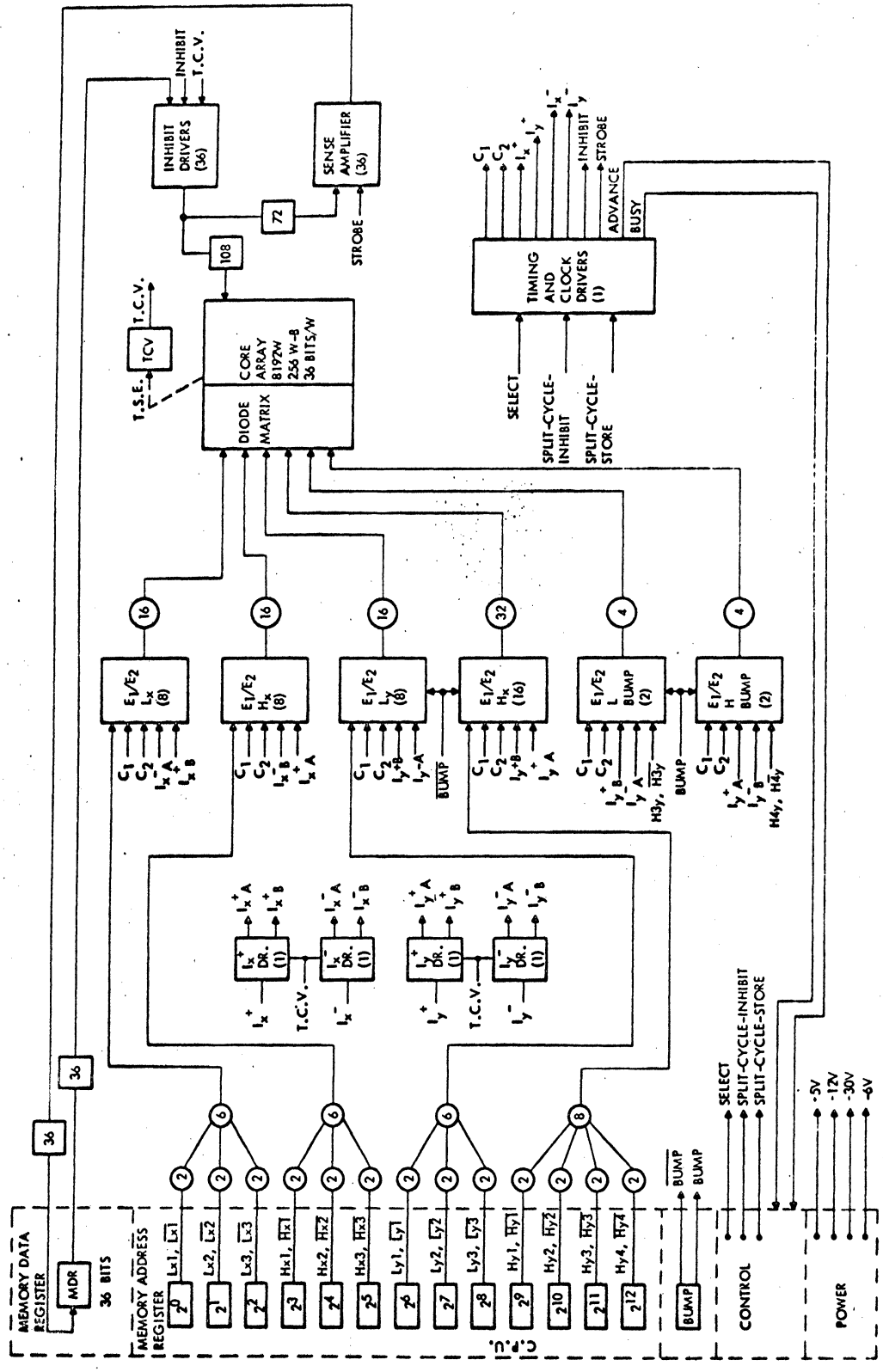


Figure 7-6. Storage Block Diagram (8K by 36 Bits)

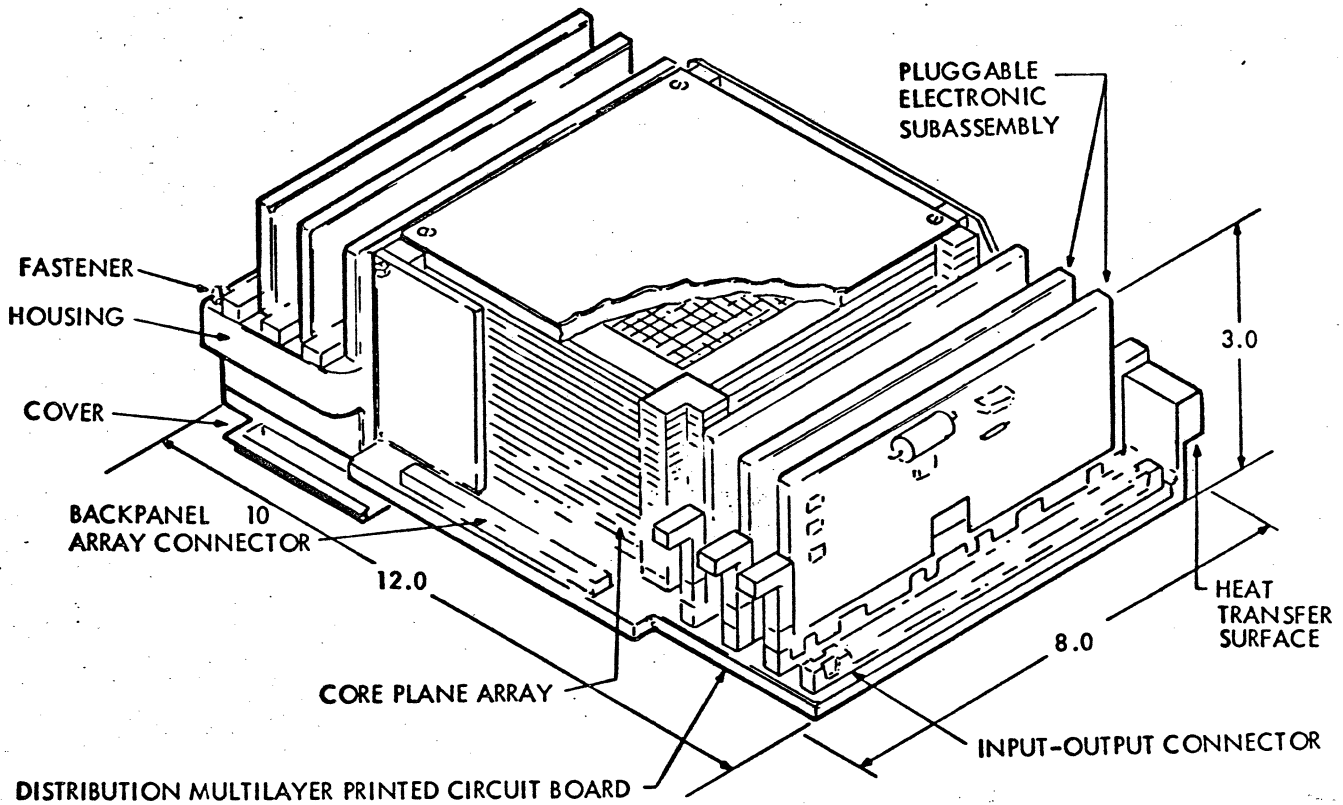


Figure 7-7. Storage Assembly

7.5 Read-Only Store (ROS)

The ROS has a storage capacity of up to 3072 words, 100 bits/word, and is operated at a continuous 416 nsec cycle. The access time is approximately 155 nsec. This store is word-organized.

The storage element is a microminiature linear ferrite core designed for operation over a wide temperature environment. The pertinent core characteristics are:

Temperature Operating Range	-55°C to +100°C
Full Select Current:	100 ma(10 nsec rise time)
Worst Case "one":	10 mv.

A block diagram of a typical ROS system is shown in Figure 7-8. The ROS-read command initiates a read cycle, and the data are available at the sense latches 155 nsec after the read command. The reset for the sense circuitry is generated within the ROS subassembly. A drawing of the ROS subassembly is included in the mechanical section.

Monolithic circuits are used in the decode, driver, detector and latch portion of the ROS. The sense amplifier is basically of monolithic design using capacitors for D.C. isolation.

The storage system power is approximately 13 watts based on continuous operation at a 416 nsec cycle time. The standby power is approximately 9.5 watts.

The ROS plane design is as shown in Figure 7-9. Core mats are mounted directly to multilayer interconnection boards to which the integrated circuits for signal driving and sensing are mounted. Planes are wired on automatic equipment which threads the cores, selects the personality for each plane by eliminating cores in word bit positions where "zeros" are desired, checks the function of remaining cores, and welds the mat terminations. A close-up of an individual mat is shown in Figure 7-10.

A typical read-only store memory assembly is shown in Figure 7-11. The memory array is constructed of multilayer printed circuit boards with core mats attached. Driver circuits are also mounted on these boards.

The array is positioned under a distribution multilayer printed circuit board. This distribution board serves to electrically connect memory circuits and array. The read-only store assembly is removable as a module. Electrical connections to the assembly are made through pluggable input/output connectors.

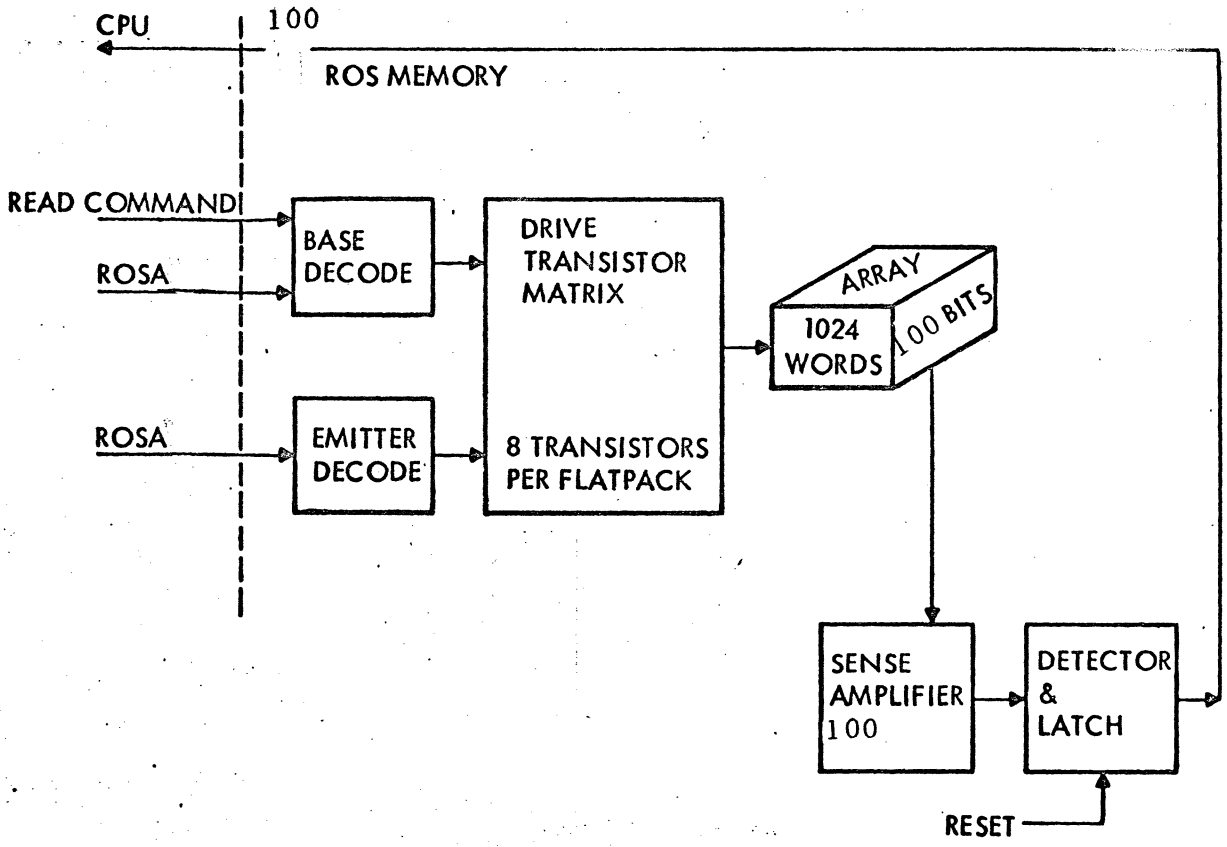


Figure 7-8. ROS Block Diagram

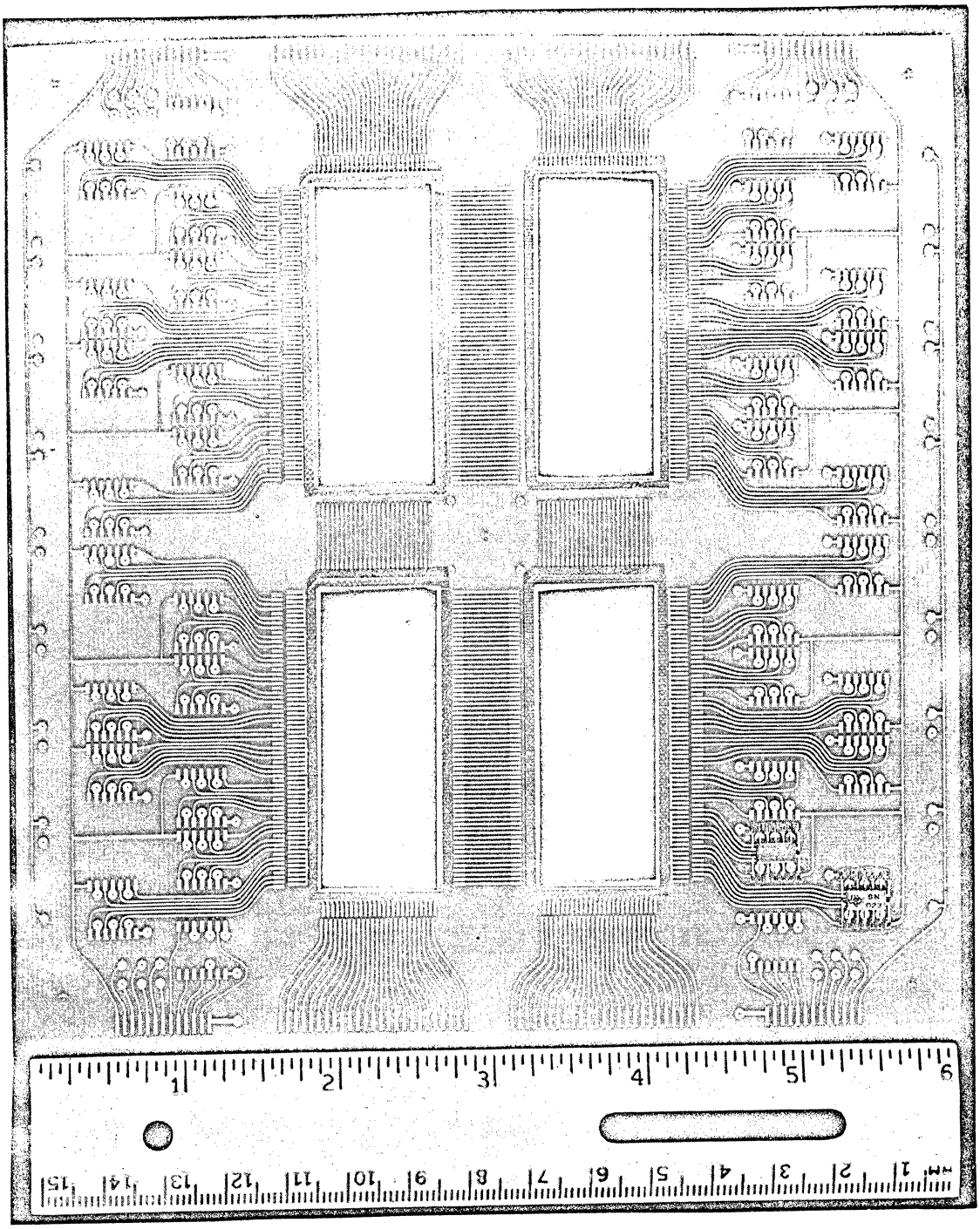


Figure 7-9. ROS Page Design

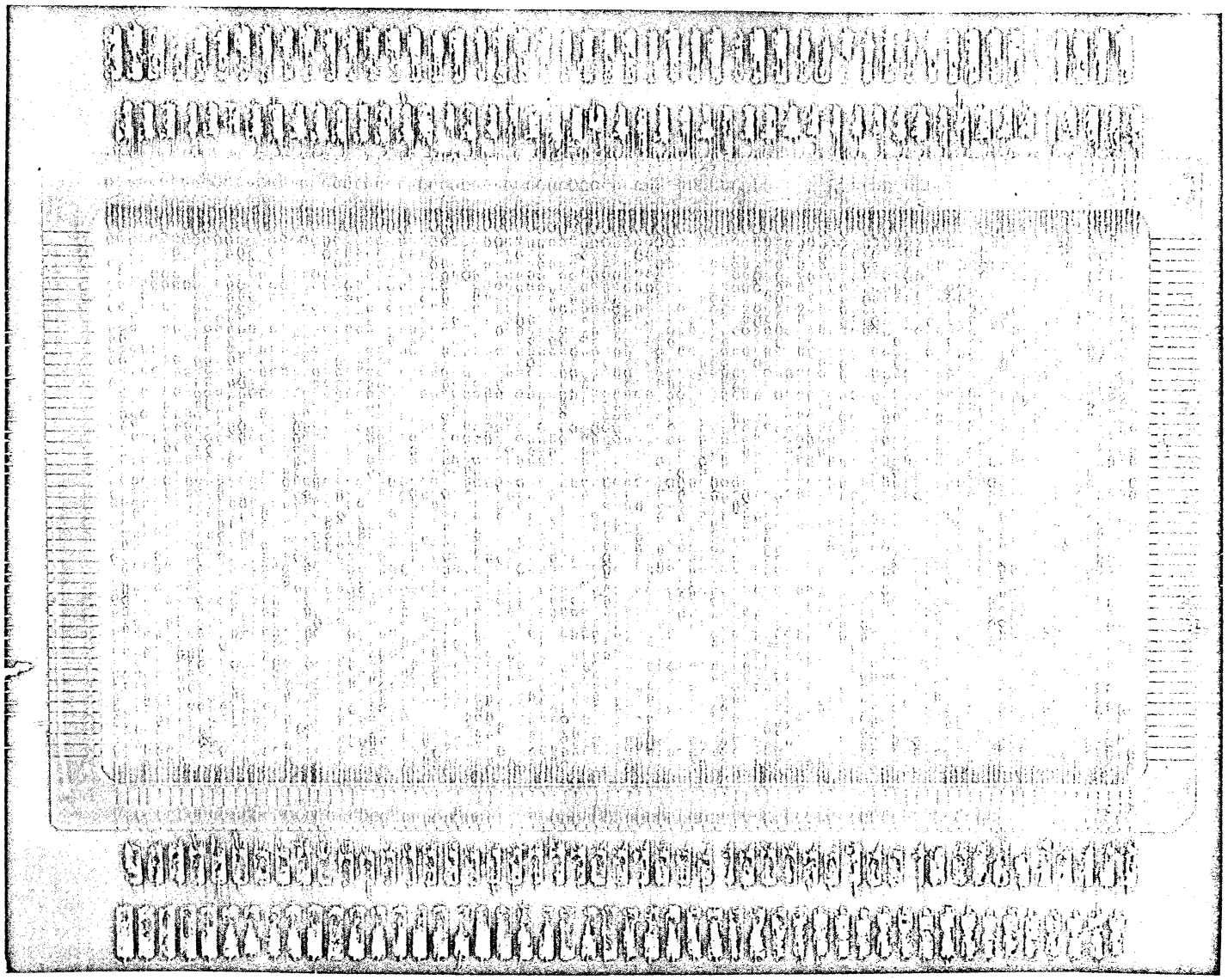


Figure 7-10. ROS Memory Plane

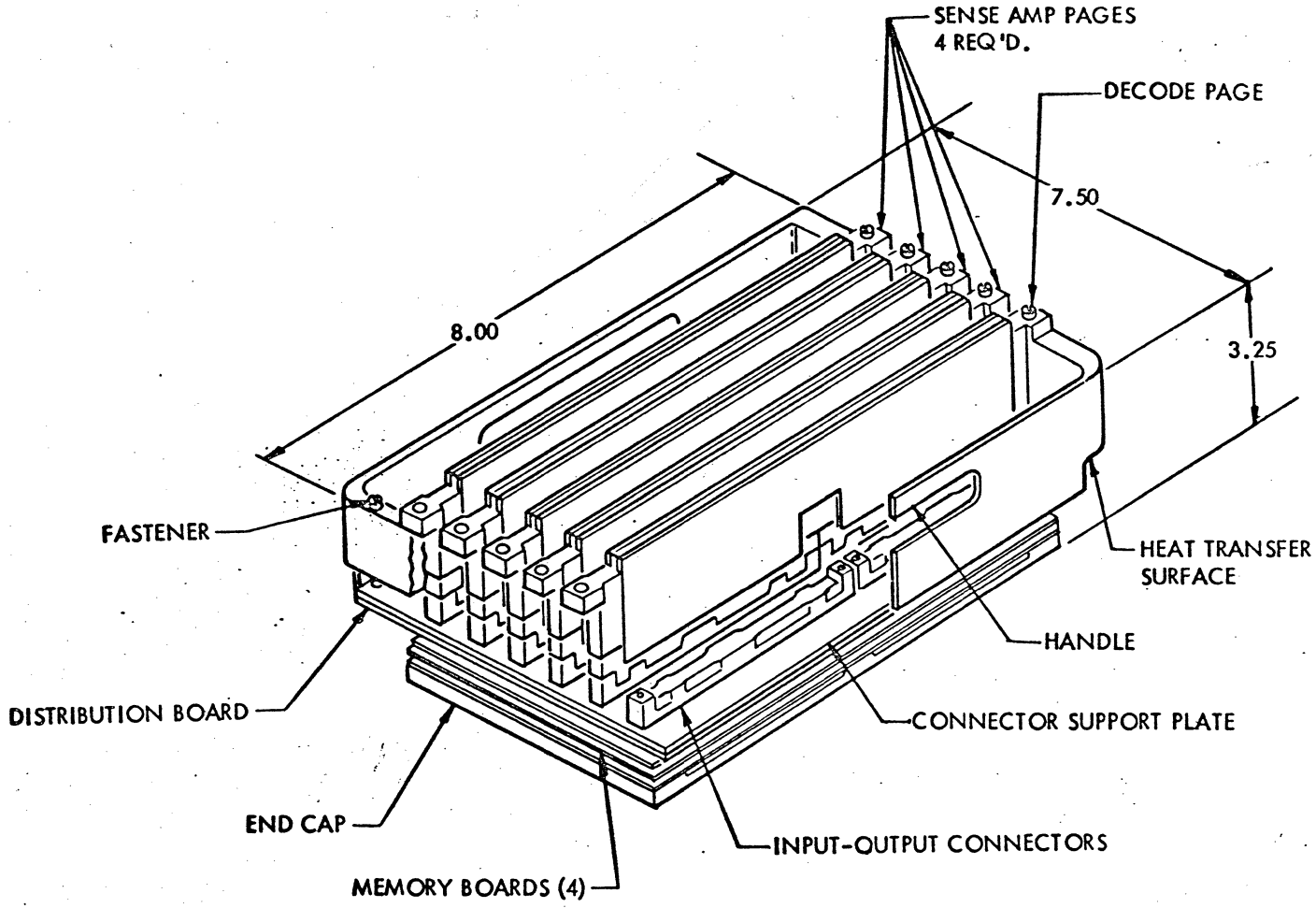


Figure 7-11. Read Only Storage Assembly

7.6 Power Supplies and Distribution

Not available at this time.

7.7 Structural Design

Not available at this time.