EAI

ELECTRONIC ASSOCIATES, INC. *West Long Branch, New Jersey*

EAI 8400

SCIENTIFIC COMPUTING SYSTEM

PRELIMINARY INFORMATION MANUAL

PRINTED IN U.S.A.

March 1965

EAI 8400 SCIENTIFIC COMPUTING SYSTEM

TABLE OF CONTENTS

SECTION		PAGE
II (cont.)		
7.0	SIMULATION PROGRAMS GROUP	7-1
7.1	Hybrid Mode Control	7-1
7.2	Integration Control	7-1
7.3	Function Generator Loader	7-1
7.4	Hybrid Computer Set-Up	7-1
7.5	Hybrid Debug	7-2
8.0	HYTRAN PROGRAMS GROUP	8-1
8.1	Static Check	8-1
8.2	Report Generator	8-2
8.3	Equipment Check-Out	8-2
9.0	DIAGNOSTIC SYSTEM	9-1

1.0 BASIC SYSTEM DESCRIPTION

The EAI 8400 is a new, exceptionally fast, scientific computing system that features a unique combination of capabilities for Real-Time Computation --

HIGH SPEED PROCESSING
FLOATING POINT OPERATION, and
FORTRAN LANGUAGE PROGRAMMING.

With these capabilities, the 8400 is uniquely suited for real-time applications in -- scientific simulation, hybrid computation, laboratory or industrial on-line monitoring and control, and batch scientific processing as well.

1.1 SUMMARY OF 8400 CHARACTERISTICS

GENERAL

- Stored-program, scientific computer
- Autonomous organization see p. 1-4
- Parallel mode
- Silicon and micrologic circuitry
- 16 word fast memory, 250 nanoseconds cycle time or access time ?
- Priority interrupt system with mask registers
- Power fail safe
- Save register See p. 3-1, 3-14
- Real-time clock

PROCESSOR

- Powerful instruction list - over 750 commands
- Floating-point arithmetic, 32 and 56 - bit

  24-bit mantissa, 8-bit exponent
  48-bit mantissa, 8-bit exponent

- Fixed-point arithmetic, 16 and 32-bit

fixed + floating

- Integer, or mixed-mode arithmetic with

  16-bit fixed-point integers and
  32-bit floating-point operands

- Index arithmetic, 16-bit 2^{16} = 64 K (full capacity)

- Typical instruction execution times:

32-bit FLOATING ADD	3.50 usec
32-bit FLOATING MPY	6.25 usec
32-bit FLOATING DIV	9.50 usec
56-bit FLOATING ADD	6.00 usec
16-bit FIXED ADD	3.25 usec
16-bit FIXED MPY	5.25 usec
16-bit FIXED DIV	7.50 usec
32-bit FIXED ADD	4.00 usec

16 bits for analog work?

- Byte manipulations with 1, 2, 4, 8, or 16 bits

- Seven index registers

- Indirect addressing

STORAGE

- Magnetic core memory

- Capacity to 65,536 words, directly addressable 2^{16} words

- Word size and utilization

  32 data bits, 2 EXEC bits, and 2 parity bits;
  half-word or full-word, and byte addressing

- Expansion with 4k, 8k, and 16k banks

  2 usec complete cycle time
  750 nanosecond access time

- Independent bank read write control

- Storage access by up to four processors four processors operating simultaneously? Yes

INPUT/OUTPUT

- Bi-directional buffered data channels -- up to eight available -- each handling up to fifteen access devices

not Analog channels

buffer

. Automatic Data Channel Processor available, providing simultaneous data exchange and compute capability direct memory access

Analog tie-in → . Flexible systems interface for real-time, device-systems integration

. Peripherals

magnetic tape systems, card readers and punches, line printers, paper tape reader and punch.

CONSOLE

. The EAI 8400 System includes an operator's console with complete register display and on-line typewriter.

SOFTWARE

. 8400 Standard Monitor System

MACRO Assembler, FORTRAN IV, subroutine library, and programs for problem preparation, de-bugging, up-dating and modification

. 8400 Simulation Monitor System

Hybrid mode control, integration control, function generation, and other programming aids--for digital and hybrid scientific simulation

. 8400 HYTRAN[S] Monitor System

Designed primarily for processing programs used for preparation and check-out of analog and hybrid computer programs

Assembler - PHAP?
Debugg I - SPECTRA
Debugg II - CASPER?

[S] a service mark of Electronic Associates, Inc.

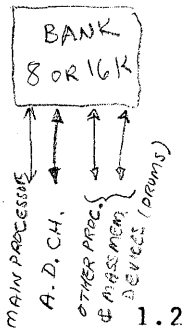
1.2 BASIC SYSTEM ORGANIZATION

The basic system organization of the EAI 8400 integrates the operation of three autonomous subsystems; Memory, Floating-Point Processor, and Exchange Module. A fourth subsystem, the Automatic Data Channel Processor, is an optional expansion for the system. Each of these subsystems has independent timing and control facilities. System interrelationships are on a request/response basis. The control autonomy feature provides an unusual expansion flexibility for increasing machine throughput, as well as capacity. Figure 1-1 illustrates the internal system organization and various means of interfacing with external devices and systems.

1.2.1 The Memory

4×16 = 64 K

The Memory is structured with one, two, three, or four independent Memory Banks, each having autonomous control and data handling facilities for processing storage read and write requests from the system processors. Banks of 8192 and 16,384 word capacities are available, and may be used in any combination. Each Memory Bank has four storage access channels. In a typical multi-bank system, the first access channel of each bank is connected to a bus from the Floating-Point Processor and the second channel is connected to a separate bus from the Automatic Data Channel Processor. This configuration provides overlapped memory access by the Floating-Point Processor, as well as simultaneous Input/Output and Computation. The third and fourth access channels may be used for multiprocessor expansion and/or interfacing with external mass memory devices...



1.2.2 The Floating-Point Processor

The Floating-Point Processor employs a 32-bit wordlength which provides for a powerful instruction repertoire (over 750 commands) and direct addressing of up to 65,536 words of memory. It has unusually extensive capabilities for both arithmetic and logical operations. The Processor is designed with floating-point as a basic, rather than expansion, capability. This concept is fundamental to the 8400's exceptional floating-point speed and storage efficiency characteristics.

1.2.3 The Exchange Module

analog

The Exchange Module contains a Data Channel System for interfacing with standard external devices and a System Interface for special device and Systems integration. The Data Channel System provides communication paths and control for up to eight bi-directional Data Channels, each capable of handling fifteen device controllers (including A-D and D-A conversion equipment). The channels are designed for the new 8-bit ASCII and EBCDIC peripheral codes and have internal logic for byte assembly and disassembly, parity generation and checking, and collating

card m tape etc

for plotting probably

code conversion. Independent channel operation may be under program control, or under control of the Automatic Data Channel Processor. The Systems Interface includes a directly addressable input/output bus system and provision for control lines and external interrupt lines as required for hybrid or other system integration.

1.2.4 The Automatic Data Channel Processor

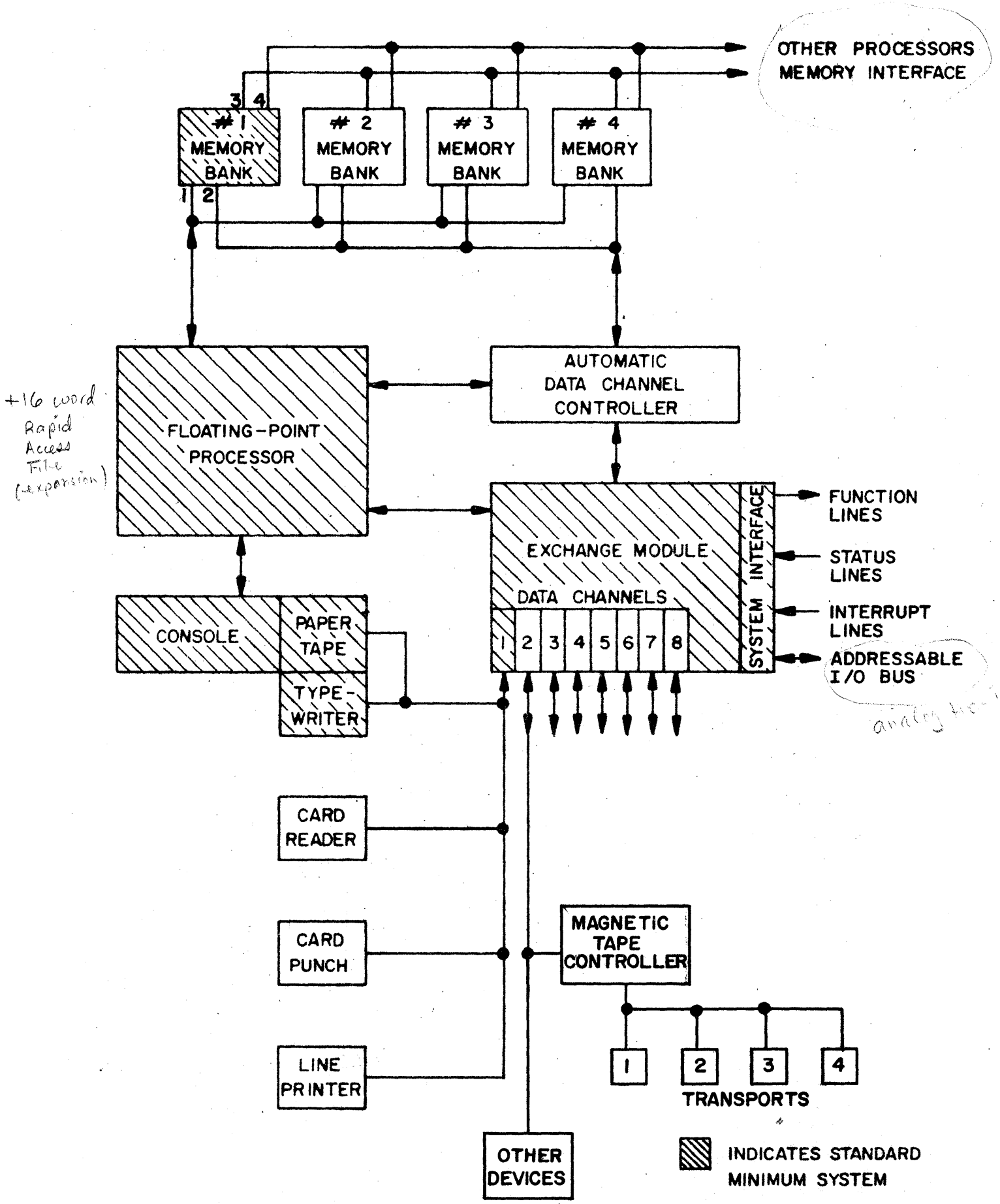
The Automatic Data Channel Processor provides a means of control for the Data Channels that permits block data transfers independent of the Floating-Point Processor. Once initialized it executes a complete block data transfer between the selected peripheral devices and memory, with data transmission occurring over a separate ~~memory bus~~. access channell
└ Separate from the processor↔memory access channel

1.2.5 Expansibility

Modular expansibility in both capacity and speed are fundamental to the 8400's basic design.

Throughput of the Floating-Point Processor can be increased by increasing index register speed with optional conversion paks, or by the addition of an optional 16 word fast memory (Rapid Access File) for scratch-pad and high-speed looping techniques. Faster storage processing is obtained by over-lapping when the initial Memory is expanded with additional banks and Exchange Module throughput is increased by simultaneous channel operation when new Data Channels are added. Further increases in speed can be realized by expansion to a multi-processor system in which several Floating-Point Processors operate in parallel.

The control autonomy feature of the 8400 provides an unusual expansion flexibility permitting the up-dating of individual subsystems with new technological advances, without obsoleting the existing initial system hardware.



EAI 8400 SCIENTIFIC COMPUTING SYSTEM

FIGURE 1-1

2.0 STORAGE

The Memory provides high-speed, random access storage of instructions and data used by the Floating-Point Processor and Exchange Module. It has a maximum directly addressable storage capacity of 65,536 words, 131,072 half-words, or 262,144 8-bit bytes. The capacity is provided by independent banks each having control and data handling facilities for processing storage requests from four system processors. The banks are of a non-volatile ferrite core construction and are characterized by a 2 microsecond complete cycle time and a storage access time that is 750 nanoseconds.

2.1 STORAGE CHARACTERISTICS

2.1.1 Storage Word

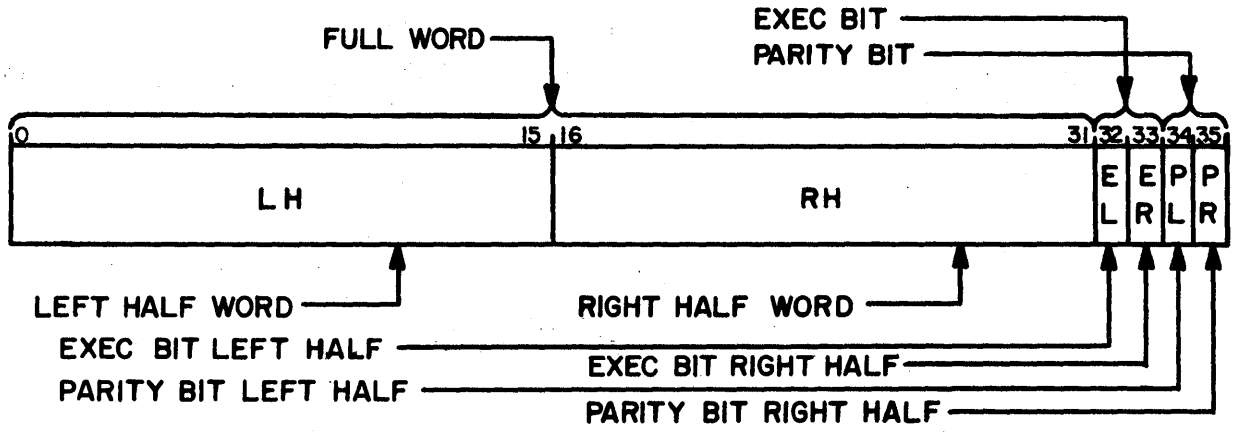
The memory word of the 8400 is comprised of 32 bits for information storage, 2 EXEC bits for special control functions and 2 parity bits; thus a memory word is 36 bits in length. The information portion may be used alternatively as one full-word location, for the storage of a 32-bit operand or instruction, or as two half-word locations for the storage of 16-bit operands or address fields. A parity bit and an EXEC bit are assigned to each half-word. The EXEC bits, an exclusive feature of the 8400, are used as markers for such purposes as dynamic re-location, list processing and table manipulation. Figure 2-1a shows the memory data word format.

2.1.2 Storage Addressing

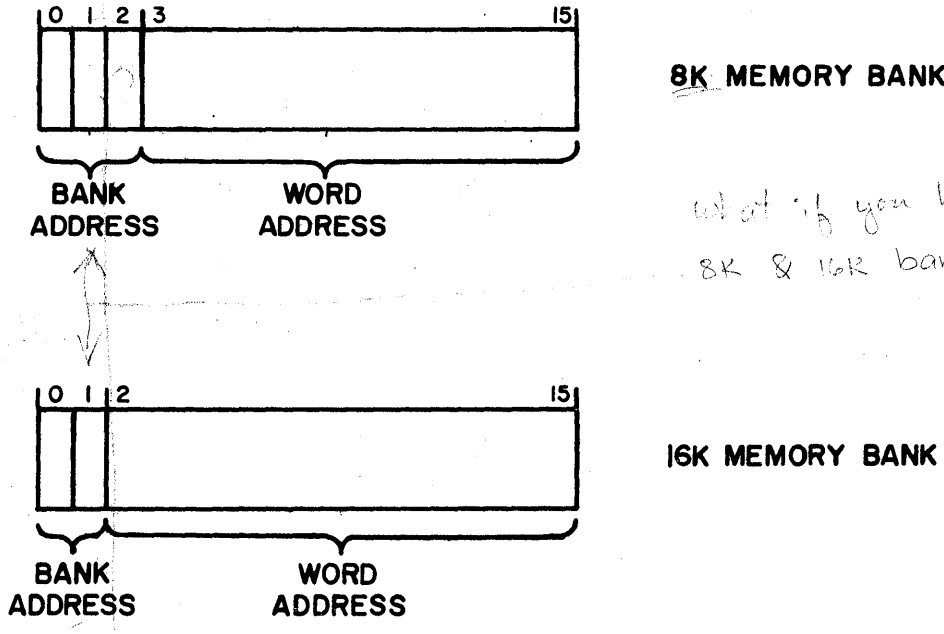
The location of data in storage is identified by a 16-bit memory address word. The bit designations of the address word are interpreted differently by memory banks of different storage capacities. Figure 2-1b shows word format and its interpretations by an 8192 word bank and 16,384 word bank.

2.1.3 Storage Access

The Memory Banks in an 8400 Memory may be accessed by up to four processors. These may be Floating-Point Processors, Automatic Data Channel Processors or external mass memory devices. The processor requesting access to storage transmits a request signal and a 16-bit address field to storage. The request signal specifies the read or write operation to be performed and whether full-word, half-word and/or EXEC bits are to be transfered. If the same bank is accessed simultaneously by two request sources, the bank control unit services the requests sequentially.



MEMORY DATA WORD FORMAT
FIGURE 2-1A



ADDRESS WORD FORMATS
FIGURE 2-1B

2.1.4 Storage Parity

One parity bit accompanies each half-word transfer in the 8400 and two parity bits accompany full-word transfers. Odd parity is employed; that is, the parity bit is set such that the number of "1's" in a half-word, plus the parity bit, is always an odd number. During a write cycle, the correct parity bit is generated and stored automatically. During a read cycle, the detection of a parity error causes the Memory Parity Indicator to flash at the console and initiates an internal Parity Failure interrupt. The memory bank causing the error may be located by the interrupt subroutine or, alternatively, by the operator using the Bank Select Switch on the console maintenance panel.

2.2 EFFICIENT CAPACITY UTILIZATION

Directly addressable data units which can be stored in one 8400 memory location include 32-bit full-words and 16-bit half-words, as previously noted. In addition, the Processor provides Double Precision Floating-Point Instructions which permit the direct sequential addressing of a contiguous word-pair and a set of Logical Connective Instructions which enable the direct addressing of 16, 8, 4, 2, and 1-bit bytes. Thus, the effective addressable storage capacity of the memory depends not only on the number of memory locations; but also on the sizes and mix of the data units to be stored. For example, a 64k memory has storage capacity for 64k floating-point or fixed point operands of 32-bit word length, 128k fixed-point operands of 16-bit word length or 256k ASCII or EBCDIC peripheral code characters. Figure 2-2 summarizes the effective storage capacities for all of the 8400 storage unit sizes and indicates the types of program information that utilize each of the sizes. The variety of useful types of information which can be stored in storage units of an exactly matching size results in highly efficient utilization of the available storage, by permitting dense packing of information with almost no waste capacity.

All memory addresses are available for general program use with the exception of a few addresses reserved for special purposes. The reserved addresses are listed below (in octal notation):

8 words	Addresses 00000-00007:	Reserved for the Accumulator, Save Register and 6 Index Registers
16 words	Addresses 00010-00027:	Reserved for the Rapid Access File
17 words?	Addresses 00040-00060:	Reserved for 16 Internal Interrupt line locations
256	Addresses 00061-00460:	Reserved for 256 External Interrupt line locations

The Accumulator and Save Register are standard arithmetic hardware registers

EFFECTIVE ADDRESSABLE STORAGE CAPACITY

ADDRESSABLE DATA UNIT	DATA UNITS/ MEMORY WORD	STORAGE CAPACITY -WITH 64K MEMORY	TYPE OF INFORMATION
64-BIT WORD PAIRS	½	32K	DOUBLE PRECISION FLOATING POINT OPERANDS
32-BIT FULL WORDS	1	64K	FLOATING POINT AND EXTENDED FIXED POINT OPERANDS; or INSTRUCTIONS
16-BIT WORD	2	128K	FIXED POINT, INDEX AND INTEGER OPERANDS; or ADDRESS FIELDS
16-BIT BYTES	2	128K	4 HEXADECIMAL CHARACTERS, 4-DIGIT BCD CODES; or OTHER 16-BIT FIELDS
8-BIT BYTES	4	256K	ASCII AND EBCDIC 8-BIT CHARACTER CODES; or 6-BIT ALPHANUMERIC CODES
4-BIT BYTES	8	512K	BCD NUMERIC CHARACTER CODES; or OTHER 4-BIT FIELDS
2-BIT BYTES	16	1024K	2-BIT TEST OR DECISION MAKING STATUS CONDITIONS
1-BIT BYTES	32	2048K	1-BIT TEST OR DECISION MAKING STATUS CONDITIONS

FIGURE 2-2

addressable as locations 00000 and 00001. The index registers and Rapid Access File are optional hardware registers whose functions are fulfilled with core memory locations in systems in which these options have not been elected. Any of the locations reserved for external interrupts may be used for other purposes in systems not requiring the full external interrupt line capacity of the 8400.

2.3 EFFICIENT CYCLE TIME UTILIZATION

Each of the independent banks of which the Memory is comprised is an autonomous storage module capable of responding to read and write requests from up to four storage request sources. The control autonomy provided for the banks makes possible the use of several operational techniques that effectively increase the processing speed of the system. The techniques are as follows:

2.3.1 Concurrent Memory Operation

In this mode of operation, words in different memory banks are accessed simultaneously by different subsystems; for example a Floating-Point Processor and an Automatic Data Channel Processor.

2.3.2 Overlapped Memory Operation

In this mode of operation words in different memory banks are accessed in "overlap" fashion by one subsystem; as in the case where a Floating-Point Processor while storing data in one bank begins fetching the next instruction from a different bank.

2.3.3 Combined Concurrent-Overlap

In this mode of operation the requesting subsystems operate concurrently and overlap their individual memory accesses by addressing the same banks alternatively; and different banks simultaneously.

The use of these three modes of operation can significantly reduce operating time in multi-processor, multi-user and single user systems.

3.0 FLOATING-POINT PROCESSOR

The Floating-Point Processor employs a 32-bit word length which provides for a powerful instruction repertoire (over 750 commands) and direct addressing of up to 65,536 words of memory. It has unusually extensive capabilities for both arithmetic and logical operations, as well as extensive control capabilities. The Processor is designed with Floating-Point as a basic, rather than expansion capability --- a concept fundamental to the 8400's exceptional floating-point speed and storage efficiency characteristics.

The processor provides system control for the 8400 system of autonomous functional modules, integrating its own operation with the operation of the Memory and Exchange Module. This role of systems control is illustrated by the diagram of Figure 3-1.

As the central processing unit of the 8400 Computing System, the Floating-Point Processor provides all of the capabilities for control and execution of the stored program. An indication of its powers and capability in this role is given by the following descriptions of the principal registers affecting operation:

Processor Registers

1. Instruction Register (I) contains the instruction currently being executed.

2. Location Counter (L) contains the address of the next instruction to be executed. The register is addressable under program control.

3. Accumulator is designed to be "universal"; i.e. every variety of arithmetic and data manipulation is performed with the one register, making programming simpler. It consists of four sections: a 16-bit accumulator, a 16-bit extension for 32-bit fixed-point, a 16-bit extension for 32-bit floating-point, and a 24-bit second extension for 56-bit floating-point. All manipulations between the accumulator sections are handled automatically. The accumulator is addressable as memory location zero.

4. Save Register ($) saves the current contents of the Accumulator concurrently with executing an arithmetic instruction. It is addressable as memory location one.

5. Index Registers (X) provide automatic address modification. Six core index registers and accumulator index capability are basic to the computer. The accumulator is index register 1, the rest are X2 thru X7.

6. Flag Register (F) contains indicator bits, set as the result of arithmetic operations, exchange and interrupt status signals. The Flag Register is addressable. as what location?

7. Internal Mask Register (IM) contains a 16-bit priority pattern specifying which interrupt conditions should be acknowledged.

8. External Mask Register (EM) contains a 16-bit pattern as described above but for external interrupt conditions....

9. Interval Timer Register (T), provides an optional real-time clock whose contents are decremented by one, every millisecond. When contents are reduced to zero, an interrupt signal is generated, the initial value is reset, and the clock continues. The Timer register is addressable.

10. Rapid Access File provides 16 hardware registers of 32 bits and 2 EXEC bits. These registers have a 250 nanosecond access time for high speed data storage or instruction execution. When this option is not present in the system, 16 memory locations respond to Rapid Access File instructions.

11. Console Register (C) is accessible by the program and by the operator. It allows monitoring, data display, and data input while the program is running.

The control and arithmetic-logical capabilities of the Floating-Point Processor are described separately, in the immediately succeeding sections; 3.1 and 3.2, respectively.

EAI 8400 SYSTEM BLOCK DIAGRAM

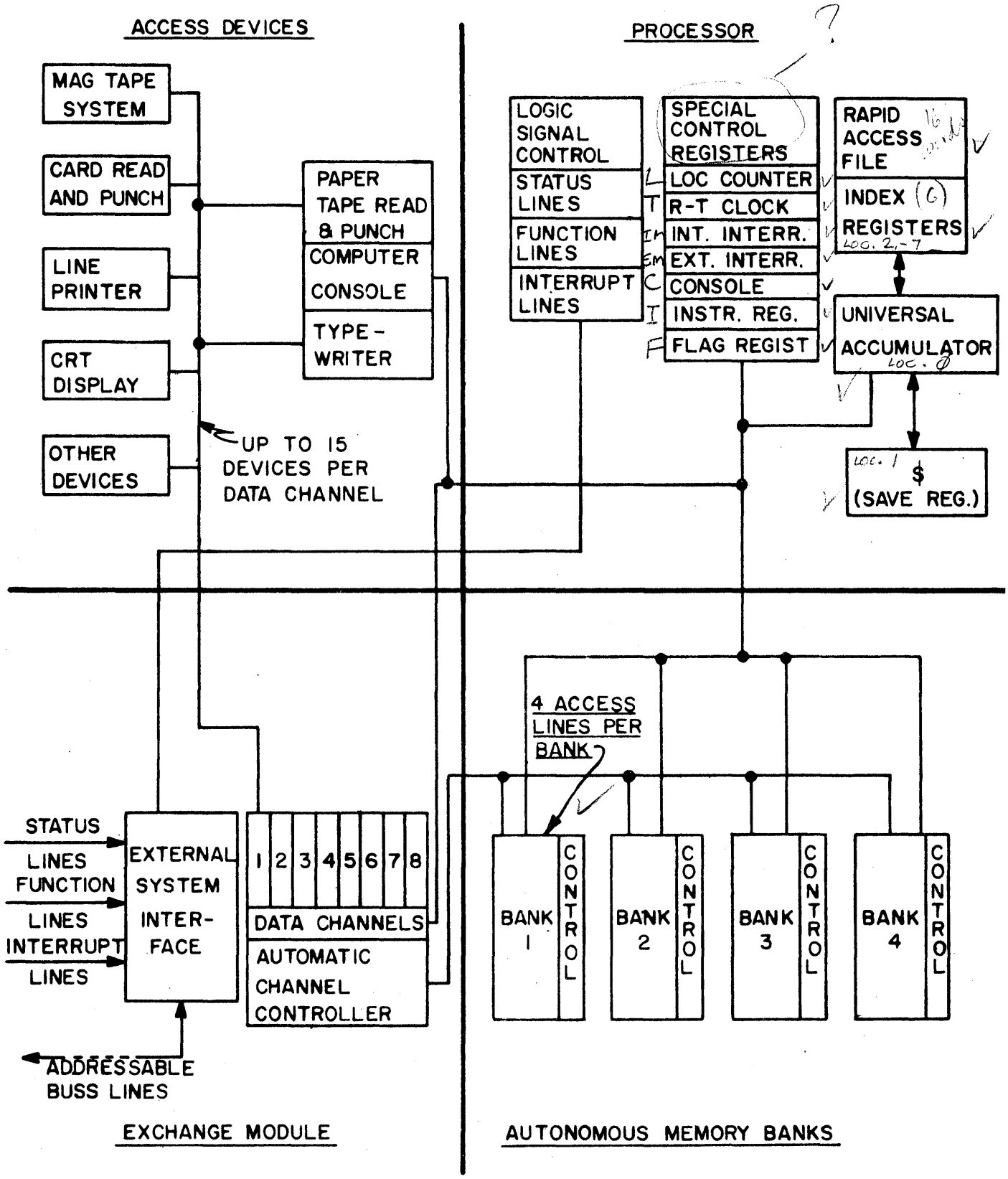


FIGURE 3-1

3.1 CONTROL FUNCTIONS

The control functions performed by the Processor include the sequencing of the computer through its program, the interpretation of instructions, the timing and gating of data flow, and the establishing of control relationships and priorities for the 8400 System. These functions are carried out through the use of the following control techniques:

a) Logic timing and gating of the various system data busses, enabling the unique data paths required for the execution of each instruction; hardware control of other system elements as required by the operation being performed using the decoded instruction as its guide.

b) Automatic Internal Interrupt System which provides the facility for continuously monitoring various conditions of the computer or its environment, and notifying the main program when certain conditions occur. If an interrupt signal is acknowledged, the main program will stop and transfer control to a subroutine, which services the condition. Use of the interrupt system permits immediate detection of system faults and facilitates the operation and coordination of asynchronous external devices.

c) Program-controlled Internal Status and Function Lines which allow the programmer to monitor and set control lines throughout the system.

Several special registers are employed in some of the control functions. These include the Flag Register which provides on-line monitoring of the status of sixteen machine conditions during the course of a program and optional Interval Timer Register and Rapid Access File. An exclusive feature of the 8400 is its unique EXEC BIT Control System which enables the use of the EXEC bits marking each half word storage location in memory for special control functions, such as dynamic relocation of subroutines and coding sequences from one location in memory to another.

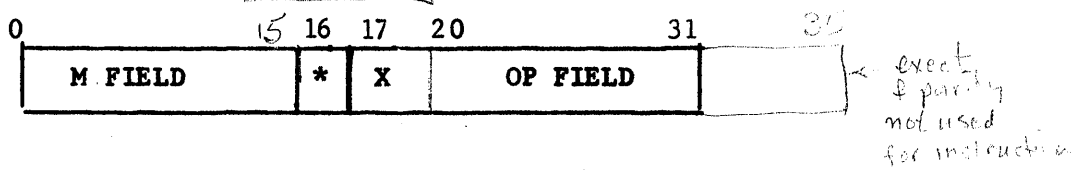
but still requires a pretty sophisticated exec routine - not like RCA's hardware dynamic relocation via content-addressed memory!

3.1.1 Instruction Characteristics

The 8400 is a single address computer employing a 32-bit instruction word. Under the normal mode of control the stored program is executed sequentially. The current instruction is contained in the Instruction Register (I) and the address of the next instruction is contained in the Location Counter (L). The execution of an individual program step is determined by the current instruction which is interpreted by logic and timing circuitry that implements the various functions

to be performed. The normal program control capabilities therefore are indicated by the 8400 instruction word format which is shown below. These capabilities include addressing, address modification, and instruction interpretation and control.

8400 INSTRUCTION WORD FORMAT



M is a 16-bit address field. * is an indirect address bit.
X is a 3-bit index register field. OP is a 12-bit command.

Addressing

In arithmetic and logical instructions the 16-bit M Field may contain the address of a direct or an indirect operand, an immediate operand, or a shift count. In immediate addressing the address field itself is the operand. Both immediate operands and shift counts are signed numbers in two's complement notation. bit 0 = sign bit ?

The addressable locations in which data may be found include any of the 32-bit memory locations, the Accumulator and the Save Register. Immediate addressing and half-word or byte positions in an addressable location are specified by bits in the operation field.

In control instructions the M Field may contain the address of an internal or external sense or function line and registers of external devices that are interfaced to the Addressable Input/Output Bus of the Exchange Module.

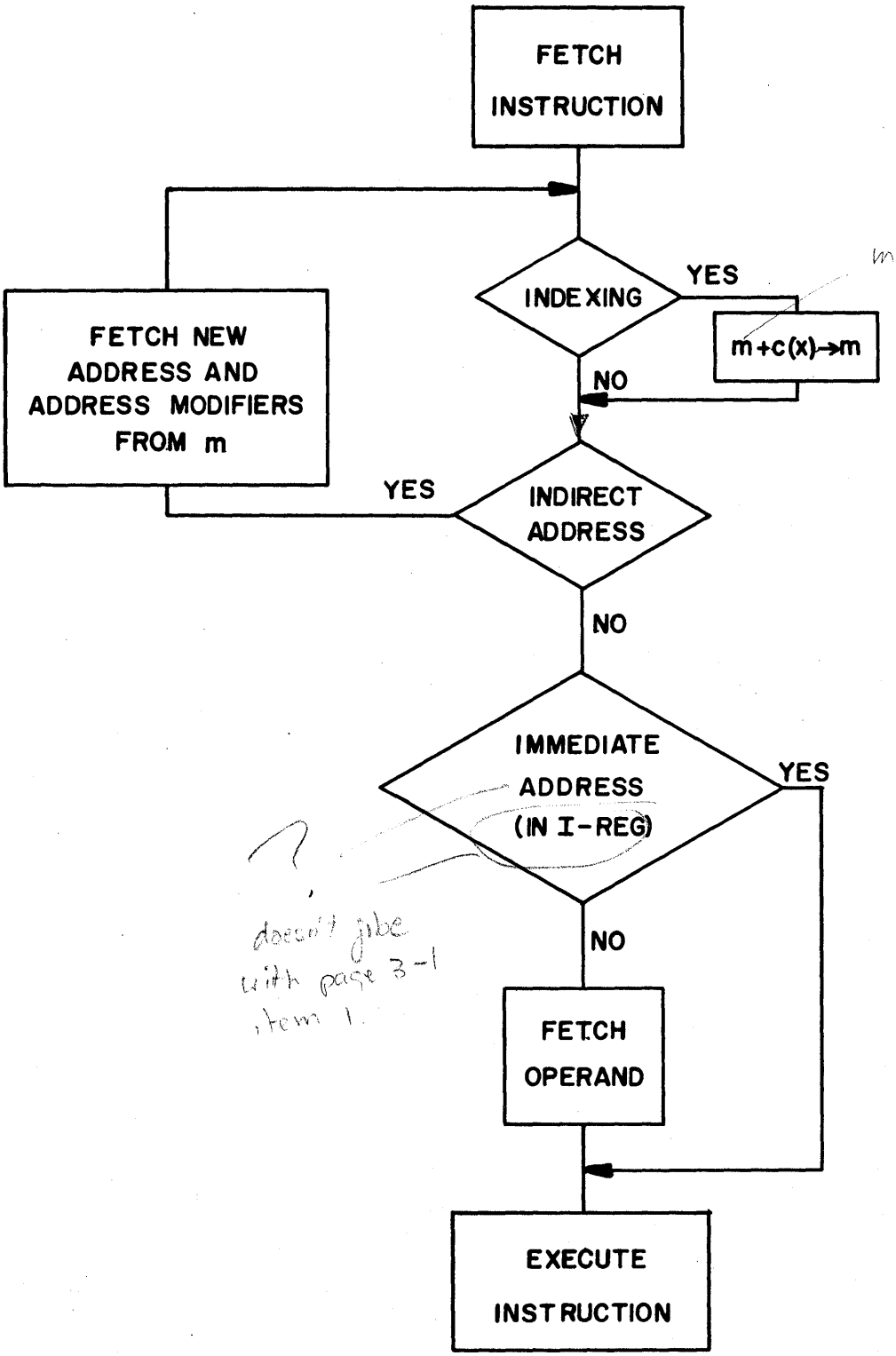
Address Modification

The * Bit in the instruction word determines whether or not indirect addressing is to be performed before an operand fetch. Multi-level indirect addressing is permissable, with optional address modification at every level.

The X Field specifies the index register to be used for address modification. This 3-bit field can address seven index registers. When indexed address modification is specified, the effective address is formed by adding (in two's complement notation) the contents of the selected index register to the base address contained in the M Field. A zero code in the X Field indicates that no modification is to be performed. That leaves 7 index registers accumulator + 6 hardware

FLOW CHART FOR ADDRESS MODIFICATION

WHERE INDEXING, INDIRECT ADDRESSING AND IMMEDIATE ADDRESS IS PERMISSABLE.



m field

Indexing proceeds indirect and DISPLACEMENT also

doesn't jibe with page 3-1 item 1.

If both indirect addressing and address modification are specified, the modified effective address is used for indirect addressing at each level.

Instruction Interpretation and Control

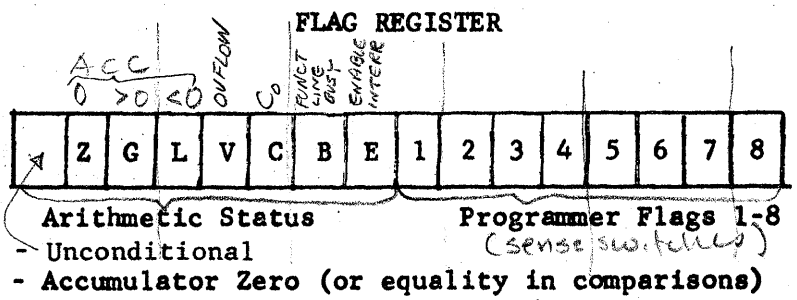
The 12-bit OP Field contains the operation code for the instruction to be performed. Instruction decoding logic interprets the OP code and sets the appropriate data paths and control circuitry required to execute the instruction.

The length of the OP Field in the 8400 provides an unusually extensive instruction repertoire of over 750 commands. The ~~order~~ OP code employed utilizes certain bits to designate Basic Operations (B) and other bits to designate Operation Modifiers (M). With this bit organization the programmer need remember only B + M codes to specify B x M instructions. Two special modifiers specify unnormalized floating-point operation and the saving of the contents of the accumulator in the Save Register prior to instruction execution.

In data addressing, bits in the OP Field augment the address field, specifying immediate addressing; or, in the case of operations with half-word data or smaller bytes, specifying the position of the data at the addressed location.

3.1.2 The Flag Register

The 8400 control system contains a 16-bit Flag Register and associated circuitry that provide continuous monitoring of sixteen machine conditions during the course of a program. The individual bits of the Flag Register indicate the status of these conditions at the end of each instruction. Logic circuitry controlled by the current instruction enables testing, setting and resetting of the Flag Bits and provides a capability for conditional modification of the normal sequential control of the program in progress on the basis of a program condition monitored by one of the Flag Bits.



() - Unconditional
Z - Accumulator Zero (or equality in comparisons)
G - Accumulator Greater than Zero (or Greater in comparisons)
L - Accumulator Less than Zero (or Less in comparisons)

V - Overflow of Accumulator (Cumulative)
C - Carry-out (Most Significant Bit)
B - Function Line or Data Channel Busy (not available)
E - Internal and External Interrupt System Enabled
1-8 - Console Programmer Flag 1-8 set

Note: complement testing of flag bits is also provided; Z bit can alternatively indicate a high (one) result for a Test Sense Line or Test EXEC Bit instruction

The Flag Register can be stored by the program, thereby enabling the internal status of the machine to be retained and later retrieved after program interruptions are completed.

3.1.3 Interrupt System

The fast and extensive interrupt system of the EAI 8400 provides the capability for altering the normal flow of sequential program control in response to the occurrence of any one of sixteen computer internal conditions and operating modes monitored by sixteen internal interrupt lines; and, up to 256 external conditions monitored by the addition of external interrupt lines to the Exchange Module. When an interrupt condition occurs, the interrupt system automatically notifies the main program and, if acknowledged, stops the program and transfers control to an interrupt subroutine. The subroutine services the interrupt and returns control to the main program.

Conditions Monitored

External interrupt conditions are assigned at the users option when external interrupt lines are installed. The internal interrupt conditions are fixed by the design of the machine. These internal conditions are listed below in order of assigned priority (from highest to lowest).

INTERNAL CONDITIONS

0. Power Failure - Detection of a drop in AC line voltage below operating level

1. Parity Failure- Parity check error in a memory bank or data channel

2. Reserved

3.	EXEC Mode -	Presence of set EXEC bits in a word read from memory
4.	Exponent Fault-	Presence of an exponent overflow or underflow in floating-point operations
5.	Memory Protect-Mode	Attempt to write in a protected location in memory
6.	Interval Timer-	Down counting of Interval Timer register to zero
7.	Console -	Operator depressing one of four console interrupt control buttons
8-15.	Exchange Module Data Channels	Data Channel 1-8 interrupt control lines (last four may be used as external lines at users option if data channels not installed)

typewriter tape etc.

System Structure and Priorities

The System interrupts are arranged in Groups of 16, each group having 16 individual interrupt levels, as follows:

Group 0	Internal interrupts 0-15
Group 1	External interrupts 0-15
Group 2	External interrupts 16-31
.	.
.	.
.	.
Group 16	External Interrupts 240-255

Each group of interrupts has priority over each succeeding group, and each level in a particular group has priority over lower levels. At the beginning of every instruction execution cycle, an interrupt scanning system sweeps through each group of interrupts, starting with Group 0.

Because of the scanning action of the interrupt system, interrupts of higher priority are always detected before interrupts of lower priority. Once an interrupt routine is given control, only interrupts of higher priority may interrupt the interrupt subroutine although this arrangement may be masked if desired. Detection of an active interrupt line takes place during the scan sweep, but the interruption of the normal program cycle by recognized interrupts takes place only after an instruction has been executed and before the next instruction is fetched. Exceptions are the EXEC and parity internal interrupts occurring on an instruction fetch which will interrupt before the instruction is executed.

System Enable-Disable

The entire interrupt system, both external and internal levels, may be enabled or disabled by setting or resetting the E-bit of the Flag Register. This bit may be appropriately set by the following instructions.

(1) LDF - load flag register

(2) JSE, JRE, JTE, LRE and their complements JSNE, JRNE, etc. (See Flag Test Instructions in Instruction Repertoire Section).

An exception is the Power Failure interrupt which can never be disabled.

Dynamic Priority Allocation

A very powerful feature of the 8400 interrupt system is the capability provided for Dynamic Priority Allocation of both external and internal interrupt lines, under program control. This is accomplished by means of Internal and External Mask Registers which provide masking control of the basic interrupt registers that continuously monitor the interrupt conditions assigned for detection.

The Internal Mask Register contains 16 masking bits corresponding to Interrupt Group 0 (internal conditions 0-15) and the External Mask Register contains 16 masking bits corresponding to Interrupt Group 1 (external conditions 0-15). The priority sequence of the interrupt lines in these two groups may be altered by resetting (to zero) the masking bits of lines to be inhibited; and setting (high) the bits of the lines to be recognized. This is accomplished by LDM and LDE instructions used to load the internal and external mask registers with the appropriate bit patterns. Additional groups of external interrupt can be inhibited under program control by means of the STATUS/FUNCTION LINE INSTRUCTION (SFL).

Through the use of mask control it is possible for the programmer to achieve the following results:

(1) Ensure an interrupt subroutine is not interrupted by one or more higher priority interrupts by resetting the corresponding mask bits for those higher order interrupts to be prevented.

(2) Restructure the normal priority sequence by loading and "safe-storing" mask configurations conforming to the order of priority desired.

(3) Establish a dynamic "priority" level for the main program whereby only selected priority interrupts will be able to interfere. This is accomplished by altering the masking bits with instructions in the main program sequence and can be changed as the program progresses.

Interrupt Memory Locations

A unique memory location is assigned for each of the internal and possible external interrupt lines. Upon detection of an interrupt condition the normal program cycle is broken and an unconditional jump is effected to the memory location assigned to the particular interrupt line that is signalling for the program's interruption. The instruction which the programmer has stored in the interrupt's assigned memory location now determines the computers response. This instruction, which cannot be interrupted, controls the action taken by the interrupt logic in executing the interrupt subroutine. Three choices of instructions which may be used for this purpose are:

(1) a "Link" Instruction: This is the normal interrupt subroutine linkage. The address of the next unexecuted instruction of the interrupted program (location counter contents) are stored in the memory address specified by the link instruction and the subroutine starts in the next memory cell. This avoids problems encountered in computers permitting only one such preassigned cell in handling interrupts of other interrupt routines.

(2) an "Execute" Instruction: The instruction contained in the address specified by the Execute command will be singly executed without affecting the Location Counter (unless that instruction is a Jump or Link). This choice is used for single or chained "Execute" interrupt subroutines.

(3) any Other Instruction: The instruction contained in the interrupt memory location is executed and control returns to the next instruction using the address in the Location Counter.

The variety of actions available through the choice of instructions that can be stored in an interrupt memory location adds another dimension of flexibility to that provided by the masking systems.

3.1.4 Status and Function Line Control

The capabilities provided by the Flag Register and Interrupt Systems, for modifying program control on the basis of internal and external conditions, are augmented in the 8400 by Status and Function lines which can be tested, set and reset under program control. Four banks of status and function lines are available; the first two of which are reserved for internal control purposes and the remainder for external systems control.

By using a set of Status and Function line commands (see Instruction repertoire section), internal conditions can be tested, and logic and data flow circuitry can be manipulated under program control. Control commands includes the selection of the appropriate bank and specific line or lines to be employed, using immediate addressing. When testing Status Lines, the Z bit of the Flag Register is made to correspond to the state of the line (s). The Busy Bit (B) of the Flag Register is set if a function line cannot be set, as a result of conflicting requirements.

The Status and Function Line Instructions for Bank 0 pertain to the Processor and Memory, while the instructions for Bank 1 pertain to the Exchange Module. The functions performed are listed below:

BANK 0

1. Testing parity error flip-flops in the various memory banks.
2. Testing Console interrupt flip-flops.
3. Turning on/off the Real-Time Clock.
4. Establishing EXEC interrupt conditions.

BANK 1

1. Clearing a data channel.
2. Initializing a data channel.
3. Testing parity error flip-flops in the various data channels.
4. Enabling/disabling interrupt signal gates.
5. Device function control.

3.1.5 EXEC Bit Control System

The EXEC bit Control System is an exclusive feature of the EAI 8400, which provides the programmer with many powerful programming techniques. The system operates in conjunction with the EXEC bits associated with each halfword in the 8400 memory. These word marking bits may be set, reset or tested by a group of EXEC bit control instructions. The result of a test of any half-word will set the Z bit in the Flag

Register if the EXEC bit is high. Programmed decisions using the Flag Test Instruction Set can therefore be made, based on the state of any word's EXEC bit.

EXEC Mode Interrupt

The EXEC Bit Instructions enable the use of the EXEC bits under the direction of the stored program. In addition, the EXEC bits are monitored by the 8400's Internal Interrupt System. This allows automatic recognition of the EXEC bits for a variety of purposes.

The EXEC Mode Interrupt occurs whenever a word is read from memory and the EXEC bits for the word are set. The EXEC Mode Interrupt can be recognized at three points during the instruction cycle as follows:

1. After instruction fetch and before address modification; this would result from reading an instruction with one or both of its EXEC bits set.

2. After address modification and before operand fetch; this would result from indirect addressing.

3. After execution and before the next instruction fetch; this would result from reading an operand with the EXEC bits set.

The interrupt subroutine determines which case occurred, and acts accordingly.

Applications

The 8400 Programming Systems use the EXEC Bit Control System for a variety of purposes, the most significant of which is dynamic relocation of programs stored in memory. Both the Assembler and FORTRAN IV Compiler provide convenient means for setting the EXEC bits in individual half-words and word blocks, as appropriate. Thus other uses for the EXEC System are limited only by the programmer's imagination. Suggested possibilities are:

1. Special simulations of other computers by trapping instructions marked with EXEC bits and executing them by interrupt software subroutines.

2. Implementation of special programming languages, such as list processors, compilers, interpreters, and generalized translators.

3. On-line breakpoint debugging for monitoring the progress of programs during execution.

4. Data Tagging for a special processing during input-output operations or table updating.

5. Implementation of push-down stack techniques using EXEC control.

3.1.6 Interval Timer Register (down counter with presetting)

As an optional control feature, the 8400 has available a 16-bit interval timer register which decrements the register once every millisecond.. When the contents of the timer register, T, becomes zero, a Real-Time Clock Interrupt is generated. The clock does not stop counting when it reaches zero, but "returns" to its maximum value, and continues to decrement.

X = 65.535 seconds

The Interval Timer has a maximum range of 65,535 milliseconds per count down and can be used for calculating elapsed time for periodic program interruption. Interruption can be programmed to occur every X seconds by loading the timer register with the binary equivalent of X in milliseconds. The timer counts down until it reaches zero and then generates an interrupt signal, which will initiate a subroutine to perform the desired services, and resets the timer for another X seconds. This feature permits flexible system integration, such as:

1. Synchronizing a program with a real-time device.

2. Outputting data periodically to certain peripheral devices.

3. Time-sharing multiple programs, or multiple consoles.

4. Periodically testing Sense Lines as an alternate to automatic interrupts.

3.1.7 Rapid Access File

Another optional feature of the 8400 is the Rapid Access File, containing 16 high-speed registers. The individual locations in this file are specified in the same fashion as memory locations and can be used for the storage of instructions and operands. In the case of instruction storage, short high speed loops can be preloaded into the file and then operated upon from these high speed storage locations. This provides an increase in throughput for such functions as table searching with a wide variety of test criteria. Scratch pad memory programming techniques are another application of the Rapid Access File.

3.2 ARITHMETIC

The Floating-Point Processor contains the logic and circuitry for performing the arithmetic and logical operations necessary for executing the stored program instructions. Some of the processor's important capabilities are:

1. High-speed processing - obtained by augmenting fast arithmetic circuitry with logic capabilities for powerful single instructions that enable a reduction in the total number of instructions necessary to perform a given function.

2. Floating-point operations - designed to be the normal arithmetic mode of operation for high speed, real time applications.

3. A complete set of logical operations - for fast and efficient programming language translation, input-output data handling and non-arithmetic problem requirements.

4. Programming language features - that provide ease of programming and reduce processing time as well as off-line preparation time.

5. A "Universal Accumulator" - that eliminates programmer concern with inter-register transfer hardware considerations.

6. High speed temporary storage - that provides a simple effective means of holding intermediate computational results for subsequent reuse without additional memory referencing.

The Universal Accumulator and Save Register are particularly illustrative of the special programming features provided in the EAI 8400. The Accumulator is universal and directly addressable. It provides very high speed processing of both floating and fixed-point data in a variety of word formats. The Universal Accumulator concept saves the programmer the burden of transferring the result of a previous operation to the proper arithmetic register, of ensuring that the correct register is loaded or unloaded in transferring data to and from storage. All these functions are performed automatically by high-speed parallel logic. Not only are programmed instructions to accomplish these transfers unnecessary, but also one of the most frequent sources of programming errors is eliminated. The addressable nature of the Accumulator enables very high speed squaring and doubling. In addition, operations on data in the Accumulator can use the general set of instructions relating to memory. For

example, the "Store after Rounding" instruction can be used to round quantities in the Accumulator by addressing location zero (the Accumulator's assigned address).

The SAVE Register provided in the Floating-Point Processor is a flexible high-speed storage register with configuration identical to that of the Universal Accumulator. It is addressable as location one. The SAVE Register allows the programmer to save the contents of the Accumulator prior to the execution of an arithmetic instruction. When this information is again required, it can be restored to the Accumulator by directly addressing the SAVE Register, an operation which requires 250 nanoseconds, considerably less than core memory access. The data is returned to the Accumulator automatically in the proper format for the arithmetic operation to be performed; all standard 8400 data formats may be accommodated.

3.2.1 Arithmetic Characteristics

Operating modes and data formats

The Floating-Point Processor can operate in a variety of modes including the following:

Floating-point, 32 and 56-bit
Fixed-point, 16 and 32-bit
Integer, 16-bit
Index, 16-bit
Boolean, 16, 8, 4, 2 or 1-bit

The data formats for these operating modes are diagrammed in figure 3.2-1, shown relative to the memory data word format. Supplemental numerical information is provided in the table below. All binary formats are in two's complement notation with sign bits high (one) for negative quantities.

Same as DDS 30 convention

Single Prec. **Floating-Point** 32 bits

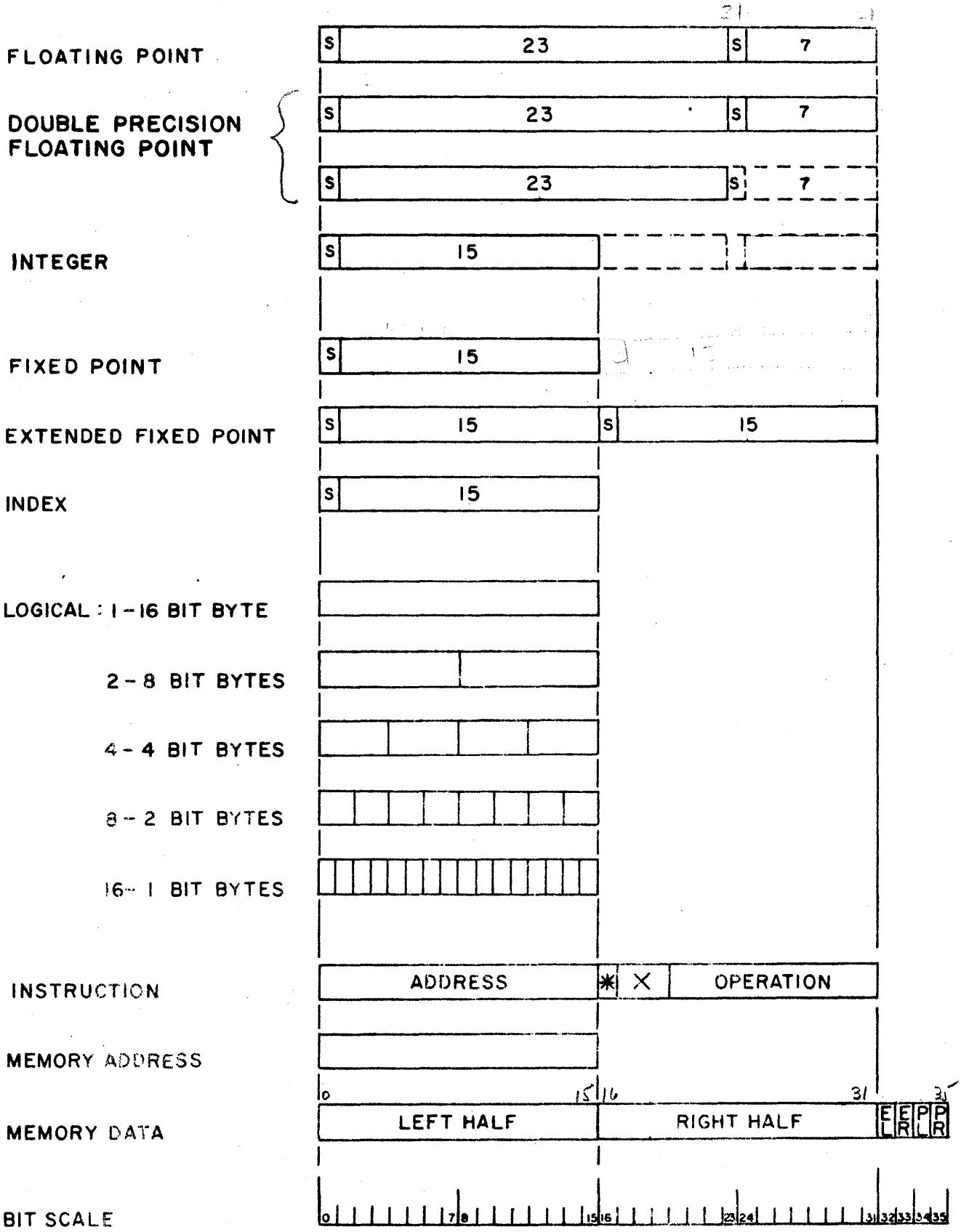
Binary format: [23 bit + sign] Mantissa — from 1 to 2^{-23} $\pm$
[7 bit + sign] Exponent — from 2^{+128} to $(2^{-23})2^{+128}$

normalized scaling

Memory Storage: one 32-bit location

Decimal Capacity: $8.388^{9} \times 10^{6}$ max. Mantissa
1.280×10^{2} max. Exponent

overall range 10^{-50} to 1



EAI 8400 WORD FORMATS

FIGURE 3.2-1

Double Precision Floating Point 56 bits

Binary Format: 46 bit + 2 sign Mantissa
7 bit + sign Exponent

Memory Storage: two 32-bit locations

Decimal Capacity: 7.056×10^{13} max Mantissa
1.280×10^{2} max Exponent

Fixed Point

Binary Format: 15-bit + sign

Memory Storage: one half-word location

Decimal Capacity: 3.277×10^{4} max. magnitude

Extended Precision Fixed Point

Binary Format: 30-bit + 2 signs

Memory Storage: one 32-bit location

Decimal Capacity: 1.073×10^{9} max. magnitude

Byte Modes

Binary Formats: 16, 8, 4, 2, 1 bits

Memory Storage: 1, 1/2, 1/4, 1/8, 1/16 of one half-word location

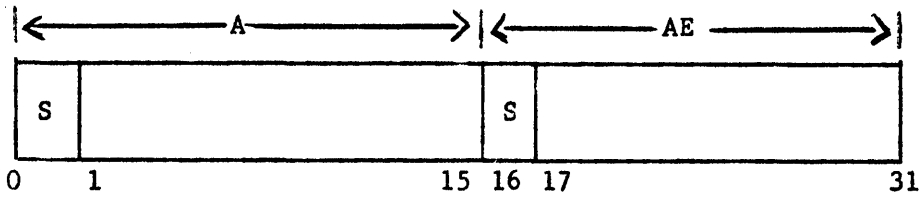
As can be seen, all data words are based on the 16/32 bit balanced-word framework. This structure allows optimal hardware organization for rapid execution of arithmetic-logical operations in the various word sizes and conversions from one format to another.

Accumulator Configuration

As in all single address machines, arithmetic-logical operations are performed on information in the Accumulator and operands from storage (core memory, Rapid Access File, SAVE Register, the Accumulator itself).

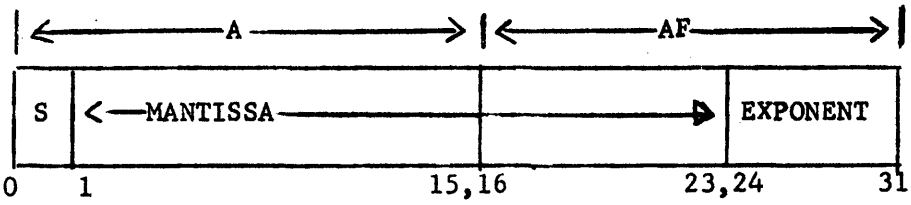
The 8400 Universal Accumulator automatically forms the proper configuration of its internal registers, to handle the four arithmetic data formats involved in the operations to be performed, as follows:

16 Bit Fixed-Point, Integer, Index, Logical Connectives and Shifts: A Register
~~A Register~~ 32 Bit Extended Precision Fixed Point, Extended Shifts:
A + AE Register

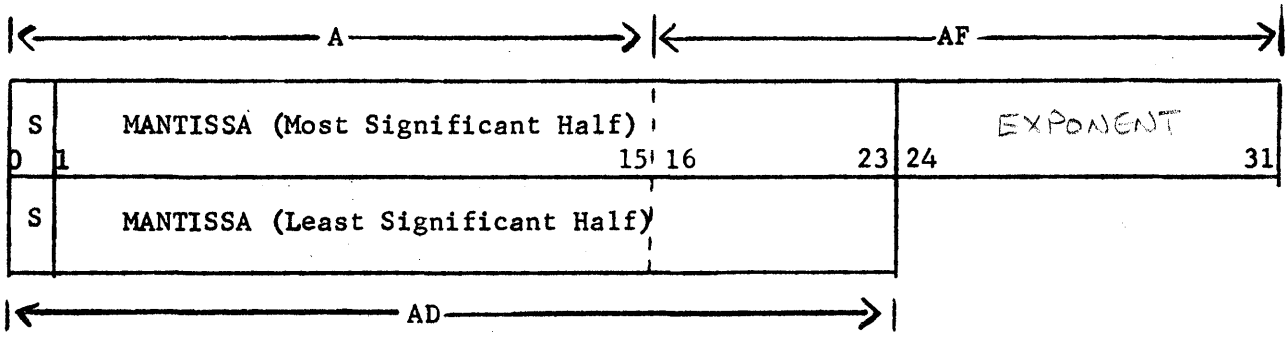


The AE Register is also used to hold 32 bit double length products and dividends of 16 bit Fixed-Point multiply/divide computation.

32 Bit Floating-Point A + AF Register



56 Bit Double Precision Floating-Point: A + AF + AD Register



The AD Register is used to hold double precision products and dividends of single precision floating-point multiply/divide computations.

The SAVE Register has a matching arrangement which is similarly adaptable. Other arithmetic registers are used to hold the operand and intermediate results.

Arithmetic Execution Speed

Figure 3.2-2 summarizes execution times for selected operation under several conditions. The significant feature is that Floating-Point execution times are equivalent to the times for comparative Fixed-Point operations. If the necessary scaling shifts are added to a typical fixed-point instruction mix, the 8400's floating-point solution will be faster. The differences in problem analysis and coding are, of course, appreciable.

Arithmetic Status Flags

In the course of running typical programs, a number of comparison operations and control steps are required. These include the ability to algebraically compare operands with respect to other operands, or with respect to zero. For example, the IF statement in algebraic compilers asks if an operand is "equal to", "greater than", or "less than" zero. Coupled with comparison operations are the actions desired as a function of the result. In the 8400 the Flag Register and its related Flag Test Transfer Instructions provide these capabilities.

The Flag Register's Zero (Z), greater than (G) and less then (L) bits continuously reflect the result in the Accumulator at the end of each instruction execution. In the COMPARE instructions included in the 8400 Arithmetic repertoire, their flag bits indicate that the data word in the Accumulator is equal to, greater than, or less than the referenced data word in memory. Flag settings are the only result of COMPARE operations; the Accumulator and Memory are unchanged. The Flag Test Transfer Instructions enable control transfer conditional upon arithmetic status through Jump, Link, Halt, and Execute operations.

Arithmetic Fault Detection

Three arithmetic fault indicators are provided in the 8400.

1. Exponent Fault Interrupt - occurs whenever the exponent of a floating-point number has become so large (positively or negatively) that the capacity of the exponent (Bits 24-31 of A + AF) is exceeded and Exponent Overflow or Underflow takes place. Because of the large range of exponents allowed in floating-point, this type of fault is relatively rare.

2. Carry-Out Flag (C Bit) In Flag Register - indicates that a carry of the most significant "magnitude" bit has occurred. Not truly a fault, carry-out is useful for initiating multiple step precisions in fixed-point operations.

3. Overflow Flag (V Bit) In Flag Register - signals that the result of an operation exceeded the capacity of the Accumulator in a positive or negative sense. Illegal multiplication, division, and "integerization" also set the V flag, which is cumulative and can be reset only by programmed Flag Test instructions. Due to the nature of two's complement notation, such a fault is sensed by the overflow logic whenever a carry of the most significant "magnitude" bit occurs with NO carry-out of the sign bit or a carry-out of the sign bit with NO carry from the first "magnitude" bit.

ARITHMETIC SPEED CHART
(times in microseconds)

Type of Operation	Memory Overlapped Minimum	Memory Overlapped Maximum	No Memory Overlap Minimum	No Memory Overlap Maximum	Comment Notes
Load/Store	2.25	2.75	4.00	4.00	32 bit
Floating Add	3.50	4.00	4.00	4.00	32 bit
FloatingCompare	4.00	4.50	4.50	4.50	32 bit
Floating Mult.	6.25	7.50	6.75	7.50	32 bit
Floating Div.	9.50	10.00	10.00	10.00	32 bit
Double Float Add	6.00	6.50	6.00	6.50	56 bit
Integer Add	3.50	4.00	4.00	4.00	16 bit
Fixed Add	3.25	3.75	4.00	4.00	16 bit
Fixed Multiply	5.25	6.25	5.75	6.25	16 bit
Fixed Divide	7.50	8.00	8.00	8.00	16 bit
Extended Fix Add	4.00	4.50	4.50	4.50	32 bit
Test/Branch	1.75	2.25	2.00	2.25	-
Logical	3.25	3.75	4.00	4.00	-
Shift	2.00	2.50	2.00	2.50	Note 5
No Ref to MM.	na	na	na	na	Note 6

(1) Use of Save Register or addressable accumulator will decrease times shown for which all operand fetches are from core memory.

(2) Minimum times assume no address modification and minimum operation execution time where applicable.

(3) Maximum times assume indexing and maximum operation times where applicable.

(4) For floating point operations add .25 u-sec for each pre-shift and normalize post shift.

(5) For shifts add .25 u-sec for each place; however, with 8400 fast floating point fixed point scaling is nearly eliminated.

(6) Inter-register transfers for arithmetic operations are not required with 8400 Universal Accumulator.

Figure 3.2.2

All arithmetic operations must produce a result which lies in the $1 > R \geq -1$ range or a fault is produced. The 8400 handles arithmetic faults as shown in the following table.

(where R is the characteristic only, of course!

Machine Operation(s) Causing Fault	Action Taken	Final State of Overflow Flag
All Fixed Point, Index Arithmetic Instructions	Operation Completed	SET
Floating Point and Integer Clear & Subtract Add Subtract Multiply	Operation is Completed; Result is corrected.	RESET
Floating Point and Integer Clear & Divide Divide	Operation is Completed; 1. Answer meaningless if Divisor << Dividend, yielding $R \geq 2$. 2. Result is corrected if $2 > R \geq +1$.	SET RESET
Floating Point Store Rounded	Operation is completed; Result is corrected.	RESET
Integer Store and Store Rounded	Operation is completed.	SET

3.2.2 Arithmetic Operations

Processing speed in the 8400 is achieved by augmenting fast arithmetic execution times with an extensive instruction repertoire, containing singly powerful instructions that result in shorter programs. Over 80 arithmetic instructions are provided.

The arithmetic instructions are readily learned and easily remembered through the use of a coding scheme which employs 10 basic operations and 6 class modifiers so that only 10 + 6 = 16 mnemonic symbols are required for 10 x 6 = 60 instruction mnemonics. The coding scheme is indicated by the table on the following page which shows all of the 8400 arithmetic operations. Operations suffixed with a "U" indicate unnormalized floating-point. There are 24 of these operations. Characteristics and utilization of the six classes of operations are discussed below.

Floating Point

Standard precision floating-point computations are the workhorse of the 8400. The word length and accumulator configuration are fundamentally designed to allow single memory access parallel floating-point capability. The precision and range provided by the standard floating-point class of operations is adequate for most problem variables.

In standard precision Floating-Point operations, a full 32-bit memory word is transferred in parallel to the Accumulator where the computations are performed by the A + AF circuitry. For loading double length dividends and storing double precision products, the DCA or DCS and DST commands may be used in conjunction with two full-word contiguous memory cells. The AD register is also used in these operations.

Both normalized and unnormalized floating-point operations are provided for all instructions except Store and Compare.

Automatic scaling, implemented in floating-point operations by exponent arithmetic and normalization, is the prime advantage of normalized floating-point operations. It relieves the programmer from the time-consuming, and often complicated task of scaling his problem to the limited range of a computer. In each floating-point instruction, several operations are combined -- exponent equalization, the desired arithmetic operation, corrective measures for overflow, and post normalization. This results in the saving of processing time and storage.

While the majority of scientific applications use Normalized Floating-Point, the unnormalized capability has important applications as well. For example, this mode is used in the programming of multiple precision arithmetic operations. In such cases, the partial results are left unnormalized, and summarized to produce the multiple precision results. These final results may be normalized or still kept without normalization as the problem demands.

Pre Modifiers: 1) F = 32 bit floating
2) E = 32 bit fixed (extended)
3) D = 56 bit Double floating
4) I = 16 bit Integer
5) X = 16 bit Index
6) ?

EAI 8400 INSTRUCTION LIST

ARITHMETIC OPERATIONS

Function	Mnemonic	Function	Mnemonic
32 Bit Floating Point:		**16 Bit Fixed Point:**	
Subtract	FSB	Subtract	SB
Clear Subtract	FCS	Clear Subtract	CS
Clear Add	FCA	Clear Add	CA
Add	FAD	Add	AD
Compare	FCP	Compare	CP
Multiply	FMP	Multiply	MP
Store	FST	Store	ST
Store Rounded	FSR	Store Rounded	SR
Divide	FDV	Divide	DV
Clear Divide	FCD	Clear Divide	CD
56 Bit Double Floating Point:		**32 Bit Extended Fixed Point:**	
Subtract	DSB	Subtract	ESB
Clear Subtract	DCS	Clear Subtract	ECS
Clear Add	DCA	Clear Add	ECA
Add	DAD	Add	EAD
Compare #	DCP	Compare #	ECP
Multiply #	DMP	Multiply #	EMP
Store	DST	Store	EST
Store Rounded #	DSR	Store Rounded #	ESR
Divide #	DDV	Divide #	EDV
Clear Divide #	DCD	Clear Divide #	ECD
16 Bit Integer:		**16 Bit Index:**	
Subtract	ISB	Subtract	XSB
Clear Subtract	ICS	Clear Subtract	XCS
Clear Add	ICA	Clear Add	XCA
Add	IAD	Add	XAD
Compare	ICP	Compare	XCP
Multiply	IMP	Multiply #	XMP
Store	IST	Store	XST
Store Rounded	ISR	Store Rounded #	XSR
Divide	IDV	Divide #	XDV
Clear Divide	ICD	Clear Divide #	XCD

10 instructions

Subroutine

SHIFTING, ROTATION AND NORMALIZATION OPERATIONS

Accumulator:		Extended Accumulator:	
Arithmetic Shift	ASH	Arithmetic Shift	EASH
Logical Rotate	ROT	Logical Rotate	EROT
Normalize	NRM	Normalize	ENRM

RIGHT LEFT?

Double Precision Floating Point

The double precision floating-point instructions operate on operands occupying two memory words, i.e., 64 bits. The word of lower address contains the most significant 23 bits of the fraction with its sign and the 7 bit exponent with its sign bit. The next higher memory address stores the least significant 23 bits of the fraction plus sign, and a 7-bit plus sign exponent differing from the exponent of the most significant half by 23. This two-word operand, when transferred into the Accumulator by a double precision floating-point instruction, will have a 46 bit fraction with duplicate signs and one exponent. The data word from the first memory word is loaded in the A + AF portion of the Accumulator, and the signed fraction from the higher memory location is transferred into AD. The exponent of the least significant part of the operand is ignored. During memory storage (DST) operations, the second exponent is generated automatically and inserted into the low end of the second memory cell, enabling the programmer to perform standard or double precision floating-point operations on the same operand word with no extra formatting necessary. It is also useful when multiple precision processing is required.

With the exception of word length and word format, these instructions are defined identically with their equivalent in the standard precision floating-point class, including the choice of normalized and unnormalized results.

Fixed Point

The 8400 standard precision fixed-point class of computations employ a 16-bit half-word which is transferred in parallel from the Right or Left Half of a memory cell to the A register in the Accumulator. For loading double length (Extended Precision) dividends and storing Extended Precision products, the ECA and EST commands may be used, addressing a full-word memory location. The AE circuitry holds the least significant half of these numbers.

Because of the speed and efficiency of the 8400's floating-point circuitry, fixed-point operations take on a relatively minor significance. Standard precision fixed-point is used mainly for address arithmetic in conjunction with the Index class of operations, for function-generation, and for manipulation of constants and digitized analog or low-accuracy digital data before and after conversion to floating-point.

The storage efficiency of the 8400 for these 16-bit data has been described in the Memory Section. The immediate operand option is especially suitable for 16-bit fixed-point data since it matches exactly the standard address field format.

Extended Precision Fixed-Point

Full 32-bit memory words are transferred to the Arithmetic Section where the computations are performed by the A + AF circuitry. This class of operations serves the primary purpose of allowing computations of double length operands generated by standard precision fixed-point multiply computations.

Five of the ten Extended Precision instructions are performed by software subroutines.

Integer Arithmetic

As part of the 8400's floating-point capability, a class of Integer (mixed-mode) Arithmetic operations is provided. This unique set of commands bridges the gap between the floating and fixed domains.

Integer operations involve two data types:

(a) 16 Bit Fixed-point notations carrying an implied binary exponent of 15.

(b) Standard Precision Floating-Point notation with a 23 bit + sign fraction and 7 bit + sign exponent.

The integer set of arithmetic operations permits arithmetic operations on operands in these two notations and facilitates conversion from one notation to the other. When stored in memory, the Integer is a 16 bit half-word number. When operated on in the Accumulator, the Integer is in standard precision Floating-Point format, and is handled by the A + AF register. Conversion between these two notations is accomplished by high speed logic, in combination with other arithmetic computations, and at no increase in execution times.

Memory to Accumulator transfers of operands include automatic floating conversion by appending 0's in the eight least significant places of the mantissa and setting the exponent to $+15_{10}$ ($+ 017_8$). This is followed by performance of the indicated arithmetic computation in floating-point.

Store operations (IST) start with an automatic "integerization" operation. The floating-point contents of the A + AF portion of the Accumulator are shifted until the exponent becomes $+15_{10}$ ($+017_8$); then the most significant 16 bits of the mantissa are transferred to the designated memory half-word location. The contents of the Accumulator (A +AF) are returned to their floating-point format prior to "integerization", and remain unchanged.

The floating-point computations performed in the Integer Class are identical to those of the standard precision floating-point group. As in the latter class, the option of normalized or unnormalized results can be selected except for store (IST or ISR) operations.

Integer operations may be used by the programmer for:

(a) Integer Arithmetic computations which may be done in floating-point and converted back to integers upon completion. The advantages of this technique are the vast gain in allowable range of variables and the automatic scaling which normalized

floating-point embodies plus the double efficiency storage characteristics of the 16-bit integers.

(b) Mixed-Mode arithmetic for solving scientific problems having intermingled variables (in floating-point) and integer constants. One particular advantage is the hardware manipulation of FORTRAN Mixed-Mode statements with its attendant speed advantages.

Index Arithmetic

The Index class of arithmetic makes use of the 8400's seven index registers. Index operations start with an automatic parallel transfer to the accumulator of the 16-bit contents of the index register addressed by the X field in the instruction. This number is then combined with the contents of the addressed memory location (immediate operands may be used effectively here to reduce execution time and memory usage). Any of the basic arithmetic operations are performed in accordance with the specific instruction executed. The result is then automatically transferred back to the index register. These three steps are performed by the hardware in response to a single Index Arithmetic instruction. The only exceptions are the XCA and XCS operations which load index registers from memory, and the XST and XSR which store the contents of index registers.

3.2.3 Logical Operations

As a companion to the 8400's comprehensive arithmetic operations, a complete set of Boolean Connectives is provided. All sixteen boolean connectives are provided. The Shift-Rotate-Normalize set of shift operations are provided for manipulation of fixed point arithmetic or logical data.

Boolean Connective

Logical operations may be performed on 16, 8, 4, 2 and 1 bit bytes between half-word memory locations and the Accumulator (A Register). The operations provided are summarized on the table on the following page.

The result of a logical operation may be placed back in the memory byte location, leaving the Accumulator unchanged, or may be put in the Accumulator, leaving memory unchanged. In any case, only the specific byte selected for the logical operation is altered.

The choices of connective, byte size and position, and memory or Accumulator result, are made by one instruction. Immediate operands may be used with the 16-bit byte size. All Boolean operations except "Byte Equality Test", defined below, set the Z bit of the Flag Register if the result is all zeros.

EAI 8400 INSTRUCTION LIST

LOGICAL BYTE OPERATIONS [1,2]

Function	Mnemonic	Function	Mnemonic
Boolean Connectives - Results to Accumulator:		**Boolean Connectives - Results to Memory:**	
Set (all ones in A)	SAn	Set (all ones in M)	SMn
Reset (all zeros in A)	RAn	Reset (all zeros in M)	RMn
Memory High (load M)	MHAn	Accumulator High (store A)	AHMn
Accumulator Low (complement A)	ALAn	Accumulator Low (complement A)	ALMn
Memory Low (complement M)	MLAn	Memory Low (complement M)	MLMn
Both High (AND)	BHAn	Both High (AND)	BHMn
Either High (OR)	EHAn	Either High (OR)	EHMn
Either Low (NAND)	ELAn	Either Low (NAND)	ELMn
Both Low (NOR)	BLAn	Both Low (NOR)	BLMn
Both Different (EXCL OR)	BDAn	Both Different (EXCL OR)	BDMn
Both Same (EQUIV)	BSAn	Both Same (EQUIV)	BSMn
Complement Both High (AND $\bar{A}$)	CBHAn	Complement Both High (AND $\bar{A}$)	CBHMn
Complement Either High (OR $\bar{A}$)	CEHAn	Complement Either High (OR $\bar{A}$)	CEHMn
Complement Either Low (NAND $\bar{A}$)	CELAn	Complement Either Low (NAND $\bar{A}$)	CELMn
Complement Both Low (NOR $\bar{A}$)	CBLAn	Complement Both Low (NOR $\bar{A}$)	CBLMn
Byte Equality Test (set Z flag if bytes identical	BEQTn	Memory High (set Z flag if byte in M is zero)	MHMn

Note 1. n= 16,8,4,2, 1 bit byte size

Note 2. ELAn for example requires that if either the A bits or M bits are low, the result is put in the accumulator (the A bits are set) and the byte in memory is left unchanged.

Among the logical operations provided are the following (See Instruction List for complete set):

1. Set Memory or Accumulator bits (all one's in result)
   Any 16-bit location or portion thereof can be set.

2. Reset Memory or Accumulator (all zero's in result)
   Any 16-bit location or portion thereof can be zeroed.

3. Complement memory or Accumulator bits.
   Any memory location may be complemented without affecting the Accumulator.

4. Byte Equality Test - Compare Memory and Accumulator and set Z-bit in Flag Register if Equal. Memory and Accumulator unchanged.

5. Byte Zero Test - Test memory location of portion thereof for state of bits and set Z bit in Flag Register if memory is zero. Accumulator and memory unchanged.

Shift-Rotate-Normalize

Arithmetic Shifts (Bits 1-15 of A Register) and Extended Arithmetic Shifts (Bits 1-15 and 17-31 of A + AE Register) are used in the Standard and Extended Precision Fixed-Point modes respectively to shift data bits without affecting the sign bit(s). The 2's complement address field (M Field) as modified by the contents of a specified index register (X), if any, determines the number of shift places. This quantity can be + for shifts to right or - for shifts to left. If a left shift causes Overflow, the V bit of the Flag Register will be set. When shifting left, 0's are entered to the right of the word.

Logical Rotate and Extended Logical Rotate (A + AE) Register) shift the entire word, rotating about on itself. The sign bit(s) are acted on as information bits. Bits shifted out of either end are brought around and entered in the just vacated places on the other end. Because no information can be lost, the Overflow Flag (V) is not involved in the process. The number of places rotated is determined exactly like the Arithmetic Shift operations.

Normalize and Extended Normalize are always left arithmetic shifts until Bit 1 of A differs from Bit 0. These instructions are used in arithmetic scaling operations to remove all leading zeros. The specified index register (X) tabulates the shift count required and may be saved as a scale factor.

4.0 INSTRUCTION REPERTOIRE

Fundamental to the speed and flexibility of the EAI 8400 is its powerful instruction repertoire. The repertoire has over 750 commands - in contrast with the 100^{+} commands of most machines in the same price range. The basic instructions provided may be classified as follows:

A. Arithmetic Instructions
   1. Floating Point
   2. Double Floating Point
   3. Fixed Point
   4. Extended Fixed Point
   5. Integer Arithmetic
   6. Index Arithmetic

B. Logical Instructions
   1. Rotate - Shift - Normalize
   2. Boolean Connectives

C. Transfer and Control Instructions
   1. Index Jump Transfers
   2. Flag Test Transfers
   3. EXEC Bit Control
   4. Special Register Transfers
   5. Status/Function Line Control

D. Exchange Instructions
   1. I/O Register Transfers
   2. Automatic Channel Control

Table 4-1, at the end of this section, is a guide to the programming of the 8400 and includes the full instruction list with mnemonic and binary coding, and equation descriptions for each of the operations that can be performed.

4.1 PROGRAMMING EASE AND POWER

The repertoire is unusually comprehensive yet readily understood and remembered. Instruction codes are classified into basic mnemonic group codes, with prefixes and suffixes to pinpoint the specific operation desired. The programmer need remember only a fraction of the mnemonic codes usually associated with a machine of this instruction power.

4.1.1 Arithmetic Instructions

Using the Arithmetic Class as an example, by remembering the basic add operation mnemonic, "AD", the programmer can describe all the add operations by prefixing:

FAD Single-Precision Floating-Point Add

DAD Double-Precision Floating-Point Add

IAD	Mixed Mode and Integer Add
AD	Fixed-Point Add
EAD	Extended Fixed-Point Add
XAD	Index Register Add

Then special modifiers are used to indicate indirect (*) and "immediate" (=) addressing, and saving of accumulator contents ($). For example, $FAD* specifies, "save accumulator contents, then perform a floating-point add using an indirect address". The modifiers applicable to Arithmetic Instructions are summarized below:

ADDRESS MODIFIERS

OPN*	M	Indirect Address
OPN	M,X	Index with Register X
OPN	M/	16 bit left half address
OPN	/M	16 bit right half address
OPN	=M	Immediate address

OPERATION MODIFIERS

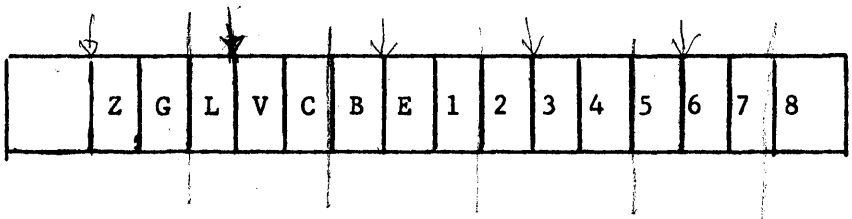
$OPN	M	Save A prior to execution
OPNU	M	Unnormalized Floating Point

4.1.2 Logical Instructions

The Logical connective mnemonic codes are descriptive of the logical manipulation performed. The mnemonic code describes, for a given bit position in both operands, the condition that causes a 1 bit to be placed in the result. For example, BHA is interpreted as follows: "For a given bit position, if the bits in memory and the Accumulator are Both High, put a 1 bit in the corresponding position in the Accumulator." This is the logical AND operation. The size of byte in a logical operation is appended to the mnemonic and the position of the byte is shown in the variable field. For example, BHA8 M,3,1 means "perform a logical AND between the 8-bit byte position 1 of the Accumulator and the corresponding bits in memory location determined by 'M', modified by the contents of index register 3." The results of the AND will appear in the accumulator. The instruction BHM8 M,3,1 designates the same AND operation, but causes the results to appear in the memory location. All of the Boolean Connective operations are included in the repertoire.

4.1.3 Flag Transfer Instructions

Test-conditional operations are based on the status of bits in a 16-bit flag register. The flag bits are set by the programmer or automatically as a result of internal conditions. For instance, the Zero, Greater than zero, and Less than zero flags are set or reset automatically after the execution of arithmetic instructions.*



( )	-	Unconditional	C	-	Arithmetic Carry
Z	-	Accumulator zero	B	-	Data Channel Busy
G	-	Greater than zero	E	-	Interrupt Mode Enabled
L	-	Less than zero	1-8	-	Programmer flags
V	-	Arithmetic Overflow			

Nine basic instructions are provided:

HJf	Halt, then Jump if flag f set
EXf	Execute instruction at specified location if flag f set
Lf	Link to subroutine if flag f set
LRf	Link to subroutine, Reset flag
Jf	Jump if flag f set
JRf	Jump, reset flag
JSf	Jump, set flag
JTf	Jump, trigger flag

The "f" following the basic mnemonic codes indicates any one of the sixteen flag conditions. For example, HJZ designates a Halt Jump if Zero operation. By prefixing the flag condition mnemonic with the letter N the complement state of the flag can be used. Thus, HJNZ designates a Halt Jump if Not Zero.

In the Link and Jump instructions the link and jump operations are conditional on the status of the flag tested; but the setting, resetting or triggering (complementing) of the flag is unconditional.

* for details see Table 4-1.

4.1.4 Index Jump Transfers

The Index Jump Test instructions increments or decrements a specified Index Register then performs a conditional jump depending on the sign of the count modifier and the sign of the resultant in the Index Register.

This capability allows fast list processing, table searching, and Function Generation by allowing manipulation in both directions - up and down a table.

4.1.5 Assembly and Machine Language Programming

Table 4-1 is a twenty page guide to assembly or machine language programming with the 8400 repertoire. The Instruction format, addressing modes, operation modifiers and notation conventions are summarized prior to a listing of all of the 8400 instructions with their respective binary and mnemonic codes and operational descriptions.

The Exchange instructions included in the table are discussed in more detail in Section 5, which describes the Exchange Module and its operation.

TABLE 4-1

SUMMARY OF INSTRUCTIONS WITH MNEMONIC AND BINARY CODING

This summary is intended for use as a programmer reference to the 8400 Instruction repertoire.

TABLE INDEX

Instruction Format

The 32-bit instructions for the 8400 are defined by an OPERATION and an ADDRESS. Using a mnemonic for the OPERATION, instructions are expressed symbolically as follows:

OPN M

where M specified an addressable location within the computer, such as a word in core memory, rapid access file, accumulator, or the save register.

The 8400 employs a basic set of operations, plus a set of modifiers that can alter the operations in various ways. Similarly, the address portion of an instruction can be modified in several ways, at the option of the programmer.

ADDRESS MODIFIERS

Modifier	Name	Format	Remarks
*	Indirect Address	OPN* M	The address for the given instruction is taken from the address portion of the 32-bit word at location M. Multiple indirect addressing is possible. *All* instructions may use an indirect address.
X	Address Modification	OPN M,X	The effective address is obtained by adding the contents of the specified index register, X, to the address, M. That is, M + C(X) → M. All instructions *except* the Index Register Class can have address modification. Indexing precedes indirect addressing at every level if both are specified.
/	Halfword Address	OPN M/ OPN /M	The operand for 16-bit operations comes from the left half of M by using M/ and the right half of M by using /M. The slash (/) has no effect on indexing or indirect addressing. A 16-bit operation written OPN M is interpreted by the assembler as OPN M/.
=	Immediate Address	OPN =M	The operand for this instruction is taken from the address field of the instruction itself. The immediate address may *not* be used with /. The immediate address is applicable to all 16-bit operations except Store and Store After Rounding.

REMARKS: The various legal combinations of the address modifiers are illustrated below:

OPN M/
OPN /M
OPN M/,X
OPN /M,X
OPN M

OPN* M/
OPN* /M
OPN* M/,X
OPN* /M,X
OPN =M
OPN =M,X
OPN* =M,X
OPN* = M

OPERATION MODIFIERS

Modifier	Name	Format	Remarks
$	Save	$OPN M	The contents of the Accumulator are saved prior to the execution of the instruction. The Save modifier may be used with *arithmetic and shift instructions*.
U	Unnormalized	OPNU M	Floating point operations are usually normalized automatically at the completion of the operation. The U modifier inhibits the automatic normalization. This modifier may be used with all floating point operations (but has no meaning in Compare, Store, and Store After Rounding).
N	Operate When --- Not Set		The N Modifier inserted into *flag* instructions causes the operation to occur when the specified flag is *not* set.

FLAG

The Processor contains a 16-bit flag register (F). Eight bits (flags) of this register are activated by the internal status of the computer, and the remaining flags can be controlled by the programmer. The 16 flags are the following:

FLAG	DEFINITION
( )	Unconditional (Always High)
Z	Accumulator Equal to Zero
G	Accumulator Greater than Zero
L	Accumulator Less than Zero
~~C~~ C	Carry-out is Generated
~~V~~ V	Overflow Occurred (Cumulative)
B	Busy Signal
E	Interrupt Enable
1 - 8	Program Flags 1 - 8

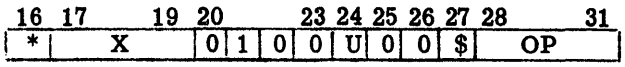
The arithmetic flags except V are reset and updated following the execution of every instruction, or may be set by the programmer. The Test-Conditional operations are based on the status of the Flag Register bits. The V bit is cumulative and must be reset by programmed testing.

NOTATION

R(i:j)	bits i through j of register R. (Register referenced is clear from the context of its use or is explicitly stated.)
m	effective memory address
⟶	replaces
A	the Universal Accumulator
A, AE, AF, AD	the registers that make up the Accumulator
$	Save register (which saves Accumulator registers A, AE, AF, AD)
RF	Rapid Access File memory cells
F	Flag register
L	Location counter
T	Real-Time clock register
M	Internal Interrupt Mask
E	External Interrupt Mask register
C	Console register, or count field used with index-jump instructions
M	a symbol used to indicate the numerical address of a memory word
I	Instruction Register
K	a symbol used to indicate a data channel number
B	a symbol used to indicate Bank number, Bus number, or Byte position
n	a symbol used to indicate byte size

FLOATING-POINT INSTRUCTIONS

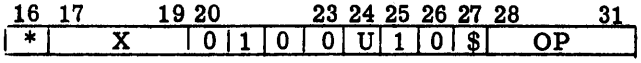
Format: OPN M,X



Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
FAD	0011	Floating Add	*, $, U	Z, G, L, C	A:AF + m → A:AF
FCA	0010	Floating Clear and Add	*, $, U	Z, G, L	m → A:AF
FSB	0000	Floating Subtract	*, $, U	Z, G, L, C	A:AF - m → A:AF
FCS	0001	Floating Clear and Subtract	*, $, U	Z, G, L	-m → A:AF
FMP	0101	Floating Multiply	*, $, U	Z, G, L	A:AF x m → A:AF:AD
FDV	1000	Floating Divide	*, $, U	Z, G, L, V	A:AF:AD ÷ m Quotient → A:AF Remainder → AD
FCD	1001	Floating Clear and Divide	*, $, U	Z, G, L, V	First clear → AD then Perform FDV
FCP	0100	Floating Compare	*, $,	Z, G, L	G set for A:AF > m Z set for A:AF = m L set for A:AF < m A:AF unchanged
FST	0110	Floating Store	*, $		A:AF → m
FSR	0111	Floating Store After Rounding	*, $	C	A:AF (rounded) → m A:AF unchanged

DOUBLE PRECISION FLOATING-POINT INSTRUCTIONS

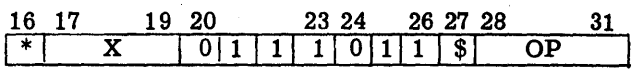
Format: OPN M, X



Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
DAD	0011	Double Floating Add	*, $, U	Z, G, L, C	A:AF:AD + m:m+1 → A:AF:AD
DCA	0010	Double Floating Clear and Add	*, $, U	Z, G, L	m:m+1 → A:AF:AD
DSB	0000	Double Floating Subtract	*, $, U	Z, G, L, C	A:AF:AD - m:m+1 → A:AF:AD
DCS	0001	Double Floating Clear and Subtract	*, $, U	Z, G, L	-m:m+1 → A:AF:AD
DMP	0101	Double Floating Multiply	*, $, U		Compat Subroutine Operation
DDV	1000	Double Floating Divide	*, $, U		Compat Subroutine Operation
DCD	1001	Double Floating Clear and Divide	*, $, U		Compat Subroutine Operation
DCP	0100	Double Floating Compare	*, $		Compat Subroutine Operation
DST	0110	Double Floating Store	*, $		A:AF:AD → m:m+1
DSR	0111	Double Floating Store and Rounding	*, $, U		Compat Subroutine Operation

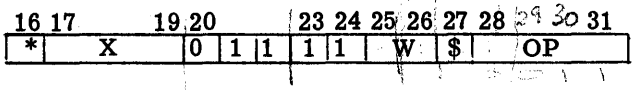
32 BIT EXTENDED FIXED POINT ARITHEMTIC

Format: OPN M,X



Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
EAD	0011	Extended Add	*, $	Z, G, L, V, C	A:AE + m → A:AE
ECA	0010	Extended Clear and Add	*, $	Z, G, L	m → A:AE
ESB	0000	Extended Subtract	*, $	Z, G, L, V, C	A:AE-m → A:AE
ECS	0001	Extended Clear and Subtract	*, $	Z, G, L, V	-m → A:AE
EMP	0101	Extended Multiply	*, $		Compat Subroutine Execution
EDV	1000	Extended Divide	*, $		Compat Subroutine Execution
ECD	1001	Extended Clear and Divide	*, $		Compat Subroutine Execution
ECP	0100	Extended Compare	*, $		Compat Subroutine Execution
EST	0110	Extended Store	*, $		A:AE → m
ESR	0111	Extended Store and Rounding	*, $		Compat Subroutine Execution

Format: OPN M/,X



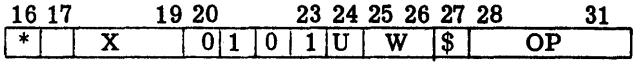
Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
AD	0011	Add	*,/,=,$	Z,G,L,V,C	A + m → A
CA	0010	Clear and Add	*,/,=,$	Z,G,L	m → A
SB	0000	Subtract	*,/,=,$	Z,G,L,V,C	A - m → A
CS	0001	Clear and Subtract	*,/,=,$	Z,G,L,V,C	- m → A
MP	0101	Multiply	*,/,=,$	Z,G,L,V	A x m → A:AE
DV	1000	Divide	*,/,=,$	Z,G,L,V	A:AE ÷ m Quotient → A Remainder → AE
CD	1001	Clear and Divide	*,/,=,$	Z,G,L,V	Clear AE Then Perform DV
CP	0100	Compare	*,/,=,$	Z,G,L	G set for A > m Z set for A = m L set for A < m A unchanged
ST	0110	Store	*,/,$		A → m
SR	0111	Store After Rounding	*,/,$	V,C	A (rounded) → m A A Unchanged

W = 00 is immediate operand
01 is left-half memory word
10 is right-half memory word
11 is illegal

NOTE: 16 bit operations assume a left-half memory word unless modified by /.

INTEGER ARITHMETIC INSTRUCTIONS

Format: OPN M ,X



Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
IAD	0011	Integer Add	*, /, U, =, $	Z, G, L, C	A:AF + mf → A:AF
ICA	0010	Integer Clear and Add	*, /, U, =, $	Z, G, L	mf → A:AF
ISB	0000	Integer Subtract	*, /, U, =, $	Z, G, L, C	A:AF - mf → A:AF
ICS	0001	Integer Clear and Subtract	*, /, U, =, $	Z, G, L	-mf → A:AF
IMP	0101	Integer Multiply	*, /, U, =, $	Z, G, L	A:AF x mf → A:AF:AD
IDV	1000	Integer Divide	*, /, U, =, $	Z, G, L, V	A:AF:AD ÷ mf Quotient → A:AF Remainder → AD
ICD	1001	Integer Clear and Divide	*, /, =, $	Z, G, L, V	Clear AD then Perform IDV
ICP	0100	Integer Compare	*, /, =, $	Z, G, L	G set for A:AF > mf Z set for A:AF = mf L set for A:AF < mf c(AAF) Unchanged
IST	0110	Integer Store	*, /, $	V	A:AF → mi A:AF Unchanged

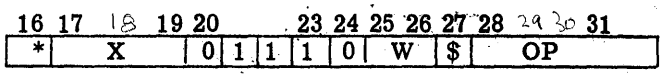
W = 00 is immediate operand
01 is left-half memory word
10 is right-half memory word
11 is illegal

mf = 16-bit memory operand "floated" as it enters the arithmetic section. It is converted to a floating-point format by attaching an exponent of +15.

mi = 16-bit memory operand "integerized" as it leaves the arithmetic section on the way to memory. It is converted from a floating-point format to integer by shifting A:AF until the exponent is +15.

INDEX REGISTER INSTRUCTIONS

Format: OPN M ,X



X = 1, 2, ...7

Remarks: X specifies which index register is modified. Instruction is non-indexable.

Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
XAD	0011	Index Add	*, /, =, $	Z, G, L, V, C	X + m → X
XCA	0010	Index Clear and Add	*, /, = $	Z, G, L	m → X
XSB	0000	Index Subtract	*, /, =, $	Z, G, L, V, C	X - m → X
XCS	0001	Index Clear and Subtract	*, /, =, $	Z, G, L, V	-m → X
XMP	0101	Index Multiply	*, /, =, $		Compat Subroutine Execution
XDV	1000	Index Divide	*, /, =, $		Compat Subroutine Execution
XCD	1001	Index Clear and Divide	*, /, =, $		Compat Subroutine Execution
XCP	0100	Index Compare	*, /, =, $		G set for X > m Z set for X = m L set for X < m X Unchanged
XST	0110	Index Store	*, /, $		X → m
XSR	0111	Index Store After Rounding	*, /, $		Compat Subroutine Execution

W = 00 is immediate operand
01 is left half memory word
10 is right half memory word
11 is illegal

BOOLEAN CONNECTIVE INSTRUCTIONS

Format: OPNn M ,X,B Options: *,/,=,n,B

Mnemonic Code: If mnemonic ends in "A", the result is put in the Accumulator. Memory and the unselected bits of the Accumulator are unchanged.

If mnemonic ends in "M", the result is put in Memory. The Accumulator and the unselected bits of the Memory are unchanged.

Remarks: n = 1, 2, 4, 8, or (blank) to specify byte size.
B = 0, 1, ..., 15 to specify byte position (See Table).
"=" = an immediate address; this option applicable only when instruction does not end in M, and a 16 bit byte is specified.

The mnemonic code describes, for each bit position of the operands, the condition under which bits will be set in the result. If the condition is not met, the bits are reset. For example, BHA is interpreted as follows: For a given bit position, if the corresponding bits in Memory and the Accumulator are Both High, set the corresponding bit in the Accumulator. If the condition is not met, reset the corresponding bit in the Accumulator.

Flags: Z All Boolean operations, except BEQT, set the Z flag when the result of the operation produces all zeros. BEQT operation sets the Z flag when accumulator and memory bytes are identical.

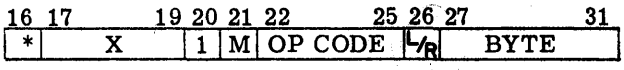
---

n-B Mapping for Byte Size and Byte Position

A Register	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M/ Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
/M Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
n-Byte Size																
n = 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n = 2	0		1		2		3		4		5		6		7	
n = 4	0				1				2				3			
n = 8	0								1							
n = 16	0															

B-Byte Position

BOOLEAN CONNECTIVE INSTRUCTIONS

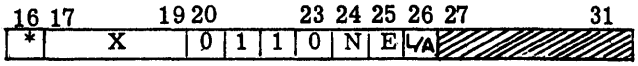


Mnemonic OPN Code	Binary OP Code	Operation
RA RM	0000	Reset {M bits / M bits}
BLA BLM	0001	Where both low, set {A bits / M bits}
CBHA CBHM	0010	Complement A bits, then if both high, set {A bits / M bits}
ALA ALM	0011	Where A bits low, set {A bits (Complement Accumulator) / M bits}
CBLA CBLM	0100	Complement A bits, then if both low, set {A bits / M bits}
MLA MLM	0101	Where M bits low, set {A bits / M bits (Complement Memory)}
BDA BDM	0110	Where both different, set {A bits / M bits}
ELA ELM	0111	Where either low, set {A bits / M bits}
BHA BHM	1000	Where both high set {A bits / M bits}
BSA BSM	1001	Where both same, set {A bits / M bits}
MHA MHM	1010	Where M bits high, set {A bits (Load Memory bits into Accumulator) / M bits (Zero Memory Test)}
CEHA CEHM	1011	Complement A bits, then if either high, set {A bits / M bits}
BEQT AHM	1100	Byte equality test (Z bit in Flag Register set if equal) Where A bits high, set M bits (Store Accumulator bits in Memory)
CELA CELM	1101	Complement A bits, then if either low, set {A bits / M bits}
EHA EHM	1110	Where either high, set {A bits / M bits}
SA SM	1111	Set {A bits / M bits}

M (I(21)) = 1 = Result to Memory
M (I(21)) = 0 = Result to Accumulator
L = I(26) = 1 = Left Half
R = I(26) = 0 = Right Half
B = I(27-31) = 0 = Immediate operand in "M" Field
BH equivalent to AND
BL equivalent to NOR
EH equivalent to OR
EL equivalent to NAND
BS equivalent to EQUIVALENCE = XOR
BD equivalent to EXCLUSIVE OR
C equivalent to COMPLEMENT

For A bits 1100 and M bits 1010, Binary Op Code shows resultant bit pattern.

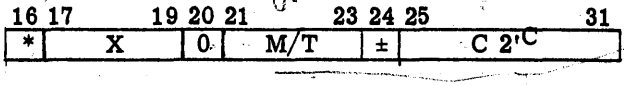
ROTATE SHIFT NORMALIZE INSTRUCTIONS



Mnemonic OPN Code	Title	Options	Flags	Operation
ROT±M, X	Logical Rotate	*		Rotate A by m (modulo 32) places + rotates to right - rotates to left
ASH±M, X	Arithmetic Shift	*	Z, G, L, V Left Only	Shift A (1:15) by m (modulo 32) places + shifts to right - shifts to left A(0) Unchanged
NRM, X	Normalize	*	Z, G, L	Normalize A by left shifting until A(0) ≠ A(1) The number of shifts → X A(0) Unchanged
EROT±M, X	Extended Logical Rotate	*		Rotate A:AE by m (modulo 32) places + rotates to right - rotates to left
EASH±M, X	Extended Arithmetic Shift	*	Z, G, L, V Left Only	Shift A(1:15):AE(1:15) m (modulo 32) places + shifts to right - shifts to left
ENRM, X	Extended Normalize	*		Normalize A:AE by left shifting until A(0) ≠ A(1) Number of shifts → X

* = Indirect Address
X = Index Register
E = Shift Double Operand I(25) = 1
L = Perform Logical Shift I(26) = 1
A = Perform Arithmetic Shift I(26) = 0
N = Shift until normalized
I (0) = 1 = >Shift left
I (0) = 0 = >Shift right

INDEX JUMP INSTRUCTIONS

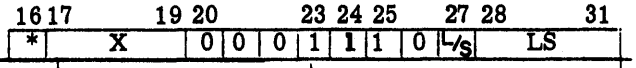


Mnemonic OPN Code	Binary OP Code	Title	Options	Flags	Operation
XJ M, X, ±C	010	Index Jump	*	Z, G, L	First X ± C → X then Jump to m
XJT M, X, ±C	011	Indexed Jump Test	*	Z, G, L	First X ± C → X then Jump to m if: +C specified and X < 0 -C specified and X ≥ 0

NOTE: C is in 2's complement if negative.

RAPID ACCESS FILE INSTRUCTIONS

Format: OPN M, X, N



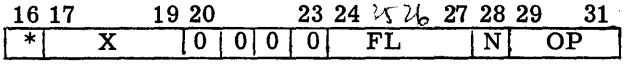
Mnemonic OPN Code	L/S Field Code	Title	Options Flags	Operation
LDRF	1	Load Rapid Access File	*, N	m → n
STRF	0	Store Rapid Access File	*, N	n → m

N = Location in Rapid Access File = 0, 1, 2, ..., 15
34 bit word transfer
32 data bits plus 2 exec bits
I(27) = 1 = Load
I(27) = 0 = Store

TEST-CONDITIONAL INSTRUCTIONS

Format: OPNf, M, X

Options: *, N, f



Example: HJZ M, X
HJNZ M, X

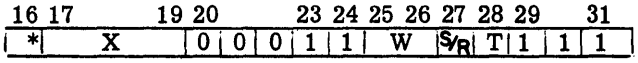
Mnemonic OPN Code	Binary OP Code	Title	Operation
HJ(N)f	000	Halt	Halt if flag f is (not) set, then jump to M on external signal or interrupt.
EX(N)f	001	Execute	Execute instruction at M if flag f is (not) set. Location counter unchanged.
L(N)f	010	Link	If flag f is (not) set, then store location counter at m/ and take next instruction from m + 1.
LR(N)f	011	Link and Reset	Same as L(N)f, but flag f is unconditionally reset.
JT(N)f	100	Jump and Trigger	Jump to M if flag f is (not) set and unconditionally trigger flag.
JS(N)f	101	Jump and Set	Jump to M if flag f is (not) set and unconditionally set flag.
JR(N)f	110	Jump and Reset	Jump to M if flag f is (not) set and unconditionally reset flag.
J(N)f	111	Jump	Jump to M if flag f is (not) set.

* = Indirect Address
N = (I(28)) = 1 = Operation to occur when selected flag (f) is not set.
N = (I(28)) = 0 = Operation to occur when selected flag (f) is set.
f = FLAG =( ), Z, G, L, V, C, B, E, 1 - 8.

EXEC BIT CONTROL INSTRUCTIONS

Format: OPN M ,X

Options: *, /, M

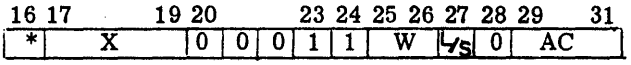


Mnemonic OPN Code	Binary OP Code	Title	Flags	Operation
SEX	S/R = 1	Set EXEC		EXEC bit at location m(M/ or /M) is set.
REX	S/R = 0	Reset EXEC		EXEC bit at location m(M/ or /M) is reset.
TEX	T = 1	Test EXEC	Z	EXEC bit at location m(M/ or /M) is tested. Z flag set (=1) when EXEC bit is high (=1). Z flag reset (=0) when EXEC bit is low (=0).

* = Indirect Address
X = Index Register
W = 00 - Illegal
01 = Left half word
10 = Right half word
11 = Illegal

LOAD/STORE PROCESSOR REGISTER INSTRUCTIONS

Format: OPN M ,X



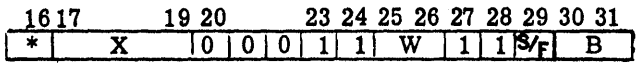
Mnemonic OPN Code	AC Field Code	Title	Options	Flags	Operation
LDAE	000	Load AE	*,/,=		m → AE
STAE		Store AE	*,/		AE → m
LDF	001	Load Flag Register	*,/,=		m → F
STF		Store Flag Register	*,/		F → m
LDL	010	Load Location Counter	*,/,=		m → L
STL		Store Location Counter	*,/		L → m
LDT	011	Load Timer	*,/,=		m → T
STT		Store Timer	*,/		T → m
LDM	100	Load Interrupt Mask	*,/,=		m → M
STM		Store Interrupt Mask	*,/		m → m
LDE	101	Load External Mask	*,/,=		m → E
STE		Store External Mask	*,/		E → m
LDC	110	Load Console Register	*,/,=		m → C
STC		Store Console Register	*,/		C → m

W = 00 = Immediate (=)
01 = Left Half (M/)
10 = Right Half (/M)
11 = Illegal
L/S = 1 = Load
0 = Store

STATUS/FUNCTION LINE INSTRUCTIONS

Format: OPN M/,X,B

Options: *,/,B,=



Remarks: B = bank = 0, 1, 2, 3
B = 0 relates to Floating-Point Processor
B = 1 relates to Exchange
B = 2, 3 relates to External devices

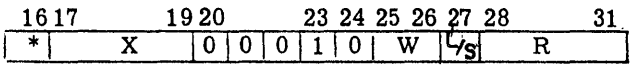
Mnemonic OPN Code	S/F Field Code	Title	Flag	Operation
TSL	0	Test Status Line	Z	Test status line, in bank B, specified by m. Z flag is set/reset if status line is set/reset.
SFL	1	Set Function Line	Z	Set function line, in bank B, specified by c(m). B flag is set if function line already set, or cannot be set because of conflict.

W = 00 = Immediate (=)
01 = Left Half
10 = Right Half
11 = Illegal

LOAD/STORE ADDRESSABLE I/O BUS INSTRUCTIONS

Format: OPN M/,X,R

Options: *,/,=



Remarks: R = bus = 0, 1, 2, ..., 15
17 bit transfers, 16 bit
plus EXEC bit

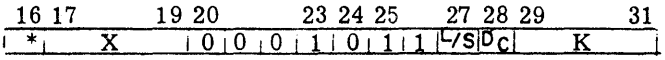
Mnemonic OPN Code	L/S Field Code	Title	Flags	Operation
LDOB	1	Load Output Bus	None	m → Bus R
STIB	0	Store Input Bus	None	Bus R → m

W = 00 = Immediate
01 = Left Half
10 = Right Half
11 = Illegal

DATA CHANNEL INSTRUCTIONS

Format: OPN M, X, K

Options: *



Remarks: R = data channel = 0, 1, 2, ..., 7

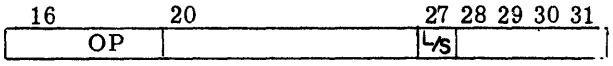
Mnemonic OPN Code	L/S D/C Field Codes	Title	Flags	Operation
LDCD	11	Load Channel Data Register	None	m → data register K format specified by SFL.
STCD	01	Store Channel Data Register	None	Data register K → m format specified by SFL.
LDCC	10	Load Channel Control Register	None	m → control register K 32 bit transfer.
STCC	00	Store Channel Control Register	None	Control register K → m 32 bit transfer.

* = Indirect Address
X = Index Register
L Load = I (27) = 1
S Store = I (27) = 0
D Data Register = I (28) = 1
C Control Register = I (28) = 0

AUTOMATIC CHANNEL CONTROL

Format: OPN M, C

Options: None



Remarks: C = count $C \leq C < 4096$

No flags affected.

These instructions control word transfers to or from contiguous memory locations starting at M.

Record length is controlled by count C which is decremented after each word transfer until C = 0, or by a signal included in the data.

Mnemonic OPN Code	Binary OP Code	Title and Operations
TCD	0100	Transmit until Count then Disconnect
SCD	1100	Skip until Count then Disconnect
TCI	0101	Transmit until Count then Interrupt
SCI	1101	Skip until Count then Interrupt
TSD	0010	Transmit until Signal then Disconnect
SSD	1010	Skip until Signal then Disconnect
TSI	0011	Transmit until Signal then Interrupt
SSI	1011	Skip until Signal then Interrupt
TED	0110	Transmit until Either then Disconnect
SED	1110	Skip until Either then Disconnect
TEI	0111	Transmit until Either then Interrupt
SEI	1111	Skip until Either, then Interrupt

Either = Either Count or Signal
L/S = 1 = Load
0 = Store

6.0 SYSTEM ACCESS DEVICES

Access Devices for the EAI 8400 Computing System include the 8400 Console Desk and a complete complement of peripheral equipments.

6.1 CONSOLE DESK

The operation of the 8400 Computing System is consolidated at the operator console (see Figure 6.1-1). The elements of the Console Desk are:

1. Display Panel - Provides a display of the system operating registers.

2. Operator's Panel - Contains those pushbuttons and indicators necessary to monitor and de-bug a program.

3. Maintenance Panel - Contains those controls and status indicators, needed for checkout and maintenance.

4. Typewriter - Used as a peripheral input/output device for communication between the computer and the operator.

Provision is included, also, for mounting an optional Paper Tape Station with 500 characters per second read and 110 characters per second punch.

The display panel and operator's panel are shown in Figures 6.1-2 and 6.1-3 respectively. All operating controls are pushbuttons within easy reach of the operator when seated at the console. The controls also function as indicators.

6.2 PERIPHERAL EQUIPMENT

Peripheral devices for the EAI 8400 may be selected in accordance with user needs. High and low speed magnetic tape systems, card readers and punches, line printers and a CRT display monitor are available. Illustrations of some of these devices are included on the pages following the console illustrations.

CHAPTER 7

CONTROL DESK

7.1 INTRODUCTION

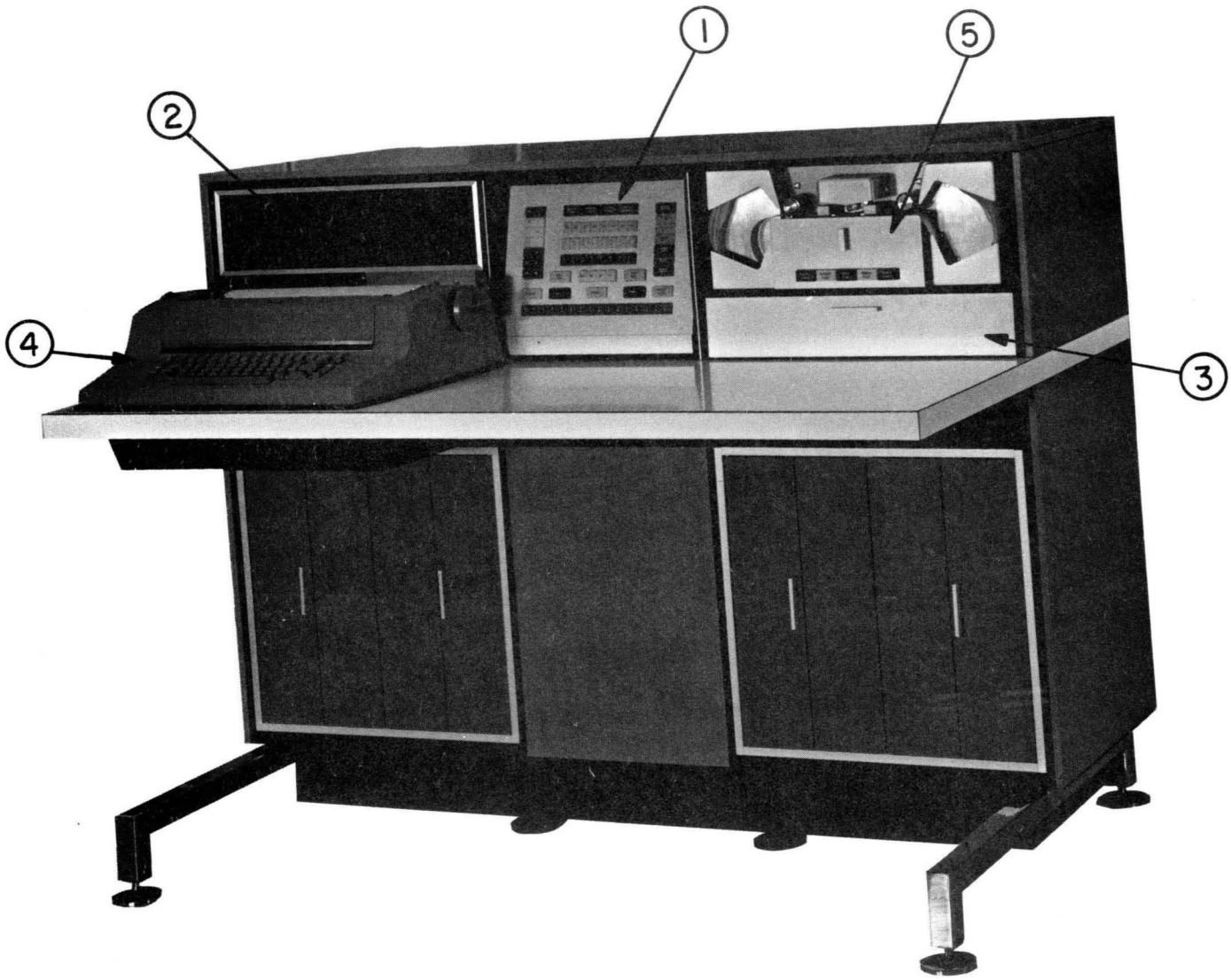
Operation of the 8400 Computer is consolidated at the system Desk Console (Figure 7-1). The Console provides complete display and control of all elements in the system. Console operating controls are indicator/pushbuttons, all within arms reach of the operator when seated at the Console.

The Desk Console includes an "on line" typewriter with an operating speed of 15 characters per second. The 8441 Paper Tape Station - reading 500 characters per second and punching 110 characters per second is also housed in the Console. Figure 7-2 shows the Control Panel used for complete system control including a special set of register controls to facilitate direct man-to-machine communication. Figure 7-3 shows the typewriter and the display panel for 9 different registers and 1 counter. Figure 7-4 shows the maintenance panel used for monitoring and manual control of system elements.

A thorough understanding of the controls, indicators and operating procedures for the Desk Console is a prerequisite for complete maintenance of the 8400 Computer. This chapter discusses the function of each control and indicator, necessary operating procedures, and presents a brief description of internal component layout.

7.2 SYSTEM CONTROLS AND INDICATORS

This section lists and briefly defines all pushbutton controls and indicators located on the System Control Panel. Circled numbers are location keys for Figure 7-5.



1. CONTROL PANEL
2. INDICATOR PANEL
3. MAINTENANCE PANEL (BEHIND COVER)
4. TYPEWRITER (IBM 735)
5. PAPER TAPE READER

FIGURE 7-1. CONTROL CONSOLE

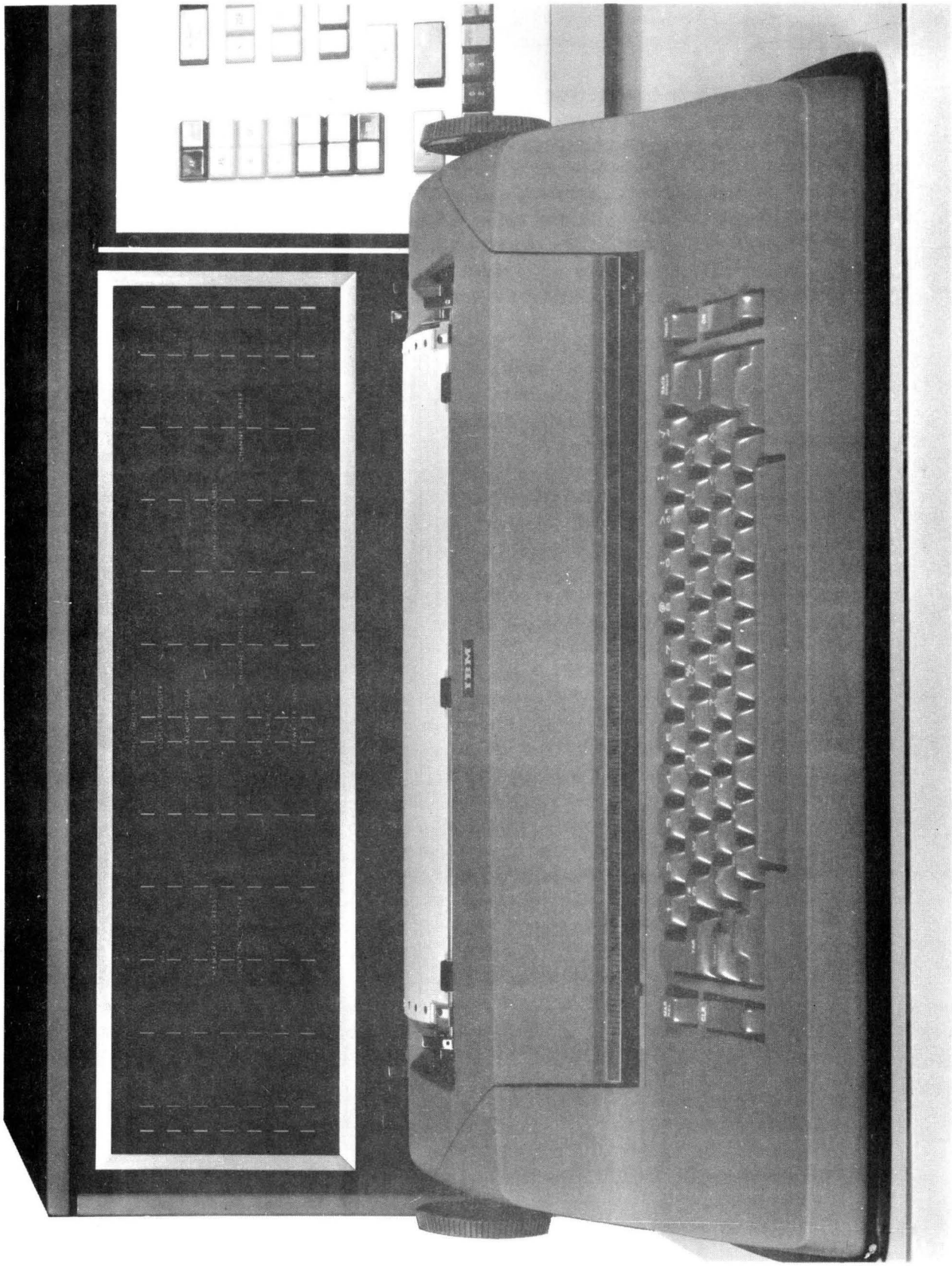


FIGURE 7-2. REGISTER DISPLAY/INPUT-OUTPUT TYPEWRITER

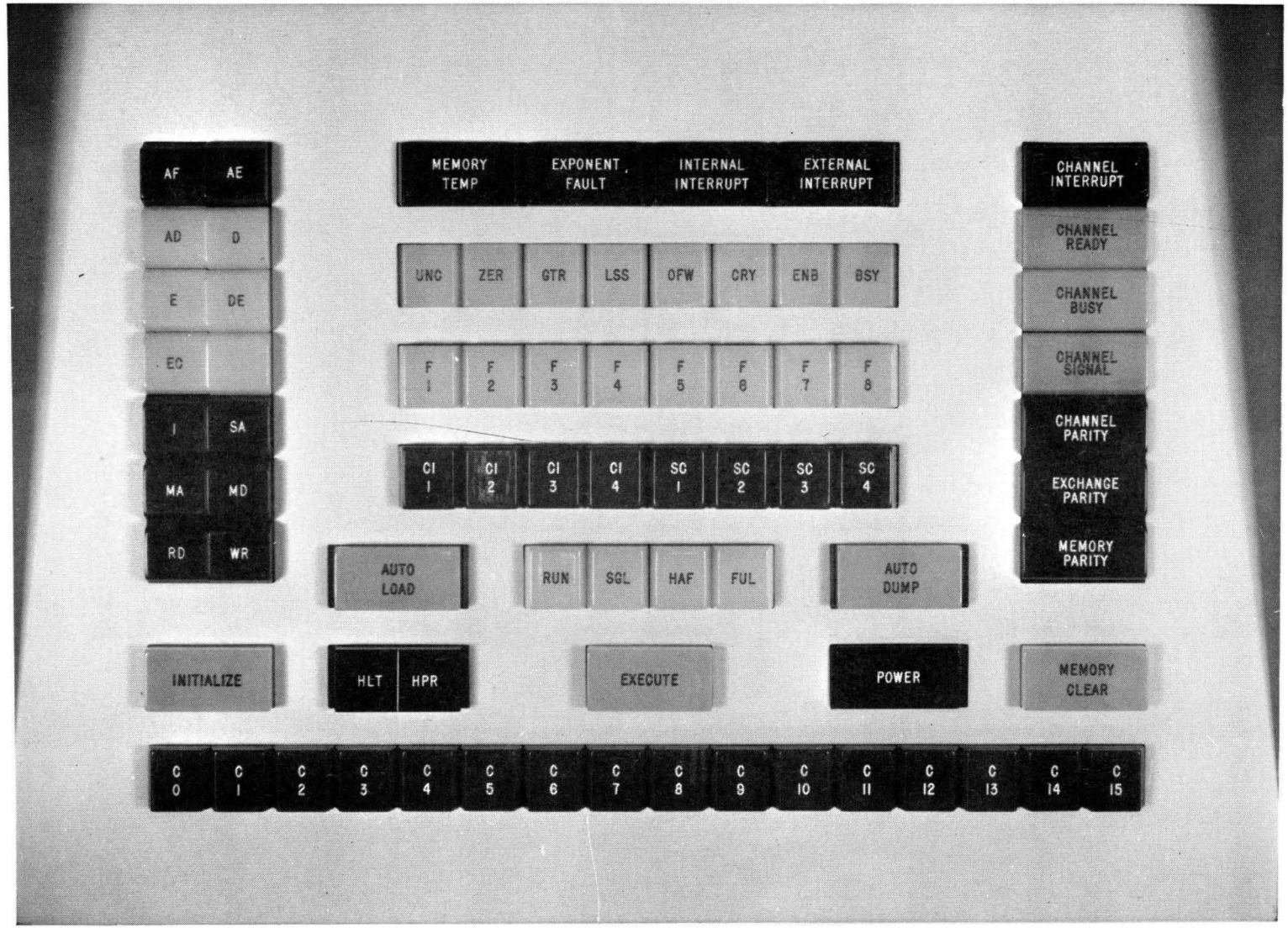


FIGURE 7-3. CONTROL PANEL

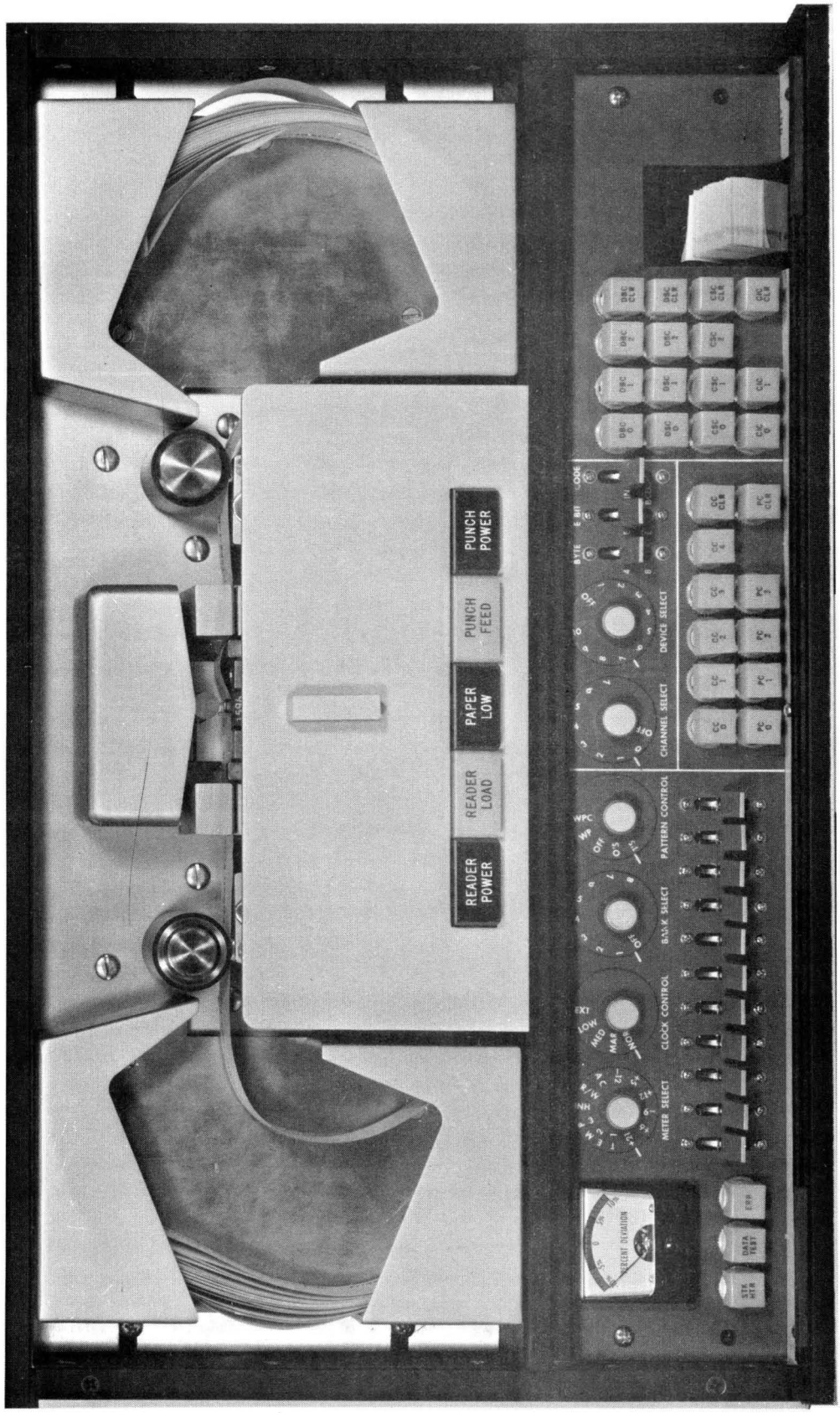


FIGURE 7-4. PAPER TAPE READER AND MAINTENANCE PANEL

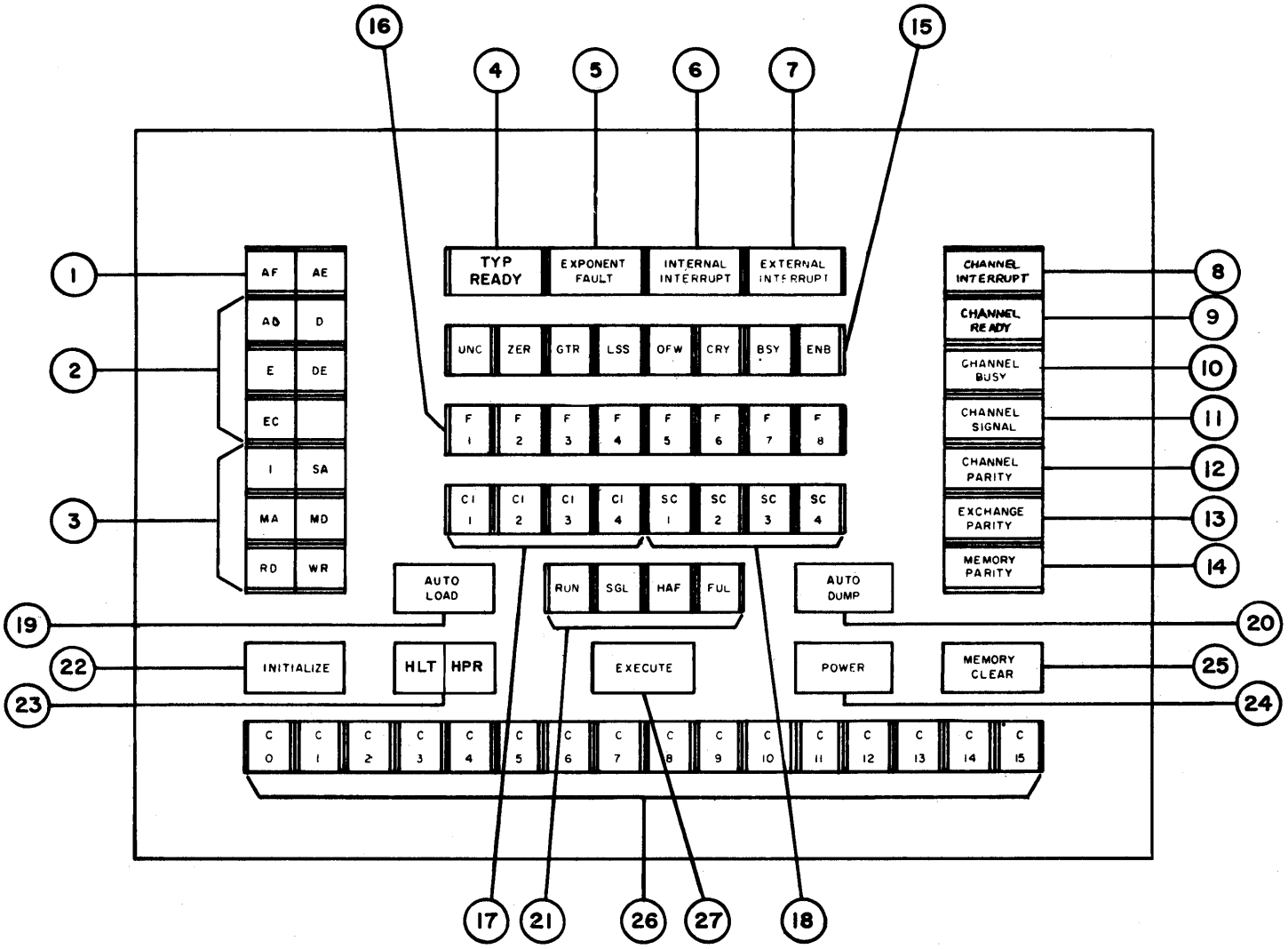


FIGURE 7.5 SYSTEM CONTROL PANEL

Register Controls.

AF, AE (1) These two pushbuttons determine which data is displayed on the ACCUMULATOR portion of the system display panel. Depressing the AF pushbutton causes bits 0 through 31 of the A register to be displayed. In this mode the A register contains one of the operands of an arithmetic operation. Depressing the AE pushbutton causes bits 0 through 15 of the A register and bits 0 and 9 through 23 of the Q register to be displayed. This configuration is used in extended arithmetic operations.

AD, D, E, DE, EC (2) These five interlocking pushbutton indicators determine which data is displayed in the DISPLAY REGISTER portion of the system display panel.

AD displays bits 0 through 23 of the Q register.

D displays bits 0 through 31 of the D register which accepts incoming data from memory and acts as an operand register for all arithmetic operations.

E displays bits 0 through 31 of the E register, which is used to store the contents of the A register during some arithmetic operations, and as an extension to the D register during double precision operations.

DE displays bits 0 through 15 of the D register in the first half of the display register and bits 0 through 15 of the E register in the second half.

EC displays bits 0 through 31 of the Exchange Control register. This pushbutton is functional only with the Automatic Data Channel Processor. The Exchange Control register controls the sequence of operations within the Exchange Module without writing each instruction through the Control Module.

Typewriter Input Controls.

I, SA, MA, MD, RD, WR (3) These six pushbutton indicators determine where typewritten instructions are routed within the computer. They are electrically interlocked so only one may be operational at a time.

I (Instruction) Typewritten information is transferred from the "W" register (0:31) to the Instruction Register.

SA (Starting Address) Typewritten information is transferred from the W register (0:15) to the Location Counter and the M field of the Instruction register when the CR key is depressed.

MA (Memory Address) Typewritten information is transferred from the W register (0:15) to the M field of the Instruction register when the CR key is depressed.

RD (Read From Memory) Information read from addressed memory location is displayed on the Memory Data register (0:35).

MD (Memory Data) Typewritten information is transferred from the W register (0:33) to the Memory Data Bus when the CR key is depressed.

WR (Write Into Memory) Contents of the Memory Data Bus (0:35) are transferred to the Memory Data Register and then written into the addressed memory location.

Typewriter Ready.

TYP RDY (TYPEWRITER READY) (4) This indicator when illuminated indicates the typewriter has been selected as an I/O device.

Exponent Fault (5)

This indicator when illuminated indicates an exponent overflow or underflow within the Arithmetic Module.

Internal Interrupt (6)

This indicator when illuminated indicates an unserviced internal interrupt.

External Interrupt (7)

This indicator when illuminated indicates an unserviced external interrupt.

Channel Interrupt (8)

This indicator when illuminated indicates an unserviced exchange channel interrupt. The affected channel is determined by the setting of the CHANNEL SELECT switch on the Maintenance Panel. When the CHANNEL SELECT switch is in the OFF position an interrupt on any exchange channel will illuminate this indicator.

Channel Condition Indicators.

These three indicators when illuminated indicate various conditions as listed in the following paragraphs. In the AUL (Auto Load) or AUD (Auto Dump) mode the condition indicated applies to the channel selected by the CHANNEL SELECT switch. In the program control mode the condition indicated is on any addressed channel. The particular channel may be determined by the CHANNEL SELECT switch.

Channel Ready (9) This indicator when illuminated indicates that the Exchange channel selected is ready to accept information from or load information into Memory.

Channel Bus (10) This indicator when illuminated indicates that an exchange channel has been addressed and is accepting information from or loading information into Memory.

Channel Signal (11) This indicator when illuminated indicates that a peripheral device has sent a fault signal. (A gap on magnetic tape; STOP code on paper tape reader, low paper on paper tape punch, etc.)

Parity Indicators

Channel Parity (12) This indicator when illuminated indicates that a Parity error has occurred in the selected device.

Exchange Parity (13) This indicator when illuminated indicates that a parity error has occurred within the exchange module.

Memory Parity (14) This indicator when illuminated indicates a memory parity error has occurred. In the AUL or AUD mode the error occurred in the particular channel selected by the CHANNEL SELECT switch. In the program control mode the parity error has occurred in any addressed bank, the particular bank may be determined by the CHANNEL SELECT switch.

NOTE

The system will not halt when a parity error occurs unless programmed to do so.

System Flag Indicators (15)

These eight indicators display machine conditions that occur during the course of a program.

UNC (Unconditional) Illuminated when the rounding flip-flop of the ACCUMULATOR is in the 1 state. This flag may be used to generate an unconditional jump instruction.

ZERO (Zero) Illuminated when the contents of the ACCUMULATOR are zero, or as a true result of TSL'S and Boolean Connect Instruction (BEQT).

GTS (Greater) Illuminated when the contents of the ACCUMULATOR are greater than zero.

LSS (Less) Illuminated when the contents of the ACCUMULATOR are less than zero.

OFW (Overflow) Illuminated when there is an overflow condition in the accumulator.

CRY (Carry Out) Illuminated when there has been a carry out of the most significant bit of the accumulator (A1, not A0).

BSY (Busy) Illuminated when an addressed function line or data channel is busy.

ENB (Enable) Illuminated when the system interrupt lines are enabled.

Programmer Flag Controls and Indicators

F1 Through F8 (16) These eight indicator pushbuttons are the programmer flags. Depressing the pushbutton complements the flag. The indicator is illuminated when the flag bit is a "1".

Console Interrupt Controls and Indicators

C1 Through C4 (17) These four indicator pushbuttons are manual console interrupts. When illuminated they indicate an unserviced console interrupt. The indicator is extinguished when the interrupt is serviced or the push-button is depressed a second time.

Configuration Switches

SC1 Through SC4 (18) These four pushbutton indicators are control switches for special 8400 configurations.

Auto Load and Auto Dump

Auto Load (19)

This indicator pushbutton is used to load information from a single peripheral device into memory during manual operation. It is illuminated during the Auto Load operation. (See Operating Procedures.)

Auto Dump (20) This indicator pushbutton is used to transfer information from Memory onto a single peripheral device during manual operation. It is illuminated during the Auto Dump operation. (See Operating Procedures.)

Clock Controls

RUN/SGL/HAF/FUL/ (21) These four indicator pushbuttons establish the mode of operation of the 8400. The RUN, HAF, FUL pushbuttons are electrically interlocked so only one is functional at a time. The SGL pushbutton mechanically locks when depressed.

RUN When depressed sets the system in normal sequential control cycle.

SGL When depressed sets the system to operate clock pulse by clock pulse. Each time the Execute pushbutton is depressed one clock pulse is generated.

HAF When depressed the system will perform half the instruction control sequence. The first time the Execute pushbutton is depressed the instruction word will be transferred to the I register and any address modification

called for will be performed. Depressing Execute a second time allows the system to finish the instruction cycle and halt.

FUL When depressed sets the system to perform the complete instruction and halt, each time the Execute pushbutton is depressed.

Initialize (22)

This indicator pushbutton is used to clear the system. The CC and CIC counter are cleared and the PC counter is set to PC12. Memory is not cleared by this control.

Halt and Halt/Proceed (HLT/HPR) (23)

HLT This indicator pushbutton is used to manually halt the system. It is illuminated when the system is in the halt condition.

HPR When illuminated indicates the system has been halted by a Halt/Proceed instruction. Depressing the Execute pushbutton restarts the system.

Power (24)

Depressing this pushbutton energizes the system. The indicator is illuminated when system power is on. Depressing the pushbutton again, de-energizes the system.

Memory Clear (25)

Depressing this pushbutton clears all non-protected Memory locations.

Console Register

C0 Through C15 (26) These pushbutton indicators represent bits 0 through 15

of the console register; when illuminated they indicate that the particular bit is in the 1 state. Depressing a pushbutton complements the particular bit.

EXECUTE (27)

This pushbutton indicator is used to start operational sequences after all other controls have been preset. It starts the system in any mode.

7.3 SYSTEM DISPLAY PANEL

Circled numbers are keyed to Figure 7-6.

ACCUMULATOR (1)

This area displays bits 0 through 31 of the ACCUMULATOR when pushbutton AF on the SYSTEM CONTROL PANEL is depressed. When pushbutton AE on the SYSTEM CONTROL PANEL is depressed bits 0 through 15 of the ACCUMULATOR and bits 0 and 9 through 23 of the Q register are displayed.

DISPLAY REGISTER (2)

This area is a general purpose display. The data display is determined by five pushbuttons as indicated below:

Pushbutton	Data Display
AD	Bits 0 through 23 of the Q register in the ARITHMETIC MODULE.
D	Bits 0 through 31 of the D register in the ARITHMETIC MODULE.

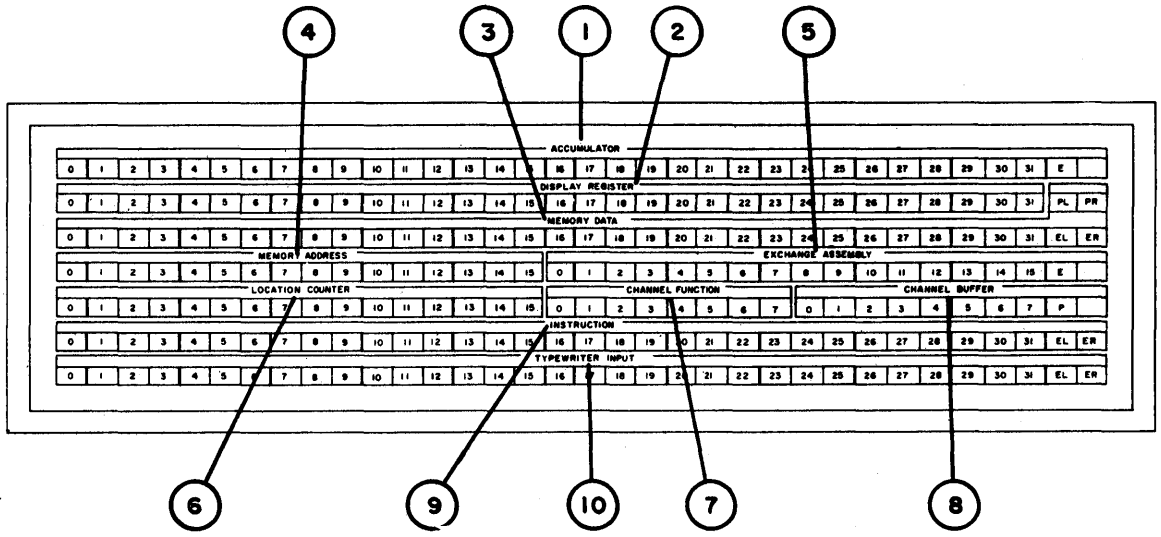


FIGURE 7.6 SYSTEM DISPLAY PANEL

Pushbutton (Cont)	Data Display
E	Bits 0 through 31 of the E register in the ARITHMETIC MODULE.
DE	Bits 0 through 15 of the D register and bits 0 through 15 of the E register. The D register is displayed in the left half of this area.
EC	Bits 0 through 31 of the Exchange Control Register in the EXCHANGE MODULE. Used only if system has ADCP (Automatic Data Channel Processor) option.

MEMORY DATA (3)

This area displays the contents of the memory location that is addressed in either manual or program controlled operation. However, it will not display if BANK SELECT switch is in AUTO position unless memory is being requested by control.

MEMORY ADDRESS (4)

This area displays the address of the memory location being accessed by the CONTROL MODULE.

EXCHANGE ASSEMBLY (5)

This area displays the contents of the Exchange Assembly Register in the EXCHANGE MODULE.

LOCATION COUNTER (6)

This area displays the address of the next instruction to be executed.

CHANNEL FUNCTION (7)

This area displays the condition of Flip-Flops within the Channel Function Register when data is being transferred to or from a peripheral device by the EXCHANGE MODULE as follows:

Indicator	Function
Bit 0	This indicator when illuminated specifies that an EXEC bit accompanies each data word to or from memory. When extinguished an EXEC does not accompany data.
Bit 1	This indicator when illuminated specifies that data is being transferred in Binary code. When extinguished data is in BCD form.
Bit 2	The indicator specifies which half of the memory word is being addressed. When illuminated it specifies that the left half of the memory word is being addressed.
Bit 3	When illuminated this indicator specifies that alternate left and right half words are being transferred to memory or to a peripheral device by the Exchange Module.

Indicator (Cont)	Function
Bit 4	This indicator specifies the direction of data transmission. When illuminated it indicates transmission to the EXCHANGE MODULE from the memory.
Bits 5,6,7	These indicators display the Byte size and number of bytes per half word of data being transmitted as follows:

5	6	7	Bits/Byte	Bytes/Half Word
0	0	0	8	0
0	0	1	8	1
0	1	1	16	1
0	1	0	8	2
1	0	0	4	4
1	0	1	4	1
1	1	0	4	2
1	1	1	4	3

CHANNEL BUFFER (8)

This area displays the condition of the Channel Buffer register which is located in the Exchange Module. Data enters the register during program control operation.

INSTRUCTION (9)

This area displays the contents of the Instruction register located in the Control Module.

TYPEWRITER INPUT (10)

This area displays the contents of the "W" register located in the Console Desk. Data enters the register during manual input operation of the typewriter.

7.4 MAINTENANCE PANEL

The maintenance panel contains controls and indicators which may be used for both test and normal operating purposes. The circled numbers in the following descriptions are keyed to Figure 7-7.

LAMP TEST, (1) ON/OFF

In the ON position, this two position toggle switch enables all light drivers in the console, providing a quick check of all lights and light drivers.

KEYBOARD, (2) UNLCK/LCK

This two position toggle switch controls the entry of data from the typewriter. In the "UNLCK" position the typewriter keyboard is unconditionally unlocked and may be used at any time to enter data. In the "LCK" position the keyboard is under program control and cannot be used unless so designated by the program.

CLOCK CONTROL (3) NOR/MAR/MED/LOW/EXT

This 5 position rotary switch determines the clock frequency of the system.

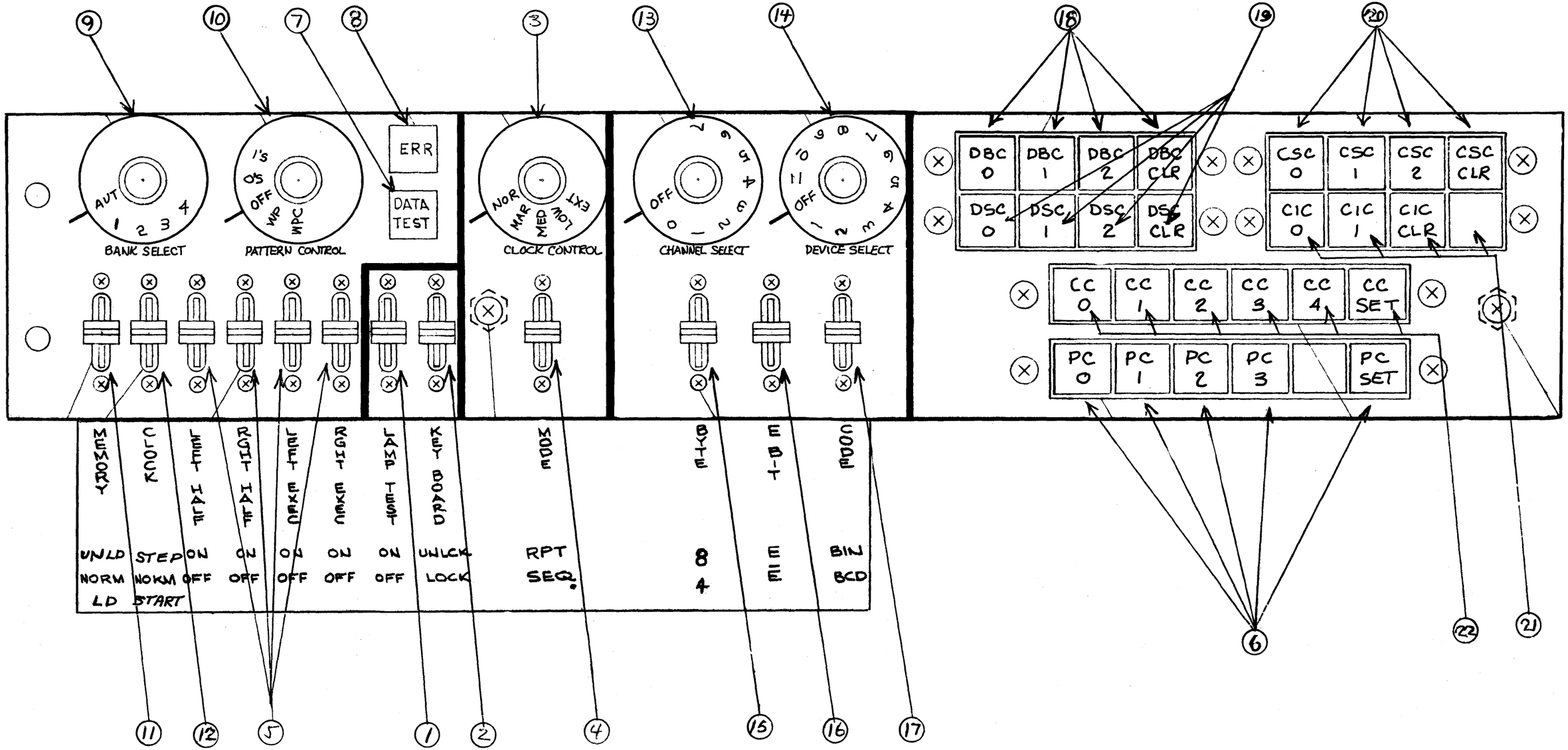


FIGURE 7.7
MAINTENANCE
CONTROL PANEL

NOR
Normal clock frequency of 4.0 megacycles

MAR
Marginal clock frequency of 4.4 megacycles used for system testing and maintenance

MED
Medium clock frequency of approximately 1.2 kilocycles

LOW
Low clock frequency approximately 1.5 cycles per second

EXT
System clock supplied by an external device

MODE (4) SEQ/RPT

This two position toggle switch determines the operational mode of the CONTROL MODULE. In the sequential (SEQ) position, upon completion of an instruction the location counter in the CONTROL MODULE supplies the next instruction. In the repetitive (REP) position the instruction fetch portion of the program is omitted and the LOCATION COUNTER is not incremented; the present instruction is repeated.

LH (Left Half), RH (Right Half), LE (Left EXE), RE (Right EXC) ON/OFF (5)

These four, two position toggle switches determine the memory word format during manual data entry. When in the OFF position, information in memory is protected; no manual entries or modifications can take place. In the ON position, manual entry of data is permitted according to the format determined by the switches in the ON position.

PC0, PC1, PC2, PC3 PC SET (6)

These five locking indicator pushbuttons control the manual presetting of the phase counter in the CONTROL MODULE. The first four (PC0 through PC3) operate in a BCD fashion, as follows:

Switch	PC0 - PC1 - PC2 - PC3
BCD Value	8 - 4 - 2 - 1

Proper manipulation of these switches allows the operator to preset the phase counter to any value from PC1 to PC14. The PC-SET pushbutton must be depressed to preset the phase counter after the proper count has been selected on PC0 through PC3. To manually set the counters the MODE SWITCH on the CONTROL PANEL must be on SGL.

DATA TEST (7)

This indicator operates in conjunction with the MEMORY PATTERN GENERATOR and the MEMORY PATTERN CONTROL SWITCH. The indicator is illuminated when testing with a 1's pattern and extinguished when testing with a 0's pattern.

ERR (8)

This indicator is illuminated when a memory error is detected during a memory test pattern.

BANK SELECT AUT 1 2 3 4 (9)

This five position rotary switch selects one of the memory banks when running a memory pattern test. It performs three functions: 1) selects the memory bank. 2) displays the contents of the bank on the MEMORY DATA

display area during UNLD,,3) displays the address of the memory location on the MEMORY ADDRESS display area during UNLD. In the AUT position memory data and address are under program control.

PATTERN CONTROL (10)

This five position rotary switch is used to generate a memory self test pattern into the memory bank selected by the BANK SELECT switch. The patterns are as follows:

1's: All 1's are written into each location of the memory bank selected.

0's: All zeros are written into the memory bank selected.

OFF: The memory self test function is disabled.

WP: Worse case pattern is written into the memory bank selected as follows:

1's into memory location 00001
1's into memory location 00002
0's into memory location 00003
0's into memory location 00004
1's into memory location 00005
1's into memory location 00006
0's into memory location 00007
0's into memory location 00008

etc., until all memory locations are full.

WPC: Worse case pattern complement is written into the memory bank.

CAUTION

The test patterns destroy the contents of all memory locations in the selected memory bank, including protected locations. The PATTERN CONTROL must therefore be in the OFF position during normal computer operation.

MEMORY (LD/NORM/UNLD) 11

This three position toggle switch is used in conjunction with the PATTERN CONTROL switch to test the bank selected by the BANK SELECT switch.

CLOCK (START/NORM/STEP) 12

This three position momentary switch controls the memory clock during memory test. Depressing the switch momentarily to the START position stops the clock. Depressing the switch again to the STEP position generates one clock pulse, permitting clock pulse by clock pulse testing of the memory.

CHANNEL SELECT 13

This nine position rotary switch is used to select the Exchange Module Data channel to be used during an AUL (Auto Load) or AUD (Auto Dump) operation. The Channel Function register displays the conditions of transfer, and the CHANNEL BUFFER register displays the information being transferred. In the OFF position, channel selection is under program control.

DEVICE SELECT 14

This eleven position rotary switch selects the device (on the channel designated by the CHANNEL SELECT switch) to be accessed during manual operation.

BYTE 4/8 15

This two position toggle switch selects the byte size between memory and the selected device during AUL (Auto Load) and AUD (Auto Dump) operation.

E Bit E/E 16

This two position toggle switch determines whether or not an EXEC bit accompanies data during an AUL (Auto Load) or an AUD (Auto Dump) operation.

CODE, BIN BCD 17

This two position toggle switch controls code conversion during manual operations. When in the BIN position, data is transferred without code conversion during an AUL or AUD OPERATION. In the BCD position, code conversion takes place during an AUL or AUD operation.

DBC, DBC0, DBC1, DBC2, DBC CLR 18

These four momentary pushbutton indicator switches are used to preset the Exchange Module Device Buffer Counter which controls the transfer of data from device to memory or from memory to device. Depressing a pushbutton sets the corresponding counter stage. Depressing DBC CLR clears the counter. Each indicator is illuminated when the corresponding bit is in the 1 state.

DSC, DSC0, DSC1, DSC2, DSC CLR 19

These three momentary pushbutton indicator switches are used to preset the Exchange Module Data Stack Counter. This counter is used in controlling information flow from the Exchange Assembly Register (EAR) to the Channel Buffer Register (CBR) and from EAR to the data stack. Depressing a push-button sets the corresponding stage. Depressing DSC CLR clears the counter. The indicator is illuminated when the corresponding stage is in the 1 state.

CSC, CSC0, CSC1, CSC2, CSC CLR (Operation with ADCP Option only) 20

These four momentary indicator pushbutton switches are used to preset or clear the Exchange Module Control Stack Counter. The counter is used to control the operational sequence and information flow within the Auto Data Channel Processor. Depressing a pushbutton sets the corresponding stage in the counter. Depressing CSC CLR clears the counter. The indicator is illuminated when the corresponding stage is in the 1 state.

CIC, CIC0, CIC1, CIC CLR 21

These three pushbutton indicator switches are used to preset or clear the Exchange Module Control Interface Counter. The counter is used to control the instruction Sequence between the Exchange Module and the Control Module. Depressing a pushbutton sets the corresponding counter stage. Depressing CIC CLR clears the counter. The indicator is illuminated when the corresponding stage is in the 1 state.

The following Exchange Counter pushbutton switches are coded as indicated.

EXCHANGE COUNTER COUNTS	DBC0 DSC0 CSC0	DBC1 DSC1 CSC1 CIC0	DBC2 DSC2 CSC2 CIC1
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0

CC0, CC1, CC2, CC3, CC4, CC SET 22

These five locking indicators display the state of the Arithmetic Module Cycle Counter. The counter is used to control the sequence of arithmetic operations within the Arithmetic Module. The indicators are illuminated when the corresponding counter stage is in the 1 state. The pushbuttons are coded in extended BCD as follows:

CC0	CC1	CC2	CC1	CC0	CC CLR
16	8	4	2	1	CLEAR

7.5 OPERATING PROCEDURES

7.5.1 Pre-Operational Control Setting

The MAINTENANCE PANEL controls listed in Table 7.1 should be set to the indicated position prior to system operation. These control settings insure that system operation is not impaired and that all facilities are functioning. The remaining controls on the MAINTENANCE PANEL may be left in any position since these switches will be set prior to a definite operation.

TABLE 7.1 PRE-OPERATIONAL CONTROL SETTINGS

Control	Setting
LAMP TEST	OFF
KEYBOARD	UNLK
MODE	SEQ
CLOCK CONTROL	NOR
BANK SELECT	AUT
PATTERN CONTROL	OFF
CHANNEL SELECT	OFF

7.5.2 Manual Operating Sequences

7.5.2.1 Machine Power. Depress the POWER switch on the system control panel. All control sequences are internally generated. The system is in halt status and is initialized.

7.5.2.2 Typewriter. When the 8400 is in the halt status, only 12 typewriter keys are legal, they are +, -, 0 through 7, CR and TAB. Only

these keys may be used to manually input data or instructions. Other keys may be used for operator notations but they will not be recognized by the 8400. Under program control all typewriter keys are legal.

The + and - keys are used to set the sign of the input information and to set the EXEC bit in an instruction or a memory data word. When entering an EXEC bit the + key is used to generate a binary 0 and the - key to generate a binary 1.

The 0 through 7 keys input data in Octal code.

Depressing the TAB key sets the 8400 to the right half of a data word.

Depressing the CR key transfers information from the TYPEWRITER REGISTER to the desired location.

When inputting data, the following word format is used.

SA (Starting Address)

$\pm N_1, N_2, N_3, N_4, N_5$ CR

I (Instruction Word)

$\pm, N_{1L}, N_{2L}, N_{3L}, N_{4L}, N_{5L}, \pm N_{1R}, N_{2R}, N_{3R}, N_{4R}, N_{5R}, \pm, \pm$, CR

MA (Memory Address)

$\pm, N_1, N_2, N_3, N_4, N_5$, CR

MD (Memory Data)

$\pm N_{1L}, N_{2L}, N_{3L}, N_{4L}, N_{5L}$, CR, $\pm N_{1R}, N_{2R}, N_{3R}, N_{4R}, N_{5R}, \pm, \pm$, CR

N_1 in each case represents a legal number represented in Octal code 0-7. N_1 would be the first octal character, N_2 the second, etc.

N_{1L} represents first octal character in the left half of the word. N_{1R} the second octal character in the right half of the word.

The $\pm$ at the end of the instruction and Memory Data words represent the EXEC bits.

7.5.2.3 Auto Load (AUL) — Machine Functions

1. Depress HLT — HLT light, PC-12, HFF Sets

2. At the MAINTENANCE PANEL

   a. Select Channel
   b. Select Device
   c. Select Byte size
   d. Select (E or $\overline{E}$)
   e. Select Code (BIN/BCD)

3. Depress SA

   a. Type SA

      $\pm N_1, N_2, N_3, N_4, N_5$, CR — W(0:15) I(0:15) LC(0:15)

4. Depress AUL

   Auto load operation will start and run to completion. The Location Counter will

automatically be incremented
from the starting address.
The system will halt if any
one of the following events occur:

a. All information from the selected device has been exhausted. — HLT light, PC-12, HFF Set

b. Stop code is detected.

c. HLT switch is depressed.

d. All Memory locations are full.

7.5.2.4 AUD (Auto Dump) — PC-10, DSC ( )c I(29:31), LDCD I

The Auto Dump procedure is identical to Auto Load except in Step 3 above, depress the AUTO DUMP switch.

7.5.2.5 Manual Instruction Insertion

1. Depress HALT switch — HLT light, PC-12, HFF Sets

2. Depress SA switch.

3. Enter SA (use procedure outlined in 3.2.2)

   $\pm$, N_1, N_2, N_3, N_4, N_5, CR — W(0:31)c I(0:31)

4. Depress I switch

5. Enter I (use procedure outlined in 3.2.2)

   $\pm N_{1L}$, N_{2L}, N_{3L}, N_{4L}, N_{5L}

   $\pm N_{1R}$, N_{2R}, N_{3R}, N_{4R}, N_{5R}

   $\pm$, $\pm$, CR

   a. Set RPT

6. Select Mode

   (RUN, SGL, HAF, FUL)

7. Depress EXECUTE

7.5.2.6 Memory Clear

1. Depress HALT — HLT light, PC-12, HFF Sets

2. Depress SA

3. Type SA

   $\pm$N , N , N , N , N , CR — W (0:31) c I (0:31)

4. Depress MEMORY CLEAR

Memory clear operation will take place from SA to zero. (The Location Counter Increments). MEMORY CLEAR pushbutton will remain illuminated until operation is complete.

7.5.2.7 Memory Write

1. Depress HLT	HLT light, PC-12, HFF Sets
2. Select format LH/RH/LE/RE	
3. Select bank	
4. Depress MA	
5. Type address $\pm N_1$, N_2, N_3, N_4, N_5, CR	W (0:31) c I (0:31)
6. Depress MD	
7. Type data as selected by format in step 2, i.e., $\pm N_{1L}$, N_{2L}, N_{3L}, N_{4L}, N_{5L} $\pm N_{1R}$, N_{2R}, N_{3R}, N_{4R}, N_{5R} $\pm$, $\pm$, CR	W (0:31) c I (0:31)

8. Depress WR	I (0:15) CAR (0:15) M, CWRM, CRQM, chosen format appear in M, DW CDB (0:33) M

7.5.2.8 Memory Read

1. Depress HLT	HLT light, PC-12, HFF Sets
2. Select bank	
3. Depress MA	
4. Type address $\pm N_1$, N_2, N_3, N_4, N_5, CR	
5. Depress RD	I (0:15) CAB (0:15) M, CWRM, CRQM, chosen word appears in MDR

CAUTION

If the system is initialized, all Memory locations can be cleared. To set the memory protect flip-flops, an SFL instruction is used.

7.5.2.9 Mode Change

1. Depress HLT — HLT light, PC-12, HFF Sets

2. Select Mode
   (RUN, SGL, HAF, FUL)

3. Depress EXECUTE

7.5.2.10 Initialize Machine

1. Depress HLT — HLT light, PC-12, HFF Sets

2. Depress INITIALIZE

   Machine is now in HALT status and INITIALIZED.

NOTE: Machine is automatically INITIALIZED when power is applied by depressing the POWER pushbutton.

7.5.2.11 Console Register SetUp

1. Select any mode except SGL

2. Complement CONSOLE REGISTER bits by depressing appropriate "C" pushbutton.

7.5.2.12 Flag Register SetUp

1. Select any mode except SGL

2. Complement FLAG REGISTER by depressing appropriate "F" button.

7.5.2.13 Console Interrupt

1. Set the CONSOLE INTERRUPT by depressing the appropriate C1 pushbutton.

7.5.2.14 Control Module Counter Preset

1. Depress HLT pushbutton

2. Depress SGL

3. Check the condition of PC indicators on the SYSTEM CONTROL PANEL; if an indicator is illuminated, the corresponding counter is in the 1 state. If a different count is desired first depress "PC CLR", then select the proper count by depressing the proper pushbutton. The pushbuttons operate in extended BCD fashion.

PC0	PC1	PC2	PC3	PC CLR
8	4	2	1	0

4. Select desired mode of operation as indicated in 7.5.2.10.

5. Depress EXECUTE

7.5.2.15 Exchange Module Counter Preset

1. Depress HLT — HLT light, PC-12, HFF Sets

2. Depress SGL

3. Check the settings of DBC0 through 2, DSC0 through 2, CSC0 through 2, and C1C0 through 1. If a different setting is desired, first depress the appropriate "CLR" button and second depress the desired combinations of push-buttons. Coding on these pushbuttons is as follows:

Exchange Counter Counts	DBC0 DBC0 DSC0	DBC1 DSC1 CSC1 C1C0	DBC2 (PP) DSC2 (CP) CSC2 (AP) C1C1 (IP)
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0

7.6 PHYSICAL LOCATION OF COMPONENTS

7.6.1 Introduction

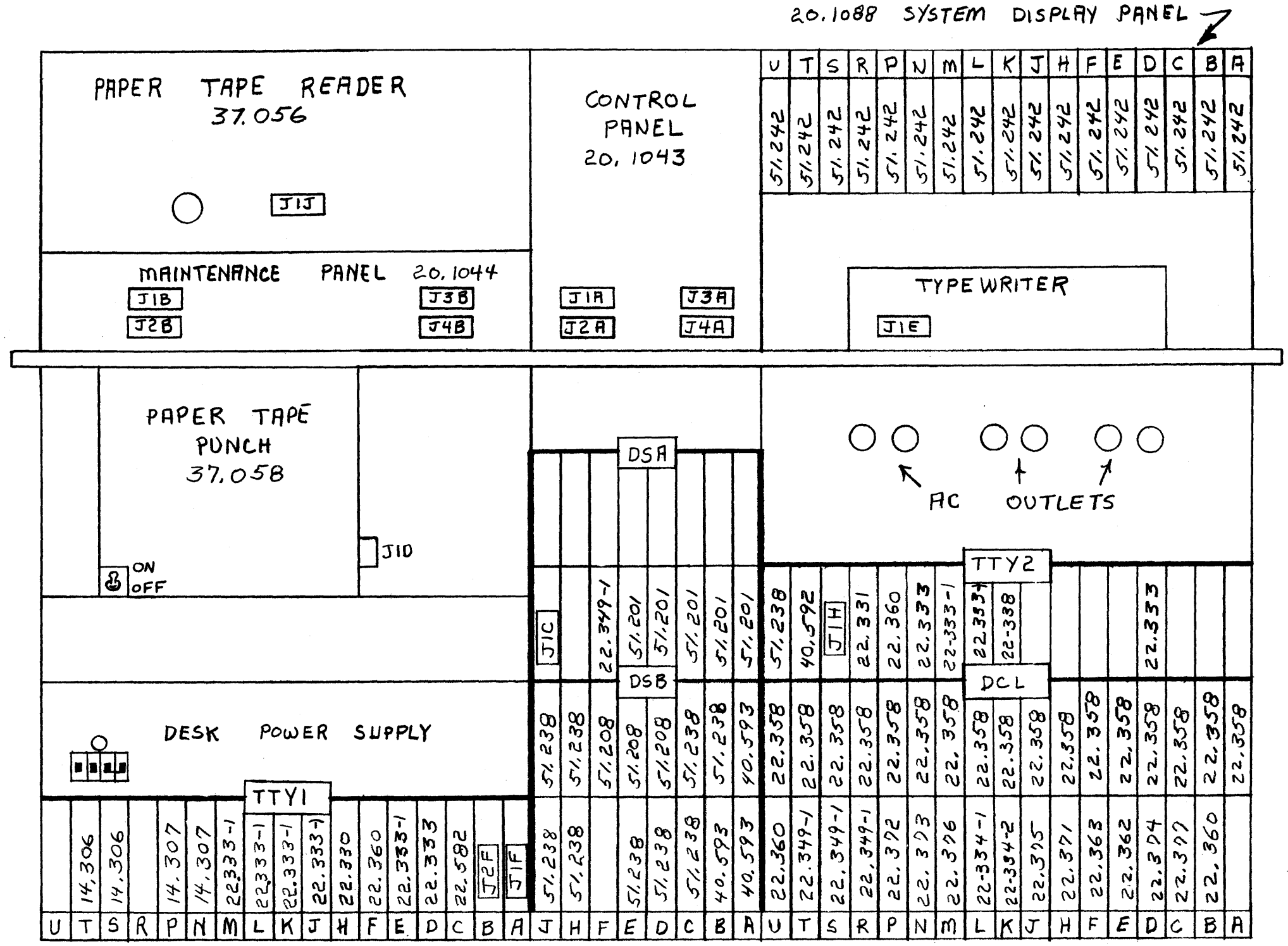
This section relates to the physical location of the desk components and the identity of the different modules as one would find by looking from the rear panel covers.

7.6.2 Component layout

Figure 7-8 is the Desk's rear view illustrating the location and layout of the control panel, the maintenance panel, and associated input-output equipment. The two row panels listed are: DSA, DSB, DCL, and TTY.

Row 1 refers to the top row and row 2 the bottom row of each panel.

Distribution panels (DSA and DSB) primarily contain Combo Cards and Wetting Cards at location "A" through "T". These cards are used to distribute logic signals from the Desk to the Floating Point Processor and vice-versa.



CABLE	RACK PANEL	PLUG	RACK PANEL	PLUG
AD1	A1-AMC-C4		D1-DSB-C2	
AD2	A1-AMC-B4		D1-DSB-D2	
AD3	A1-AMC-A4		D1-DSB-E2	
ED1	E1-CHCB1-U1		D1-DSB-E1	
ED2	E1-EMCC-R1		D1-DSB-H2	
ED3	E1-CHCB-T1		D1-DSB-J2	
ED4	E1-CHCA1-T3		D1-TTY2-U1	
ED5	E1-CHCA1-S3		D1-TTY2-T1	
CD1	C2-CMB-U2		D1-DSB-B1	
CD2	C2-CMB-U1		D1-DSB-C1	

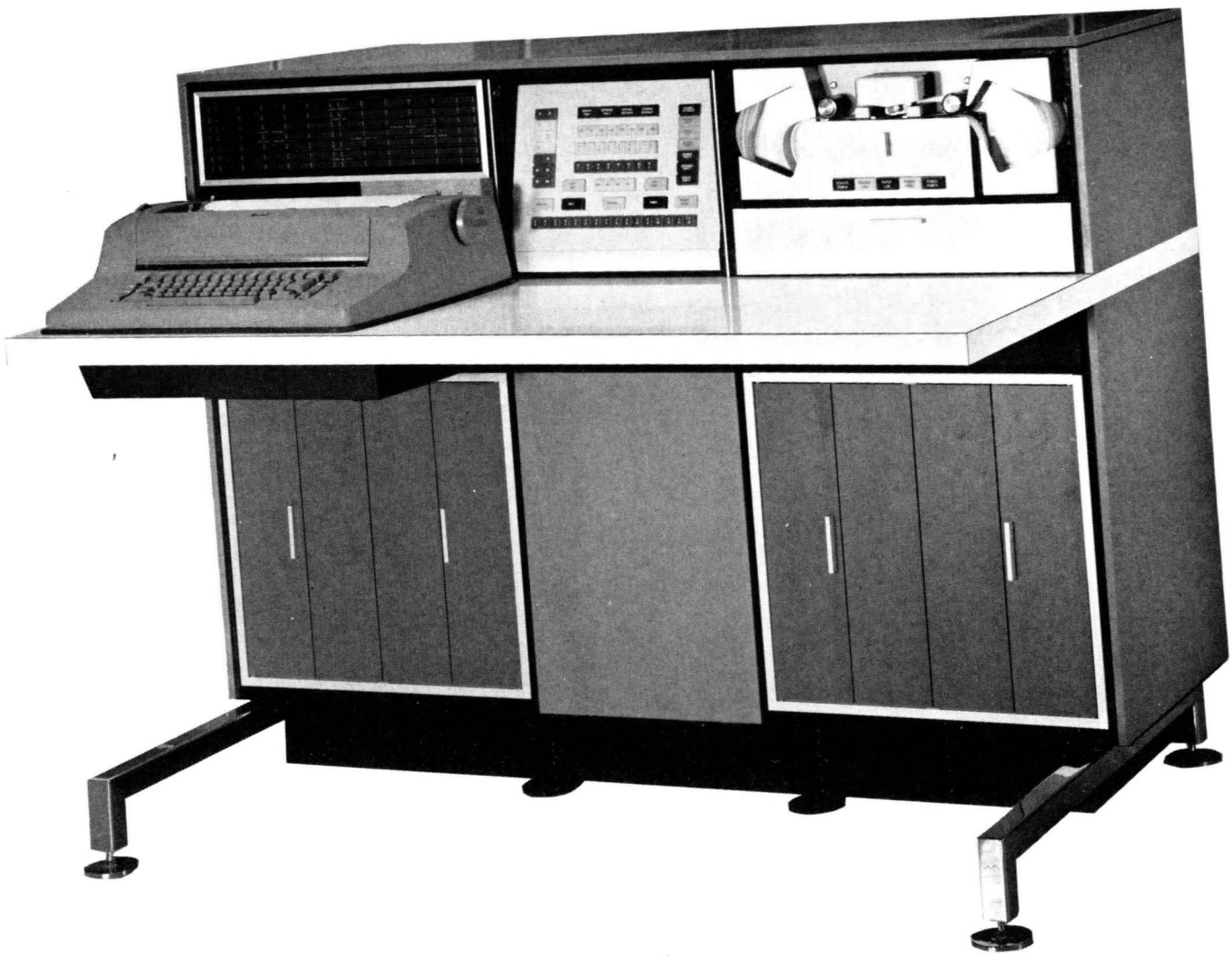
CABLE	RACK PANEL	PLUG	RACK PANEL	PLUG
MD1	M1-MBCL1-U1		D1-DSB-J1	
MD2	M1-MBCL1-T1		D1-DSB-H1	
DC1	D1-DSB-B2		C2-CMB-S2	
DC1-1	C2-CMB-S2		C1-CMD-B4	
DC2	D1-DSB-A2		C2-CMB-D1	
DC2-1	C2-CMB-D1		C1-CMD-S4	
DC3	D1-DSB-A1		C2-CMC-T1	
DC3-1	C2-CMC-T1		C1-CMD-A2	
DC4	D1-DSB-D1		C2-CMA-U1	

FIGURE 7-8 OPERATORS DESK, REAR VIEW.

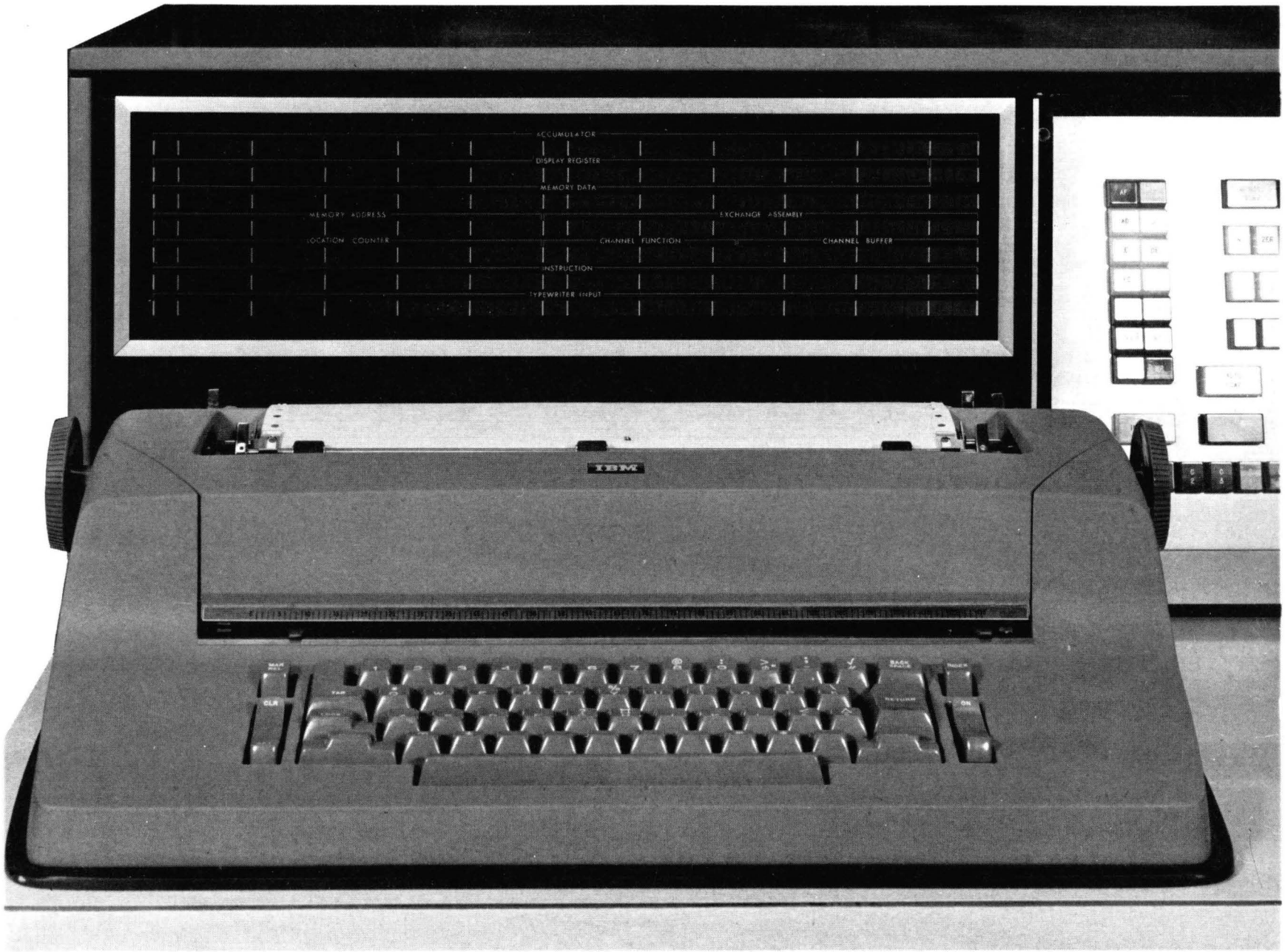
24206

The Control Logic panel (DCL) contains a variety of circuitry such as control for the Typewriter and Paper Tape Station Device Controllers. For example, the "W" register counter is located on row 2, cards F and E. These cards would be identified as DCL-F02 and DCL-E02. The TTY panels (one and two) also contain a variety of circuit cards located in slots "A" through "U". The circuitry is mainly for the Typewriter Device Controller, however, TTY-1 panel contains some circuitry for the Paper Tape Station Device Controller. The System Display Panel contains 17 identical Light Driver cards located in positions "A" through "U".

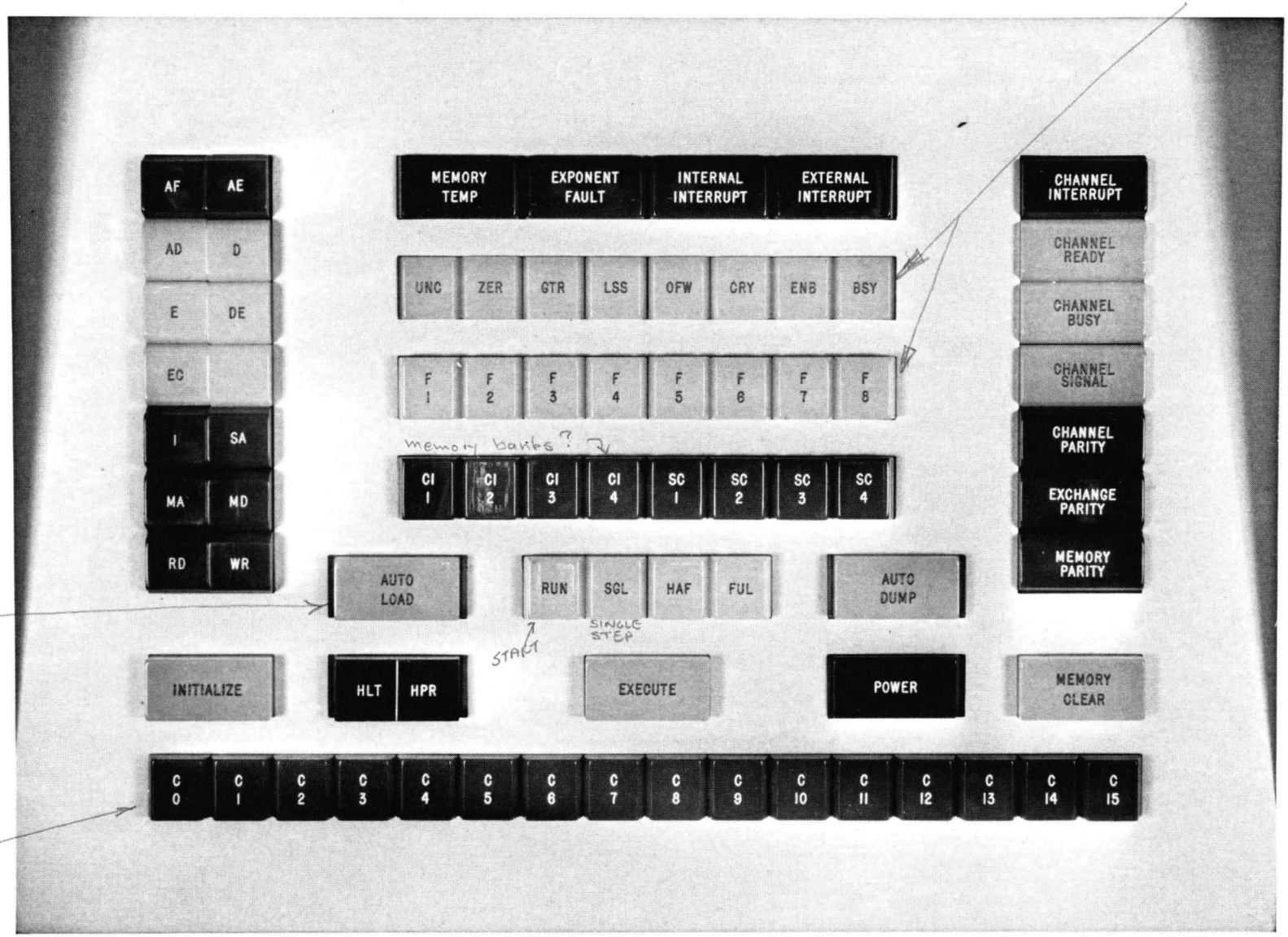
The model number of each card is shown for each card location used. A table on Figure 7-8 also provides a list of cables with the rack, panel, and plug designations for each cable.



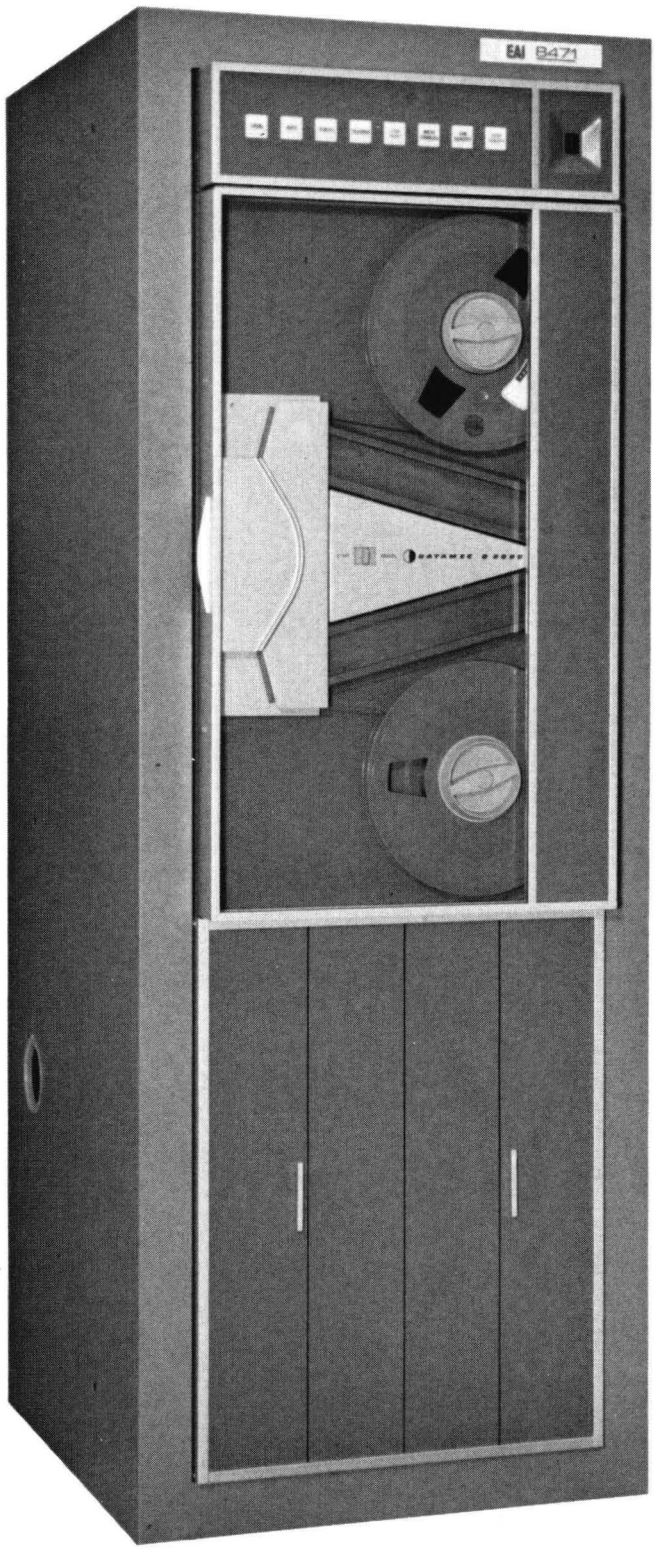
CONTROL CONSOLE



REGISTER DISPLAY / INPUT - OUTPUT TYPEWRITER



OPERATOR'S PANEL



MAGNETIC TAPE TRANSPORT

PUNCHED-CARD READER

LINE PRINTER

FLOATING-POINT PROCESSOR

1.0 STANDARD PROGRAMS AND PROGRAMMING SYSTEMS

Software is the second dimension of a computing system, and is equally vital to its effectiveness. EAI provides a comprehensive set of programming systems, programs and routines tailored to the needs of hybrid simulation, analog program setup and checkout, and general scientific computation. These range from MACRO assembly and FORTRAN IV Compiler systems, a library of relocatable subroutines that have been designed for maximum execution speed and recursive entry, and monitor system featuring console control and debugging; to a system of programs designed for increasing the efficiency of programming and operating analog and hybrid computers.

A design criteria for EAI software is its overall effectiveness in the hands of an individual user. That is to say, throughout the implementation and description is the attempt to provide software uniquely applicable in a time-shared environment or intimate man-machine relationship. Facilities for batch processing are also provided.

Efficient manpower utilization is as important as high problem throughput. The EAI 8400 Programming Systems provide a balanced capability for the scientific, research and simulation laboratory in meeting both criteria. The significant characteristics of the 8400 Software system are:

1. Dynamic Relocation of user programs and 8400 software - the unique ability of the 8400 to relocate programs once they are in core memory.

2. Highly Efficient Object Coding produced by programming systems, optimized to achieve maximum execution rate of user programs.

3. Real-Time, interruptable programs designed for recursive programming in a multi-user or hybrid environment.

4. Software design to take full advantage of the 8400's powerful instruction repertoire and storage efficiency.

The 8400 Linking Relocatable Loader has the ability to map relocation information generated by the MACRO Assembler/FORTRAN IV Compiler into the memory EXEC bits. Using this relocation information in memory, the Standard Monitor DISPLACER provides the facility to move programs physically about without destroying their ability to be executed. Some computers have hardware that "relocates" an instruction as it is being executed. The EAI 8400 moves segments of programs about in memory, updating relative cross-referencing between segments sweeping through memory and changing the links. Sections of program and data storage can be reclaimed during execution of a program, thus increasing available memory space; new sections of coding, of variable or undetermined lengths, can be brought in to replace the old.

Dynamic relocatability is the basis for Dynamic Storage Allocation, the analysis and optimization of storage requirements of programs, both before and during execution.

2.0 8400 MONITOR SYSTEMS

Efficient utilization of a digital computing system is of prime interest in the modern scientific computation laboratory. Experience has shown that setup time" is drastically reduced by the incorporation of standard procedures in the computer's own operation capabilities.

The 8400 Scientific Computing System is equipped with three Monitor Systems which relieve the programmer of the machine dependent aspects of digital computation.

The 8400 Standard Monitor System handles the real-time aspects of the machine such as I/O, interrupt routing, program loading, debugging control and storage allocation. An Operating System includes as well accounting routines and software facilities for scheduling and executing a series of unrelated computer runs without human intervention.

The Simulation Monitor System is a complete entity within the EAI 8400 Monitor System. It utilizes some of the routines described under the Standard Monitor System and has control of a group of programs designed specifically to aid in the operation of either all-digital or hybrid scientific simulation programs.

The HYTRAN Ⓢ Monitor System is another parallel system of programs designed to provide digital computer assistance in the preparation and check-out of analog and hybrid computer programs.

A unique feature of the EAI 8400 monitor systems is the modularity afforded by the exclusive dynamic relocatability of programs. The Executive section of the Standard Monitor is written to occupy absolute locations in a contiguous lower section of memory. The remainder of the Standard Monitor and the Simulation and HYTRAN Monitors are composed of subroutines which are dynamically relocatable and hence may be called in or removed by the Executive. This means that sections are brought into core memory only as required and unneeded sections may be removed. The Monitors can be reorganized automatically to assume a minimum configuration in all cases, thereby optimizing user storage. Even though the Simulation and HYTRAN Monitors and Debug Executive System may be called into play during execution, the limitation on **memory size is a function of only what is needed and how efficiently it was coded.**

The 8400 Monitor Systems and the software they control are as follows:

EAI MONITOR SYSTEMS

STANDARD MONITOR	SIMULATION MONITOR	HYTRAN MONITOR
1. MACRO Assembler 84	1. Linking Loader 84	1. HYTRAN Programs Group
2. FORTRAN IV Compiler 84	2. Debug System 84	a. Analog static check
3. Linking Loader 84	3. Subroutine Library 84	b. Analog report Generator
4. Debug System 84	4. Simulation Programs Group	c. Analog Equipment Check-out
5. Subroutine Library 84	a. Hybrid Mode Control	
	b. Integration Control	
	c. Function Generator Loader	
	d. Hybrid Computer Set-up	
	e. Hybrid Debug	
	f. Hybrid Mnemonic Addressing	

Ⓢ EAI Service Mark

2.1 STANDARD MONITOR SYSTEM 84

The 8400 Standard Monitor consists of an Executive controller and a series of functional packages which perform a variety of tasks as requested by the Executive. The Executive is a non-relocatable (absolute) program which resides in a lower section of memory; the remainder of the Monitor which is dynamically relocatable is called into core when required and removed when no longer needed.

The resident Executive controller occupies a minimum of memory space. Users of 8400 systems with critical memory restrictions may choose to operate without the Standard Monitor. In these cases, the functions of memory and peripheral device storage allocation are handled by the programmer with the aid of the MACRO Assembler, FORTRAN Compiler, Relocatable Loader, and Subroutine Library.

The functions of the resident Executive controller are as follows:

1. Dynamic Storage Allocation at the request of the user or one of the Monitor Systems.

This routine is responsible for loading programs via the Linking Relocatable Loader, assigning memory locations for the program or segments thereof, and designation of input-output devices. The generalized dynamic relocator known as DISPLACER, handles the moving of program segments in memory. The programmer guides the allocation of storage in accordance with the requirements of the problem, but is freed from the necessity of assigning the specific core and peripheral storage layout. The coding of each problem is independent of the particular storage capacity of the 8400 computer being employed and of any other program which might share memory during execution.

The unique ability of the 8400, to displace programs in memory and have them remain executable and to allow symbolic references to memory locations using the symbol table, creates a system in which the programmer need have no knowledge of the specific location of programs or data. The translation of symbolic names into specific memory locations may be made either prior to or during program execution without affecting the user.

2. System Loader

The Standard Monitor System Loader is loaded by the Console Auto Load button; the System Loader in turn loads the remainder of the Monitor and the Linking Relocatable Loader and Debug System.

3. Interrupt Direction for Monitor interrupt routines or user.

4. Basic Housekeeping Routines unique to hardware oriented activities of the 8400:

Save and Restore subprogram
Rapid Access File Control
Internal Interrupt Processing
External Interrupt Routing

5. Console Typewriter Input-Output Executive communication package.

The Standard Monitor Modules perform the following additional functions, as required:

1. Input-Output Routines and scheduling system for assignment, control, and monitoring of peripheral device activity.

This I/O supervisor package calls from the subroutine library those I/O formatting and control routines it requires. Device and channel initialization and control sequences are provided for. All peripheral devices are connected to the 8400 through one of its data channels.

2. Push-Down Stack Control

As discussed in the Relocatable Subroutine Library section, a portion of contiguous upper memory is set aside by the Standard Monitor for storage of in-process data as required by recursive programming. This system routine "STACK" insures that the stack does not interfere with other programs, and handles corrective procedures in the case where the recursive stack becomes temporarily overflowed or erroneously emptied. The stack itself is dynamically relocatable and may be shifted about in core when other operating programs compete for the space.

3. System Dump subprogram for producing relocatable outputs from memory to be re-loaded by the System Loader.

4. COMPAT and EXEC protect interrupt routines.

5. A Control Statement Analyzer which accepts commands from any external device (console typewriter, control card, paper tape, etc.) and performs various functions such as:

   a. Loading other programming systems (i.e. FORTRAN IV, MACRO Assembler, Linking Relocatable Loader). A subroutine library search may be performed at the request of the loader to bring in additional program segments.

   b. Loading other monitors (Simulation or HYTRAN monitors, Debug Executive).

   c. Deleting segments of program and/or data.

2.2 SIMULATION MONITOR SYSTEM 84

The Simulation Monitor is called into the system by the Standard Monitor Executive System Loader and replaces or supplements the relocatable modules of the Standard Monitor System. The Simulation Monitor relieves the programmer of a series of generalized tasks associated with operation and control of

real-time digital or hybrid simulation problems. In so doing, the Simulation Monitor controls the Simulation Programs Group, discussed separately. The primary functions performed are:

1. Hybrid Mode Control
2. Integration Control
3. Function Generator Loader
4. Hybrid Computer Set-Up
5. Hybrid Debug
6. Hybrid Mnemonic Addressing
7. Dual Processor Control
8. Program Loading and Execution
9. Digital Computer Debug

The loading, debugging, and execution of programs is handled in the same way as the Standard Monitor System. The Linking Relocatable Loader and Debug System are called in as required.

The Dual Processing Executive routine of the Simulation Monitor permits the time sharing of the 8400 by the hybrid system and other 8400 users. Between "operate" periods of the hybrid computer, the 8400 may be used for assembly, compile, checkout, or execution of other programs. These operations are interrupted whenever the real-time problem resumes, but continue when the 8400 is again available.

Like the Standard Monitor, the Simulation Monitor is dynamically relocatable. Only those portions of the monitor required for each phase of problem solution are retained in core.

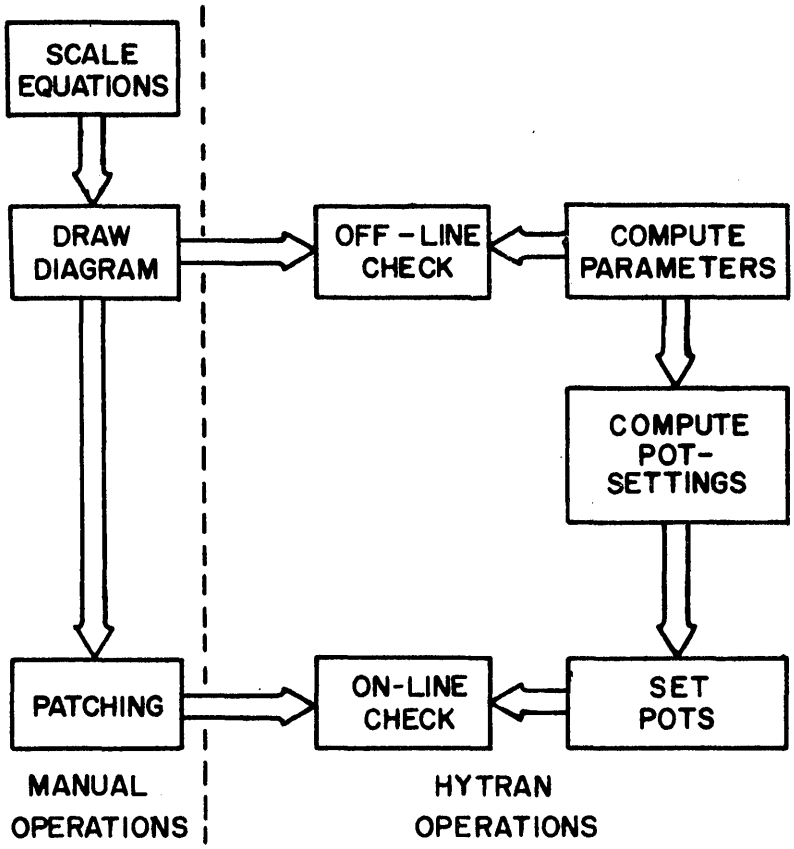
2.3 HYTRAN MONITOR SYSTEM

The HYTRAN Monitor System is called in by the Standard Monitor Executive System Loader on an operator request. It controls the HYTRAN Programs Group, a family of meta-programs to provide digital computer assistance in the programming of an analog system. The scaling of the physical equations and the preparation of the computer diagram are still performed by the programmer who thereby maintains direct control over the analog implementation of the problem. In order to permit calculation of theoretical static check values, HYTRAN also must be given the original problem statement and a set of test initial conditions. Additional data, including patching information, component setting or modes, highest derivatives, and any other expressions representing other appropriate component outputs must be provided as well. In this way, the necessary rapport between the programmer and the machine is kept. Figure 2.3-1 shows the steps required to program an analog program with the HYTRAN System.

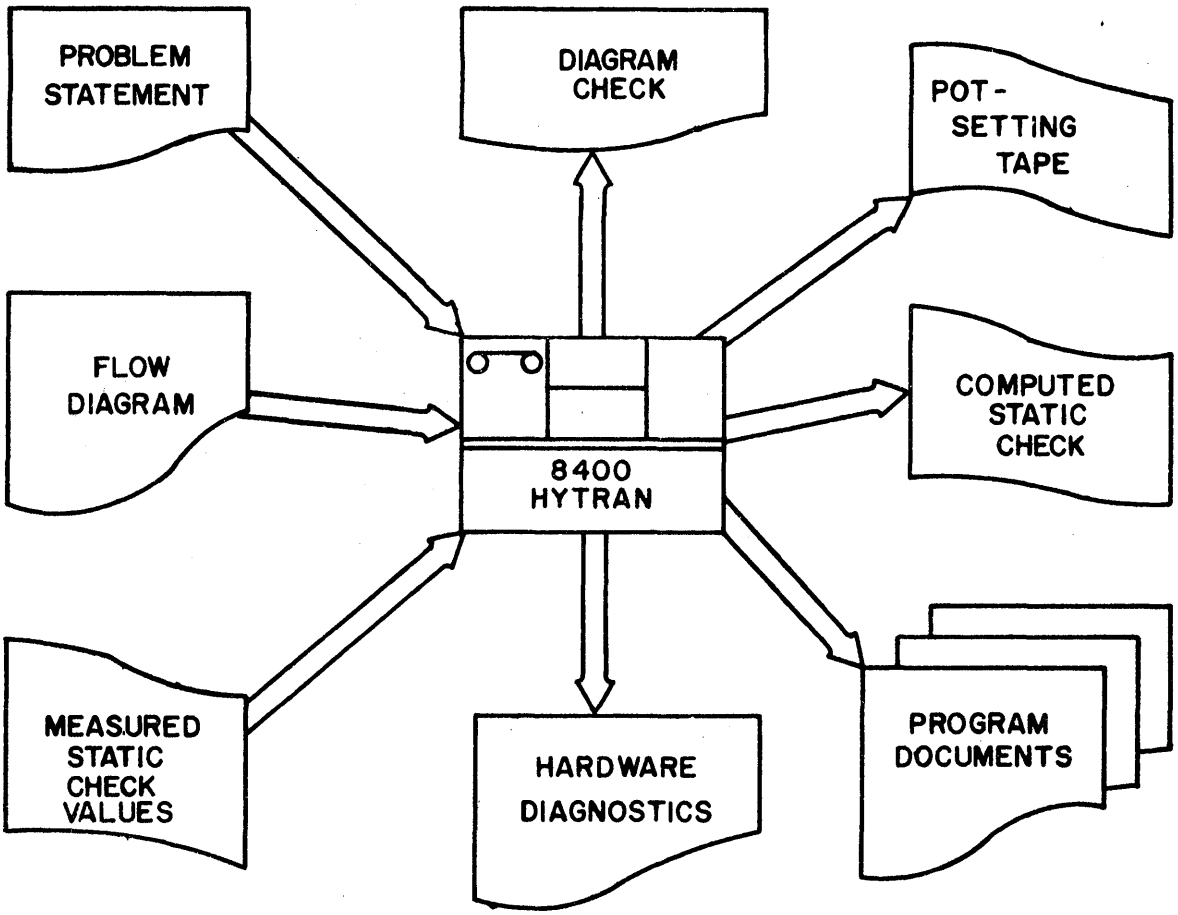
Program input is punched on paper tape in an analog-oriented language compatible with the 8400 Digital Computing System. The HYTRAN System then computes

potentiometer settings, and both a physical and voltage static check which are tested for consistency. Complete documentation of the analog program is produced (including potentiometer, amplifier, and cross-reference sheets as automatic pot-setting and static test tapes). An On-Line Diagnostic Generator checks measured static check voltages against the analog circuit diagram and the specifications of the analog components, providing a rapid means of locating patching errors or component failures. HYTRAN Outputs are shown in Figure 2.3-2.

The HYTRAN Monitor and HYTRAN Programs Group requires an 8400 System with 8K of core memory. With an 8K core, each program can be contained in memory and enough data storage is still available to process an analog computer program requiring three one-hundred and twenty-amplifier systems. In general, there are no special requirements on the analog computing system(s) since the programs cover a wide range of component configurations.



PROGRAMMING AN ANALOG PROBLEM WITH HYTRAN
FIGURE 2.3-1



OUTPUTS OBTAINED FROM THE HYTRAN SYSTEM
FIGURE 2.3-2

3.0 8400 PROGRAM PREPARATION SOFTWARE

3.1 MACRO ASSEMBLER 84

3.1.1 Introduction

The EAI 8400 MACRO Assembler 84, running under the control of the Standard Monitor provides the user with all 8400 machine operations as well as an extensive set of pseudo-operations. The coding procedures are based upon the IBM 7040/7044 MAP and 7090/7094 FAP assemblers to ensure programming familiarity with the techniques involved and the conventions and symbolism employed. Its specifications start with MAP and FAP but go well beyond in taking advantage of the special hardware features of the 8400 and in ensuring suitability for high efficiency and real-time programming.

The assembler is interruptable and can be used in a real-time environment. Included in the assembly language are special features for real-time and hybrid computation such as: ability to call re-entrant subroutines, special TIME and DELAY pseudo-operations for estimation of execution times (which are printed on the output listing), and the generation of real-time delay intervals.

The assembler is modularly designed to permit packaging for any customer's specific configuration of memory size and access equipment. An assembler package contains the basic assembler and has access to the special purpose input, input formatting, output, and output formatting routines contained in the Standard Monitor to tailor the assembler to each system's hardware environment. Any combination of the following peripherals is possible.

Input	Output
Paper Tape	Paper Tape
Punched Cards	Punched Cards
Magnetic Tape	Magnetic Tape
Typewriter	Typewriter
	Line Printer

3.1.2 Characteristics

The EAI MACRO Assembler 84 requires an 8400 System with 8192 words of core memory, and assembles programs in two passes of the source program at a rate of 300-500 statements per minute.

There are more than forty-five Pseudo-Operations available, which allow the programmer to communicate with and control the assembly process. These include: location-counter control, named relocatable common storage control,

data generation options, external and entry relocatable symbol definition, literal pool positioning control, macro definition operations, conditional assembly control, listing control, and binary output format control. Figure 3.1.2-1 lists the Pseudo-Operations provided.

Macro-Operations allow the repetitive insertion in the source program by the assembler of a sequence of instructions either written by the programmer or defined by system conventions (system macros). The macro-definition pseudo-instructions define the macro sequences and control their use. The 8400 Macro Assembler is oriented toward macro-operations and includes provision for iterative sequences with variable parameters.

LOCATION COUNTER
- USE
- ORG (with BEGIN and USE option)
- RESUME

DATA GENERATION
- DATA
- BCI
- ASC
- VFD
- DUP
- BCD

STORAGE ALLOCATION
- COMMON (named; relocatable)
- BSS

SYMBOL DEFINITION
- EQU
- SYN
- SET

LITERAL POOL POSITION
- LITORG

CONDITIONAL ASSEMBLY
- IFT
- IFF

OPERATION DEFINITION
- OPSYN

MACRO-RELATED
- MACRO
- ENDM
- IRP

LINKED RELOCATION CONTROL
- ENTRY
- EXTERN
- LINK
- AUTO (i.e., ABS)
- REL

LISTING CONTROL
- SPACE
- EJECT
- TITLE
- PMC
- INDEX
- UNLIST
- LIST

ASSEMBLY CONTROL
- REM
- END
- MORE
- ARG (i.e., PZE, PSV, MZE, MSV, etc.)

REAL-TIME PROGRAM ASSISTANCE
- SETIME
- UPTIME
- DELAY

BLOCK EXEC BIT SETTING
- EXEC

MACRO ASSEMBLER 84 PSEUDO - OPERATION LISTING

Figure 3.1.2-1

Many programming applications involve a repetition of a sequence of instructions, generally with variations in parameters at each iteration. Using the macro-definition pseudo-operations, a programmer can define this sequence as a macro-operation, indicating in the definition which arguments are variable. The variable arguments can appear in any field of any of the instructions within the sequence. Up to 63 variable arguments are permitted, and each parameter substituted for an argument can be up to 56 characters long.

Macro-operation definitions can be nested. That is, a macro-operation definition can be entirely included within the range of another higher-level macro-operation definition. This is extremely useful for defining new macros or redefining existing ones with a single instruction. There is no significant limit to the depth of nesting allowed.

The MACRO Assembler 84 produces a relocatable binary output compatible with the Standard Monitor, the linking Relocatable Loader, EAI 8400 FORTRAN IV programs, and the Subroutine Library. All provisions for relocation and linking are included in the assembler output. The EXTERN pseudo-operation allows the symbolic names of external subroutines and memory locations to be declared. The ENTRY pseudo-operation allows a program to declare symbolic names within it that can be external names to some other program. The COMMON pseudo-operation allows named blocks of common storage to be set up. As an option, the assembler produces an absolute binary output (AUTO pseudo-operation) in a format suitable for loading under the control of the 8400 Console Auto-Load System.

During assembly, a location counter is used to determine the next location to be assigned to an instruction. The Macro Assembler 84 provides 16 location counters which are controlled by the programmer by using various location counter pseudo-operations. He can use as many of the 16 location counters as desired and transfer control back and forth among them. This allows instructions to be listed in a sequence useful for documentation and loaded into memory in a different sequence for execution.

Symbolic language updating, usually a separate program, is combined into the first pass of the assembler, eliminating a special pass of the program for symbolic correction. The updater is engaged by a sense switch option. Corrections can be given on a tape prepared off-line or by the operator through the on-line typewriter. The number of corrections is limited only by the available memory for the assembler's single process table. Two basic updating operations are provided: omit and insert. Any necessary duplication of correct records prior to the point of omission or insertion is automatically provided.

Limited retrospective correction is possible, allowing the operator to return to a point which has already been processed and start anew from there. The update system provides clerical testing of all statements, and offers the operator the opportunity of making corrections where errors are not known to exist. The updating process provides a corrected symbolic tape as output, and a journal of corrective action on the typewriter.

The assembler produces an assembly listing properly annotated with error and warning indications. The set of error and warning messages is extensive, and designed to give the programmer the maximum amount of information concerning the nature of the irregularity discovered by the assembler. Assembly always proceeds to the end of the program, resulting in complete diagnostics on all source statements. An alphabetically arranged index of symbolic names and corresponding memory locations is optionally printed. Pseudo-operations allow controlling titles and spacing on the listing, or the suppression of the listing process.

Included in the binary-program output is a table of all symbolic names used in the program together with information about the nature of the program word to which the symbol is assigned. This table is optionally loaded into memory at execution time to allow symbolic debugging of the program or a symbolic disassembly dump (discussed in Debug System 84 section).

Flexible means are provided to specify any of the following types of constants: symbolic addresses; octal, hexadecimal, decimal integers; fixed-point and floating point numbers; and BCD, BCI, and ASCII interchange codes.

In addition to the 16-bit half-word literals (immediate operands) permitted by the 8400 hardware addressing system, the assembler allows 32-bit or 64-bit "multiple precision" literals. Such numeric constants, preceded by the equal sign (=) in symbolic machine instructions, are automatically pooled into a constant store with no duplication, and are referenced by proper addresses automatically inserted in the machine instructions.

Any symbol which is undefined within the program is automatically assigned a memory location within this literal pool. The programmer can often correct the oversight by placing in the assigned location a constant or transfer instruction that sends control to the proper point in the program. Automatic assignment also means that programmers do not have to define temporary storage for one word quantities, since the assembler will do this automatically.

The EQU pseudo-operation is designed so that symbols used in it do not have to be previously defined in the program. This often is a common restriction of other assembly programs giving rise to many errors.

System Organization

The modular construction of the assembler and its relationship to the Standard Monitor I/O package provide wide flexibility in tailoring the assembler to the 8400 main frame and available peripheral equipment. The incorporation of input/output control and formatting routines in the Standard Monitor simplifies the assembler and FORTRAN IV Compiler, and standardizes I/O transfer operations. The entire basic assembler is organized into separable packages. Any package can be replaced or modified to change or improve its operation without affect on other packages. Each pseudo-operation analyzer is independent, allowing new analyzers to be easily incorporated as the need for them arises. System macro routines may be added as the need for them arises.

To allow users flexibility in the use of the assembler, certain "configuring" constants are held in known locations. These can be changed by an assembler correction process, or by the Standard Monitor, to handle different memory sizes, data channel assignments, tape handler assignments, and similar other parameters.

The assembler uses only one process table whose space is shared by program symbols, operation codes, numeric literals, update information, macro skeletons, and macro parameter lists. This allows memory to be fully utilized according to the needs of the specific program with no arbitrarily assigned space lying in disuse. The algorithms used to construct the process table allow for maximum speed of insert and lookup operations, often faster than would be a binary search.

The size of the assembly program is kept to a minimum by designing analyzer and scan functions so that they are used in both assembly passes. Full use is made of table-driven techniques which take full advantage of the powerful 8400 instruction repertoire, especially the logical operations. It is estimated that about 30% space saving in the assembler results from these techniques and the power of the instruction set.

Every major package in the assembler contains a data word specifying the release date and revision number of the package which serves as a check that the most up-to-date versions have been incorporated into the assembler.

The assembler is designed to be operated in a real-time environment and may be interrupted at any time during the assembly process. All coding sequences employed in the assembler itself are self-initializing to ensure correct continuation when control is returned after interruption.

Off-Line Symbolic Assembler and Updater

EAI has developed an 8400 Simulator and Assembler known as the Phantom Assembly Program (PHAP). It is designed to run on the Pacific Data Systems (EAI Subsidiary) PDS 1020 Digital Computer (4K version), which is an inexpensive stored program machine. It operates in three passes, the first of which is merged with an updating program. An error report is produced during the first pass and the output assembly language listing and 8400 Auto Load object tape during the second pass. The normal limit to the number of symbols per assembly is 256, although this can be increased to 384 by an assembler revision. Many assembly language programs can be corrected and assembled off-line with this auxiliary computer.

3.1.3 Coding Procedures

Symbolic Instruction Format

The commonly accepted IBM 7040/7044 MAP and 7090/7094 FAP source language format is used, consisting of three parts:

1. The Label Field (columns 1-6) contains the definition of a symbolic address. Up to six alphanumeric characters are allowed. This field may be left blank.

2. The Operation Field (columns 8-14) contains the mnemonic machine operation codes, control pseudo-operation codes, or programmer macro-operation codes. For any machine instructions an asterisk (*) can appear in this field, indicating indirect addressing.

3. The Variable Field (columns 16-72) contains expressions for assigning storage addresses, index registers, count fields (if applicable), and EXEC bit control settings for 8400 machine instructions, or suitable expressions for pseudo-operations. Subfields are separated by commas. An equal sign (=) before an address subfield indicates immediate (operand) addressing.

The termination of the variable field is signalled either by reaching column 72, or by finding three consecutive blank characters. Any information beyond the termination of the variable field is treated as a programmer comment.

Up to two blank characters can be interspersed anywhere in the Variable Field to improve readability. (Blank characters anywhere in the Label Field and Operation Field are ignored.)

The variable field for any type of information (except a binary-coded character constant) can be continued starting in column 16 of the next card (i.e., record). This is signalled by the presence of a left parenthesis followed by three blank characters, or a combination of left parenthesis followed by two, one, or no blanks ending in column 72. There is no limit to the number of continuations that can be made.

Any statement with an asterisk (*) in column 1 is treated entirely as a programmer comment and is included on the output listing.

Expressions

The programmer writes expressions to represent the subfields of the variable field of a symbolic instruction. Expressions may be used in the address, index register, and count portions of the variable field. Expressions are also used in certain pseudo-instructions.

The smallest component of an expression is an element, which is a single symbol or a single integer less than 2^{16}. The asterisk (*) is a special element, defined to mean the address of the instruction in which the asterisk appears ("here").

A term consists of one or more single elements, connected by multiplication and division operators:

* (multiplication)

// (division)

An expression consists of one or more single terms connected by addition and subtraction operators, or logical operands (AND, OR, exclusive OR)

\+ (addition)

\- (subtraction)

** (logical AND)

++ (logical OR)

-- (logical EOR)

Expressions for addresses can contain a slash (/) to indicate which half of the word is desired. SAM/ indicates left half whereas /SAM indicates the right half of the word at symbolic location, SAM.

The expressions ** or .. are commonly used to designate a field whose value will be computed and inserted by the program. The expression ** is relocatable and of zero value. The expression .. is absolute and of zero value.

Data Items

Data Items are defined using the DATA pseudo-operation. They may also be used in literal form in a symbolic instruction if preceded by an equal sign (=).

A data description containing a slash (/) defines the contents of each half (16 bits) of a data word. The portion of the description to the left of the slash defines the contents of the left half of the word. The portion to the right of the slash defines the contents of the right half of the data word. Thus, the description 2/7 results in a data word as follows: +00002+00007. If no slash is used, a full word is defined. A slash is not permitted if the literal is preceded by an equal sign.

Decimal Integer is a string of digits from 0 to 9 which may be preceded by a plus or minus sign. The maximum value for a full word is $2^{30} - 1$. Integer binary scaling is assumed in order to produce a number less than unity.

Value	Decimal Integer	Internal Octal
$-3x2^{-15}$ / $27x2^{-15}$	-3/27	-77775+00033
$271x2^{-30}$	271	+00000+00417

Decimal Fixed-Point Number is a string of digits from 0 to 9, which may be preceded by a plus or minus sign, and may be written with or without a decimal point, and is always followed by a scale indicator, B, followed by a signed or unsigned integer. The maximum length for a full word is 10 digits. The scale factor, after B, specifies the location of the implicit binary point within the data-word. The value of each number must be less than unity.

Value	Decimal Fixed Point Format	Internal Octal
$7.5x2^{-3}$ / $6x2^{-9}$	7.5B3/6B9	+74000+00600
$-7x2^{-16}$	-7B16	-77774-40000

Decimal Floating-Point Number is a string of digits from 0 to 9, which may be preceded by a plus or minus sign, and may be written with or without a decimal point, and is usually followed by an exponent indicator, E or EE, followed by a signed or unsigned integer. The E indicates single-precision, and EE double-precision. The resulting binary number is normalized (i.e., minimum binary scaling is assumed in order to produce a number less than unity). The exponent is a power of ten by which the number is multiplied during conversion.

The scale indicator, B, followed by a signed or unsigned integer can be included to override the assumption of minimum binary scaling (i.e., to give a scale for an unnormalized number). If no B appears in the number and there is a decimal point, the E or EE can be omitted.

The maximum length for a double-precision number is 16 digits.

Value	Decimal Floating Point
$2.675x2^{-2}$*	2.675
$2.675x2^{-2}$	.2675E1
$2.675x2^{-2}$	2675E-3
$2.675x2^{-7}$	26.75E-1B7
$2.675x2^{-2}$	267.5EE-2

*The minimum (normalized) binary scale of 2.675 is 2^{-2}.

Octal Integer is indicated by an apostrophe (') followed by a plus, minus or no sign, then by a string of digits from 0 to 7. Maximum size is 12 characters including sign. The sign indicates the setting of the high-order bit of the data-word or half-word and has no algebraic meaning. However, if the apostrophe (') is preceded by a minus sign, the two's-complement is formed.

Octal Integer	Internal Octal
'17/'-7766	+00017-07766
'-7766	-00000+07766
/-'12	+00000-77766

Hexadecimal Integer is indicated by a double apostrophe (") followed by a string of digits from 0 to 9, or letters A to F (representing 10 to 15) or a combination of these. Maximum size is 8 characters. If the double apostrophe (") is preceded by a minus sign, the two's complement is formed.

Hexadecimal Integer	Internal Hexadecimal
"A7/"2A	00A7002A
"A72A	0000A72A
/-"2A7F	0000D581

Binary-coded Character Constants of four or more 8-bit characters are formed using the BCI and ASC pseudo-operations. This kind of constant of one to four characters can also be indicated by preceding the constant with an apostrophe (') and following the constant with an apostrophe ('). Blank characters are added on the right to fill out a data-word or half-word. Truncation, if necessary, occurs on the right.

Binary-coded Characters	Internal (b=blank character)
'A'/'OB'	AbOB
'ABC'	ABCb
/'ABCD'	OOAB (with warning)

Symbolic Constants are names of memory locations in a program. The relocation mode of a data-word or half-word is determined by the relocation mode of the symbol. Right-half word relocation causes a warning flag. Negative relocation is an error.

Certain reserved symbols (of the form .xxxxxx) represent system constants for input, output, and executive communication. These are recognized by the assembler.

Let SAM be relocatable octal 2776, and .LA82 be a system constant.

Symbolic Constant	Internal Octal
SAM/	+02776+00000
SAM	+00000+02776 (with warning)
-SAM/	+00000+00000 (with error flag)
.LA82/	appropriate system constant is inserted.

Mixed Data Definitions

In definitions containing a slash (/) the definition of the right-half word is completely independent of the definition of the left-half. Some examples are shown:

```
SAM+2/'OK'

-'-2766/56B9

2/"A7C2
```

EXEC Bit Setting

The assembler can cause EXEC bits to be set in the binary output.

To indicate an EXEC bit setting, the programmer writes one or the other of the system symbols, .E or .NE, followed by a slash (/), followed by one or the other of the system symbols, .E or .NE.

.E means "set the EXEC bit", .NE means "do not set the EXEC bit"; the slash (/) separates the spectification of left and right EXEC bits of the word.

EXEC specification	EXEC bit setting LEFT	RIGHT
.NE/.NE	0	0 (assumed if no EXEC specification is given)
.NE/.E	0	1
.E/.NE	1	0
.E/.E	1	1

EXEC bits can be set for an entire block using the EXEC pseudo-operation.

3.2 FORTRAN IV COMPILER 84 SYSTEM

The EAI 8400 FORTRAN IV System running under control of the Standard Monitor will accept and interpret the IBM 7090/7094 FORTRAN IV Language (See IBM Systems Reference Library) which is a compatible subset of EAI FORTRAN IV 84. The proposed ASA Standard FORTRAN IV Language (Comm. ACM, Oct. 1964) is also a subset of the 8400 FORTRAN IV.

3.2.1 Introduction

The 8400 FORTRAN IV requires an 8400 System with 8192 words of core memory.

The minimal I/O equipment configuration consists of paper tape reader, paper tape punch, and an on-line typewriter. Additional access devices which may be used are as follows:

Input	Output
Card Reader	Line Printer
Magnetic Tape Transport	Card Punch
	Magnetic Tape Transport

EAI FORTRAN IV 84 is designed to compile programs of up to 2000 source statements in one pass at a rate of 300-500 statements per minute. Internal Load-and-Go capability under control of the Standard Monitor will be available on machines having more than 8K of memory.

The compiler is interruptable and can be used in a real-time environment. Included in the language are special features for real-time and hybrid computation such as: recursive subroutine library, compilation of user-defined recursive subroutines, complete mixed-mode capability in expressions and assignment statements, extended logical operations implemented by means of Boolean connectives, TIME and DELAY statements for estimation of execution times and generation of real-time delay intervals, and in-line assembly language type statements which make all special 8400 hardware instructions available to the FORTRAN IV programmer.

3.2.2 Characteristics

The sizes of all constants allowed on the 8400 FORTRAN are identical to those employed on 7090/7094 FORTRAN; truncation is performed by the compiler as necessary. In accordance with standard 8400 word lengths, the following constants are used:

Integer - 1 to 5 decimal digits
Extended (Double Precision) Integer - 1 to 10 decimal digits (Not available on 7090/7094 FORTRAN IV)
Real (Standard precision Floating-Point) - 1 to 8 significant decimal digits
Double Precision (Floating-Point) - 1 to 16 significant decimal digits
Complex (ordered pair of Real constants)
Logical - True or False

Source Statements permissible in 8400 FORTRAN IV include 7090/7094 compatible Input/Output control and Format statements (Figure 3.2.2-1).

The 8400 FORTRAN IV System has features that increase the suitability of FORTRAN programming for Real-Time problem solving and take advantage of the special hardware capabilities of the 8400.

1. Mixed Expressions Integers and logical variables will be automatically floated when mixed with Real and Complex variables; Extended Integers will be floated by open subroutine. Free intermixing of Extended, Integer, and Logical variables takes place without conversion because of the word formats in the 8400. In the evaluation of relationships using the logical operators, "Greater Than", "Less Than", etc., full mixed mode capability is allowed between the Integers, Extended Integers and Logicals with the Real forms. In Arithmetic Statements a comprehensive mixed mode capability is allowed in replacement since Logical values are easily converted to Real and Real to Complex. In Figure 3.2.2-2 below, Y indicates a valid statement, N shows an invalid statement.

(See page 3-15 for Figure)

EAI FORTRAN IV SOURCE STATEMENTS
Fig.3.2.2-1

ARITHMETIC
- Arithmetic Assignment Statement
- Logical Assignment Statement

CONTROL
- Unconditional GO TO
- Computed GO TO
- Assigned GO TO
- ASSIGN
- Arithmetic IF
- Logical IF
- DO
- CONTINUE
- CALL
- RETURN
- STOP
- PAUSE
- SET TIME
- UPDATE TIME
- DELAY
- SAVE
- RESTORE

INPUT-OUTPUT
- FORMAT
- READ
- PRINT
- PUNCH
- WRITE
- ENDFILE
- REWIND
- BACKSPACE
- ACCEPT
- TYPE

DECLARATIONS
- SUBROUTINE
- FUNCTION
- Arithmetic Function Definition Statement
- BLOCK DATA
- DATA
- INTEGER
- REAL
- COMPLEX
- LOGICAL
- DOUBLE PRECISION
- DIMENSION
- COMMON
- NAMED COMMON
- EQUIVALENCE
- EXTERNAL

Right Side of Equal Sign

Left Side of Equal Sign

Variable \ Expression	Real	Integer	Extended	Complex	Double Precision	Logical
Real	Y	Y	Y	N	Y	Y*
Integer	Y	Y	Y	N	Y	Y*
Extended	Y	Y	Y	N	Y	Y*
Complex	Y*	Y*	Y*	Y	Y*	Y*
Double-Precision	Y	Y	Y	N	Y	Y*
Logical	Y*	Y*	Y*	N	Y*	Y

* - Compiler flags statement as "mixed-mode"

Fig. 3.2.2-2

2. Unlimited number of dimensions in arrays of subscripted variables:

3. Floating point incrementing in DO loops.

4. Improved input-output conversion capabilities
   (1) A method for specifying Hollerith strings without counting is included.
   (2) On data input from paper tape, commas may be used for terminating the data field to eliminate unnecessary punching of leading blanks.

5. Expressions for control values, I/O units and DO parameters.

6. Extended logical operations, implemented by means of Boolean arithmetic, with provisions for octal constants. Definition of signed octal constants is provided through a special format. Signed octal strings may be used in integer definitions. Example:

```
I = '7765          (integer)
E = '7723411177 (extended integer)
```

This facilitates the creation of masks for Boolean operations. Other logical operators such as EOR (exclusive or), EQV (equivalence), IMP (implication), NAND, NOR, are provided since these instructions are part of the 8400 repertoire.

7. A symbol table can be output as part of the FORTRAN compilation. This table is used in conjunction with disassembly and symbolic debugging features being provided the user. A symbol table "edit and dump" post-compilation program is used to produce the symbol table.

In addition to symbolic debugging (see Debug System 84 section), the 8400 FORTRAN IV provides a special optional in-line "TRACE" mode. Here the user

specifies that all computed variables (i.e., FORTRAN, any variable found on the left hand side of an equation) be output at execution time in a format equivalent to:

VARIABLE = (value in special format)

This option is chosen during compilation and extra instructions are generated in the translation process to cause these printouts. At execution time the output can be suppressed by sense switch control, but the instructions remain.

8. The 8400 allows arithmetic operations to take place on 30 bit signed integers and they will be allowed to be declared. These are "double precision" integers and are declared as, EXTENDED INTEGER.

9. The insertion of a full complement of valid mnemonic machine coded sequences between FORTRAN statements is permitted. These in-line assembly language instructions are flagged by the programmer by an "S" preceding the statement and are translated by an assembler program in the compiler.

10. Each compiled statement or block of statements will output on the listing, a calculated time estimate based on fixed rules of the execution time for that statement or block. SET TIME, UPDATE TIME, and DELAY statements are included for hybrid computational processes which require program timing. A timing block is defined to be those statements bounded by SET TIME and/or UPDATE TIME statements. The DELAY statement is used to synchronize the program to real time. These statements are equivalent to those provided in the Symbolic Macro Assembler for the same purpose.

Other features of the 8400 FORTRAN IV System are: FORTRAN generated assembly language output listing. Dump, Partial Dump and Trace debugging at execution time and Link capability for "chain" processing.

3.2.3 FORTRAN System Organization

The EAI FORTRAN IV Compiler 84 System is a complete integrated programming system consisting of:

Compiler
Relocatable Loader
Object Time Package
Subprogram Library

These systems completely equip the EAI 8400 to compile, load and execute FORTRAN IV programs in real time.

The Compiler

The conversion from source program to relocatable binary object program is accomplished in one pass by the compiler. A main program and any number of subprograms may be compiled in sequence without compiler reloading. The object programs are then processed by the loader. Required library subprograms are loaded. The object time package is called into memory, and execution is initiated.

During compilation, statements found to be in error are discarded. Both syntax errors, such as missing parentheses and semantic errors, such as misused identifiers, are noted. An error message consists of the statement in its original form with the erroneous phrase or character undermarked by a $ sign. This is followed by a comment indicating the type of error. Illegally nested DO loops, undefined or multiple-defined statement numbers, and memory allocation conflicts are summarized at the end of the program listing.

Compilation always proceeds to the end of the program, resulting in complete diagnostics. In general, the diagnostics are superior to those of commonly used compilers.

The Relocatable Loader

The loader places the compiler output into memory in a form suitable for execution. The FORTRAN IV Relocatable Loader is identical to the Relocatable Loader 84 (discussed separately), thus ensuring loading compatability of compiler and assembler object codes.

Object Time Package

The object time package provides the computer with the capability to execute object programs. It includes all routines of compiler origin such as double precision arithmetic routines, and input-output conversion and format scanning routines.

Subprogram Library

The standard Intrinsic and External Functions indicated in Fig. 3.2.2-3 and Fig. 3.2.2-4 are included in the subprogram library. The External Functions are written in FORTRAN IV and will be compiled to produce a library tape for subsequent object program executions. The Intrinsic Functions are written in EAI 8400 Assembly Language and may be used independent of the FORTRAN IV.

Because a program may be used in a real-time environment where subprograms may operate under interrupt control, the subroutines for the library allow recursive entry if they can be loaded only once by the FORTRAN Loader. The user has the option of loading recursive subroutines (once) and their non-recursive written counterparts (more than once) by an appropriate notation in the compiler statement which alerts the loader.

3.2.4 System Design

The 8400 FORTRAN IV compiler is designed to be operated in a true multi-level priority interrupt processing environment. By utilizing a general method for saving the registers, temps, and intermediate results of the object time package, the compiler is able to have control taken from it and correctly resume when control is returned. All subroutines used within

the translator itself are self-initializing - that is, the first time they are entered any necessary presetting of switches, instructions, temporaries, etc. is performed. No-time-dependent (non-interruptible) sets of instructions are included.

Console switch settings may be changed at any time. All binary tapes consist of check summed blocks. Reading routines check all tape dependent stores to prevent destruction of good information and to allow full recovery from reader failures.

The entire system is designed to be tolerant of input-output and operator errors.

All I/O will be taken care of in closed subroutines incorporated in the Standard Monitor 84, thereby making FORTRAN programs independent of Access Device configurations.

Intrinsic Function	Symbolic Name	Type of	
		Agrument	Function
Absolute Value	ABS	Real	Real
	IABS	Integer	Integer
	DABS	Double Real	Double Real
	CABS	Complex	Real
	DCABS	Double Complex	Double Real
Truncation	AINT	Real	Real
	INT	Real	Integer
	IDINT	Double Real	Integer
Remaindering	AMOD	Real	Real
	MOD	Integer	Integer
	DMOD	Double Real	Double Real
Choosing Largest Value	AMAXO	Integer	Real
	AMAX1	Real	Real
	MAXO	Integer	Integer
	MAX1	Real	Integer
	DMAX1	Double Real	Double Real
Choosing Smallest Value	AMINO	Integer	Real
	AMIN1	Real	Real
	MINO	Integer	Integer
	MIN1	Real	Integer
	DMIN1	Double Real	Double Real
Float	FLOAT	Integer	Real
Fix	IFIX	Real	Integer
Transfer of Sign	SIGN	Real	Real
	ISIGN	Integer	Integer
	DSIGN	Double Real	Double Real
Positive Difference	DIM	Real	Real
	IDIM	Integer	Integer
Obtain Most Significant Part of Double Precision Argument	SNGL	Double Real	Real
	CSNOL	Double Complex	Complex

FORTRAN IV INTRINSIC FUNCTIONS

FIG. 3.2.2-3

Intrinsic Function	Symbolic Name	Type of	
		Argument	Function
Obtain Real Part of Complex Argument	REAL DREAL	Complex Double Complex	Real Double Real
Obtain Imaginary Part of Complex Argument	AIMAG DIMAG	Complex Double Complex	Real Double Real
Express Single Precision Argument in Double Precision Form	DBLE CDBLE	Real Complex	Double Real Double Complex
Express Two Real Arguments in Complex Form	CMPLX DCMPLX	Real Double Real	Complex Double Complex
Obtain Conjugate of a Complex Argument	CONJG DCONJG	Complex Double Complex	Complex Double Complex

FIG. 3.2.2-3 (cont.)

External Function	Symbolic Name	Type of	
		Argument	Function
Exponential	EXP	Real	Real
	DEXP	Double Real	Double Real
	CEXP	Complex	Complex
	DCEXP	Double Complex	Double Complex
Common Logarithm	ALOG10	Real	Real
	DLOG10	Double Real	Double Real
Natural Logarithm	ALOG	Real	Real
	DLOG	Double Real	Double Real
	CLOG	Complex	Complex
	DCLOG	Double Complex	Double Complex
Trigonometric Sine	SIN	Real	Real
	DSIN	Double Real	Double Real
	CSIN	Complex	Complex
	DCSIN	Double Complex	Double Complex
Trigonometric Cosine	COS	Real	Real
	DCOS	Double Real	Double Real
	CCOS	Complex	Complex
	DCCOS	Double Complex	Double Complex
Hyperbolic Tangent	TANH	Real	Real
Square Root	SQRT	Real	Real
	DSQRT	Double Real	Double Real
	CSQRT	Complex	Complex
	DCSQRT	Double Complex	Double Complex
Arctangent	ATAN	Real	Real
	DATAN	Double Real	Double Real
	ATAN2	Real	Real
	DATAN2	Double Real	Double Real

FORTRAN IV EXTERNAL FUNCTIONS

FIG. 3.2.2-4

4.0 8400 PROGRAM LOADING AND RELOCATION SOFTWARE

The normal program loading sequence for the 8400 is, as follows:

1. Standard Monitor Executive (including System Loader) is put into core memory by the Console Auto Load system.

2. The System Loader then loads the remainder of the Standard Monitor (or other monitor system) and the Linking Relocatable Loader 84.

3. The Linking Relocatable Loader performs the basic operations of loading relocatable MACRO Assember and/or FORTRAN IV object (binary output) taped programs in memory, scanning the subroutine library tape automatically for standard subroutines, and creating linkages between the subroutines and the main program. The Relocatable Loader includes the option of mapping the relocation bits into the EXEC bits to maintain relocatability after the program has entered core memory.

4.1 AUTO LOAD/DUMP SYSTEM

The 8400 Console includes a hardware Auto Load/Dump system for loading or dumping relocatable or non-relocatable binary object programs to or from absolute memory positions.

The starting address for the operation is entered from the Console I/O typewriter. The system then loads or unloads consecutive memory cells until a stop code is detected in the data (Auto Load only), the Console Halt switch is depressed, or all memory locations are filled or empty (used in absolute reload operation and core dump).

At the option of the programmer, EXEC bits may be loaded or unloaded along with full data-word transfers. Relocation information previously mapped into these bits will allow program dynamic relocation by the Standard Monitor Displacer routine after initial reloading by the Auto Load System.

Any peripheral device connected to the system may be selected for Auto Load/Dump operations. No bootstrap loading sequence is required ahead of the loaded program in this all-hardware system.

4.2 LINKING RELOCATABLE LOADER 84

The Linking Relocatable Loader is entered into memory by the System Loader in response to an operator control directive. It relocates and integrates programs, subroutines, and sub-programs, into one object program in memory.

It determines the sub-program required for object program execution, searches the subroutine library for these sub-programs, and loads them into memory together with the necessary linkages, and prints or types out a memory map of core assignments. Provisions are made for multiple naming of subroutines and library functions to allow multiple loading of the same sub-program.

The binary machine language output produced by the MACRO Assembler or FORTRAN IV Compiler may be loaded as several segmented programs. The Relocatable Loader will allocate their respective storage automatically filling in missing references between the segments. The Loader is aware of the contents of memory at the time that a new program or subroutine is to be loaded. The Loader determines the first memory location that is unused and automatically relocates the new program to that position, with proper adjustments to it so that it will operate properly there. The Loader accepts a specially devised input format which determines which addresses are relocatable, which symbols cross-reference, how the EXEC bits are to be handled and verifies correct loading by recalculating logical check-sums stored on the external medium.

Approximately 1000 words of memory are used by the Relocatable Loader itself; when loading is complete, control returns to the monitor which dynamically relocates the object program to overlay the Loader, so equivalent memory space is made available for buffer areas and temporary storage.

The Loader handles a variety of object programs:

1. The object program is relocatable and is placed in memory, using space assigned by the programmer or the Loader itself. A program's being "relocatable" implies that upon loading, certain half words will be algebraically adjusted upwards (or downwards) in memory by a fixed amount determined by the Loader as part of its function. A "relocation bit" for each half word is included in the MACRO Assembler and FORTRAN IV Compiler binary output for each instruction, indicating if the address field should be modified during relocation. Relocatable object coding produced by the assembler and compiler consists of an OP code followed by a pointer to various produced tables (constants, etc.); once the memory of an item in the tables is established by the loader, the pointer is replaced by the address of that item.

The 8400 MACRO Assembly language and FORTRAN IV is written so that the user follows simple rules to get such a formatted output. Relocatable assemblies enable many programs which otherwise would interfere in storage allocation to be stacked end-to-end in memory. Furthermore, the programmer is allowed to symbolically "link" together sub-programs by cross-referencing the symbolic locations of one segment with those of another. The actual lines of communication are set up by Linking Relocatable Loader which not only relocates but cements the program segments as loading takes place.

2. The object program is non-relocatable and includes specific memory assignments. Such "absolute" programs may be loaded using the Console Auto Load hardware.

3. Relocation information bits recorded on the external medium may be mapped into storage EXEC bits permitting dynamic relocatability - the ability to move programs about in storage. EXEC bit assignments for other than relocation information; i.e., memory protection or object time breakpoint debugging, may also be set in the object code stream. The mapping of relocation bits is exercised by the following codes:

00	no mapping
01	Map into right EXEC only
10	Map into left EXEC only
11	Map into both

The normal mode is left only. Here the right EXEC bit, assigned for some control purpose, appears in storage together with a relocation bit mapped into the left EXEC.

The symbol table which may be included in the object code is also relocatable. Symbol cross-references to memory addresses are updated by the standard Monitor DISPLACER routine during dynamic relocation.

5.0 PROGRAM CHECKOUT SOFTWARE - DEBUG SYSTEM 84

5.1 GENERAL

The Debug System 84 provides the 8400 user with a powerful and human-engineered set of tools for bringing an untested computer program into full operational capability. The package will enable him to examine, modify, dump, and test instructions, program segments, and data in a flexible and convenient way. Two notable features are the ability to use symbols from the symbolic version of a program in stating debugging commands, and the ability to insert instructions into a code sequence by program displacement without an elaborate patching procedure. In debugging language the most frequently-performed operations are expressed simply and concisely, yet the flexibility and power of each is extensive.

There are four classes of debugging aids: Examine/Modify, Load/Dump, Breakpoint/Trace, and Monitor/Control.

The system consists of a Debug Executive Program and a collection of independent closed debug subroutines performing specific debugging functions. The Debug Executive enables the 8400 programmer to use the on-line typewriter for such functions as creating a program, inserting and deleting coding, adding to a symbol table, executing a program, inserting breakpoints, opening a cell or special register, etc.

The Debug Executive is entered into memory by one of the control monitors in response to an operator directive. The Debug Executive in turn loads the subordinate routines requested.

5.2 SYSTEM OPERATION

In a typical operation, the debugging library would be stored on some external medium (i.e., a magnetic or paper tape). The programmer, having loaded his program and the Debug Executive program, would proceed to set-up the run for debugging by entering a series of instructions through the typewriter (or by pre-punched paper tape, control cards) which would insert debug subroutine calling sequences using program displacement.

The control directives indicate the debug function, the address range over which the function is to operate, and the point in the object program at which the function is to be called into play (if not implied by the address range). Generally for closed shop installations, all directives are pre-punched and are stored in memory for execution when indicated. Otherwise, directives are given via the console I/O typewriter, and control returns to the typewriter after a function is completed. The operator can intervene at any time by depressing the Console interrupt button, and can indicate certain routine options using the Console Register sense switches.

Calling sequences for Debug subroutine packages may be either:

1. Explicit Commands - On-line insertion and execution of the debugging control directive during program run time directly from one Input Device, or,

2. Implicit Commands - Debugging sequences are inserted directly in the coding before execution. The Debut Executive will prepare the debug calling routine and execute it at the appropriate time.

Examples of debugging functions using both types of commands are:

1. Implicit - Assembly coding entries, dumps, traces, register status, return to console control, set debugging mode.

2. Explicit - Load/Unload (program), insert patch, erase, replace (implicit coding), mark/unmark (Exec bits) search, start and other control commands (implicit and explicit both) from typewriter, card, paper tape or any other input medium as long as the proper debug formats are used.

If the "required" debugging subroutines are unavailable when called, an error complement results. Erroneously stated debugging requests are spotted by diagnostic aids.

Automatic debugging is achieved by pre-setting debugging sequences within a program just prior to execution time using the various functions provided. During execution, modifications can be made by direct interrupt or by having set a point to return control to the debugging system in the main program. Certain alternate debugging modes are controllable internally or overriden by console switch settings, - the trace, dump, and breakpoint (Exec) functions particularly.

The debugging commands make use of indicators that state the format in which output response is desired. Within an individual debugging command the output mode may be altered, and this alteration can be made to apply either for all subsequent commands or for just the command being processed. In symbolic format, the output is in the form of instructions, provided that the instruction decoding subroutine is present in memory. Symbolic addresses are used where appropriate if the program symbol table is available in core.

5.3 ORGANIZATION

The Debug System 84 Executive program will be stored in memory at the high end; a set of interlinked user programs will be stored above it. Each program will consist of a control segment, the body of the program proper, and a symbol table, if desired. The control segment specifies the name of the program, the starting location of the body of the program, the starting location of the symbol table, and the starting location of the next control segment. Each entry in the symbol table occupies two

words, and specifies the name of the symbol, its value, its type, and certain information about how it is used in the program. The standard monitor controls the location of each program and its relationships to its neighbors.

A section of memory is reserved by the Standard Monitor for a programmed push-down stack where the debug functions are executed.

The debug subroutines when in memory are organized in a parallel fashion to the problem routines described above.

5.4 SYMBOLIC DEBUGGING

Symbolic debugging implies that variables are referred to at execution time by the names assigned to them during assembly or compilation. In most computers all such information is lost when the object programs are loaded. The loader provides a memory map showing the name and territorial limits of each program loaded. Other symbolic information, however, is contained only in the compiler/assembly listing and cross-reference symbol table printout and is not loaded.

In the 8400, symbolic cross-reference information produced by the loader assembler and compiler is optionally introduced into memory along with the associated programs. The Debug Executive is capable of accessing both data and data names, enabling the programmer to take advantage of the mnemonic facilities provided in automatic programming systems during execution of his program. He need not be aware of absolute addresses during debugging just as he was not fully concerned with the exact assignment of addresses during program preparation by the assembler or compiler.

The symbol table is continuously updated as symbols are modified and when programs are dynamically relocated.

5.5 DEBUGGING FUNCTIONS

1. Examine/Modify Class - are used to examine the contents of a single cell or register, several consecutive cells, or segments of memory; modify the contents of a cell or group of cells; search, or zero blocks of memory; delete or introduce one or more data-words or instructions. A single command in many cases performs several of these functions at once.

There are two types of searchs, both of which use a mask register and a criterion register. The "ones" in the mask register specify the bits to be compared; if all the masked bits of the cell being tested agree with the corresponding bits of the criterion register, the cell meets the criterion. On one search, all cells that meet the criterion are printed out; on the other search, those that do not meet the criterion are printed out.

The deletion or insertion of data-words, instruction, or debugging cell sequences into a program has in the past called for leaving gaps in the coding or patching in additional coding using jump instructions. In the 8400 this important function of instruction correction is accomplished by direct insertion. Using direct insertion, one or several instructions are inserted into the program at the point indicated, after the original program body has been dynamically relocated to provide the space. Likewise, removal of an instruction or section of coding is followed by a dynamic relocation of the program to fill the gap. This capability is exercised by a program which forms a part of the 8400 Standard Monitor, called the "Displacer" and uses the EXEC bit system. The dynamic relocatability feature - the ability to move programs around in memory at will - is one of the most important software features of the 8400. Ha! a dud

2. Load/Dump Class - provides three types of dump and a relocatable load..

There are three types of dump within the Debug System 84: a symbolic disassembly of a single program, a relocatable dump of a single program, and a relocatable systems dump of all of memory.

If the symbol table produced during the program assembly or compilation has been loaded together with the object program, a symbolic disassembly dump can be made. The symbolic disassembly provides printing of a stored program in symbolic, or Assembler-84, language for analysis or record keeping of the current symbolic listing of a modified program. Each cell of the program is decoded into an instruction in Macro Assembly language format. All flag bits and modifiers appear in the same format as is used for assembly input. The operation part is generated by the Debug System's operation decoded. The address part is expressed in symbolic form, using the symbol table for cross-referencing.

The relocatable dump normally outputs computer words to paper copy or to magnetic tape; the format is the same as the output of the assembler; thus the dumped program can be reloaded by the standard loader. The relocatable dump includes the symbol table, if it is present in memory. The relocatable dump can reduce the number of program reassemblies that may be required for a difficult program.

The system dump of all of memory provides a means of saving the state of the machine as rapidly as possible, and reloading without requiring the facilities of the relocatable loader. This dump contains a literal copy of the contents of memory, including EXEC bits, symbol table, stack, and other cross referencing information.

3. Breakpoint/Trace Class - provides the capability for inspecting the status of a program during execution.

Breakpoints may be inserted in a program by one of two means: debugging control commands may be included in-line on the original source program listing, indicating those points where the program will halt and an output of arithmetic and control registers is to be made. Alternatively, the EXEC bit control system may be used. The breakpoint supervisor routine receives control whenever a cell appropriately marked by a high EXEC bit is about to be executed or a store is about to take place into marked cell. Control commands are used to set or remove EXEC markings and to establish breakpoint counts (program interrupt does not occur until a break has occured the specified number of times). The breakpoint supervisor can distinguish between breakpoint and other kinds of interrupts.

Program Tracing causes a readout of pertinent registers after execution of each instruction. Print-out may be suppressed by sense switch. Tracing of selected segments of coding is permitted.

4. Monitor/Control - provides a comprehensive set of commands to inspect a program and control the debugging process. The following are representative debug commands for this purpose:

Execute any instruction
Print-out any data-word in one of many formats
Search memory for instructions with a specified effective address
Transfer control to a specified address and continue program execution
Output symbolic name of octal address requested with the program name
Delete named program
Remove symbol table of named program
Proceed following a breakpoint halt
Establish new program names (used to program subroutines on-line)

6.0 RELOCATABLE SUBROUTINE LIBRARY 84

6.1 GENERAL

The Relocatable Subroutine Library 84 is compatible with the MACRO Assembler, FORTRAN IV Compiler, Linking Relocatable Loader, and Monitor Systems. The routines of the library are called by pseudo-instructions in the assembler and compiler and are linked to the main program during loading by the Linking Loader 84. The input/output control and formatting subroutines form a portion of the Standard Monitor System and are called in as appropriate when the storage allocation requirements of a program have been established.

Two forms of subroutines are provided - one for re-entrant (recursive) operations and a faster non-re-entrant version. Re-entrant subroutines are the generalization of recursive ones. A single subprogram may be entered by two or more calls at the "same time"; that is, a program using a re-entrant subroutine, is interrupted and the interrupt program proceeds to use this very same subroutine.

When a re-entrant subroutine is interrupted, the working registers and temporary core memory locations which are unique to the particular call on the subroutine in progress (such as the return address) are safe-stored in a section of core reserved by the Standard Monitor for this purpose. This memory area is a software implementation of a push-down stack - a portion of contiguous memory together with a pointer which references the "top most" cell in the stack. Index Register seven is used exclusively for the pointer. This technique will allow the same routine to be interrupted at many levels and used by higher levels and yet allow each level to restore its temporary cells and complete the routine at the completion of the higher interrupt. The re-entrant versions of the subroutine library include the save-restore routines required for "recursive" programming in a real-time multi-programming environment.

A program in the Standard Monitor is provided to update or edit the Subroutine Library. Sections or individual programs may be added, deleted, modified or listed through typewriter input instructions from the operator.

The routines included in the library are:

6.2 ARITHMETIC SUBROUTINES - SINGLE AND DOUBLE PRECISION FIXED AND FLOATING-POINT

1. Sine)
2. Cosine) - radian input
3. Tangent)

4. Arctangent - radian output
5. Logarithm - natural and common
6. Exponential
7. Square Root

6.3 MATHEMATICAL SUBROUTINES

1. Function Generation - One, two, or three variables. Fixed and Floating -Point, using linear interpolation between variable breakpoints

   8400 subroutines employ fixed spacing of data values to provide maximum execution speed. Any desired accuracy can be achieved by appropriate selection of "breakpoint" spacing. When several functions of the same variables are to be evaluated the execution time of the second and subsequent function is less than for the first, by the time required to process the arguments. The table size limitations are appropriate for a large class of functions, and are imposed in order to optimize the execution time.

   The pseudo-op specifies the data tables location and one, two, or three arguments. The tables may be loaded with pre-formatted data by the regular loading routine, or the Function Generator Loading Program in the Simulation Monitor System may be used to calculate the table from arbitrary input data.

   The Fixed-Point Subroutine may be used in a hybrid computer program where 12-14 bit arguments are derived from the analog computer in fixed point format. When a two variable function is called for, the data table is given by a two dimensional array, up to 32 x 32 values. A three variable function is given by a 32 x 16 x 8 (or smaller) array.

   The Floating-Point subroutine is a general purpose routine for one, two and three variable functions; data is stored as 32 bit floating point numbers. Up to 256 breakpoints may be selected for any variable. The range and scaling of the independent variable is handled automatically by the 8 bit exponent of the argument.

2. Digital Integration - Several numerical integration algorithms with automatic time scaling (step size changing):

   Euler, Henn, Runge-Kutta, Milne, Parabolic Predictor-Corrector, Adams (four different types).

   The Integrator Control Program of the Simulation Monitor System provides ready access to the several integration routines. Upon typewritten request for a different integration algorithm this program will make the proper linkages and calculate any required starting

values for the new routine, permitting the programmer to choose the method exhibiting the best speed and accuracy for his particular problem.

6.4 CONVERSION SUBROUTINES

1. Data Conversion - Single and Double Precision: BCD to Binary, Binary to BCD

2. Format Conversion - Single and Double Precision: Fixed-Point to Floating-Point, Floating-Point to Fixed-Point

3. Character Converstion:

   IBM BCI (6-bit Alphanumeric) to ASCII (7-bit alphanumeric), ASCII to IBM BCI, ASCII to EBCDIC, EBCDIC to ASCII

6.5 INPUT/OUTPUT AND DATA DISPLAY SUBROUTINES

1. Interrupt System - Line Sorting, Priority Masking
2. Status Line - Sorting
3. Peripheral Device Control and Data Formatting - Typewriter I/O, punched paper tape I/O, punched card I/O, Line printer output, magnetic tape I/O labelled multi-file, multi-reel, with users own coding options), Display Register Control (CRT Display monitors, etc.).

6.6 COMPAT MODE SUBROUTINES

Programmed operators for Double Precision Floating-Point, Extended Precision Fixed-Point, Index Register classes of arithmetic. These routines may be called by the MACRO Assembler, FORTRAN IV Compiler or hardware COMPAT interrupt system.

7.0 SIMULATION PROGRAMS GROUP

In addition to the Linking Relocatable Loader 84, Debug System 84, and Relocatable Subroutine Library 84, the Simulation Monitor controls a group of simulation programs, as follows:

7.1 HYBRID MODE CONTROL

This is the program that gives the digital computer the structure of a simulator in like manner to an analog console by exercising interface capabilities (including interrupts, sense lines, delays, and mode control) to achieve real-time synchronization between the stored and the parallel computer programs, and between the various computing system elements needed in the solution of most simulation problems. The program controls a dual-processing mode by which it is possible to work on a second program located in a protected portion of memory when the computing system is not employed on the primary simulation program.

(The monitor permits dual processing when the simulation program is not in the "Operate" mode.)

7.2 INTEGRATION CONTROL

The Digital Integration Control Program is one that is requested from the monitor via the console I/O Typewriter by the operator to change the integration algorithm. Selection of Algorithms is made from the Subroutine library; initialization and time scale changes can be made also.

7.3 FUNCTION GENERATOR LOADER

The Digital Function Generator Loading Program permits the operator to load new data or to make changes in data previously stored in tables associated with the particular function generator subroutine.

7.4 HYBRID COMPUTER SET-UP

The Hybrid Computer Set-up and Check Out Program, selected by typewriter through the monitor, provides the capability for control of analog computer mode and time scale selection, component selection, read-out, pot set and checkout. Sample commands to the Analog I/O Computer are, as follows:

Control

a. OP - Operate)
IC - Initial Condition) mode control
HLD - Hold) commands for the
ST - Static Test) analog computer
PS - Pot Set)

b. CS - Select Analog Console for Hybrid Operation.

c. LTAB - Load table of addresses to Analog I/O Computer.

d. LTAD - Load table with data to Analog I/O Computer.

e. SETT - Set table of potentiometers in table, output results; if setting differs by more than the selected percentage, print errors.

f. SET - Set individual potentiometer selected and output results; indicate error if setting differs by more than selected percentage.

Monitor

a. STCK - Check column of data vs. state of analog computer.

b. SCAN - Readout component values selected.

c. DTAB - Dump complete table selected, for set-up and static check at a later time.

d. CHK - Check selected amplifier output for specified voltage to required accuracy.

e. RD - Read output of selected multiplier.

7.5 HYBRID DEBUG

The Hybrid Debug program provides for typewriter control of the linkage and interface system.

Diagnostic tests of the Analog and Interface subsystems are conducted from the 8400 Console.

8.0 HYTRAN® PROGRAMS GROUP

The HYTRAN Monitor controls the following group of programs:

8.1 STATIC CHECK

The practice of computing two independent sets of check values has been used as the basis for the HYTRAN Off-Line Static Check. The theoretical static-check values in volts are computed from expressions, provided as part of the program input, which specify component outputs in terms of the scale factors, parameters, and variables of the problem. Further input defines the analog component interconnections of patching information which is used to calculate the voltage check values. In this computation, all input voltages to a component, the kind of input to which the component is connected, and the transfer function of the component are used to determine its voltage output.

When both voltage and theoretical values are available for a component output they are compared. If in agreement, they yield the off-line static check value for that component. If the values are not in agreement, the error is isolated by retaining the theoretical value as the static check input for all subsequent calculations and an error message is given. Values exceeding the voltage range of the computer also will cause error messages but will be retained for further static check calculations.

The On-Line Static Check

While in systems without digital access to the analog computer, the on-line static check must be performed by manual comparison, the availability of a digital input-output system provides the HYTRAN user with a choice of two automatic procedures for such on-line checking. One method is to feed the HYTRAN-generated static-check tape into the Analog I/O Computer to obtain an automatic comparison between calculated and measured values. This method is used whenever the 8400 is not available at the time of analog on-line check.

If the 8400 computer is available, the use of the second HYTRAN method allows an improved consistency check of debugging complex problems as well as for preventive-maintenance checks. This method is implemented by feeding a paper tape (generated on the Analog I/O computer) containing the measured pot settings and/or outputs of all components into the digital computer. HYTRAN then checks the transfer value of each individual component and compares it with the measured voltage at the component

output. Thus, errors can not propagate but are pin-pointed at the component level.

8.2 REPORT GENERATOR

This is a Documenting Program which sorts and converts the information from the intermediate tape to component work sheets that contain a list of the analog computing components in an orderly sequence, together with their modes and their outputs or settings in terms of problem parameters, variables, and scale factors. In addition, an alphabetic list of the values of parameters and variables is typed out, and Analog I/O Computer tapes are generated.

One tape contains the potentiometer settings in a format which allows automatic pot setting; the other tape contains the computer static-check values for on-line check with the Analog I/O Computer.

The Documenting Program also generates a cross-reference sheet containing an alphabetic list of symbols for parameters and variables. Each of these symbols is followed by a listing of the components whose output or settings include that symbol. This feature provides assistance in changing parameters or scale factors manually.

8.3 EQUIPMENT CHECK-OUT

This is an On-Line Diagnostic Program utilizing basically the same processing procedures as that of the Off-Line Static Check Generator, described as in 8.1 above, in the HYTRAN Monitor System. The program accepts an Analog I/O Computer tape containing a complete read-out of all static-check voltages and potentiometer settings and generate diagnostics which serve to locate analog components which are either improperly patched or do not perform satisfactorily. Thus, a permanently-programmed analog computer pre-patch panel can be used with this program for daily preventive maintenance checks.

9.0 DIAGNOSTIC SYSTEM

The 8400 Diagnostic System produces a go-no-go indication of the machine, and attempts to analyze machine faults to assist in computer maintenance. Thorough tests are performed on all instructions and program controlled machine functions. The tests can be executed without operator intervention or he can specify tests and their order of execution. For preventative maintenance, tests can be run with the machine subjected to variations in clock and voltage margins. The 8400 Diagnostic System also tests all peripheral equipment.

Machine diagnostic routines include the following:

1. Memory Module

   a. Addressing tests that test the integrity of each address.
   b. Worst case pattern sensitivity tests on logic and driving circuits.

2. Floating-Point Processor, Control Section

   a. All instructions are tested with fixed data in a logical sequence that aids in localizing faults.
   b. All instruction options are tested.
   c. Pattern sensitivity tests are made on all internal control registers and data transfer paths by generating all possible bit patterns and register state transitions.
   d. All internal interrupts are tested by generating the interrupt conditions and checking for the correct hardware transfer.

3. Floating-Point Processor, Arithmetic Section

   a. Complete control and data transfer path tests using fixed and randomly generated data.
   b. Noise and pattern sensitivity tests using fixed and random data.
   c. Complete check on the integrity of results and arithmetic flags.

4. Exchange Module

   a. All modes of data assembly and transmission are checked.
   b. Pattern sensitivity tests using random data.
   c. Full load tests using varying combinations of available peripheral.

5. Peripheral Equipment

   a. Check basic execution of instructions, i.e., Magnetic tape read, write forward and reverse etc., with fixed data; typewriter, output of all possible characters and manual input with visual verification of output, etc.

b. Stressing tests - i.e., high speed read/write or randomly generated characters and block lengths.
c. Maximum load tests to check for no interaction between peripherals.