EAI®

SCIENTIFIC COMPUTATION

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

8400 SCIENTIFIC COMPUTING SYSTEM

REFERENCE HANDBOOK

Publ. No. 00 800 9049-0

CONTENTS

	Page
CHAPTER 1 - SYSTEM DESCRIPTION	1-1
1.1 INTRODUCTION	1-1
1.2 EXPANSIONS	1-1
1.2.1 8402 Basic Computing System	1-3
1.2.2 8403 Basic Computing System	1-3
1.2.3 8410 Central Processor	1-3
1.2.4 8420 Memory Module	1-4
1.2.5 8430 Exchange Module	1-4
1.2.6 8440 Desk Console	1-4
1.2.7 8490 Power Module	1-4
1.3 PROCESSOR	1-6
1.3.1 Memory Word	1-6
1.3.2 Instruction Word	1-7
1.3.3 Data Word Formats	1-7
1.3.4 Processor Registers	1-10
1.4 ADDRESSING	1-12
1.4.1 Direct Addressing	1-12
1.4.2 Indexed Addressing	1-13
1.4.3 Indirect Addressing	1-13
1.4.4 Immediate Addressing	1-13
1.5 INTERRUPT SYSTEM	1-13
CHAPTER 2 - INSTRUCTION REPERTOIRE	2-1
2.1 INTRODUCTION	2-1
2.2 EFFECTIVE ADDRESS CALCULATIONS	2-1
2.2.1 Direct Addressing	2-1
2.2.2 Indexing	2-1
2.2.3 Indirect Addressing	2-2
2.2.4 Summary	2-2
2.2.5 Combinations of Addressing Options	2-4
2.3 ARITHMETIC INSTRUCTIONS	2-6
2.4 NOTATION	2-6
2.4.1 Addressing Conventions	2-7
2.4.2 Register Conventions	2-7
2.5 THE FIXED POINT INSTRUCTION CLASS	2-8
2.5.1 The Save Register	2-8
2.5.2 The Accumulator Address	2-9
2.6 THE EXTENDED PRECISION INSTRUCTION CLASS	2-9
2.7 THE INDEX INSTRUCTION CLASS	2-10
2.8 FLOATING POINT INSTRUCTION CLASS	2-11

	Page
2.8.1 Floating Divide	2-12
2.8.2 Floating Multiply	2-13
2.9 THE DOUBLE PRECISION INSTRUCTION CLASS	2-13
2.10 THE INTEGER INSTRUCTION CLASS	2-14
2.10.1 Floating	2-15
2.10.2 Integerizing	2-15
2.11 BOOLEAN CONNECTIVE INSTRUCTIONS	2-16
2.11.1 The Mnemonics	2-17
2.11.2 Addressing	2-18
2.12 CONDITIONAL INSTRUCTIONS	2-21
2.12.1 The Flag Operations	2-21
2.12.2 Index Jumps XJ, XJT	2-22
2.13 INSTRUCTIONS TO LOAD AND STORE SPECIAL REGISTERS	2-22
2.13.1 Load Register or Bus	2-22
2.13.2 Store from Register or Bus	2-23
2.13.3 The Flag Register	2-23
2.13.4 Location Counter	2-23
2.13.5 Timer	2-24
2.13.6 Mask Register	2-24
2.13.7 Console Register	2-24
2.14 EXEC BIT INSTRUCTIONS	2-24
2.14.1 Exec Bit Controls	2-24
2.14.2 Accumulator Exec Bits	2-25
2.15 INPUT/OUTPUT INSTRUCTIONS	2-25
2.15.1 SFL Instruction (Set Function Line)	2-25
2.15.2 TSL Instruction (Test Status Line)	2-27
2.15.3 LDCD, STCD Instructions	2-27
2.15.4 LDCC, STCC Instructions	2-28
2.15.5 LDOB, STIB Instructions	2-29
2.16 SHIFT, ROTATE AND NORMALIZE INSTRUCTIONS	2-29
2.16.1 Arithmetic Shift	2-29
2.16.2 Rotates	2-30
CHAPTER 3 - PRIORITY INTERRUPT SYSTEM	3-1
3.1 INTRODUCTION	3-1
3.2 BASIC OPERATION	3-1
3.3 PRIORITY	3-1
3.4 INTERRUPT CONTROL	3-3
3.5 MASKING	3-5
3.6 USER/MONITOR MODE AND THE INTERNAL INTERRUPTS	3-6
3.6.1 User/Monitor Modes	3-6
3.6.2 Internal Interrupts	3-7

Page

3.7 EXTERNAL INTERRUPTS 3-10
3.8 CONSOLE INDICATORS 3-10

CHAPTER 4 - INPUT/OUTPUT SYSTEM 4-1
4.1 INTRODUCTION 4-1
4.2 DATA CHANNELS 4-1
4.2.1 Function 4-1
4.2.2 Structure 4-2
4.2.3 Instructions 4-4
4.2.4 Programming 4-6
4.2.5 Byte Assembly/Disassembly 4-7
4.2.6 Code Conversion 4-8
4.3 AUTOMATIC DATA CHANNEL PROCESSOR 4-8
4.3.1 Function 4-8
4.3.2 Structure 4-8
4.3.3 Control Words 4-8
4.3.4 Operation 4-10
4.4 SYSTEM INTERFACE 4-11
4.4.1 Function 4-11
4.4.2 Structure 4-12
4.4.3 SFL/TSL Instructions 4-13
4.5 PERIPHERAL DEVICES 4-14
4.5.1 Typewriter 4-14
4.5.2 Card Reader (Models 8452, 8453, and 8454) 4-16
4.5.3 Paper Tape Reader 4-20
4.5.4 Paper Tape Punch 4-21
4.5.5 Line Printer (Models 8461, 8462, and 8463) 4-22
4.5.6 Card Punch (Models 8455 and 8456) 4-26

CHAPTER 5 - COMPUTER CONSOLE OPERATIONS 5-1
5.1 INTRODUCTION 5-1
5.2 CONTROLS AND INDICATORS 5-2
5.2.1 Register Controls 5-2
5.2.2 Typewriter Input Controls 5-3
5.2.3 Exponent Fault 5-5
5.2.4 Interrupt Indicators 5-5
5.2.5 Channel Condition Indicators 5-5
5.2.6 Parity Indicators 5-6
5.2.7 System Flag Indicators 5-6
5.2.8 Programmer Flag Controls and Indicators 5-6
5.2.9 Console Interrupt Controls and Indicators 5-6
5.2.10 Configuration Switches 5-6
5.2.11 AUTO LOAD and AUTO DUMP 5-6

		Page
5.2.12	Clock Controls	5-7
5.2.13	System Controls and Indicators	5-7
5.2.14	Console Register	5-7
5.3	CONSOLE DISPLAY	5-7
5.3.1	Accumulator	5-7
5.3.2	Display Register	5-7
5.3.3	Memory Data	5-8
5.3.4	Memory Address	5-8
5.3.5	Exchange Assembly	5-8
5.3.6	Location Counter	5-8
5.3.7	Channel Function	5-8
5.3.8	Channel Buffer	5-9
5.3.9	Instruction	5-9
5.3.10	Typewriter Input	5-9
5.4	Maintenance Panel	5-9
5.4.1	Lamp Test	5-9
5.4.2	Keyboard	5-9
5.4.3	Clock Control	5-9
5.4.4	Mode	5-10
5.4.5	Left Half, Right Half, Left Exec, and Right Exec	5-10
5.4.6	PC0, PC1, PC2, and PC3	5-10
5.4.7	Data Test	5-10
5.4.8	ERR (Error)	5-10
5.4.9	Bank Select	5-11
5.4.10	Pattern Control	5-11
5.4.11	Memory - LD/NORM/UNLD	5-11
5.4.12	Clock - STEP/NORM/START	5-11
5.4.13	Channel Select	5-11
5.4.14	Device Select	5-11
5.4.15	Byte 4/8	5-12
5.4.16	E BIT E/E	5-12
5.4.17	Code-BIN/BCD	5-12
5.4.18	DBC0, DBC1, DBC2	5-12
5.4.19	DSC0, DSC1, DSC2	5-12
5.4.20	CSC0, CSC1, CSC2	5-12
5.4.21	C1C0, C1C1	5-12
5.4.22	CC0 Through CC4	5-12
APPENDIX 1 - WORD FORMATS		A1-1
1.	INSTRUCTIONS	A1-1
2.	LOGICAL DATA	A1-1
3.	FIXED POINT FRACTIONS	A1-1

Page

4. FLOATING POINT NUMBERS A1-2
5. INTEGERS A1-2
6. ALPHANUMERIC DATA A1-2
7. GENERALIZED DATA A1-3

APPENDIX 2 8400 INSTRUCTION AND TEST MNEMONICS A2-1

APPENDIX 3 - TABLE OF INTERRUPT ADDRESS CODES A3-1
1. ANALOG-TO-DIGITAL CONVERSIONS A3-2
2. OPERATION CODES FOR ANALOG MONITOR/CONTROL A3-2

APPENDIX 4 - TABLE OF SFL/TSL CODES A4-1
1. PROCESSOR INTERRUPT SFL A4-1
2. PROCESSOR INTERRUPT TSL A4-1
3. EXCHANGE INTERRUPT SFL A4-2
4. EXCHANGE INTERRUPT TSL A4-2
5. HYBRID SFL's A4-2

APPENDIX 5 - CHARACTER CODE EQUIVALENCE TABLE A5-1

APPENDIX 6 - POWERS OF TWO A6-1

APPENDIX 7 - OCTAL-DECIMAL INTEGER CONVERSION A7-1

APPENDIX 8 - HOLLERITH CARD CODES A8-1

APPENDIX 9 - LINKING LOADER TEXT BINARY CARD FORMAT A9-1

APPENDIX 10 - PAPER TAPE FORMAT A10-1

APPENDIX 11 - TWO'S COMPLEMENT ARITHMETIC A11-1
1. THE TWO'S COMPLEMENTS SYSTEM A11-1
2. RANGE OF NUMBERS A11-1
3. TRUNCATION AND ROUND-OFF A11-1
4. SHIFTS A11-3
5. OVERFLOWS A11-4
6. MULTIPLE PRECISION A11-4

APPENDIX 12 . . . . . A12-1

ILLUSTRATIONS

Figure Number	Title	Page
1-1	Typical 8400 Scientific Computing System	1-2
1-2	8400 System Diagrams	1-3
1-3	Memory Word Format	1-6
1-4	Instruction Word Format	1-7
1-5	Summary of 8400 Word Format	1-8
1-6	Processor Registers	1-10
1-7	Universal Accumulator Formats	1-11
2-1	8400 Registers	2-2
2-2	Effective Address Calculation	2-5
3-1	Interrupt Register Mask Enable Configuration	3-2
3-2	Interrupt States	3-4
3-3	Multi-Level Interrupts	3-4
3-4	Internal Interrupt Conditions	3-7
3-5	Console Interrupt Buttons	3-9
4-1	Exchange Module	4-1
4-2	The Elements of a Data Channel	4-3
4-3	Data Channel SFL Instructions	4-5
4-4	Data Channel TSL Instructions	4-6
4-5	Program-Controlled Data Transfer	4-7
4-6	Byte Size/Byte Count Variation	4-8
4-7	Channel Control Word Format	4-9
4-8	ADCP Action for a TCD Operation	4-11
4-9	Typewriter Keyboard	4-15
4-10	Connection of Typewriter to the Channel Buffer Register	4-15
4-11	Typewriter Character Position in Memory	4-15
4-12	Typewriter SFL Codes	4-15
4-13	Hollerith-BCD Code on a Card	4-17
4-14	Position of Binary Card Characters in an 8-bit Byte	4-17
4-15	Card Reader TSL Codes	4-18
4-16	Card Reader SFL Codes	4-19
4-17	Paper Tape Reader SFL Instructions	4-21
4-18	Paper Tape Punch SFL Instructions	4-22
4-19	Vertical Format Codes	4-24
4-20	Line Printer TSL Instructions	4-25
4-21	Line Printer SFL Instructions	4-25
4-22	Card Punch SFL Instructions	4-28
4-23	Card Punch TSL Instructions	4-29

Figure Number	*Title*	*Page*
4-24	*Magnetic Tape SFL Instruction*	*4-30*
4-25	*Magnetic Tape TSL Instruction*	*4-31*
5-1	*Control Console*	*5-1*
5-2	*Control Panel*	*5-2*
5-3	*Register Display/Input-Output Typewriter*	*5-3*
5-4	*Paper Tape Reader and Maintenance Panel*	*5-4*
5-5	*System Control Panel*	*5-4*
5-6	*System Display Panel*	*5-8*
5-7	*Maintenance Control Panel*	*5-10*

CHAPTER 1

SYSTEM DESCRIPTION

1.1 INTRODUCTION

The 8400 Scientific Computing System (Figure 1-1) Organization is made up of three autonomous subsystems; memory, processor, and exchange, which operate together from one control. Figure 1-2 illustrates, in block diagram form, a typical 8400 Scientific Computing System.

The memory consists of from one to four banks with individual controls. Each bank has four storage access channels for multiple access communication. In the typical configuration shown in Figure 1-2, the first channel of each bank is connected to a bus from the processor. Another separate bus ties together the second channel of each bank. This bus is connected to each optional Automatic Data Channel Controller used. The banks' third and fourth channels are available for bus connection to external processors and mass memory devices. With this arrangement, the banks can be accessed in an overlapped fashion by the central processor and by the external processors in an expanded multiprocessor system. Each bank can also exchange information with external mass memory devices for efficient time-sharing processes. With this configuration, the processors may continue computation during input/output activity.

The central processor, functioning as the heart of the system, has two-way communication with all subsystems and optional Automatic Data Channel Controllers. Provided with a complete capability of performing all required arithmetic and logical operations, it performs the major part in control and execution of the stored program. The achievement is accomplished by an accumulator and an extensive complement of registers, control lines, and logic circuitry. In general, the basic items of this subsystem are: logic signal control, status lines, function lines, interrupt lines, special control registers, location counter, interval timer control, instruction register, flag register, high-speed save register, and seven index registers including the Universal Accumulator.

The last subsystem, the exchange, consists of a data channel control system and a system interface. The data channel control system provides a fully buffered interface with external input/output (I/O) devices. It includes up to eight two-way data channels which can be controlled by either the program or one of the optional Automatic Data Channel Controllers. Each channel has the capability of controlling up to fifteen external devices. The system interface includes an I/O bus system that is directly addressable, as well as provision for status control lines, function control lines, and external interrupt lines; as required for the integration of hybrid or other systems with the 8400.

Two additional units, the Automatic Data Channel Controller and the console, are shown separately in the diagram. The first of these, an optional expansion in the exchange, provides data channel control for data transfer (independent of processor operation) between external devices and memory. The console, which is considered as part of the processor, includes: system controls, register displays, an on-line typewriter and a paper tape station.

1.2 EXPANSIONS

One important aspect of the 8400 is the versatility of configurations. The expansions may be factory installed or added in the field when the 8400 users' requirements change. A complete listing of expansions may be found in Section 1.1 of the *8400 Maintenance Series - System Information* manual (EAI Publication Number 00 800 9002-0). This section provides

Figure 1-1. Typical 8400 Scientific Computing System

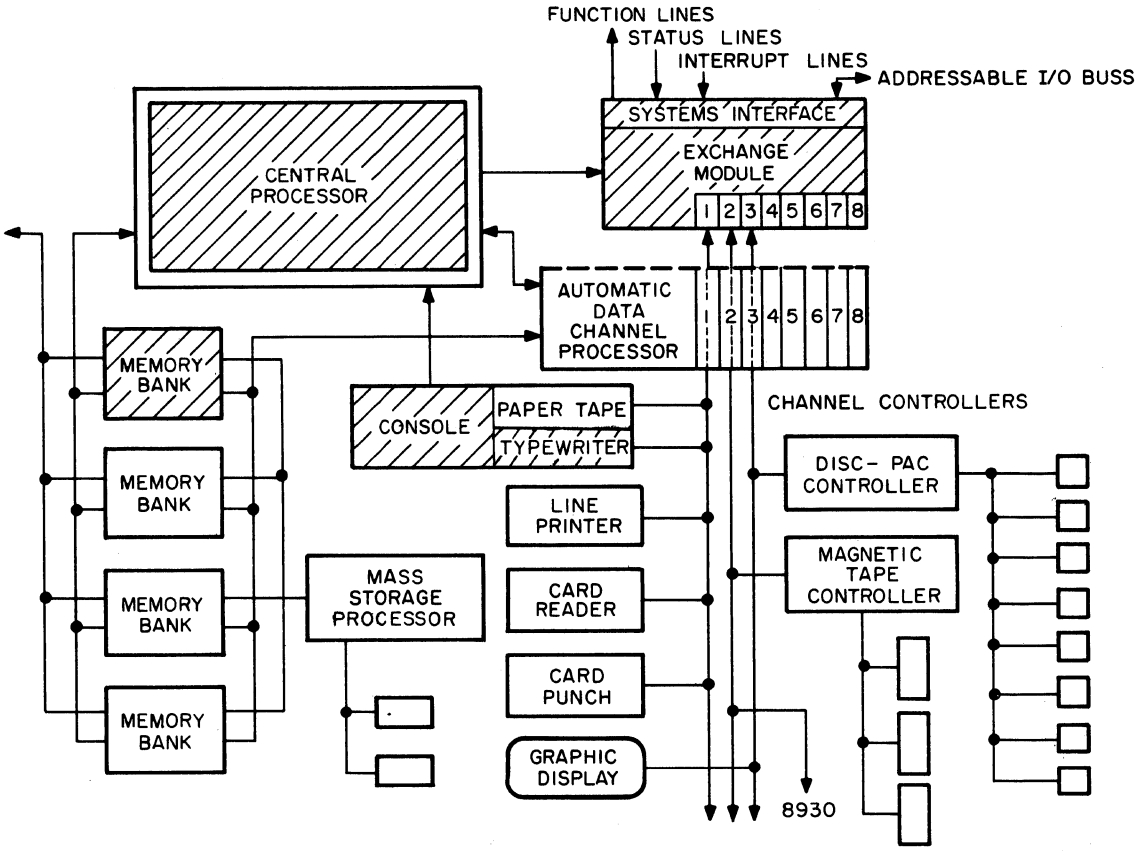


Figure 1-2. 8400 System Diagrams

a brief outline of the basic systems (8402, 8403) along with the standard and optional components available. Figure 1-2 illustrates the various configurations.

1.2.1 8402 Basic Computing System - Includes the following:

8410 Floating Point Processor;

8420 Memory Module - 8K capacity;

8430 Exchange Module;

8440 Desk Console;

8490 Power Module.

1.2.2 8403 Basic Computing System - Same as the 8402 System except that the memory module has a 16K storage capacity.

1.2.3 8410 Central Processor - Including:

Hardware for performing fixed and floating point arithmetic;

Three high-speed index registers and four index registers in core†;

Masked priority interrupt system with 16 internal and 16 external levels;

Power fail-safe system;

Exchange module (8430) with one (8431) data channel, one (8440) desk console with on-line input-output typewriter and one (8490) power system;

Indirect, immediate and byte addressing capability; and, SAVE register with 560 nanosecond cycle time.

---

† *Four, high-speed index registers referred to as the Quad Index Register Pak may be optionally deleted. However, the system always has seven index registers. After the deletion is made, the system index registers include the accumulator, two high-speed registers and four registers in core memory.*

Masked priority interrupt system with 16 internal and 16 external levels; and

Power fail-safe system tied to the highest interrupt level.

1.2.4 8420 Memory Module - Includes:

Core storage capacity of 8192 words, each containing 32 bits for information, 2 EXEC bits for special control functions and 2 parity bits;

A 650 nanosecond access time;

A 1.75 microsecond cycle time;

Independent read/write control enabling overlapped operations with other memory banks; and,

The capability for handling independent busses from up to four request sources.

Maximum one unit to be added in field.

1.2.5 8430 Exchange Module - Includes:

A channel control system that can accommodate up to eight 8431 data channels;

Two bi-directional buffered data channels each capable of handling 16-bit parallel communication;

The capability for handling 15 device controllers per channel;

The capability for controlling 16, 8 and 4-bit byte assembly or disassembly sequences, including parity checking or generation as well as conversion of BCD to processor collating codes;

The capability for independent channel control from the processor or from the optional, 8435-1 Automatic Data Channel Control System (permitting simultaneous multi-channel operation);

An availability for four channel interrupt lines when less than five 8431 data channels are used;

A systems interface with up to 16, fully-buffered, 16-bit parallel input/output busses - up to 128 groups of status lines with 8 lines per group - and up to 128 groups of function lines with 8 lines per group; and,

Interface terminations for optional external priority interrupt system expansions, for up to 256 interrupt levels.

1.2.6 8440 Desk Console - Including:

Operator's panel with complete display and control facilities including console, status line control and processor access for on-line parameter changing;

A maintenance panel;

An on-line, Selectric Typewriter for manual and program-controlled input/output.

1.2.7 8490 Power Module - Including:

A capability for providing the 8400 system's full power requirements;

Provisions for the manual, marginal testing of memory;

Provisions for power-fail monitoring; and,

Provision for over/under power protection.

1.2.8 The following list includes optional peripheral devices and system expansion components:

8441 Paper Tape Station - with a 500 cps read and 110 cps punch capability. (cps = char/sec).

8412 Quad Index Register Pak - adds to the processor four high-speed index registers (registers 4 to 7).

8417 Timer Register - provides addressable, real-time millisecond clock.

8422 8K Memory Bank - with same features as 8422-E Memory Module. (Maximum four banks per 8400.)

8423 16K Memory Bank - with a 16,384 word, core storage capacity; other features are the same as those for the 8420 Memory Module. (Maximum four banks per 8400.)

8431 Program Control Data Channel - provides a data channel capability for any exchange module channel position, from 1 to 7; handles up to 15 peripheral device controllers.

8435-1 Automatic Data Channel Processor - provides independent block data transfer control for the 8431 data channel of channel position 0 in the exchange module; requires the use of an 8420 Memory Interface Pak; independent of central processor.

8435-2, 3, 4 Automatic Data Channel Processor Expansions - each adds independent block data transfer control for one 8431 data channel occupying any channel position between 1 and 7 in the exchange module; 8435-1 Automatic Data Channel is required in order to use the expansion. (Maximum of three.)

8420-21, 22, 23, 24 Memory Interface Pak - provides coupling interface between 8435-1 Automatic Data Channel Processor and Memory Banks 1, 2, 3, and 4, respectively. Maximum of four; one required per memory bank. Necessary if an ADCP is to be used.

8437-2 through 16 External Interrupt System Expansion Group - each group adds 16 interrupt lines to basic external interrupt system.

8438-1 through 128 Status Line Package - provides in the exchange additional status line groups of 8 lines each (two 8-line groups per unit). Each package provides fully buffered flip-flop storage for sense input from external devices. (Maximum 64 groups.)

8439-1 through 128 Function Line Package - provides in the exchange additional function groups of 8 lines each (two 8-line groups per unit). Each package provides fully buffered flip-flop storage for function line output to external devices. (Maximun 64 groups.)

8441 Paper Tape Station - 500 character-per-second read and 110 character-per-second punch. Mounting provisions are included in the 8440 Central Console.

8452 Card Reader - 400 cards-per-minute; 12 row cards, 80 column read.

8453 Card Reader - 800 cards-per-minute; 12 row cards, 80 column read.

8454 Card Reader - 1400 cards-per-minute; 12 row cards, 80 column read.

8455 Serial Card Punch - 100 cards-per-minute to 316 cpm.

8456 Parallel Card Punch - 300 cards-per-minute.

846l Line Printer - 300 lines-per-minute; 132 columns-per-line, 64 characters, buffered printer.

8462 Line Printer - 600 lines-per-minute; 132 columns-per-line, 64 characters, buffered printer.

8463 Line Printer - 1000 lines-per-minute.

8472 Magnetic Tape System - provides controller handlin up to four transports (8473); one is included, maximum of four. The tape transport uses 7-track, IBM compatible tapes and operates at 45 ips and 556 and 800 bpi, respectively.

8474 Magnetic Tape System - provides controller handling up to four transports (8475);

one is included (maximum of four). The tape transport uses 7-track, IBM compatible tapes and operates at 75 ips and 556 and 800 bpi, respectively.

8476 Magnetic Tape System - provides controller handling up to four transports (8477); one is included. The tape transport uses 7-track, IBM compatible tapes and operates at 120 ips and 556 and 800 bpi, respectively.

8478 Magnetic Tape System - provides controller handling up to four transports (8479); one is included. The tape transport (8479) uses 7-track, IBM compatible tapes and operates at 150 ips and 556 and 800 bpi, respectively.

NOTE

Model Numbers 8472-9, 8474-9, 8476-9, and 8478-9 are the same as the models listed above except that they use 9 track IBM compatible tapes.

8481 Display Monitor - provides point, line and character plotting on a 10" x 10" display of 1024 points along each axis. Light pen is included.

1.3 PROCESSOR

1.3.1 Memory Word

The 8400 Computer's memory word consists of 36 bits: 2 bits for parity check, 2 bits for program control (EXEC bits), and 32 data bits. The memory word format is shown in Figure 1-3.

The parity bits are generated and stored on a half-word basis during the write cycle. Parity is then checked during the read cycle. If an error is located, the console indicator lights and a parity interrupt is initiated. The 8400 System uses odd parity; this means that whenever the number of logic ONE's

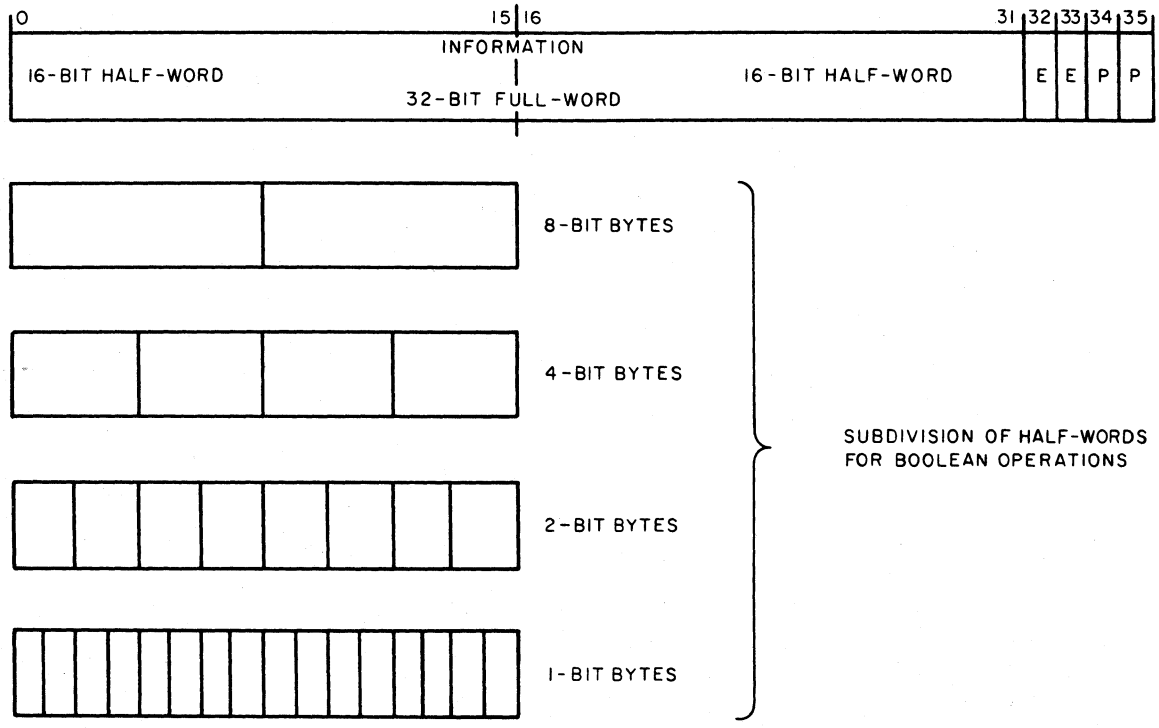


Figure 1-3. Memory Word Format

making up a word is even, a parity bit is generated so that the result is odd. (Using odd parity, the parity bit is always the opposite when all 1's or 0's are used.)

The EXEC bits, in effect, expand the system's software capability. Used by the programmer to tag selected memory words, these two bits are also capable of the following:

... enabling interrupt control for memory protection,

... dynamic relocation of object programs,

... stack or table pointing, and so forth.

EXEC bit control is discussed in Chapter 2.

The information portion of the word may contain a full (32-bit) word, two half (16-bit) words, or portions thereof (8, 4, 2, or 1-bit) for Boolean operations.

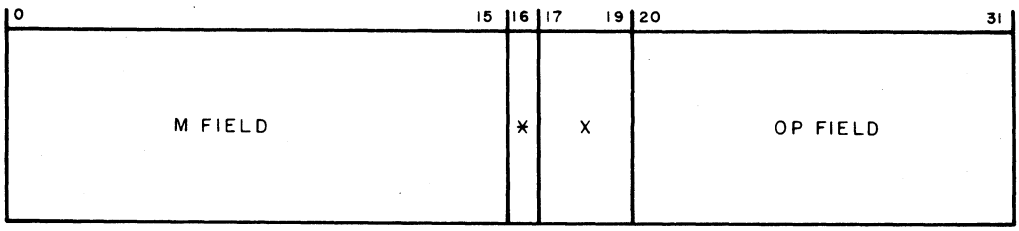
1.3.2 Instruction Word

Instructions are executed in sequence by the 8400 Instruction Register (I). Each instruction has a 32-bit word format as shown in Figure 1-4. This figure indicates the normal program control capabilities of the instruction word; for example, addressing, address modification, and instruction interpretation. The first sixteen bits (M field) in the word format represent the operand address during a data fetch. It may also signify: an instruction address during an instruction fetch, an immediate operand, or a shift count. The next four bits designate any address modification required. If bit 16 (*) is a binary 1, the M field contains the address of another location in memory that will replace the present M field, rather than the address of an operand. Bits 17 through 19 (X, where X = 1 to 7) specify the number of an Index register. Either or both may be used to change the interpretation of the instruction address during execution.

The last 12-bit (OP field) portion of the word format denotes the operation to be performed.

1.3.3 Data Word Formats

This section describes the word formats used in the 8400 Computer. The brief descriptions refer to Figure 1-5. Arithmetic formats are in a *two's complement* notation with the + sign (binary 1) indicating a negative quantity. The instruction, memory data and memory address word formats are included in Figure 1-5 for comparison.



M 16 - BIT (MEMORY) ADDRESS FIELD
* 1 - BIT INDIRECT ADDRESS MODIFIER
X 3 - BIT INDEXING MODIFIER
OP 12 - BIT COMMAND (OPERATION CODE)

Figure 1-4. Instruction Word Format

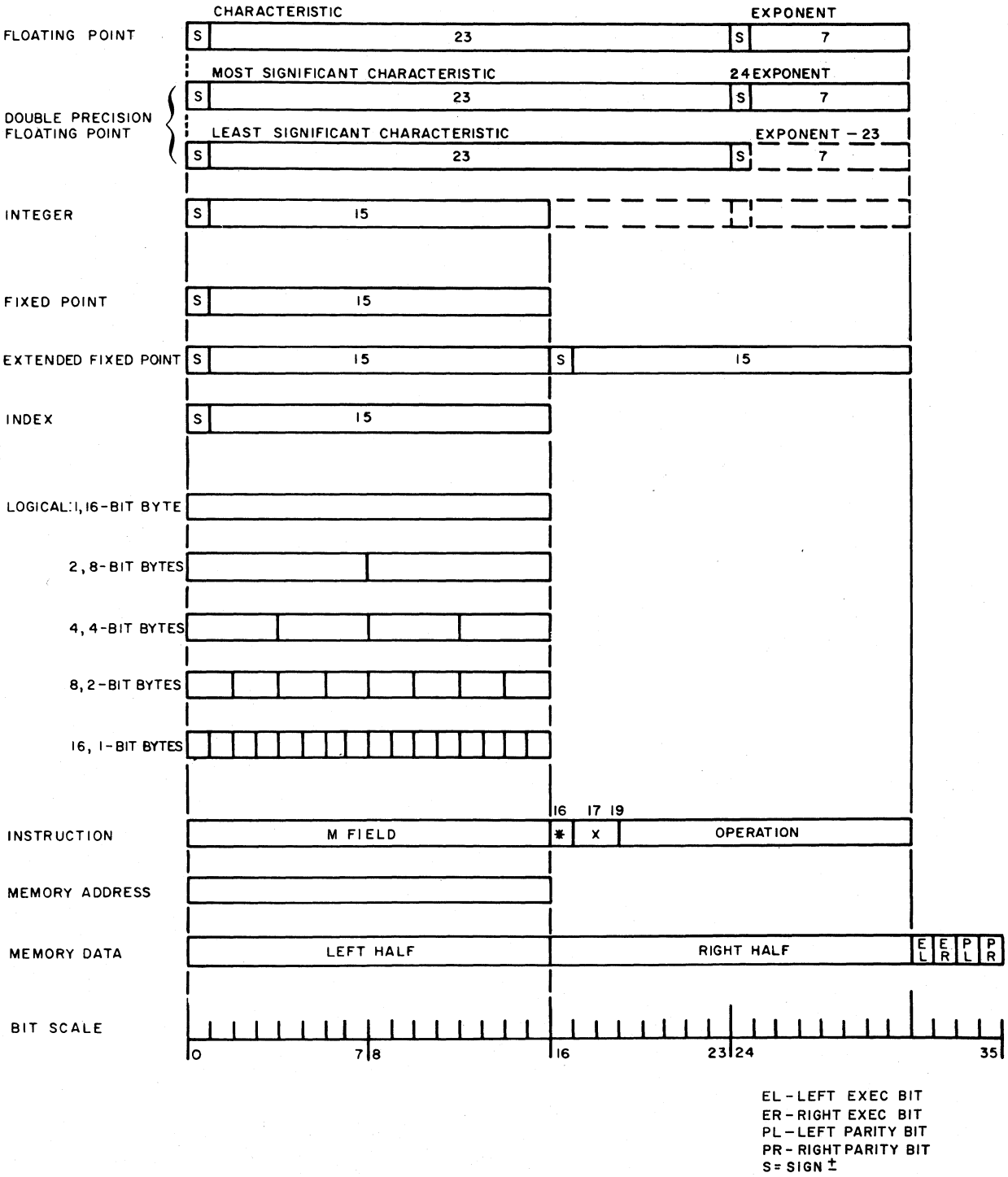


Figure 1-5. Summary of 8400 Word Format

1.3.3.1 *Floating-Point.* Floating-point numbers are either single word (32-bit), or double precision (56-bit) quantities. The single-precision floating-point number consists of:

... a fractional part (23 magnitude bits),

... a sign bit,

... and an exponent part (7 magnitude bits) with its own sign bit.

This single-precision floating-point notation provides an accuracy of six decimal digits.

The double-precision floating-point number occupies two consecutive memory word locations. The word with the lowest address contains the most significant fraction and exponent bits. The signed exponent part of the word (eight bits) with the higher address is adjusted during memory store to EXP-2^{23}. Double floating-point notation provides an accuracy of thirteen decimal digits.

The double-precision floating-point word format has direct correspondence with the single floating-point format. For example, when executing a 32-bit floating-point multiply, the product will be in the double-precision word format. Therefore, the results of several *32-bit floating-point multiply* operations can be accumulated using *double-precision floating-point* add operations. The results may be operated on individually since the sign and exponent for each of the *most significant* and *least significant* portions are preserved.

Floating-point operations are normalized (adjustment of the mantissa and floating-point number so that the mantissa lies in the prescribed normal range) automatically after each operation unless the instruction is post-modified by the unnormalized symbol (U). Normalization is accomplished by using left shifts to remove all leading zeros from the number in the accumulator. The shifting continues until the contents of the first two bit positions (0, 1) in the accumulator differ.

1.3.3.2 *Fixed-Point.* Formats for the standard (16-bit) and extended (32-bit) fixed-point quantities are illustrated in Figure 1-5. The standard fixed-point format consists of a 15-bit fraction along with a sign bit and may occupy either half-word position of the memory word. The extended fixed-point format contains two 15-bit fractional parts and a sign bit for each. Its left half-word contains the fifteen most significant bits and the sign of the entire 30-bit fractional quantities. In standard fixed-point arithmetic operations, the half-words are addressed individually.

1.3.3.3 *Integer.* Integer arithmetic instruction involve operations with two types of data words:

1. Standard, 16-bit fixed-point
   and
2. Single, 32-bit floating-point.

The data word associated with the system memory is standard, 16-bit, fixed-point notation. The operand in the Accumulator is in single 32-bit, floating-point notation. In the *integer* mode, a 16-bit, fixed-point number is automatically converted from the half-word memory location to the floating-point format. Likewise, a floating-point number in the accumulator which represents the result of a series of floating-point operations, is integerized and stored in the designated half-word memory location. Operations in this mode may be either normalized or unnormalized by post modifying the associated instructions.

1.3.3.4 *Index.* In this operation, the contents of a specified index register is arithmetically combined with the contents of a half-word memory location. The result, obtained in the accumulator, is automatically transferred back into the specific index register and the previous contents of the accumulator are restored.

1.3.3.5 *Logical Byte.* Logical Byte operations between half-word memory locations and the accumulator may be performed in 16, 8, 4, 2, or 1-bit bytes. A single instruction selects the desired byte size,

byte positions, logical connective and recipient (either memory or accumulator) of the operation results.

1.3.4 Processor Registers

The following registers in the 8400 Computer provide the major portion of the Processor's capability for control and execution of the stored program. (See Figure 1-6).

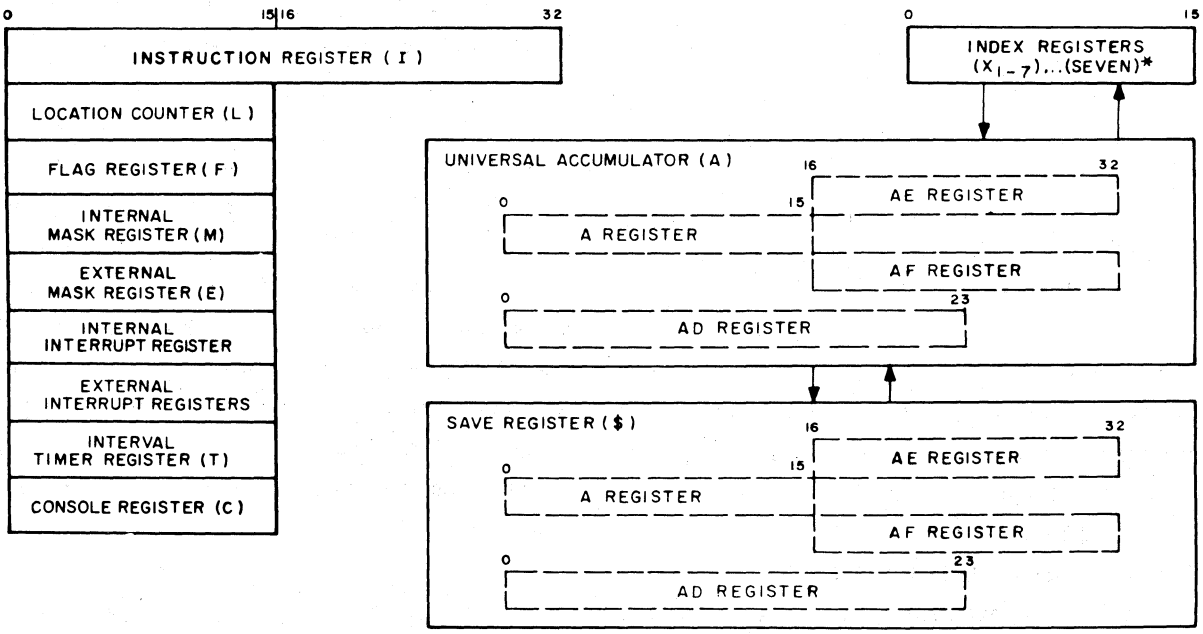
1.3.4.1 *Instruction Register (I).* This 32-bit register stores each instruction as it is executed. The register format is the same as the 8400 instruction word as shown in Figure 1-4.

1.3.4.2 *Location Counter (L).* This 16-bit register contains the address of the next instruction to be loaded into the Instruction Register. Its primary function is to provide system program control by sequentially directing the flow of instructions into the system. The contents of the Location Counter may be stored when necessary.

1.3.4.3 *Universal Accumulator (A).* The Universal Accumulator as shown in Figure 1-7, consists of four separate registers which carry out the 8400's arithmetic and data operations.

The first section, the 16-bit A Register, is used by itself for 16-bit fixed-point operations and as the most significant 16 bits of all other operations. The second section, the AE Register, is used when extended fixed-point and extended shift operations are needed. It enables the A Register to Handle 32-bit fixed-point quantities such as: 32-bit, double-length products; and dividends of standard, 16-bit fixed-point multiply and divide operations. The AF Register, the third section, is a 16-bit A Register extension and is used for single-word (32-bit) floating-point quantities. The final section, the AD Register, provides a 24-bit extension to the AF Register. This enables the accumulator to handle 56-bit, double-precision floating-point quantities; such as the double-precision products and dividends of a single-word floating-point multiplication and division operations.

The accumulator provides several special 8400 programming features. It provides a single reference location for the implicit operand and the result of all arithmetic and logical operations. The accumulator is universal in that it automatically handles all inter-register transfers after each arithmetic operation. Another convenience is that it may be used as an index register. Finally, by virtue of its self-addressing



*The A Register of the Universal Accumulator is Index Register X_1

Figure 1-6. Processor Registers

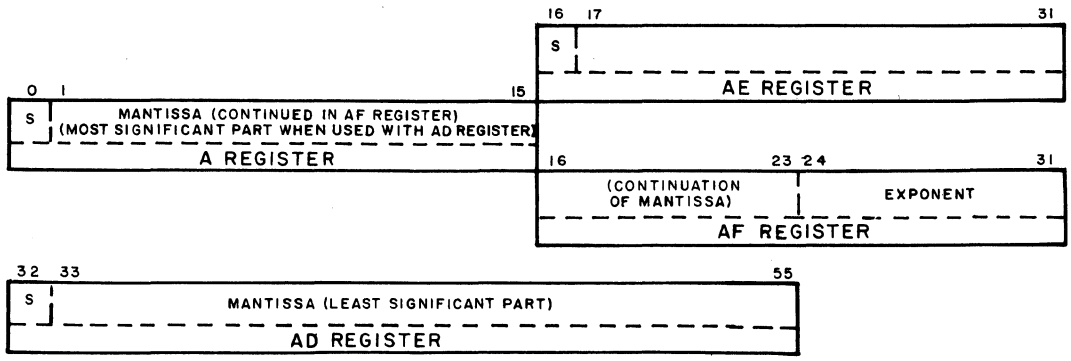


Figure 1-7. Universal Accumulator Formats

capability, the accumulator enables the performance of doubling and squaring at high speeds. This self-addressing capability also enables data transfer between the accumulator and all index registers. The Universal Accumulator is addressable as memory location zero.

1.3.4.4 *Save Register ($).* The Save Register is a high-speed storage register similar to the Universal Accumulator. (See Figures 1-6 and 1-7.) This register is used to retain the entire contents of the accumulator prior to the execution of any arithmetic or shift instruction. The *Save* option is designated by the & symbol and may be used with any arithmetic or shift operation.

The programmer, by using the Save Register, is able to store or read operands in 560 nanoseconds, less time than it takes using core memory. The data is automatically arranged in the proper format when recalled by the Universal Accumulator. Similar to a memory cell, this register retains data until a subsequent instruction containing the $ symbol stores new data; the data is NOT destroyed during the save-write cycle.

1.3.4.5 *Index Register (X_{1-7}).* Seven index registers including the Universal Accumulator (index register one) provide automatic address modification. These registers retain half-word numbers that are expressed in two's complement notation.

When indexed address modification is specified, the effective address is formed by adding the contents of the selected index register to the contents of the M field. This operation has no effect on index register content.

Index arithmetic instructions allow direct operation between the respective contents of a specified index register and an addressed memory location. A single instruction effects the following: An automatic parallel transfer of the contents of the addressed index register to the Universal Accumulator; an arithmetic operation, as specified, combining this quantity with that contained in the addressed memory location; and, an automatic transfer of the result back to the same index register. The transfer from the index register is made to the parallel A Register of the Universal Accumulator with the previous A Register contents being stored. Since the index register has no extension (AE Register), its use is restricted to operations giving a half-word (16-bit) results.

1.3.4.6 *Flag Register (F).* The addressable, 16-bit Flag Register continually monitors machine conditions as well as those specified by the programmer. At the end of each instruction, the status of these conditions is indicated by the register's sixteen flag bits.

Tested by a set of transfer operations, the flags provide the basis for the 8400's extensive program-control capability. They signify modifications of the normal sequential control for the program. Basic control instructions affected include the following:

HJf	HALT if flag f set and JUMP when execute button depressed;
EXf	EXECUTE instruction at specified location if flag f set;
Lf	LINK to subroutine if flag f set;
LRf	LINK to subroutine if flag f set, RESET flag; JUMP if flag f set;
Jf	JUMP if flag f set;
JRf	JUMP if flag f set, RESET flag;
JSf	JUMP if flag f set, SET flag;
JTf	JUMP if flag f set, TRIGGER flag.

The LINK and JUMP operations are conditional; they depend upon the status of the flag tested. The setting, resetting, or triggering (complementing) of the flag, however, is unconditional.

The Flag Register bits indicate the status of 16 internal machine conditions. Eight of the bits serve as programmer console flags and are set by either console switches or the program. Internal machine status conditions can be preserved at any particular time by storing the entire register contents in memory. This enables the programmer to retrieve internal machine status after the occurrence of subsequent interrupt conditions.

1.3.4.7 *Mask Registers, Internal (M) and External* (E). The Internal Mask Register and External Mask Register permit the programmer to select the interrupts a program will respond to and, to establish a priority among these interrupts. These 16-bit registers are loaded and stored by the use of special instructions (see Chapter 2).

1.3.4.8 *Console Register (C).* This 16-bit register enables monitoring, data display and data input while the program is in progress. It may be loaded by the operator or by the program.

1.3.4.9 *Interval Timer Register (T).* Enabled by the LOAD INTERVAL TIMER (LDT) instruction, this 16-bit register decrements one count each millisecond† providing computer *real-time* control. As the register goes through zero, an interrupt is generated and the register is reset to its maximum value. At this point, unless reloaded by the interrupt subroutine, the register continues to decrement as before.

With all its bit positions occupied, the Interval Timer Register will decrement through a maximum time range of 65,536-1 milliseconds $\left(\left(\sum_{n=0}^{15} 2^n\right)\right)$ Interrupts may be programmed to occur at any selected time interval within this range. Consequently, the register is extremely useful for: program synchronization, periodic output of data, time-sharing programs or consoles, periodic sense line testing, and many other purposes.

1.4 ADDRESSING

The extensive addressing capability in the 8400 Scientific Computing System facilitates the handling of all normally encountered address manipulations involving core memory locations. Direct, indexed, and indirect addressing have been made available to the programmer. In addition, an immediate or literal addressing capability provides programming flexibility for fast efficient processing.

1.4.1 Direct Addressing

With direct addressing, the 16-bit address specified by the instructions' M field refers directly to the memory location of the data (operand) that is to be

† *Other factory-set timing intervals are also available.*

used in the specified operation. In arithmetic operations, either full-word or half-word operands may be used. With full-word operands, the entire contents of the specified memory location are involved. With half-word operands, either the right or left half of the full-word location is used. The half-word to be used is designated in the instruction by a / (slash) post modifier (see Chapter 2).

For double precision arithmetic operations, the contents of both the specified memory location and the next memory location (M + 1) are accessed. In Boolean operations, the specific half-word (including its byte size and position) is specified by using post modifiers in the associated instruction, i.e., AHM4/SAM,,3. In this instruction: 4 specifies a four-bit byte; SAM designates the memory location and, being to the right of the slash, the right half-word of this full-word location is specified. The instruction states, "where each of the four bits in the third byte position are high, set the corresponding bits in the right half-word of memory location SAM".

1.4.2 Indexed Addressing

Indexed addressing represents an important and highly useful variation of direct addressing. The 8400 contains seven index registers providing an efficient, flexible means of address modification.

Indexing adds the contents of an index register to the address portion of an instruction, Bits 17, 18, and 19 of the instruction word specify which one of the seven index registers is to be activated. If bit positions 17, 18, and 19 are zero, no indexing is specified. The contents of any one of the computers' seven index registers are added to the 16-bit address field (base part) to form the "effective address".

1.4.3 Indirect Addressing

Multi-level indirect addressing may be used without being restricted by any 8400 instructions. The * bit (bit 16) of the instruction word is used as the indirect indicator. When indirect addressing is specified, the 16-bit address gives the memory location where the address of the data may be found. Thus, the address of the data is given indirectly.

If both an index register and indirect addressing are required by the programmer, the effective address is computed as previously discussed and then the indirect address is computed.

1.4.4 Immediate Addressing

The 16-bit address of an arithmetic or logical instruction serves as the operand when immediate addressing is used. This operand is a signed number represented in two's complement notation. Immediate addressing is specified in the 12-bit operation (OP) field of the instruction word and is accomplished in symbolic notation by placing the = symbol in this field.

This form of addressing saves instruction time since it permits the direct use of data from the Instruction Register; no memory access required. This form of addressing also saves memory space since no operand memory locations are required and; in addition, the operand may be modified by the contents of a specific index register since the immediate operand is located in the instruction word address field. With this modification accomplished prior to using the immediate operand, the *effective immediate operand* concept can be used by the programmer to provide greater programming flexibility. (A store command with immediate addressing capability is called an NOP.)

The immediate operand notion extends to shifting operations as well. In a SHIFT instruction, the desired number of shifts is specified as an arithmetic operand. Direction of shifting is determined by the shift count sign. And, as an immediate operand, this shift count may be modified by the contents of a specified index register.

1.5 INTERRUPT SYSTEM

The true multilevel interrupt system with mask control permits multilevel interruption to *any* depth

without a loss of return-continuity. When an internal or external interrupt condition occurs and the interrupt action is not inhibited by masks, an instruction in a reversed interrupt location is executed (each interrupt condition has a reversed location). The execution of this instruction does not change Location Counter contents. LINK, the normally executed instruction, transfers control to the interrupt routine while preserving the return address of the interrupted program. Once started, an interrupt action may be interrupted by the occurrence of a subsequent system interrupt condition.

The interrupt system responds to internal conditions and operating modes monitored by sixteen internal interrupt lines. It is also responsive to any of up to 256 external conditions monitored by an expanded complement of external interrupt lines. There are 16 external interrupt lines in the first group with a capability of up to 15 additional external groups. The complete system is arranged in 17 groups containing 16 individual interrupt levels apiece. Group 0 includes the internal interrupt levels and groups 1 through 16, the external interrupt levels. Each group has priority over the succeeding group and each group level over lower levels.

Basically, 16 flip-flops (each set by a particular interrupt condition) are scanned. A scanner will accept a flip-flop output only if the Interrupt Enable bit in the Flag Register and the corresponding mask register bit are both high. The scanner begins scanning at selected points in the instruction flow, continues until an interrupt condition is detected, and then locks onto that position. The priority of a scanner is established when no "higher-priority" scanner has locked up. After priority has been established, interrupt logic determines the address of the reserved interrupt location.

After an interrupt subroutine has been given control and the return address of the interrupted program has been saved, scanning of the higher priority interrupt levels resume. This insures that the operation of the subroutine will be interrupted only when the higher interrupt conditions are detected. Such interruption may be eliminated by using the appropriate masking.

CHAPTER 2

INSTRUCTION REPERTOIRE

2.1 INTRODUCTION

This chapter contains a complete and precise definition of the operations performed by every 8400 instruction. Further details on input/output instructions for peripheral devices are given in Chapter 4. The emphasis in this chapter is on programming rules and conventions; moreover, as an aid in learning the entire repertoire, the instructions are presented in classes and sets. The programming conventions and notation used are taken from the 8400 Macro-Assembler manual (EAI Publication Number 07 800.0001-3). The actual OP code numerical values are given in the appendix and in the 8400 Programmer's Card.

Consistent with the objective of presenting the instruction repertoire as the programmer will use it, the view of the computer is offered as functional - appropriate for the programmer. Therefore, Figure 2.1 illustrates the essential programmable elements of the central processor. Only the 16-bit input and output busses are missing. The registers required for momentary storage of data traveling to the accumulator from the memory is only one explicit inter-register transfer and that is between the accumulator and the Save ($) Register. The memory-to-register transfer paths are obvious; therefore, not shown. Other register transfers which are closely related to the instruction and their options are developed in the text below.

Figure 2.1 illustrates the following important facts:

- ...The Save Register can hold a copy of the accumulator contents except for the exec bits (shown crosshatched);
- ...the EA and AF Registers are alternative extensions of the A Register;
- ...the A Register is also treated as Index Register one;
- ...the instruction address field in the left portion of the instruction word;
- ...other special registers are a half-word in size.

2.2 EFFECTIVE ADDRESS CALCULATIONS

2.2.1 Direct Addressing

All instructions referencing an operand in memory specify one of several ways to address the operand. The most basic, *direct addressing*, uses the 16-bit left hand number of the instruction word as the operand address. Other addressing modes are termed: *index modification, indirect addressing,* and *immediate* addressing. *Half-word* addressing, a variation on each of the above, is allowed when the operand is a 16-bit word. The left or right half-word selection is made after a memory address. This left/right option is an integral part of the OP code (bits 20 to 31 of the I Register as shown in Figure 2.1).

2.2.2 Indexing

One of the primary uses of index registers arises from their ability to modify instruction addresses. For this to occur, the instruction must specify the particular index register that is to take part in the modifying activity. Indexing adds the contents of an index register to the address portion of an instruction.

When indexing is specified, the contents of the index registers are added algebraically, in two's complement notation, to the address portion of the instruction. This new address, the effective address, is then to be used as the operand.

If the index register contains a negative (two's complement) value, the result of address modification to subtract this value from the address portion of the instruction.

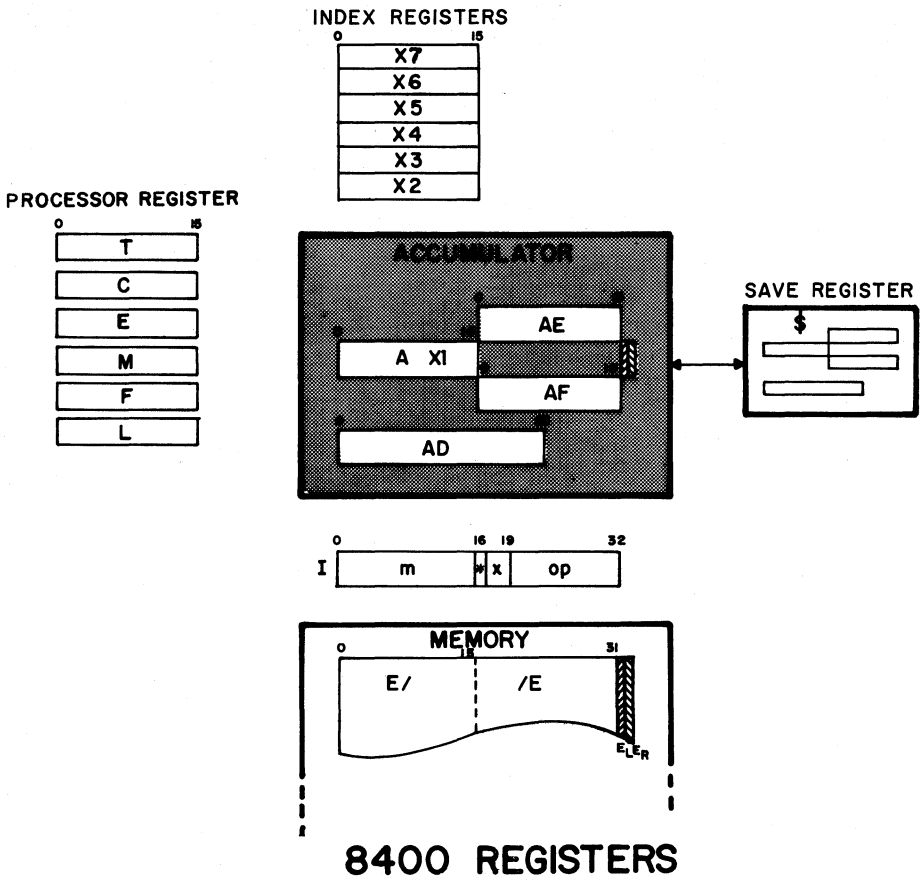


Figure 2.1

As an example of address modification, assume that memory address 500 contains the instruction AD 124,2 and this instruction has a two in the appropriate index register position. If the contents of the index register are 27, then the number stored in memory location 151 (124 + 27 = 151) is added into the accumulator when the add AD instruction is executed. Note that memory location 500 still contains the instruction AD 124,2 in its original form. Memory address 151 is called the *effective address,* and the process is called address modification; that is, the address of the instruction is modified in the central processor for execution purposes, but is unaltered in memory.

2.2.3 Indirect Addressing

When processing data is located in several different areas of memory, it is at times convenient to operate upon an *indirect* address as opposed to an actual address. For example, if the programmer wishes to perform an add instruction at memory location 3000 (assume that memory location 123 already contains 1862), he would write location 3000

AD 123.

However, if he wishes to add to the accumulator, not the contents of memory location 123 but the *contents of the contents* of memory location 123, then the computer would add the contents of location 1862 to the accumulator. This procedure is known as *indirect addressing,* and occurs when a 1 is placed in bit position 16 (*) of the instruction word. Mnemonically, it is written as:

AD* 123

Note that if an index register and indirect addressing are requested by the programmer, the effective address is computed as previously discussed and then the indirect address is computed.

2.2.4 Summary

Table 2.1 lists the entire 8400 Instruction Repertoire.

An instruction that has a number (m) in the left half-word and no address modifier in the right half-word is said to address memory core location, m, which contains the operand. If the *Immediate* option is

Table 2.1. 8400 Instruction Repertoire

8400 INSTRUCTION LIST

ARITHMETIC OPERATIONS

32-BIT FLOATING-POINT	MNEMONIC	16-BIT FIXED POINT	MNEMONIC
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	MNEMONIC	32-BIT EXTENDED FIXED-POINT	MNEMONIC
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	Multiplyt	EMP
Store	DST	Store	EST
Store Rounded †	DSR	Store Rounded †	ESR
Divide †	DDV	Divide †	EDV
Clear Divide †	DCD	Clear Divide †	ECD

†*denotes compatible subroutine operations*

16-BIT INTEGER	MNEMONIC	16-BIT INDEX	MNEMONIC
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

Operation modifiers extend the basic list. The post modifier "U" specifies unnormalized operation for F, D and I Classes. The premodifier "$" specifies a Save Register store of the accumulator contents prior to the execution of a modified instruction of any class. Examples: FADU; $EAD or $FADU.

The prefixes or suffixes in the mnemonic symbol denote the class of operation or operation conditions. The basic symbol indicates the operation itself.
Example: In FAD, F stands for Floating-Point Arithmetic (32-bit precision). AD indicates the ADD operation.

SHIFTING, ROTATION AND NORMALIZING OPERATIONS

ACCUMULATOR	MNEMONIC	EXTENDED ACCUMULATOR	MNEMONIC
Arithmetic Shift	ASH	Arithmetic Shift	EASH
Logical Rotate	ROT	Logical Rotate	EROT
Normalize	NRM	Normalize	ENRM

LOGICAL BYTE OPERATIONS

BOOLEAN CONNECTIVES – RESULTS TO ACCUMULATOR	MNEMONIC	BOOLEAN CONNECTIVES – RESULTS TO MEMORY	MNEMONIC
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 $\overline{A}$)	CBHAn	Complement Both High (And $\overline{A}$)	CBHMn
Complement Either High (Or $\overline{A}$)	CEHAn	Complement Either High (Or $\overline{A}$)	CEHMn
Complement Either Low (Nand $\overline{A}$)	CELAn	Complement Either Low (Nand $\overline{A}$)	CELMn
Complement Both Low (Nor $\overline{A}$)	CBLAn	Complement Both Low (Nor $\overline{A}$)	CBLMn
Byte Equality Test (Set Z Flag if Bytes identical)	BEQTn	Memory High (Set Z Flag if Byte in M is Zero)	MHMn

Example: ELAn specifies that the result of a NAND, with an n-bit byte in A and an n-bit byte in M, replace the A-byte. The mnemonic is interpreted as follows: "For corresponding bit positions in the A-byte and the M-byte, where EITHER the A-bit or the M-bit is LOW, set resultant A-bit. n= 1,2,4,8 or 16 bits.

CONTROL OPERATIONS

TEST BRANCH – ON "FLAG". f	MNEMONIC
Halt – Jump	HJf
Execute	EXf
Link (To Subroutine)	Lf
Link – Reset Flag	LRf
Jump – Trigger Flag	JTf
Jump – Set Flag	JSf
Jump – Reset Flag	JRf
Jump	Jf

f denotes one of the 16 conditions monitored by a special flag register. These conditions are listed below with their respective mnemonics. Complement conditions can be specified by prefixing the condition mnemonics with the modifier, "N".

Examples: Halt Jump on Overflow is coded HJV. Halt Jump on no Overflow is HJNV.

INDEX JUMPS	MNEMONIC
Index Jump – Unconditional	XJ
Index Jump Test	XJT

EXEC BIT CONTROL	MNEMONIC
Set EXEC Bit	SEX
Reset EXEC Bit	REX
Test EXEC Bit	TEX

Accumulator Equals Zero	Z	Carry (Or Borrow)	C
Accumulator Greater than Zero	G	Busy Data Channel	B
Accumulator Less than Zero	L	Enabled Interrupt	E
Overflow (Cumulative)	V		
Unconditionally True	blank	Programmer Flags	1-8

I/O OPERATIONS

I/O REGISTER LOAD	MNEMONIC	I/O REGISTER STORE	MNEMONIC
Load Output Buss	LDOB	Store Input Buss	STIB
Load Channel Data Register	LDCD	Store Channel Data Register	STCD
Load Channel Control Register	LDCC	Store Channel Control Register	STCC
Set Function Line	SFL	Test Status Line	TSL

AUTOMATIC DATA CHANNEL CONTROL – DATA BLOCK TRANSMISSION	MNEMONIC	AUTOMATIC DATA CHANNEL CONTROL – NO DATA BLOCK TRANSMISSION	MNEMONIC
Transmit on Count – Disconnect	TCD	Skip on Count – Disconnect	SCD
Transmit Until Signal – Disconnect	TSD	Skip Until Signal – Disconnect	SSD
Transmit on Either – Disconnect	TED	Skip on Either – Disconnect	SED
Transmit on Count – Interrupt	TCI	Skip on Count – Interrupt	SCI
Transmit Until Signal – Interrupt	TSI	Skip Until Signal – Interrupt	SSI
Transmit on Either – Interrupt	TEI	Skip on Either – Interrupt	SEI

SPECIAL REGISTER TRANSFER OPERATIONS

REGISTER TRANSFER – LOAD	MNEMONIC	REGISTER TRANSFER – STORE	MNEMONIC
Load A Extended	LDAE	Store A Extended	STAE
Load Flag Register	LDF	Store Flag Register	STF
Load Location Counter	LDL	Store Location Counter	STL
Load Timer Register	LDT	Store Timer Register	STT
Load Mask Register Internal	LDM	Store Mask Register Internal	STM
Load External Mask Register	LDE	Store External Mask Register	STE
Load Console Register	LDC	Store Console Register	STC

specified, the number (m) is "immediately" treated as data (operand) rather than as an address. If the *Indirect* option is specified, core memory address m contains the "address" of the operand rather than the operand itself. If the *Index* option is specified, a number is taken from one of the seven index registers and added to m to produce the effective address of the operand; while the left half contents (m) remain the same. Several options may be used simultaneously as described in the following section. In any case, the action of address modification is often referred to as "effective address calculation". Since this is a common occurrence and is possible with the majority of instructions, the E notation is used regularly throughout for the contents of the effective address.

2.2.5 Combinations of Addressing Options

The various address modifiers and the legal combinations thereof are shown in Table 2.2 in the format used when writing instructions for the assembler.

The precedence of the addressing options is: X, *, =, *or* /. This means that index modification (X) takes place before indirect addressing (*), and the final option is the half-word selection of immediate, left, or right (= and / cannot appear simultaneously). This sequence is illustrated in Figure 2.2. The notation in this figure is that parentheses around a register name specify the register contents; the arrow reads as "replaces", and subscripts indicate specific register bits.

The flow diagram of Figure 2.2 is interpreted as follows:

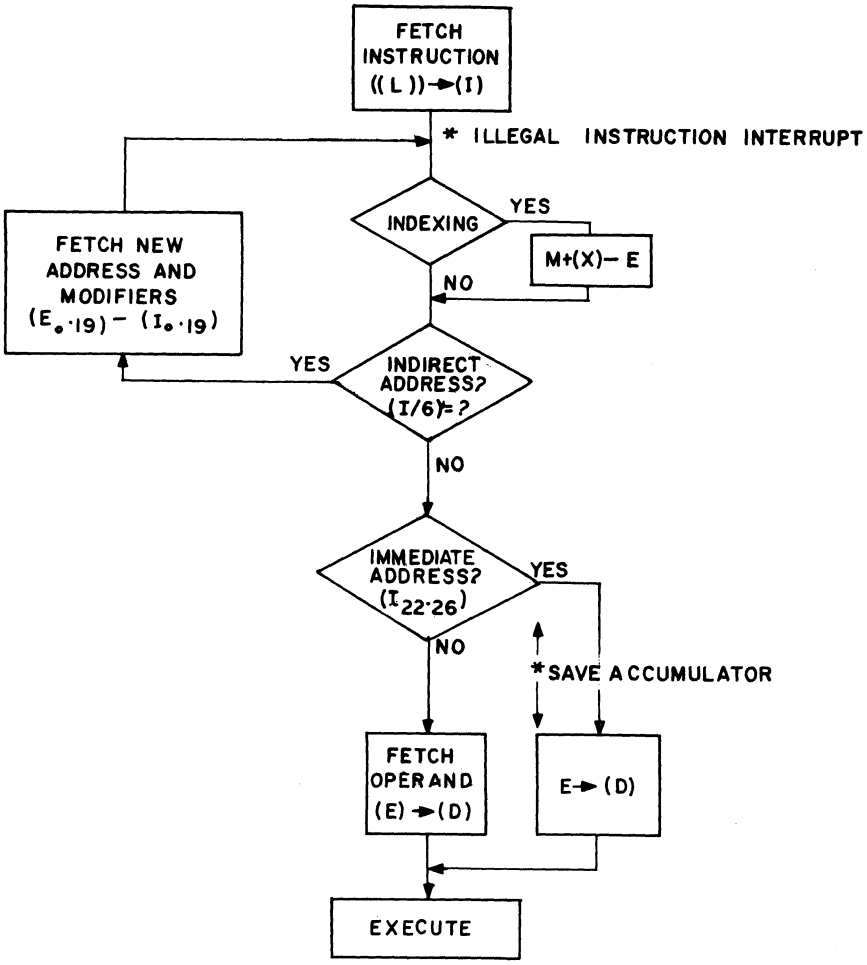
1. The instruction cycle starts with an instruction fetch, which is denoted as ((L)) ⟶ (I), or "The contents of the memory address that is the contents of L replaces the contents of the instruction register, I".
2. If indexing is specified (by 1 to 7 in bit positions $I_{17, 18, 19}$) then the sum of the number *m* in the instruction address field and the contents of the specified index register forms (tentatively) the effective address E.

Table 2.2

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 + (X) ⟶ E.A. 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	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: All legal combinations of the address modifiers are illustrated below:

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



EFFECTIVE ADDRESS CALCULATION

Figure 2.2

3. If indirect addressing is specified (by $I_{16} = 1$) then a new word, located at address E, is obtained from the memory. Only the first 19 bits of this word are used, and they replace the contents of bit $I_{0:19}$. Now, the original value *m* has been replaced in I by $(E_{0:15})$, and the index and indirect bits have also been replaced. Steps 1, 2, and 3 are repeated until for some indirect address, indirect addressing is *not* specified. It is possible, through a programming error, for the above loop to be a closed path, which "hangs-up" the computer. Such a loop can be broken only by a manual halt or an interrupt. However, in the normal course, indirect references can be made (at the expense of time for each memory fetch) and indexing (for different index registers) can be performed at each level of indirectness.

4. With a new value for E, bits $I_{25:26}$ in the original instruction are tested for immediate addressing (half-word operands only). If *immediate* is specified, the value of E itself is placed in the intermediate data register (D) before the execution of the accumulator instruction. Otherwise E is

used as the effective memory address for the final gathering of data from the memory. After this gathering, the *left/right* option selects the specified half-word for execution.

5. Note that the *Save* option takes place just after the immediate address test on either path.

The different *effective address* calculations may be specified as follows, where OPN signifies any operation code for which the address options are valid.

OPN m	operand = $(E_{0:15})$ or $(E_{0:31})$, E = m
OPN m/	operand = $(E_{0:15})$, E = m
OPN /m	operand = $(E_{16:31})$, E = m
OPN m, X	operand = $(E_{0:15})$ or $(E_{0:31})$, E = m + (X)
OPN /m, X	operand = $(E_{16:31})$, E = m+ (X)
†OPN*m	operand = $(E_{0:15})$ or $(E_{0:31})$, E= (m)
†OPN*/m	operand = $(E_{16:31})$, E = (m)
†OPN*m, X	operand = $(E_{0:15})$ or $(E_{0:31})$, E = (m + (X))
†OPN*/m, X	operand = $(E_{16:31})$, E = (m + (X))
OPN = m	operand = E = m (16-bit operand only)
OPN = m, X	operand = E = m + (X) (16-bit operand only)
†OPN*=m	operand = E = (m) (16-bit operand only)
OPN* = m, X	operand = E = (m + (X)) (16-bit operand only)

2.3 ARITHMETIC INSTRUCTIONS

There are ten basic arithmetic instructions that occur in each of six classes of operation. The classes differ in the form of arithmetic, word size, and the registers affected. The resultant sixty mnemonics and their functions are readily committed to memory. There are numerous variations to these basic instructions and they follow a consistent and logical pattern. All arithmetic instructions may exercise the *Save* option prior to execution; and they set Z, G, and L decision flags after execution. All floating-point operations may terminate with an unnormalized result. All multiply and divide operations require double length registers, hence the double precision instructions are executed by subroutines. These mnemonics (and some others in the set of sixty basic operations) are recognized by the assembler and replaced by the appropriate Link instruction. Alternatively, the actual codes for these instructions are recognized by the computer and cause an interrupt (number 2 interrupt). Software is provided to select the right subroutine. All compare, store, and store-rounded instructions leave the entire accumulator unchanged. The add, subtract, and store-rounded conditions generally result in bit (C) of the flag register being set. The carry flag (C) indicates that an arithmetic carry has been produced. Divide conditions can result in setting the overflow (V) flag (see Paragraph 2.8.1).

2.4 NOTATION

The following shorthand notation is used throughout this text. Some are assembly language conventions.

---

† *one level of indirect addressing assumed for illustrations.*

2.4.1 Addressing Conventions

E	effective address
(E)	contents of E
m	contents of address field of instruction word
/E, /m	specifies right half-word for 16-bit operands
E, E/, m, m/	specifies left half-word for 16-bit operands
OP = m	immediate address, *m* is a *literal*, a constant
OP m,X	indexing, X is an integer 1 to 7; E = m + (X) except Index Class instructions
OP* m	indirect addressing, E = (m)
OP* m,X	indexing plus indirect, E = (m + (X)) (Note: the * is part of the OP field)

2.4.2 Register Conventions

(A)	contents of the 16-bit A Register of the Accumulator
(AAE)	contents of the Extended Fixed-Point Accumulator
(AAF)	contents of the Floating-Point Accumulator
(AAFAD)	contents of the Double Precision Accumulator
($)	contents of the Save Register
($A)	contents of the 16-bit A portion of the Save Register
($AAE)	contents of the Extended Fixed-Point Save Register
($AAF)	contents of the Floating-Point Save Register
($AAFAD)	contents of the Double Precision Save Register
$(A_{0:7})$	contents of bits 0 to 7 in A Register
$(AD_{0,\ 9:23})$	contents of bits 0 and 9 to 23 in the AD Register
$\overline{(A_{0:7})}$	the one's complement of the contents of 0 to 7 of A
E_{32}	left Exec bit at effective address
E_{33}	right Exec bit at effective address
$A_{32:33}$	Accumulator Exec bits
int( )	the integer part of the floating-point operand. Note that for two's complement numbers the integer part is always the most positive integer that is more negative than the number, hence for negative numbers the magnitude of the integer part is larger than the magnitude of the number.
frac( )	the fractional part of the floating-point operand: frac(m) = $\lvert$ (m) - int(m) $\rvert$ is always a positive fraction

flt( )	the 16-bit operand converted (floated) to a floating-point number
sgn( )-ZGL	the *Zero, Greater than, Less than* flags are set according to value of the operand relative to zero
nrm( )	the normalized value of the floating-point operand

2.5 THE FIXED POINT INSTRUCTION CLASS

This is the most basic class of arithmetic instructions. No prefix is used before the basic mnemonics of AD, SB, etc., as in the other classes. The operand is always a 16-bit half-word. Two's complement binary arithmetic is performed. The state of the Z, G, L decision flags is determined by the resultant value of (A), "zero", "greater than" or "less than" zero after each operation (except CP as noted below). The addressing options *, /, and X are available for all instructions; and = is available for all but ST and SR. The *Save* option, $, may precede any of the mnemonics.

Load

CA	m,X	(E) ⟶ (A) (E_{32}) ⟶ (A_{32}) for m/ ø ⟶ $(AD_{1:8})$

Example:

The above reads as follows: Line 1; The contents of E (memory) replaces the contents of the A Register of the accumulator. Line 2; The left Exec bit at the effective address replaces bit 32 of the A Register for a left half-word. Line 3; bits 1 through 8 of the AD Register are replaced by ø.

Add

AD	m,X	(A) + (E) ⟶ (A) (E_{32})XOR(A_{32}) ⟶ (A_{32}) for m/

Clear, Subtract

CS	m,X	-(E) ⟶ (A) (E_{32}) ⟶ (A_{32}) for m/ ø ⟶ $(AD_{1:8})$

Subtract

SB	m,X	(A) - (E) ⟶ (A) (E_{32})XOR(A_{32}) ⟶ (A_{32}) for m/

Compare

CP	m,X sgn	[(A) - (E)] ⟶ ZGL flags $(A_{0:15,\ 32,\ 33})$ unchanged

Store

ST	m,X	(A) ⟶ (E) (A_{32}) ⟶ (E_{32}) for m/

Store, Rounded

SR	m,X	(A) + $\left\lvert (AE_1)2^{-15} \right\rvert$ ⟶ (E)

Multiply

MP	m,X	(A)x(E) ⟶ (AAE) ø ⟶ (AE_0) (AD) destroyed

Divide

DV	m,X	(AAE) ÷ (E) ⟶ (A) Remainder ⟶ (AE) (AD) destroyed

Clear, Divide

CD	m,X	ø ⟶ (AE) (AAE) ÷ (E) ⟶ (A) (AD) destroyed

2.5.1 The Save Register

The sections of the accumulator are duplicated in the Save Register. When the *save* option is exercised, the entire contents of the accumulator (except Exec

bits) are saved: (A, AE, AF, AD) → (\$A, \$AE, \$AF, \$AD) prior to execution of the instruction. The Save Register is assigned memory address number one. The actual core memory cell number one is not accessed by arithmetic instructions (except ST and SR), nor by Boolean connective instructions (except M type; see Paragraph 2.4, Boolean Connective Instructions). To gather data from the Save Register, the programmer writes either 1 or \$ in the address field (the assembler translates \$ to 1). Thus, CA1 and CA \$ result in (\$A) → (A). Since the save operation takes place first, \$AD \$ doubles the contents of A; \$SB \$ clears (A); \$MP \$ squares (A); and \$CS \$ inverts the sign of (A). In each case, the original contents of A is saved in \$A. It is not possible to store in the Save Register except by the \$ prefix; ST 1 and SR 1 both store in core memory location number one.

2.5.2 The Accumulator Address

In the same manner as above, the accumulator itself is assigned memory address zero for arithmetic instructions (which cannot access core memory location zero). This results in the following operations with the fixed point instructions: CA Ø does not change (A) but resets Z, G, L if, for example, CP were the previous instruction; CS Ø inverts the sign of (A); MP Ø squares (A); and CD Ø overflows. ST Ø does nothing and SR Ø rounds (A). The left/right *slash* option is not available for addresses zero and one: CA/1 and CA 1 are the same, as are SB /Ø and SB Ø. Neither (AE) nor (\$AE) are affected by such operations.

2.6 THE EXTENDED PRECISION INSTRUCTION CLASS

For each fixed-point instruction code and mnemonic there is a corresponding extended precision code and mnemonic. The former operate on 16-bit half-words and the latter on 32-bit whole words. All operations use fixed-point, two's complement, arithmetic. Execution of the extended precision instructions takes place in the AAE Register. The left half bits of the two halves of this register, A_0 and AE_0, are half-word sign bits. Therefore, the data word consists of sign plus 30 bits. The instructions, EMP, EDV, ECD, ECP, and ESR are not executed directly, but by programmed subroutines, since more than 32 bits of register are required. The subroutines may be entered in two ways. If the processor attempts to execute one of these instruction codes, the number two interrupt occurs, and either the Monitor or the Compat routine (EAI Publication Number 07 825 0046-0, see Preliminary Bulletin Program Information 66026) takes control and selects the proper subroutine. The more common method is for the Macro Assembler (EAI Publication Number 07 800.0001-3) to recognize the mnemonic code and substitute a *Link* instruction to the subroutine.

The state of the Z, G, L decision flags is determined by the resultant value of (AAE) after each operation. The addressing options * and X are available with all instructions in the class. The *immediate* option is available (for all but EST and ESR) indirectly through action of the Macro Assembler. The *immediate* option normally means that the address field of the instruction is treated as a *literal* (a 16-bit data word). However, since the extended class instructions operate on whole words this is not possible. The programmer may then use the = symbol to specify a 32-bit literal. During the assembly process, the literal value is placed in a special data storage area called the *Literal Pool* and the instruction address field is then given the data address.

The *Save* option may precede any of the mnemonics.

Load

ECA	m,X	(E) → (AAE)
		(AD) destroyed

Add

EAD	m,X	(AAE) ± (E) → (AAE)
		(AD) destroyed

Clear, Subtract

ECS	m,X	-(E) ⟶ (AAE) (AD) destroyed

Subtract

ESB	m,X	(AAE) - (E) ⟶ (AAE) (AD) destroyed

Compare

ECP	m,X	

Store

EST	m,X	(AAE) ⟶ (E)

Multiply

EMP	m,X	

Divide

EDV	m,X	

Clear, Divide

ECD	m,X	

Store, Rounded

ESR	m,X	

Address Zero and One are special addresses for the Accumulator and Save Register. Therefore, they are not normally used with the Extended Precision instructions, (AE) and ($AE) cannot be accessed in this manner. The result of EAD Ø is to add (A) and (AE).

2.7 THE INDEX INSTRUCTION CLASS

The same 16-bit fixed-point operations that take place in the A Register, may also be programmed for any index register. Thus, XCA m, 1 is equivalent to CA m. Note, however, that the X field of all of the index class instructions is used to select the register for action and *not* for address modification. This addressing option is not available for the index instructions; while * and / are available for all, and the *immediate* option (=) is available for all but XST and XSR. An index register must be specified.

Since index registers are restricted in size to 16 bits, operations requiring whole word registers (XMP, XDV, XCD, XSR) are performed by subroutines. Execution of the codes for these instructions results in the number two internal interrupt. The Macro Assembler substitutes a Link instruction for these mnemonics.

The state of the Z, G, L decision flags is determined by the resultant value of the specified index register, (X), or of the comparison (X)-(E). The *Save* option $, may precede any of these instructions; and (X) is saved in the A portion of the Save Register, rather than (A) being saved. However, in this case the rest of the accumulator is saved, i.e., $XCA m, 3 causes (X3, AE, AF, AD) to replace ($A, $AE, $AF, $AD) and the accumulator is unchanged.

Load

XCA	m,X	(E) ⟶ (X) then *sgn*(X) ⟶ ZGL

Add

XAD	m,X	(X) + (E) ⟶ (X) then *sgn* (X) ⟶ ZGL

Clear, Subtract

XCS	m,X	-(E) ⟶ (X) then *sgn* (X) ⟶ ZGL

Subtract

XSB	m,X	(X) - (E) ⟶ (X) then *sgn* (X) ⟶ ZGL

Compare

XCP	m,X	sgn (X) - (E)⟶ ZGL (X) unchanged

Store

XST	m,X	(X)⟶ (E)

Multiply

XMP	m,X

Divide

XDV	m,X

Clear, Divide

XCD	m,X

Store, Rounded

XSR	m,X

To clear an index register, the literal constant zero is loaded by using the *Immediate* option; XCA = ∅,X. The *Immediate* option is also used to add or subtract constants. The value of a variable just calculated in the accumulator is added to index register three by XAD ∅, 3. Note that in this case the zero is the effective address (specifically the address of A). Zero and one (or $) are used to access (A) and ($A) with all of the index instructions (except XST 1 and XSR 1). The *Slash* option has no effect with address zero and one. While XST ∅, 5 is used to move a half-word from X5 to the A Register, other means are used to move index register contents directly to other 16-bit registers. For example, LDAE = ∅, 5 loads AE from X5, LDF = ∅, 3 loads the Flag Register from X3, and LDOB = ∅, 2 moves (X2) to the output bus. In these cases the zero is a literal constant.

2.8 FLOATING POINT INSTRUCTION CLASS

This most important class of arithmetic instructions operates upon 32-bit data words, in the AAF portion of the accumulator. A floating number has a sign and a twenty-three bit mantissa, which are held in A and in the left, eight bit positions of AF (denoted as $AAF_{0:23}$ or as A, $AF_{0:7}$). The floating-point exponent has a sign and seven bits, held in $AF_{8:15}$. The range of magnitudes for floating-point numbers is between 2^{128} and 2^{-128}; operations that result in larger or smaller magnitudes cause an exponent fault interrupt. A subroutine is used to normally set the accumulator to zero or the largest possible value. The value of zero is represented by 32 zero bits; however, if a word with a zero mantissa and a non-zero exponent is loaded it is treated as zero.

The basic floating-point arithmetic instructions (FAD, FSB, FMP, FDV) perform normalized arithmetic. Normalization means that the mantissa of the operation result is shifted to the left to eliminate any leading zero bits. The exponent is then decremented once for each bit position shifted. FCA and FCS also normalize the number loaded into the accumulator. FSR (floating store rounded) normalizes *after* rounding and *before* storing. FST (floating store) does not normalize.

All floating instructions (except compare) may be used with the *Unnormalize* option by appending the letter U to the end of the mnemonic. The effect is to inhibit the post-normalization operation. Unnormalized data may then be considered as 24-bit, fixed-point data with assigned scale factors (the exponents). When adding and subtracting, the exponents are automatically adjusted to agree with the larger exponent. For multiplication and division, the exponents are added or subtracted. Care must be taken to avoid overflow on division. Note that FCAU loads a copy of (E) without normalizing, and is therefore used with FSTU to move 32-bit words from one core

location to another. See Paragraph 2.7 on moving Exec bits.

Multiple level indirect addressing (* option) and indexing is possible with all instructions; and as explained in Paragraph 2.3.3, the *Immediate* (=) option may be employed indirectly through the use of the Macro Assembler Literal Pool feature. The *Save* option ($) may be exercised with all instructions to cause (AAF) to be stored in $AAF prior to the operation. As with all arithmetic instructions, the ZGL flags are set by the result of each operation.

Load AAF

FCAU	m,X	(E) $\longrightarrow$ (AAF) $(E_{32:33}) \longrightarrow (A_{32:33})$

Clear, Add, Normalize

FCA	m,X	*nrm* (E) $\longrightarrow$ (AAF)

Add

FAD	m,X	*nrm* [(AAF) + (E)] $\longrightarrow$ (AAF)
FADU	m,X	(AAF) - (E) $\longrightarrow$ (AAF)

Clear, Subtract

FCS	m,X	*nrm*[-(E)] $\longrightarrow$ (AAF)
FCSU	m,X	$(E_{32:33}) \longrightarrow (A_{32:33})$

Subtract

FSB	m,X	*nrm* [(AAF) - (E)] $\longrightarrow$ (AAF)
FSBU	m,X	(AAF) - (E) $\longrightarrow$ (AAF)

Compare

FCP	m,X	*sgn* [(AAF) - (E)] $\longrightarrow$ ZGL

Store

FSTU	m,X	(AAF) $\longrightarrow$ (E) $(A_{32:33}) \longrightarrow (E_{32:33})$ FST = FSTU

Multiply

FMP	m,X	*nrm*[(AAF) x (E)] $\longrightarrow$ (AAFAD) (AE) destroyed
FMPU	m,X	(AAF)x(E) $\longrightarrow$ (AAFAD) (AE) destroyed

Divide

FDV	m,X	*nrm* [(AAFAD) ÷ (E)] (AAF) mantissa of Rem. $\longrightarrow$ (AD) (AE) destroyed
FDVU	m,X	(AAFAD) ÷ (E) $\longrightarrow$ (AAF) mantissa of Rem. $\longrightarrow$ (AD) (AE) destroyed

Clear, Divide

FCD FCDU	m,X	$\emptyset \longrightarrow$ (AD) then execute FDV or FDVU

Store, Rounded

FSR	m,X	*nrm* $\left[(AAF) + \left\lvert (AD_1)2^{-23} \right\rvert\right]$ (E) (AAF) unchanged
FSRU	m,X	(AAF) + $\left\lvert (AD_1)2^{-23} \right\rvert$ (E) (AAF) unchanged

2.8.1 Floating Divide

Special characteristics of FDV, FCD must be mentioned (also true for IDV, ICD). Overflow will occur if the mantissa of the divisor is less than half the mantissa of the dividend. This will not occur if the divisor is the result of a normalized operation.

The result of a floating divide operation is to leave the quotient in AAF and the 24-bit mantissa of the remainder in AD. Note, however, that the 24-bit AD cannot hold the exponent of the exponent of the remainder. The full remainder, as a proper floating point number, can only be recovered by deduction.

2.8.2 Floating Multiply

The result of a floating multiply is a number with sign bit, 46 mantissa bits, and an 8-bit exponent, that is held in the AAFAD register. This number is converted into the double word format by separating the mantissa into two 23-bit segments, and adding a positive sign bit for the lower half. An exponent is created for the lower half which is 23 less than the upper half exponent ($AF_{8:15}$). The Universal Accumulator holds the lower half mantissa in $AD_{1:23}$ and provides for the extra sign bit, AD_0. The extra exponent for the lower half is not required in the accumulator and is created only upon execution of a double floating store instruction. (AD_0) does not enter into any mathematical operations except for holding the sign of the division remainder mantissa.

After the execution of FMP, the extra sign bit AD_0 is reset to zero. Any two's complement fraction consisting of bits truncated from a larger number (either sign) is itself a positive number (with an exponent). Resetting AD_0 makes it possible to treat the lower half of the product in normal floating-point format.

The double precision product is often used in an otherwise single precision computation. For example, in the calculation ($y = \sum_{n} a_i b_i$), it is useful to accumulate the double precision results of each multiplication as illustrated in this small routine.

```
XCA     =N, 3
DCA     ZERO        initialize
$FCA    A, 3        save partial sum
FMP     B, 3        a x b product
DAD     $           add partial sum
XJT     *-3, 3, -1  index and loop
FST     Y           store single precision result
```

2.9 THE DOUBLE PRECISION INSTRUCTION CLASS

Double precision floating-point instructions operate on data consisting of a sign, 46 bits of mantissa and an 8-bit exponent. This data is held in memory as two successive 32-bit words, *each in proper 32-bit floating-point format.* That is, each word has a sign, a 23-bit mantissa, and an 8-bit exponent. Half of the double precision mantissa magnitude bits are in each word. The second sign and exponent are redundant from the point of view of the accumulator. However, for multiple precision calculations by subroutine, it is convenient to have each half of the double word in single precision format. Note that the second sign is always positive and the second exponent is 23 less than the first. Double precision operations take place in the 46-bit AAFAD register; $AF_{8:15}$ hold the exponent. The second exponent is created upon execution of double store (DST).

All operations are performed in the same manner, only with the 46-bit mantissa, and with the same options and restructions as the single precision floating-point operations. Double multiply, divide, compare, and store-rounded must be executed by subroutine. If the immediate addressing option is used, the Literal Pool feature allocates two memory cells from the data specified in the address field.

The instruction pair, DCAU and DST, may be used to move double precision data (both words) *with exec bits* from one memory location to another via the accumulator.

Load AAFAD

DCAU	m,X	$(E, E + 1_{0:23}) \longrightarrow (AAFAD)$ $(E_{32:33}) \longrightarrow (A_{32:33})$ (AE) destroyed

Clear, Add, Normalize

DCA	m,X	*nrm* $(E, E + 1_{0:23}) \longrightarrow$ (AAFAD) (AE) destroyed

Add

DAD	m,X	*nrm* $[(AAFAD) + (E, E, + 1_{0:23})]$ $\longrightarrow$ (AAFAD) (AE) destroyed
DADU	m,X	(AAFAD) + $(E, E + 1_{0:23})$ $\longrightarrow$ (AAFAD) (AE) destroyed

Clear, Subtract

DCS	m,X	-*nrm* $(E, E + 1_{0:23}) \longrightarrow$ (AAFAD) (AE) destroyed
DCSU	m,X	- $(E, E + 1_{0:23}) \longrightarrow$ (AAFAD) $(E_{32:33}) \longrightarrow (A_{32:33})$ (AE) destroyed

Subtract

DSB	m,X	*nrm* $[(AAFAD) - (E, E + 1_{0:23})]$ $\longrightarrow$ (AAFAD) (AE) destroyed
DSBU	m,X	(AAFAD) - $(E, E + 1_{0:23})$ $\longrightarrow$ (AAFAD) (AE) destroyed

Compare

DCP	m,X	

Store

DSTU	m,X	$(AAFAD) \longrightarrow (E, E + 1_{0:23})$ $(AF_{0:15}) - 23 \longrightarrow E + 1_{24:31}$ $(A_{32:33}) \longrightarrow (E_{32:33})$ DST = DSTU

Multiply

DMP	m,X	
DMPU	m,X	

Divide

DDV	m,X	
DDVU	m,X	

Clear, Divide

DCD	m,X	
DCDU	m,X	

Store, Rounded

DSR	m,X	
DSRU	m,X	

2.10 THE INTEGER INSTRUCTION CLASS

This unique set of instructions performs floating-point arithmetic operations on *fixed-point* and floating-point data. The data operand in the accumulator is always a floating-point number (AAF or AFFAD). The data operand specified by the effective address is always a 16-bit fixed-point data word. Each instruction (except the store commands) first gathers the 16-bit operand, converts it to a floating-point number, and then executes a normal floating-point operation. The conversion process is termed floating and the reverse process, upon storing, is called integerizing.

It is important to understand exactly what happens during floating and integerizing; other features (except half-word address option) of the ten instructions of this class are the same as those of the Floating Point Class.

2.10.1 Floating

A 16-bit data word is normally thought of as a fixed-point fraction, with the binary point adjacent to the sign bit. Fixed-point multiply and divide instructions perform fractional arithmetic. On the other hand, a 16-bit data word may be considered to be a fixed-point integer, with the binary point to the far right-hand position. This is of course the common practice when operating on memory address numbers.

No confusion occurs provided a correct scale factor is used when the data word enters an arithmetic operation. The floating of a 16-bit word by an Integer Class Instruction causes the contents of the half-word effective address to be loaded into A (for Integer-clear-add, ACA), zeroes to be loaded in $AF_{0:7}$, and exponent of +15 to be loaded in $AF_{8:15}$. When in memory, the operand is considered an integer, hence a scale factor of 2^{15} is entered in AAF as the number itself becomes the mantissa (a fraction) of a floating point number. The least significant eight bits of the mantissa are made zero. After this conversion, the rest of ICA is simply to normalize (AAF). In the case of IAD, ISB, IMP, etc., the above conversion is performed before presenting the floated operand to the accumulator.

2.10.2 Integerizing

The integer part of the result of any floating-point calculation (by floating, double precision, or integer class instruction) can be stored as 16-bit fixed-point integers provided the magnitude of the number is less than 2^{16}. The IST instruction (same as ISTU) first causes the contents of AAF to be shifted left or right as needed, bringing the exponent to +15. When $(AF_{8:15}) = +15$, the binary point may be considered to be to the right of A_{15}, hence (A) is the 16-bit integer part of the original contents of AAF. Then (A) is stored at the effective address and AAF is restored to its original state. If the magnitude of (AAF) is equal to or greater than 2^{16} an overflow will occur, and an incorrect number will be stored.

The half-word addressing options (= 1) are available, as well as indirect addressing and indexing. Addresses *zero* and *one* refer to (A) and ($A). The same rules for the *unnormalize* (U) option and divide overflow apply as for floating-point instructions. Use of the *Save* options will *save* (AAF).

Load, Integer

ICAU	m,X	*flt*(E) → (AAF) *which means:* (E) → $(A_{0:15})$ ø → $(AF_{0:17})$ +15 → $(AF_{8:15})$ = Exponent

Clear, Add, Normalize

ICA	m,X	*nrm* [flt(E)] → (AAF)

Add

IAD	m,X	*nrm* [(AAF) + flt(E)] → (AAF)
IADU	m,X	(AAF) + flt (E) → (AAF)

Clear, Subtract

ICS	m,X	*nrm* [-flt(E)] → (AAF)
ICSU	m,X	-flt(E) → (AAF)

Subtract

ISB	m,X	*nrm*[(AAF) - flt(E)] → (AAF)
ISBU	m,X	(AAF) - flt(E) → (AAF)

Compare

ICP	m,X	(AAF) - flt(E) → ZGL (AAF) unchanged

Store

ISTU	m,X	*int*(AAF) ⟶ (E) (AAF) unchanged IST = ISTU

Multiply

IMP	m,X	*nrm* [(AAF) x flt(E)] ⟶ (AAFAD) (AE) destroyed
IMPU	m,X	(AAF) x flt(E) ⟶ (AAFAD) (AE) destroyed

Divide

IDV	m,X	*nrm* [(AAFAD) ÷ flt(E)] ⟶ (AAF) (AE) destroyed
IDVU	m,X	(AAFAD) ÷ flt(E) ⟶ (AAF) (AE) destroyed

Clear, Divide

ICD	m,X	(Ø ⟶ (AD)) *then execute FDV, FDVU* (AE) destroyed
ICDU	m,X	

Store, Rounded

ISRU	m,X	*int*(AAF) + Δ ⟶ (E) where Δ = Ø if *frac* (AAF)<1/2 = 1 if *frac* (AAF)>1/2 (AAF) unchanged ISR = ISRU

Integer arithmetic instructions may be mixed freely within floating computations to save execution time and memory space, particularly with the use of immediate addressing. Thus, the Fortran statement, A = 3B + I might be implemented by the code:

```
FCA     B
IMP     =3
IAD     =I
FST     A
```

Where A and B are floating-point numbers, and I, a Fortran integer, is stored as a fixed-point number. Care must be taken with IDV and ICD, for the floating operand (divisor) is an unnormalized number and if the resultant mantissa is less than half the accumulator mantissa overflow occurs.

A characteristic of negative two's complement numbers, as noted elsewhere, is that when least significant bits are truncated, the result is greater in magnitude (more net negative) than the untruncated number. IST truncates the fractional part of (AAF) before storing and yields for negative numbers the "next more negative integer". ISR yields the nearest integer for both signs. The pairs; IST and ISTU, ISR and ISRU, each produce identical codes, no normalization is performed.

2.11 BOOLEAN CONNECTIVE INSTRUCTIONS

If an accumulator (a) bit and a memory (m) bit are considered as arguments of a logical function to form a resultant bit, r. Then there are sixteen possible functions that may be performed.

Table 2.3 gives the function as well as showing the four possible values of r for each combination of a and m. Basic operations used in this table is indicated by:

1. The over-bar: logical inversion of the logical value. (ONE to ZERO and ZERO to ONE)

Table 2.3. Boolean Connective Functions

a =	0	1	0	1	
m =	0	0	1	1	
r =	1	1	1	1	Set r = 1 regardless of $\underline{a}$ and $\underline{m}$
	0	0	0	0	Reset r = 0 regardless of $\underline{a}$ and $\underline{m}$
	0	1	0	1	r = a
	0	0	1	1	r = m
	1	0	1	0	$r = \overline{a}$
	1	1	0	0	$r = \overline{m}$
	0	1	1	1	r = a + m OR function
	0	0	0	1	r = a x m AND function
	1	0	0	0	$r = \overline{a + m}$ NOR function
	1	1	1	0	$r = \overline{a \times m}$ NAND function
	0	1	1	0	$r = (a + m)\ \overline{(a \times m)}$ Exclusive OR function
	1	0	0	1	$r = (a \times m) + \overline{(a \times m)}$ EQUIVALENCE function
	1	0	1	1	$r = \overline{a} + m = a \times \overline{m}$
	0	0	1	0	$r = \overline{a} \times m = \overline{a + m}$
	0	1	0	0	$r = \overline{\overline{a} + m} = a \times \overline{m}$
	1	1	0	1	$r = \overline{\overline{a} \times m} = a + \overline{m}$

2. OR: Result is a ONE if either argument is a ONE.

3. AND: Result is ONE only if both arguments are ONE.

4. NOR: Result is ONE if neither argument is ONE.

5. NAND: Result is ONE if either argument is ZERO.

6. Exclusive OR: Result is ONE if the argument values are different.

7. Equivalence: Result is ONE if argument values are the same.

The last four functions are performed by first complementing *a* and then executing the OR, AND, NOR or NAND function.

These are thirty-two basic Boolean connective instructions, two for each of the sixteen functions listed in Table 2.3. One instruction stores the result *r* in the original location of *a*. The other instruction stores *r* in the memory location *m*.

Variables *a*, *m*, and *r* are not restricted to single bits but may be in bytes. If bytes are used, the byte sizes of the three variables are all the same. The sixteen functions are performed upon individual pairs of bits within the bytes both simultaneously and independently. The byte size is specified (1, 2, 4, 8 or 16 bits) by writing the respective digit after the instruction mnemonic in the OP code field. A byte size of 16 is implied if no digits are specified. For example, RA1 resets a single bit in the accumulator; RA8 resets 8 bits; and RA resets all 16 bits of the A register.

For byte sizes of 1, 2, 4, or 8 bits, the byte position within the half-word is specified in the count field (third subfield of the address field) by a decimal number 0 to 15. For example, RM1, m, X, ∅ resets the sign bit of (E); RM1, m, X, 15 resets (E_{15}). The positions of larger bytes are denoted by numbers 0 to 7 for two bits, by 0 to 3 for four bits and 0 to 1 for eight bits. Therefore, RM8, m, X resets ($E_{8:15}$).

2.11.1 The Mnemonics

The thirty-two instruction mnemonics indicate the logical action performed and the designation of the

result. They do NOT indicate the name of the Boolean function. If the mnemonic ends in "A", the result is placed in the accumulator. If the mnemonic ends in "M", the result is put in memory. The mnemonic codes for the 16 pairs of instructions appear as shown in Table 2.4.

Table 2.4. Boolean Mnemonics

Destination		Condition†
Accumulator	Memory	
SA	SM	Set
RA	RM	Reset
≠	AHM	Accumulator High
ALA	ALM	Accumulator Low
MLA	MLM	Memory Low
EHA	EHM	Either High (OR)
BHA	BHM	Both High (AND)
BLA	BLM	Both Low (NOR)
ELA	ELM	Either Low (NAND)
BDA	BDM	Both Different (XOR)
BSA	BSM	Both Same (EQU)
CEHA	CEHM	Comp. A, Either Hi
CBHA	CBHM	Comp. A, Both Hi
CBLA	CBLM	Comp. A, Both Lo
CELA	CELM	Comp. A, Either Lo

† *The expression in the right column describes the condition for which a bit in the result is set to ONE; all other conditions produce a ZERO bit in the result.*

2.11.1.1 *Examples*

1. *MLA4 m,X,3* Memory low to accumulator; 4-bit byte; byte position 4; which means the memory bits $E_{12:15}$ are inspected. For each ZERO bit (low) the corresponding bit within $A_{12:15}$ is set to ONE and the other bits of $A_{12:15}$ are reset to ZERO. This is a byte load instruction.

2. *CEHM8 m,X,Ø* Complement A, Either High to Memory; 8-bit byte; first byte position; which means the complement of $(A_{0:7})$ is compared to $(E_{0:7})$, bit by bit. If either bit of a pair is high, the corresponding bit position in memory is set high, otherwise it remains low.

2.11.1.2 ≠ *Special Cases (BEQT Replacing AHA, MHM).* The two codes accumulator high to accumulator (AHA) and memory high to memory (MHM) appear to do nothing. This is nearly true, for the specified byte is simply restored (unchanged), to its original location. However, MHM is a useful ZERO byte test, since the Z flag is set if the result of the logical function is a byte of all ZEROES. The only action of MHM is to indicate a ZERO byte in memory which, if E = 0, refers to the accumulator. The AHA code is therefore redundant and is replaced by the more useful Byte Equality Test (BEQT) for which the byte size and position options are the same as above. The action of BEQT is to set the Z flag if *and only if* all accumulator and memory bytes are the same.

2.11.2 Addressing

Indirect addressing and half-word addressing options are valid for Boolean instructions, however immediate addressing is not possible with M-type

instructions. An effective address of ZERO refers to the accumulator. This results in each A and M pair of instructions being equivalent. An effective address of ONE refers to core memory cell number one for M-type instructions. It does not refer to $A register since storing in the Save Register is only possible by means of the $ prefix operator. See Paragraph 2.3.2.

In the following table of instructions, all of the possible combinations of byte sizes and byte positions are given by way of example. Any instruction may address any of the bit combinations illustrated.

In the following examples, if the result of the logical operations yields a byte (in the referenced position) consisting of all zeroes, the Z flag is set; otherwise it is reset.

Table 2.5. Boolean Instruction

set and reset		*(one bit/byte)*
SA1 Ø,Ø,Ø	$1 \longrightarrow (A_0)$ reset Z	
SM1 m,X,1	$1 \longrightarrow (E_1)$ reset Z	
RA1 Ø,Ø,2	$Ø \longrightarrow (A_2)$ set Z	
RM1 m,X,3	$Ø \longrightarrow (E_3)$ set Z	
≠ *byte equality test*		*(one bit/byte)*
BEQT1 m,X,4	if $(A_4) = (E_4)$ set Z, if ≠ reset Z	
store and load		*(one bit/byte)*
AHM1 m,X,5	$(A_5) \longrightarrow (E_5)$	
MHA1 m,X,6	$(E_6) \longrightarrow (A_6)$	
zero byte test		*(one bit/byte)*
MHM1 m,X,7	if $(E_7) = Ø$ set Z, if ≠ reset Z	
complement accumulator		*(one bit/byte)*
ALA1 Ø,Ø,8	$(A_8) \longrightarrow (A_8)$	
store and load complement		*(one bit/byte)*
ALM1 m,X,9	$(A_9) \longrightarrow (E_9)$	
MLA1 m,X,10	$(E_{10}) \longrightarrow (A_{10})$	
complement memory		*(one bit/byte)*
MLM1 m,X,11	$(E_{11}) \longrightarrow (E_{11})$	

Table 2.5. Boolean Instruction (Cont)

OR, AND	*(one bit/byte)*
EHA1 m,X,12	(A_{12}) OR $(E_{12}) \longrightarrow (A_{12})$
EHM1 m,X,13	(A_{13}) OR $(E_{13}) \longrightarrow (E_{13})$
BHA1 m,X,14	(A_{14}) AND $(E_{14}) \longrightarrow (A_{14})$
BHM1 m,X,15	(A_{15}) AND $(E_{15}) \longrightarrow (E_{15})$
NOR, NAND	*(2 bits/byte)*
BLA2 m,X,Ø	$(A_{0:1})$ NOR $(E_{0:1}) \longrightarrow (A_{0:1})$
BLM2 m,X,1	$(A_{2:3})$ NOR $(E_{2:3}) \longrightarrow (E_{2:3})$
ELA2 m,X,2	$(A_{4:5})$ NAND $(E_{4:5}) \longrightarrow (A_{4:5})$
ELM2 m,X,3	$(A_{6:7})$ NAND $(E_{6:7}) \longrightarrow (E_{6:7})$
XOR, EQU	*(2 bits/byte)*
BDA2 m,X,4	$(A_{8:9})$ XOR $(E_{8:9}) \longrightarrow (A_{8:9})$
BDM2 m,X,5	$(A_{10:11})$ XOR $(E_{10:11}) \longrightarrow (E_{10:11})$
BSA2 m,X,6	$(A_{12:13})$ EQU $(E_{12:13}) \longrightarrow (A_{12:13})$
BSM2 m,X,7	$(A_{14:15})$ EQU $(E_{14:15}) \longrightarrow (E_{14:15})$
complement A, then OR, AND	*(4 bits/byte)*
CEHA4 m,X,Ø	$(\overline{A_{0:3}})$ OR $(E_{0:3}) \longrightarrow (A_{0:3})$
CEHM4 m,X,1	$(\overline{A_{4:7}})$ OR $(E_{4:7}) \longrightarrow (E_{4:7})$
CBHA4 m,X,2	$(\overline{A_{8:11}})$ AND $(E_{8:11}) \longrightarrow (A_{8:11})$
CBHM4 m,X,3	$(\overline{A_{12:15}})$ AND $(E_{12:15}) \longrightarrow (E_{12:15})$
complement A, then NOR, NAND	*(8 and 16 bits/byte)*
CBLA8 m,X,Ø	$(A_{0:7})$ NOR $(E_{0:7}) \longrightarrow (A_{0:7})$
CBLM8 m,X,1	$(A_{8:15})$ NOR $(E_{8:15}) \longrightarrow (E_{8:15})$
CELA m,X,Ø	$(A_{0:15})$ NAND $(E_{0:15}) \longrightarrow (A_{0:15})$
CELM m,X,Ø	$(A_{0:15})$ NAND $(E_{0:15}) \longrightarrow (E_{0:15})$

2.12 CONDITIONAL INSTRUCTIONS

2.12.1 The Flag Operations

The execution of each of the following instructions

HJc	halt and jump to E
EXc	execute and instruction at E
Lc	link to E
LRc	reset flag, link to E
JTc	trigger (complement) flag, jump to E
JSc	set flag, jump to E
JRc	reset flag, jump to E
Jc	jump E

is conditional upon the state of the flag indicated by c; c can refer to any flag of the flag register in its set or reset state.

Example: HJ1 START

The halt will occur, with subsequent transfer to START, only if flag 1 is set. If it is not set, no halt occurs, and transfer to the instruction following HJ1 takes place. On the other hand, the reverse applies to the example:

HJN1 START

that is, the halt will occur if flag 1 is not set, etc.

LR, JT, JS, JR cause the indicated change to the referenced flag whether the instruction is executed or not; the instruction is executed conditional to the present state of the flag. If unconditional execution of the instruction is required, c is left blank.

If unconditional non-execution of the instruction is required for creating a class of no-operations, c could be set to N. For this purpose HJN, EXN, LN and JN would serve. Of these the last, JN, has been selected in the assembler as the non-operation instruction, NOP.

The condition code c, may have thirty-two values; these are the 16 codes for the bits of the flag register and the same codes prefixed by N to denote the inverse condition. They are:

Z, G, L, V, C, B,

1, 2, 3, 4, 5, 6, 7, 8

N, NZ, NG, NL, NV, NC, NB, NE

N1, N2, N3, N4, N5, N6, N7, N8

Normal indirect addressing is available with all instructions.

Halt

†HJc m,X *user* mode:
if (c) = 0, NOP; if (c) = 1, interrupt no. 2

monitor mode:
if (c) = 0, NOP; if (c) = 1, computer halts with (E) ⟶ (L)

Execute

EXc m,X if (c) = 0, NOP; if (c) = 1, execute the instruction located at E

Link

Lc m,X if (c) = 0, NOP; if (c) = 1, (L + 1) ⟶ (E), (E + 1) ⟶ (L)

LRc m,X if (c) = 0, reset flag; if (c) = 1, reset flag, (L + D ⟶ (E)), (E + 1) ⟶ (L)

† *Privileged instructions, see Chapter 3.*

Jump

Jc	m,X	if (c) = 0, NOP; if (c) = 1, (E) ⟶ (L)
JRc	m,X	if (c) = 0, reset flag; if (c) = 1, reset flag, (E) ⟶ (L)
JSc	m,X	if (c) = 0, set flag; if (c) = 1, set flag, (E) ⟶ (L)
JTc	m,X	if (c) = 0, trigger (complement) flag; if (c) = 1, trigger flag, (E) ⟶ (L)

2.12.2 Index Jumps XJ, XJT

2.12.2.1 *Unconditional Jump* - XJ m,X,Δ. The specified index register X is algebraically incremented by the value Δ, and transfer takes place to location m (NOT to location m, c).

Δ is accorded 8 bits (8-15) in the instruction and is interpreted as (-) if bit 8 is "1", or (+) if bit 8 is "0". Thus, $-128 \leq \Delta \leq 127$.

2.12.2.2 *Conditional Index Jump* - XJT, m,X,Δ. The contents of index register X are algebraically incremented by Δ, and if the sign of the resultant index register contents is found to be the same as that of Δ, transfer takes place to the instruction following the XJT instruction. If these differ, transfer takes place to m. Δ has the same significance as described above.

The specified index register is not used in the effective address determination. If indirect addressing is used with XJ or XJT, address modification by indexing does not take place at any level.

index, jump

XJ	m,X,Δ	(X) + Δ ⟶ (X), $-128 \leq \Delta \leq 127$ E ⟶ (L)

index, jump test

XJT	m,X,Δ	(X) + Δ ⟶ (X), $-128 \leq \Delta \leq 127$ then if sgn (X) = sgn Δ, (L + 1) ⟶ (L); if ≠, E ⟶ (L)

2.13 INSTRUCTIONS TO LOAD AND STORE SPECIAL REGISTERS

Each of the several 16-bit registers in the 8400 and the 16-bit input and output busses are serviced by a pair of instructions that gather and store a half-word from either half of the memory word. The addressing options *, /, and X apply in each case and the *Immediate* option (=) is available for all load instructions. Exec bits are not affected or moved by these instructions, and except for the direct effect on the flag register with LDF, there is no associated change to the flag register. The 32-bit data channel control registers (CCR) are serviced by a similar pair of instructions; however, the half-word addressing options = and / do not apply. In the case of the 16-bit channel data registers (CDR), a half-word and Exec bit are moved between memory and the registers; however, the = option does not apply and the *Left-Right* option is determined by another register (CFR) in the data channel. See 4.1.

2.13.1 Load Register or Bus

LDAE	m,X	(E) ⟶ (AE) *external accumulator register* (AE) destroyed
LDF	m,X	(E) ⟶ (F) *flag register*
LDL	m,X	(E) ⟶ (L) *location counter*
LDT	m,X	(E) ⟶ (T) *timer register*
LDM	m,X	(E) ⟶ (M) *interval interrupt mask register*
LDE	m,X	(E) ⟶ (EM) *external interrupt mask register*

LDC	m,X	(E) → (C) console register
LDOB	m,X,R	(E) → Output bus R where $R \leq 17_8$
LDCC	m,X,k	$(E_{0:31})$ → (CCRk) *channel control register k*
LDCD	m,X,k	$(E_{0:15,\ 32})$ → (CDRk) or $(E_{16:\ 31,\ 33})$ (CDRk) *channel data register k according to the L bit of channel function register*

2.13.2 Store from Register or Bus

STAE	m,X	(AE) → (E)
STF	m,X	(F) → (E)
STL	m,X	(L) → (E)
STT	m,X	(T) → (E)
STM	m,X	(M) → (E)
STE	m,X	(EM) → (E)
STC	m,X	(C) → (E)
†STIB	m,X,R	Input Bus R → (E)
†STCC	m,X,k	(CCRk) → $(E_{0:31})$
†STCD	m,X,k	(CDRk) → $(E_{0:15,\ 32})$ or $(E_{16:31,\ 33})$ *according to the L bit of channel function register*

All I/O instructions plus those that modify the limiter, console, mask register are privileged instructions and in user mode they are not executed by activate the internal interrupt number two (see 3.6.1). In a computer lacking an automatic data channel LDCC and STCC cause this interrupt in *monitor* mode as well.

†*Privileged instructions*

There are 15 bus addresses, R, which are written as octal numbers, and eight channel numbers, k (Ø through 7). When indexing is not employed, these instructions must be written with two commas, e.g., LDOB m,,1Ø loads output bus number 8. Note that LDOB = Ø,3,Ø loads bus zero from index register three.

2.13.3 The Flag Register

The sixteen bits of the flag register, F, are moved as a group, although they are set and reset individually by other instructions and machine functions. The zero bit of F is not really a flag and cannot be reset; it is always set. Hence, when tested by conditional instructions, a positive test always results (see Paragraph 2.12).

Bits $F_{1:30}$ are the Z (zero), G (greater than), and L (less than) flags that are set to the sign of A after each arithmetic instruction, except the index class. For the index class, they are set by the sign resulting from the index arithmetic operation. The Boolean connective instructions also set and reset flag Z. Bit F_4 is V (overflow flag) which remains set, until reset. Bit F_5 is C (carry flag) which indicates if a carry from the accumulator occurred with the last arithmetic instruction. Bit F_6 is B (busy flag) which indicates whether or not the last I/O instruction was executed. Bit F_7 is E (enable flag), enables the entire interrupt system when set. This flag cannot be changed in *user* mode. (See 3.6.1.). Register bits $F_{8:15}$ are eight, general purpose flags available to the programmer (called flag 1, 2, ... flag 8). These eight flags can be set and reset manually at the computer console.

2.13.4 Location Counter

The LDL instruction is quite equivalent to the jump instruction, but there are differences in addressing

options. Note the following functionally equivalent pairs:

LDL	=m,X	J	m,X
LDL	m/,X	J*	m,X to one level of indirectness

There are no direct equivalents of:

LDL*	m/,X	LDL*	/m,X
LDL*	-m/,X	LDL*	=/m,X
		LDL	/m,X

Thus, LDL can jump direct or indirect to addresses stored in right half-words. LDL*m,X reaches to one more level of indirectness than does J*m,X. Note that the *Indirect-immediate* option is possible (LDL* = m,X). This differs from LDLm,X only when the contents of the effective address m,X specifies indirect or index modification.

2.13.5 Timer

The timer may be read, but not changed in *user* mode. The operation is such that the contents of T is decremented once every millisecond. When (T) reaches zero, internal interrupt number six is activated (see Paragraph 3.6). This interrupt forces the mode into *monitor* mode and then the T register is reloaded by the monitor program.

2.13.6 Mask Register

The internal mask register (M) and the external mask register (EM or E) contain 16 bits each, which individually enable interrupt lines. These registers cannot be modified in *user* mode. See Chapter 3.

2.13.7 Console Register

Instructions LDC and STC are privileged instructions, and can be performed only in *monitor* mode. The contents of this register can be set and reset manually at the computer console.

LDOB, LDCC, LDCD, STIB, STCC, STCD Instructions

These instructions are treated in the Input/Output Instructions found in Paragraph 2.14.3.

2.14 EXEC BIT INSTRUCTIONS

Every memory word (in core memory and on the Rapid Access Drum) contains 32 data bits plus two special bits that are identified as *left* and *right* Exec bits. The left Exec bit (E_L) is in position 32, and the right Exec bit (E_R) is in position 33, however, E_L is always associated with the left half-word (bits 0:15), and E_R with bits 16:31. E_L is used by all system programs to designate a relocatable address in the left half-word. E_R is primarily used to protect a word located in core memory. E_R operates with the interrupt system to interrupt the computer when an illegal reference is made to the memory cell in *user* mode. See Paragraph 3.6.1.

Exec bits are set, reset, and tested respectively by the three instructions: SEX, REX, TEX. In each case, the contents, other than E_L or E_R, of the effective address is ignored. The *slash* option specifies left or right Exec bit; for example, REX m, X resets E_L and REX /m,X resets E_R. The test instruction sets the Z flag if the specified exec bit is set.

2.14.1 Exec Bit Controls

SEX m,X	$1 \rightarrow (E_{32})$	set E_L
SEX /m,X	$1 \rightarrow (E_{33})$	set E_R
REX m,X	$\emptyset \rightarrow (E_{32})$	reset E_L
REX /m,X	$\emptyset \rightarrow (E_{33})$	reset E_R
TEX m,X	$(E_{32}) \rightarrow$ Z flag	test E_L
TEX /m,X	$(E_{33}) \rightarrow$ Z flag	test E_R

2.14.2 Accumulator Exec Bits

The accumulator is the only register that accepts Exec bits. Bit A_{32} is associated with the E_L of a left half-word loaded from memory. Bit A_{33} is associated with the E_R of a whole word loaded into AAF. A_{32} and A_{33} are addressed by the above instructions with an effective address of zero.

The accumulator Exec bits each exist for a specific purpose. A_{33} is used when a word is to be moved with its protection bit E_r from one core location to another. The instruction pair, FCAU, FST, moves all 34 bits from core to core. Similarly, DCAU, DST is used to move double precision data (and must not be used to move pairs of words of any other type of data). All arithmetic instructions (except FCAU and DCAU) and all Boolean instructions that change the contents of the accumulator reset A_{33}. Any instruction that does not change the contents of the accumulator does not alter (A_{33}). Only FST and DST store (A_{33}) in memory.

The accumulator left Exec bit, A_{32}, is used to preserve the relocation information throughout address calculations, as follows. The difference of two relocatable address values must be an absolute value; the sum of such numbers, however, is not defined in these terms. The sum or difference between an absolute value and a relocatable one must be relocatable. Therefore, after either of the instructions AD m/,X and SB m/,X (note: *left half* option only), the contents of A_{32} is the exclusive OR of (E_{32}) with the initial (A_{32}). A_{32} is located from memory only by the instructions: FCAU, FCSU, DCAU, DCSU, CA (left half), and CS (left half). All other arithmetic and Boolean instructions that change the contents of the accumulator, reset A_{32}. All others that do not change (AAFAD), do not alter A_{32}. Only the instructions FST, DST, ST (left half) store A_{32} in memory.

A_{32} and A_{33} are not saved by the *Save* Register option. Index registers do not contain bits for holding relocated Exec bits; however, by programming system conventions, index registers 5, 6, and 7 are assumed always to contain relocatable address values.

2.15 INPUT/OUTPUT INSTRUCTIONS

The instructions in this group are:

SFL	=M,,k	Set a function line in bank k
TSL	=M,,k	Test a sense line in bank k
LDCD	M,X,K	Load channel data register, channel K (output)
STCD	M,X,K	Store channel data register, channel K (input)
STCC	M,X,K	Store channel control word, channel K (input)
LDOB	M,X,R	Load output bus R
STIB	M,X,R	Store input bus R

LDCC and STCC instructions are only available with the Automatic Data Channel Processor expansion and once initiated, govern the transfer of data without further intervention of the central processor.

Programmed half-word transfer using LDCD and STCD are available with or without ADCP. In addition, the System Interface Instructions LDOB and STIB provide data transfer between registers of external devices.

A more complete summary of these instructions is given in Chapter 4, Paragraphs 4.3 to 4.5.

2.15.1 SFL Instruction (Set Function Line)

Immediate addressing must always be used with SFL instructions. Within this instruction indexing and indirect addressing may also be used. The effective address, E, is therefore always used as an immediate operand, whose bit pattern determines the channel, device, and function required.

The data channel SFL's fall into two broad categories (1) initialize channel and connect device, or (2) channel clear, disconnect, set/reset ready interrupt, and set/reset signal interrupt. A non-zero 4-bit byte (bits 4-7) denotes the former category, whose bit pattern significance is as follows:

bit ø	always set
bit 1-3	channel number (0-7)
bit 4-7	device designation (non-zero byte, 1-15)
bit 8	transfer Exec bits if set, omit Exec bits if reset
bit 9	*binary* mode if set, *BCD* if reset
bit 10	start transfer with left half-word initially if set, right if reset
bit 11	alternate left to right, right to left, if set; if reset continue transfers from word half specified by bit 10
bit 12	transfer to memory (input) if set; if reset, transfer to device (output)
bit 13-15	code for bits per byte/number of bytes per half-word transferred.

Bits 4-7	Device
01000	Paper Tape Reader
01400	Card Reader
02000	Paper Tape Punch
02400	Card Punch
03000	Typewriter
03400	Line Printer
04000	Magnetic Tape

Bits 13-15	Byte Size/Count
00000	Exec bits only
00001	8/1 (i.e., signifies 8 bytes of 1 bit each)
00002	8/2
00003	16/1
00004	4/4
00005	4/1
00006	4/2
00007	4/3

In the second category of data channel functions denoted by a zero byte (bits 4-7), bits have the following significance:

bit ø	unconditional channel clear
bit 1-3	channel number (0-7)
bit 4-7	(zero byte)
bit 8-10	no significance
bit 11	Reset Channel signal interrupt
bit 12	Set Channel signal interrupt
bit 13	Reset Channel ready interrupt
bit 14	Set Channel ready interrupt
bit 15	Channel disconnect

compatible combinations of channel data functions may be called with one SFL instruction by setting the corresponding bits.

Each device has a set of SFL's for establishing device dependent conditions, e.g., on the typewriter - type red or type black; on the paper tape reader - read forward or read reverse; on the paper tape

punch - turn power on or turn power off; and so on. Available SFL's are specified in device descriptions, Paragraph 4.5 in this manual.

The successful completion of the SFL instruction is indicated by the busy (B) flag in the flag register, which changes to a reset condition after completion.

2.15.2 TSL Instruction (Test Status Line)

Immediate addressing is required with TSL instructions; within this restriction, indexing and indirect addressing may be used. The effective address E, is therefore always used as an immediate operand, whose bit pattern specifies which test is required, on which device, on which channel. As with SFL instructions, bit 1-3 specify the channel, and bits 4-7 the device; with the remaining bits specifying the test.

For the data channel functions the following tests are available.

00001	Test channel signal
-00001	Test channel signal and clear
00002	Test channel parity
-00002	Test channel parity and clear
00004	Test channel ready

With peripherals, device related tests are available. For example: for card reader, test if the reader is ready; test if the reading of the previous card is complete: for magnetic tape, test if tape movement has ceased; and so on. The available tests are specified in device descriptions in Paragraph 4.6 of this manual.

The Z flag is set if the result of a TSL is true, reset is false.

2.15.3 LDCD, STCD Instructions

Having connected a device for data transfer, the actual transfer of data is effected by the LDCD or STCD instruction.

There must be one of these instructions per half-word transferred. If bit 11 of data channel SFL calling for left and right alternate transfer is set, two instructions per memory location are required. Since initial and subsequent left/right half positioning is implicit in the SFL call, address specification in this case is identical for both instructions and refers apparently only to the left half.

The instruction modifier X,*, may be used with both LDCD and with STCD. The / modifier is irrelevant and should not be used with either.

The accumulator cannot be accessed directly by an STCD or LDCD instruction. (E = 0 or 1 refer to core location 0 or 1 with LDCD and STCD.)

Example:

The following example reads paper tape, punched in autoload format, for N words, starting the loading at memory location MEMORY. The device SFL connects the paper tape reader (PTR) to transfer Exec bits, binary mode, starting with left half memory word, alternating left/right/left, etc., reading from device to memory, half-words comprising four 4-bit bytes.

```
SFL     =1,,1       conditional disconnect
                    channel 0, bank 1

SFL     ='-1374,,1  connect PTR

JB      *-1         wait until device
                    connected

XCS     -N,X        count of N in index
                    register X
```

```
STCD    MEMORY +    transfer first half-
        N,X         word to memory

STCD    MEMORY +    transfer second
        N,X         half-word to memory

XJT     *-2,X,1     loop until N words
                    transferred

SFL     =1,,1       disconnect.
```

2.15.4 LDCC, STCC Instructions

Available only with ADCP, these instruction govern transfer of data according to a data control word of the form

```
ARG(*)      ADDRSS,X,COUNT
```

where ADDRSS is the starting address of a block size given by COUNT, in a manner specified by the * and X bits (16-19).

The options implied by bits 16-19 are:

bit 16 - Transfer data if set, skip if reset (T or S)

bit 17 - Disconnect at end if set, do not disconnect if not set (D or I)

bit 18 - Skip or transfer until signal (S)

bit 19 - Skip or transfer until count complete (C)

and are summarized mnemonically by one of the following OP symbols: -

TCD, SCD, TCI, SCI, TSD, SSD, TSI, SSI, TED, SED, TEI, SEI in the formal context:

```
OP      ADDRSS,,COUNT
```

Example

```
SCI     Skip data until Count is zero then
        Interrupt (but do not disconnect)

TED     Transfer data until Either count is zero
        or signal is received, then Disconnect
        (and interrupt)
```

These ADC OP symbols could be used as shown below to cause the skipping of the first 500 words of a data record, then the transfer into memory of either the remaining portion of the record if there were fewer than 1500 words, or of the next 1000 words.

```
        SFL                 connect device
                            for transfer to
                            memory

        JB      *-1         wait until
                            connected

        LDCC    OMIT        initiate skip
                            action

        --

        --

OMIT    SCI     ,,500       ADC OP symbol

READ IN TED     START,,1000 definitions
```

with an interrupt routine containing

```
        LDCC    READIN      initiate actual
                            data transfer
```

If a signal, due to the advent of a terminating character or situation (stop code on paper tape, end-of-

record gap on magnetic tape, end-of-card on card reader) precedes the completion of the specified count, the number of words transmitted can be ascertained by performing STCC after interrupt, and examining the count field.

Transfer of data, one initiated, proceeds automatically under control of the ADCP without the need of central processor intervention.

An interrupt is always generated on the completion of these ADCP functions.

2.15.5 LDOB, STIB Instructions

These system interface instructions effect the transfer of 17 bit data (16 bits plus 1 exec bit) from E via the output buss to the specified external register (LDOB), or from the specified external register via the input buss to the effective address E (STIB).

The X,*,=and / options are available with LDOB, and X,* and / with STIB

2.16 SHIFT, ROTATE AND NORMALIZE INSTRUCTIONS

These instructions are available in single or extended precision form. If extended precision, the A-AE registers are involved; if single precision, the A register only.

Each instruction in this group may use the *Save* option.

2.16.1 Arithmetic Shift

Although = is not written in the first character position of the address field of shifts or rotates, the *immediate mode* is *assumed* by the hardware. Thus, if N=3, and (X)=1, so that E=4

```
ASH         N,X
```

is a request for an arithmetic shift of the A register contents 4 places to the right (equivalent to a division by 2^4). The same affect is achieved by

```
ASH         4
```

or by:

```
ASH*        SAM
```

where the left half contents of SAM is 4. These are examples of positive shifts.

With arithmetic shifts, the sign bit $\emptyset$ is propagated for right shifts, so that a negative number remains negative, a positive one positive. Similarly for left shifts (negative shifts) the sign bit does not change, and bit 1 is lost for each left shift.

For extended shifts, the sign bit of the AE register does not change. Therefore, only bits 1-15, 17-31 are involved. For example,

```
EASH        -15
```

shifts the AE register, considered as an arithmetic quantity, to the A register.

The Z, G, L and V flags reflect the resultant state of the A, or A-AE registers.

Bits in positions 15-E to 15 for ASH right shift or 31-E to 31 for EASH right shifts, where E is the number of shifts, *are* permanently lost. For left shifts, the same applies to bits in positions 1 to E for both ASH and EASH. If a 1 bit is lost in the latter cases for initially positive value, or a $\emptyset$ bit for an initially negative value, the overflow flag is set.

The nominal effective address is truncated by hardware to provide an effective immediate address E such that $-64 \leq E \leq 63$.

The *Save* option prefix may be used to save (A) in ($A) prior to shifting. If indirect addressing is used, the immediate operator is still implied and the value of E is taken as the shift count. Actually E is truncated at 7 bits, and the effective count is:

$$-64 \leq (E_{0,\ 10:15}) \leq 63.$$

2.16.1.1 *right arithmetic shift*

		$(E_{0,\ 10:15})$ positive
		where e = $(E_{0,\ 10:15})$ modulo 15
ASH	m,X	shift $(A_{1:15})$ right by e bits
		$(A_0) \longrightarrow (A_{1:e})$
		right-most e bits are lost
		where e = $(E_{0,\ 10:15})$ modulo 31
EASH	m,X	shift $(AAE_{1:15,\ 17:31})$ right by e bits
		$(A_0) \longrightarrow$ left most e bits (excluding AAE_{16})
		right-most e bits are lost
		(AAE_{16}) unchanged

2.16.1.2 *left arithmetic shift*

		$(E_{0,\ 10:15})$ negative
		where e = $(E_{0,\ 10:15})$ modulo 15
ASH	m,X	shift $(A_{1:15})$ left by e bits
		$\emptyset \longrightarrow$ right most e bits
		$(A_{1:e})$ lost, V flag set if any lost bits $\neq (A_0)$
		where e = $(E_{0,\ 10:15})$ modulo 31
EASH	m,X	shift $(AAE_{1:15,\ 17:31})$ left by e bits
		$\emptyset \longrightarrow$ right most e bits (excluding AAE_{16})
		left most e bits lost (excluding AAE_{16})
		V flag set if any lost bits $\neq (A_0)$

2.16.2 Rotates

The contents of the A register for single precision or of the A-AE register for extended precision, are regarded as a bit pattern having no arithmetic significance, so that bit $\emptyset$ and bit 16 for extended precision are treated as any other bit.

In contrast with arithmetic shifts, no bits are lost: for single precision, right-rotation (positive E) the bit in position 15 is transferred to position $\emptyset$; the reverse applies to left (negative) rotation. For extended precision rotation, the bit in position 31 is transferred to bit position $\emptyset$; the reverse applies to left (negative) rotation.

The *save* prefix operator may be used, and the same immediate addressing rules apply as above for shifts.

The flags are unaffected by rotates. For example,

EROT 16

exchanges the contents of the A and AE registers.

The rotate instructions are formally summarized as follows:

rotate, right or left		$(E_{0,\ 10:15})$ positive or negative for e = $(E_{0,\ 10:15})$
ROT	m,X	$(A_{0:15}) \longrightarrow (A_{e:(15+e)_{\text{modulo 16}}})$
EROT	m,X	$(AAE_{0:31}) \longrightarrow (AAE_{e:(1+e)_{\text{modulo 32}}})$

Normalize

The specified index register receives a count of shifts required to normalize the contents of the A register if single precision is required, or A-AE

registers if extended precision desired. The contents of the accumulator are considered arithmetic, and the shifts follow the rules of ASH or EASH operations.

The count appearing in the specified index register is a positive quantity, so that the two instructions

```
NRM     ø, 2
ASH     ø, 2
```

would leave the A register unchanged, but with the number of shifts to achieve normalization in index register 2.

NOTE

The criterion for normalization is that bits ø and 1 of the A register must be different.

The *normalize instructions* are formally summarized as:

Single Precision

NRM	ø,X	for (A)≠ø, Arith. Single Prec. left shift until $[A_0] \neq [A_1]$, positive count of shifts c → (X). for (A)=0, 0 → (X).

Extended Precision

ENRM	ø,X	for (A)≠ø, Arith. Extend. Prec. left shift until $[A_0] \neq [A_1]$, positive count of shifts c → (X). for (AAE)=0, 0 → (X)

CHAPTER 3

PRIORITY INTERRUPT SYSTEM

3.1 INTRODUCTION

The 8400 Interrupt System provides a means of interrupting a program sequence at the occurrence of some event, and executing a routine that corresponds to the event. Conditions both internal and external to the machine can cause interrupts. Hardware is provided to maintain an assigned priority among the interrupt conditions, and to maintain the priority while an interrupt routine is in progress. When the machine is interrupted, the continuity of the instruction sequence is preserved, and interrupt routines as small as one instruction can be used.

3.2 BASIC OPERATION

The elements of the interrupt system are as follows:

1. A 16-bit Internal Interrupt Register with associated decoding logic and scan circuits.
2. Up to sixteen 16-bit External Interrupt Registers with associated decoding logic and scan circuits.
3. The Enable Flag in the Flag Register denoted by (E).
4. A 16-bit Internal Mask Register denoted by (IMR).
5. A 16-bit External Mask Register denoted by (EMR).
6. An additional external mask bit denoted by (EMB).

When an interrupt condition occurs, a bit corresponding to that condition is set in one of the interrupt registers. If the enable flag E and the Mask bit corresponding to that position in the interrupt register are set, a signal is gated to the scan circuits. At various points in the instruction cycle, the scanner searches for any interrupt signals. The order of scanning determines the priority of one interrupt condition relative to another. If an interrupt condition is detected by the scanner, and no higher priority signals were detected, the interrupt logic generates an unique address for that interrupt. At certain points in the instruction cycle, the machine acknowledges the detected interrupt by breaking the program sequence, and executing the instruction at the location specified by the interrupt logic.

The Enable Flag, E, can either enable or disable all signals from the interrupt registers. When the E flag is *reset* (set to zero, disabled, turned off), no interrupt condition can be acknowledged, except upon power failure, and the program sequence cannot be interrupted. Instructions pertaining to the flags are discussed in Chapter 2.

The interrupt registers themselves cannot be disabled. An interrupt condition will always set the corresponding bit in the interrupt register independent of the E flag and the mask bits. The interrupt registers provide a buffer to remember the occurrence of interrupt conditions until the interrupt logic can acknowledge the signal. Depending on the type of interrupt routine used, a bit in the interrupt register is reset either automatically when the interrupt condition is acknowledged or is reset by program. Figure 3-1 shows the interrupt register mask configuration.

3.3 PRIORITY

There are two aspects in determining the interrupt priority of specific interrupt requests:

1. When multiple interrupt conditions occur simultaneously, which condition will be first acknowledged by the system?

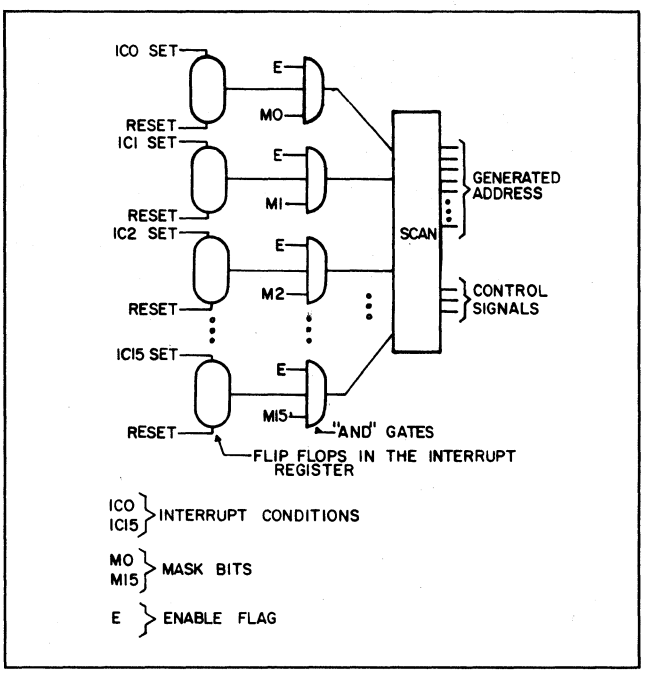


Figure 3-1. Interrupt Register Mask Enable Configuration

2. Once an interrupt has been acknowledged, an interrupt routine is in progress, what conditions will cause that routine to be interrupted?

Regarding the first, the scan action of the interrupt detection logic determines which interrupts take precedence over other interrupts. That is, if conditions occur simultaneously, the scan determines which will be acknowledged first. This feature of the interrupt system is referred to as *hardware priority*. The memory locations associated with the interrupts reflect the hardware priority. The highest priority interrupt has the lowest memory location as shown in Table 3-1.

Table 3-1. Memory Locations

Octal	Decimal	Interrupt Name
40	32	Power Failure/Memory Parity Error
41	33	Data Exec
42	34	Illegal Instruction
43	35	Instruction Exec
44	36	Exponent Fault

Table 3-1. Memory Locations (Cont)

Octal	Decimal	Interrupt Name
45	37	Memory Protect
46	38	Timer
47	39	Console
50-57	40-47	Data Channels 0-7
60-77	48-63	External Group 1
100-117	64-79	External Group 2
120-137	80-95	External Group 3
140-157	96-111	External Group 4
160-177	112-127	External Group 5
200-217	128-143	External Group 6
220-237	144-159	External Group 7
240-257	160-175	External Group 8
260-277	176-191	External Group 9
300-317	192-207	External Group 10
320-337	208-223	External Group 11
340-357	224-239	External Group 12
360-377	240-255	External Group 13
400-417	256-271	External Group 14
420-437	272-287	External Group 15
440-457	288-303	External Group 16

The first 16 interrupts (highest priority) are internal. All remaining interrupts are called interrupts. The first 6 interrupts refer in some way to the instruction sequence. The meaning of all internal interrupts is explained in Paragraphs 3.7.

Once an interrupt has been acknowledged, there are two methods by which a given priority can be maintained for the duration of the interrupt routine. One method is through the use of the mask registers. By manipulation of the mask registers, any priority sequence - not necessarily the same as hardware priority - can be achieved and maintained. Programmed priority allocation is discussed in Paragraph 3.5. If the hardware priority sequence is acceptable to the user, this priority will be maintained automatically for the duration of an interrupt routine by the hardward. If certain programming rules are observed. The automatic maintenance of hardward priority is discussed in Chapter 4.

3.4 INTERRUPT CONTROL

Interrupts can be acknowledged only at certain times during the instruction sequence:

1. After gathering of instruction,
2. After each level of address modification, and
3. After the execution of an instruction.

The actual execution of an instruction can never be interrupted. Also, interrupts can never occur at the following times:

1. When executing the first instruction of an interrupt routine,
2. After the machine has been halted from console, and
3. When the E flag is reset.

An exception to the above is the power failure interrupt which has the highest hardware priority. This interrupt can occur, independent of its mask bit and the E flag, during auto load or auto dump and after the machine has been halted manually.

When an interrupt is acknowledged, the machine is forced to execute the instruction at the interrupt location. The interrupt location address is not placed in the location counter. The resulting action depends on the type of instruction at the interrupt location. The four possible cases are as follows:

1. Link instructions (L or LR).
2. Jump instructions (J, JS, JR, JT, HJ, XJ, XJT, LDL).
3. Execute instruction (EX).
4. Other instructions.

A link instruction serves to inform the interrupt system that an interrupt route is being initiated. When the link at the interrupt location is executed, the system enters a scan-limiting state, and the interrupt condition is not reset. Once in the *acknowledged, but not reset* state, the scan is limited so that only interrupts of higher priority can be acknowledged. The use of the link instructions, therefore, permit the hardware priority to be maintained for the duration of an interrupt routine.

The Jump Trigger (JT) instruction is the means by which a program informs the interrupt system that an interrupt routine has been completed. If the system is in a scan-limiting state, and a JT instruction is executed, the system then resets the highest priority interrupt condition that has been acknowledged but not reset. There are three states for each type of interrupt.

1. Idle; no interrupt condition present.
2. Set; an interrupt condition has set the bit in the interrupt register, but the condition has not yet been acknowledged.
3. Acknowledged but not reset; a link instruction at the interrupt location was executed, and no JT instruction has yet been executed.

The possible states of the interrupt system are shown in Figure 3-2. While an interrupt is in the acknowledged state, no more interrupt of that type can be received until the JT instruction is executed. The bit in the interrupt register remains set until the JT is executed. If no interrupt is in the acknowledged state, then JT instructions have no effect on the interrupt system.

The link JT technique for interrupt routines provides multi-level interrupt capability with the proper priority without resorting the mask manipulation. An example of multi-level interrupts is shown in Figure 3-3. For each interrupt routine, the use of the lin

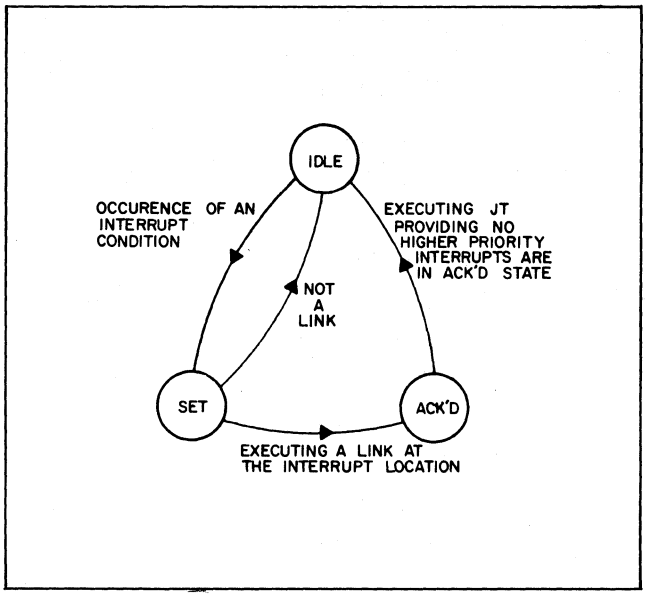


Figure 3-2. Interrupt States

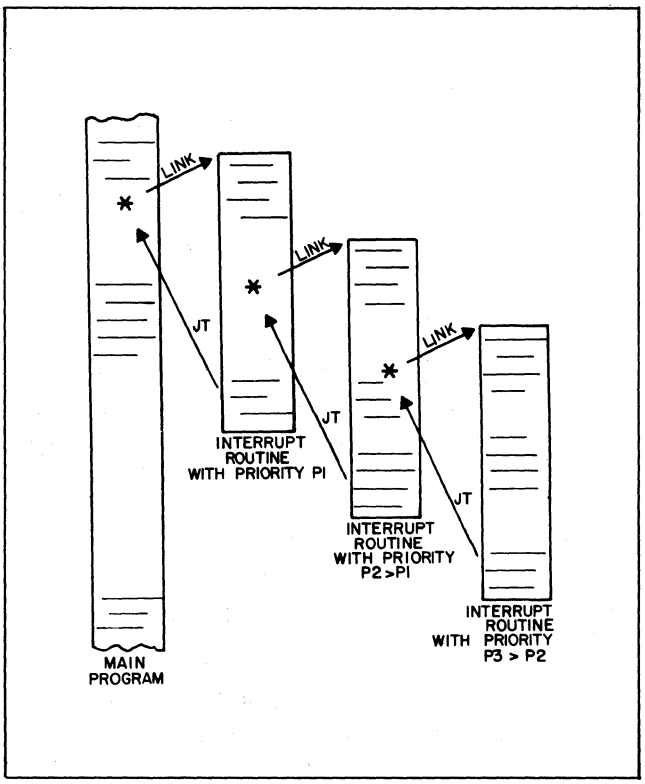


Figure 3-3. Multi-Level Interrupts

instruction guarantees that only higher priority interrupts can occur while the routine is in progress. When the routine is complete, the use of the JT guarantees that the status of the interrupt system before the interrupts will be re-established.

Note that the resetting of the interrupt is independent of any flags associated with the JT instruction. For the instruction JTf, where f is some flag, the following occurs:

1. The jump is performed if flag f is true.
2. Flag f is triggered unconditionally.
3. The highest priority acknowledged interrupt is reset unconditionally.

When the instruction executed at the interrupt location is not a link, the interrupt conditions is automatically reset. If the instruction is not a jump type instruction that alters the location counter, the interrupt instruction is then executed as a one-instruction interrupt routine. The return is made to the interrupted program to insure that no instructions will have been missed.

If the instruction at the interrupt location is a jump instruction, the interrupt is automatically reset, and the jump is executed. Since the location counter is loaded with a new address, the location counter value at the time of the interrupt is lost. In this case, it is impossible to determine by program at which point the program was interrupted.

Should the instruction at the interrupt location be an Execute (EX), the resulting action depends on the object of the Execute instruction. The interrupt condition is reset automatically, independent of the object instruction unless it is a Link. If multiple Execute instructions are chained together, the entire chain, including the object instruction, is non-interruptable.

Interrupts that refer to machine instructions (interrupts 0-5) are handled in a special way. If one of these interrupts is acknowledged either after gathering of instruction or after address modification, the location counter is incremented by one before the instruction at the interrupt location is executed.

For example, if an illegal instruction at location L causes an interrupt after gathering instruction the following occurs:

1. With a link instruction at the interrupt location, the address L + 1 is stored at the effective address of the instruction.

2. With a non-link, non-jump instruction, return would be made to L + 1, after execution of the one instruction routine.

A personalized interrupt is one in which a transfer of control takes place *immediately*, should the interrupt condition arise, where immediately means following the execution of the instruction currently being obeyed.

This is critical for such interrupts as "unvalid instruction". On the other hand, other interrupts cause the transfer of control to take place one instruction later than this and this is satisfactory for interrupts generated externally.

However, the 8800 interfacy busy interrupt is not personalized, and so the occurrence of an 8800 instruction which causes an 8800 busy interrupt to occur, causes the interrupt routine to save the address of the instruction *after* the one that caused the interrupt to occur. It is wise therefore for the programmer to follow every 8800 interface instruction that can result in an '8800 busy' interrupt, by a NOP. If the interrupt occurs the address of the NOP will be saved, and the interrupt routine can occur properly.

3.5 MASKING

If some priority sequence other than the hardware priority is needed, masking must be used. For those interrupts with individual mask bits, any sequence of priority can be achieved. The instructions pertaining to the mask bits are the following:

LDM	m	Load the *Internal Mask* Register with a half-word from memory.
LDE	m	Load the *External Mask* Register a half-word from memory.
Options:		*, X, /, =
Flags:		none
STM	m	Store the contents of the *Internal Mask* Register into a half-word in memory.
STE	m	Store the contents of the *External Mask* Register into a half-word in memory.
Options:		*, X, /
Flags:		none
SFL	='60,,0	Set the external mask bit (EMB)
SFL	='61,,0	Reset the external mask bit (EMB)

The B flag is set following either SFL instruction if the EMB was set prior to the instruction.

TSL	='60,00	Test the external mask bit (EMB)

The Z flag is set following this instruction if the EMB was set, and the flag is reset if the EMB was not set.

Dynamic priority allocation can be achieved by using the masks in the following way:

1. Determine for each interrupt condition which of the others are to have higher or lower priority.

2. During the main program, keep all mask bits set and all interrupts enabled, and perform a JT instruction to release the machine from its limited-scan control state.

3. Store the existing mask registers; then change the masks so that only bits for interrupts of priority greater than the given interrupt (in the new sequence) are set.

4. Set the E flag to re-enable the interrupt and execute the interrupt routine.

5. When the interrupt routine is completed, the interrupts should be disabled, the original masks restored, and the interrupts again enabled before returning to the point that was originally interrupted.

3.6 *USER/MONITOR* MODE AND THE INTERNAL INTERRUPTS

3.6.1 User/Monitor Modes

Every 8400 is equipped with a feature to facilitate multi-programming and time sharing. The *User/Monitor* mode feature prevents a user program from interfering with the continuous operation of the computer. In *user* mode, any instruction that initiates input/output operations or modifies the state of certain control register is not executed. In *monitor* mode all instructions are permitted.

The *monitor* mode flip-flop controls the mode of the computer. The machine is placed in *monitor* mode when any interrupt occurs, or by the INITIALIZE button from the console. The flip-flop can be reset and tested by the following instruction:

```
SFL   ='65,,0   Reset Monitor Mode

TSL   ='65,,0   Test Monitor Mode
```

The *Reset Monitor* mode instruction, in addition to placing the machine in *user* mode, also enables the interrupt system. Once in *user* mode, the interrupts cannot be disabled. The interrupts can then be disabled only after an interrupt has occurred, and the machine has been returned to *monitor* mode.

Instructions that cannot be executed in *user* mode are called *privileged* instructions. They are:

1. LDT, LDM, LDE, LDC

2. SFL, TSL

3. LDOB, STIB

4. LDCD, STCD, LDCC, STCC

5. HJ

When one of these instruction is attempted in *user* mode, the instruction is not executed, a privileged instruction flip-flop is set, and an interrupt is generated. The privileged instruction flip-flop can be tested and reset with the following instructions:

```
SFL   ='21,,0   Reset Privileged Instructions

TSL   ='21,,0   Test Privileged Instructions
```

Other illegal instructions generate this interrupt as usual, but do not set the Privileged Instruction flip-flop.

Since the interrupt system cannot be disabled in *user* mode, instructions that refer to the E flag in the flag register are not allowed to change its state. Instructions of this type are executed, but do not affect the E flag, as shown by the following:

Instruction	*Acts Like*
JSE	JE
JSNE	JNE
JRE	JE

Instruction	*Acts Like*
JRNE	JNE
LRE	LE
LRNE	LNE
JTE	JE
JTNE	JNE

In *user* mode, the LDF instruction does not change the E bit, and the JT instruction has no affect on the interrupt system.

3.6.2 Internal Interrupts

The internal interrupts are summarized in Figure 3-4. The interrupt number represents the bit number for the corresponding bit in the Mask Register. The lowest number has the highest priority. Each interrupt is discussed in detail below.

(0) Interrupt Name: *Power failure and memory parity error interrupt.*

Interrupt Location: '40

Interrupt Mask: Does not affect this interrupt.

If the voltage level varies beyond a safe limit, a power failure interrupt is generated. Memory parity is checked whenever the memory is accessed - either for normal program execution or for the automatic data channel operation. Both power failure and memory parity error are considered catastrophic. Restart may be from the beginning of the present job or from the last SAVE point in the job. The following instructions are used to determine cause: TSL = '23,,0 for Test Memory Parity Failure, and SFL = '23,,0 for Reset Memory Parity Failure.

(1) Interrupt Name: *Data Exec*

Interrupt Location: '41

Internal Mask: '40000

The Data Exec interrupt is set for any gathering of data in the *monitor* mode of a word with the left Exec

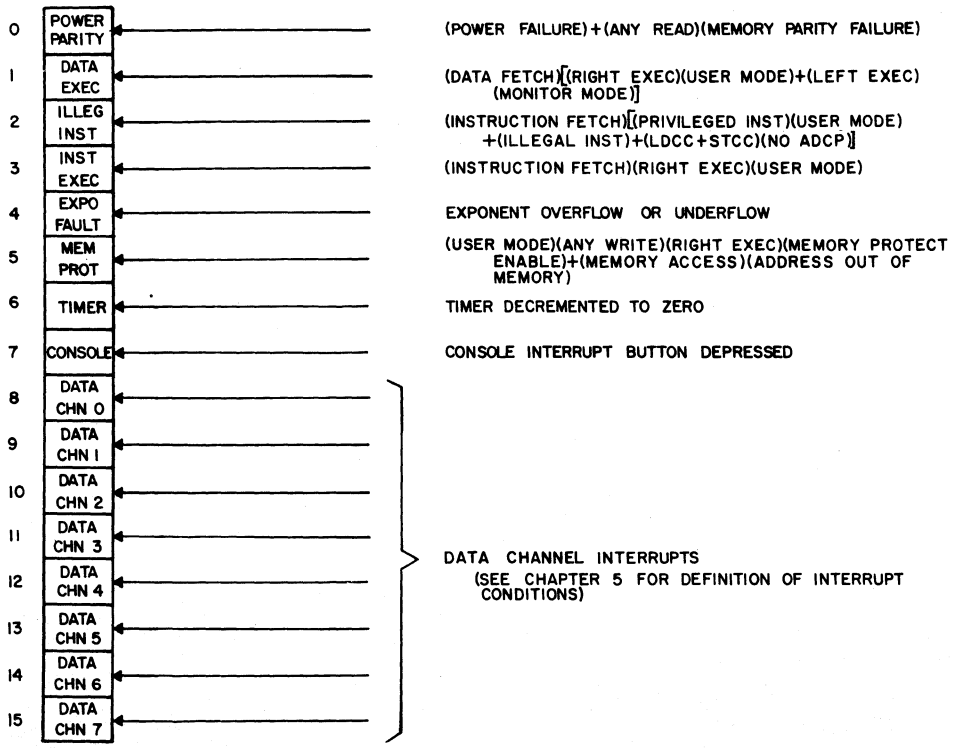


Figure 3-4. Internal Interrupt Conditions

bit set, or in *user* mode of a word with the right Exec bit set.

Exceptions: TEX, LDCC, LDCD, LDOB.

The cycle of instruction gathering lasts until the effective address has been fully calculated, including indirect addressing and indexing. Instructions that have operands in a memory location given by the effective address, have a data gathering cycle that follows the instruction gathering cycle. Immediate instructions do not have a data gathering cycle.

(2) Interrupt Name: *Illegal Instruction*

Interrupt Location: '42

Internal Mask: '20000

In general, any instruction that is undefined or would result in stopping machine operation will cause this interrupt. Immediately after gathering instruction the interrupt is generated and the instruction is not executed. The following specific instructions cause this interrupt:

1. Undefined OP codes
2. STCC or LDCC when no Automatic Data Channel Processor is present
3. Privileged instructions in *user* mode

(3) Interrupt Name: *Instruction Exec*

Interrupt Location: '43

Internal Mask: '10000

This interrupt is generated following instruction gathering in *user* mode if the right Exec bit at the location of the obtained instruction is high. The instruction is not executed.

(4) Interrupt Name: *Exponent Fault*

Interrupt Location: '44

Internal Mask: '04000

This interrupt is generated by a floating-point exponent exceeding the proper range as the result of some floating-point operation. The proper range for exponents is $-128 \leq Exp \leq 127$. Exponent overflow does not inhibit a floating-point operation.

(5) Interrupt Name: *Memory Protect*

Interrupt Location: '45

Internal Mask: '02000

This interrupt results when an instruction tries to modify any half-word, full-word, or Exec bit in a protected area of memory. Specifically, this interrupt occurs on a store instruction, in *user* mode, if and only if the right Exec bit is set at the location and the *memory protect* mode has been enabled for that memory bank. The *memory protect* mode can be enabled and disabled individually for up to four memory banks. The instructions pertaining to the *memory protect* mode are the following:

	Test	Set	Reset
Bank 1	TSL = '40,,0	SFL = '40,,0	SFL = '41,,0
Bank 2	TSL = '42,,0	SFL = '42,,0	SFL = '43,,0
Bank 3	TSL = '44,,0	SFL = '44,,0	SFL = '45,,0
Bank 4	TSL = '46,,0	SFL = '46,,0	SFL = '47,,0

The SFL instructions set the B flag if the specified mode control was set prior to the SFL action. The TSL instructions set the Z flag if the specified mode control was set, and reset the Z flag if the control was reset.

This interrupt will also occur if memory is referenced by an illegal address (for example, an address out of memory). If an instruction attempts to access

memory with an illegal address, the memory access is bypassed, the instruction cycle is completed, and the interrupt is generated.

(6) Interrupt Name: *Timer*

Interrupt Location: '46

Internal Mask: '01000

The interval timer is an optional feature on the 8400, and this interrupt cannot occur on those machines without a timer. This interrupt occurs when the timer is decremented to zero.

The basic element of the timer is the 16-bit Timer Register. The following instructions pertain to the timer:

LDT	m	Load the Timer Register with a half-word from memory
STT	m	Store the Timer Register into a half-word in memory.
Options		*, X, /, =
Flags:		none
SFL	='62,,0	Start the timer
TSL	='62,,0	Test the timer
SFL	='63,,0	Stop the timer

The B flag is set following the SFL instructions if the timer was operating prior to the SFL; the Z flag is set following the TSL if the timer was operating. Once the timer is operating, the Timer Register is decremented every millisecond. Decrementing continues until the timer is stopped with the appropriate SFL instruction. Note that the timer runs while the machine is in a manual halt condition. After zero is reached, the next decrement produces the maximum value in the Timer Register, and decrementing proceeds from there.

(7) Interrupt Name: *Console*

Interrupt Location: '47

Internal Mask: '00400

The console interrupt is generated when any one of the four console interrupt buttons (CI1-CI4) is depressed. Each button is buffered with a flip-flop as shown in Figure 3-5. Pushing the button sets a corresponding flip-flop, and generates the interrupt. An indicator in the button lights when the flip-flop is set. The flip-flops can be tested and reset by program to determine which of the buttons caused the interrupt. When the flip-flop is reset, the indicator light goes out. The instructions related to the console interrupt are listed below:

TSL	='25,,0	Test Console - Interrupt 1
SFL	='25,,0	Reset Console - Interrupt 1
TSL	='27,,0	Test Console - Interrupt 2
SFL	='27,,0	Reset Console - Interrupt 2
TSL	='31,,0	Test Console - Interrupt 3

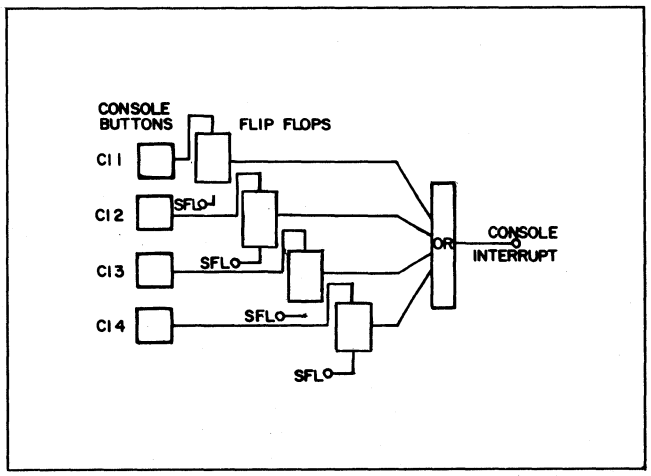


Figure 3-5. Console Interrupt Buttons

```
SFL   ='31,,0   Reset Console - Interrupt 3

TSL   ='33,,0   Test Console - Interrupt 4

SFL   ='33,,0   Reset Console - Interrupt 4
```

The TSL instructions set the Z flag if the specified flip-flop was set. The SFL instructions set the B flag if the specified flip-flop was set.

The JT instruction that resets the bit in the interrupt register has no affect on the flur flip-flops associated with the console interrupts. Following a console interrupt, both the interrupt bit and the console flip-flop must be reset before another console interrupt can be generated.

(8) - (15) Interrupt Name: *Channel Interrupt*

Interrupt Location: '50 - '57

Internal Mask: '00377

Each data channel has one interrupt. Channel 0 corresponds to location '50, channel 1 to '51, and so forth. When a device is connected to a data channel, interrupts can result from the channel itself, or from the connected device. When no device is connected to a channel, interrupts can result from those devices on the channel which have been properly enabled.

3.7 EXTERNAL INTERRUPTS

The conditions that cause external interrupts are a function of the external equipment tied to the interrupt lines. There are up to 16 groups of 16 interrupts. The external mask register pertains only to the first external group which is shown below:

Interrupt	*Location*	*External Mask*
(1, 0)	'60	'-00000
(1, 1)	'61	'40000
(1, 2)	'62	'20000
(1, 3)	'63	'10000
(1, 4)	'64	'04000
(1, 5)	'65	'02000
(1, 6)	'66	'01000
(1, 7)	'67	'00400
(1, 8)	'70	'00200
(1, 9)	'71	'00100
(1, 10)	'72	'00040
(1, 11)	'73	'00020
(1, 12)	'74	'00010
(1, 13)	'75	'00004
(1, 14)	'76	'00002
(1, 15)	'77	'00001

For those machines with no more than four data channels, the interrupts corresponding to channels 4-7 are available as external interrupts. In this case, lines for these interrupts are available through the system interface. A response line is available through the system interface that indicates when the interrupt has been serviced.

The interrupts are set by a positive transistion in a signal. Therefore, either pulse or level signals can be used to generate external interrupts.

When a signal on an interrupt line is set, an interrupt will occur. If the external signal maintains its position level, no more interrupts will result from that signal until it falls and is set again. A positive transition is required for both internal and external interrupts.

3.8 CONSOLE INDICATORS

The following items pertain to the interrupt system:

1. The INITIALIZE button on the console sets all interrupts in the interrupt registers, returns all interrupts to the idle state, and resets the console interrupt flip-flops.

2. The INTERNAL INTERRUPT indicator on the console is lit when any internal interrupt is *not* in the idle state.

3. The CHANNEL INTERRUPT indicator on the console is lit when any channel interrupt is *not* in the idle state.

CHAPTER 4

INPUT/OUTPUT SYSTEM

4.1 INTRODUCTION

The input/output of the 8400 resides in the Exchange Module which contains the following functional units:

1. An Exchange Module Central Controller (EMCC) with up to 8 bi-directional data channels for buffered data transfer to a number of devices.

2. An Automatic Data Channel Processor (ADCP) which automates the data channel operation, provides direct memory access, and permits simultaneous input/output/ compute operations.

3. A System Interface which permits direct data transfer with external data handling systems.

Each peripheral device is provided with a device controller which enables all devices to use the generalized data and control interface of the exchange module.

The operators desk, which uses the features of the Exchange Module, is discussed in detail in Chapter 5.

A functional representation of the Exchange Module is shown in Figure 4-1.

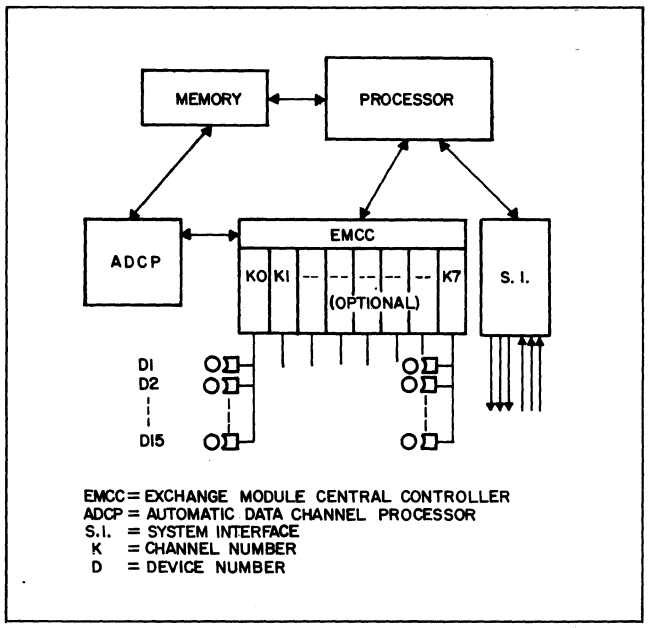


Figure 4-1. Exchange Module

4.2 DATA CHANNELS

Every 8400 is equipped with at least one data channel and may be expanded to 8. Up to 15 devices can be connected to each channel, although only one device can be selected for data transfer at a given time.

4.2.1 Function

The purpose of an 8400 data channel is to facilitate data transfers to or from peripheral devices with a minimum of programming effort. Byte assembly and disassembly is provided for handling 4 and 8-bit bytes. All transfers between memory and a channel are made as 17-bit halfwords (16 data bits and Exec bit), while transfers between the channel and a device can be in terms of 4, 8 or 16-bit bytes and an Exec bit.

Character buffering is provided so that character devices need not have their own external buffer register. Buffering is also provided for control signals and error indicators so that all external devices can be programmed and operated in a generalized and consistent manner.

Code conversion circuitry is provided so that external devices which require a binary coded decimal

(BCD) code can be handled as well as those that generate or accept the 8400 internal binary code. On input, the BCD code from a device is converted to binary code, and on output, the internal binary code is converted to the device oriented BCD code.

Parity generation and checking logic is provided for 4 and 8-bit byte transfers. On output, the parity bit is generated and sent with the data; on input, a parity bit is generated and checked against the parity bit received with the data. With binary data transfers, odd parity is used, and with BCD transfer, even parity is used.

4.2.2 Structure

The data channel complex includes the Exchange Module Central Controller (EMCC) plus up to 8 individual channels. Associated with the individual data channels are the following elements:

1. Channel Function Register (CFR)

   This 8-bit register holds a code word which specifies the type of operation to be performed on the channel--input or output, binary or BCD, 4, 8, or 16-bit bytes, etc. Details of the Channel Function Register format are discussed with the SFL instructions. Associated with the Channel Function Register is the channel control and device selection logic which actively connects one device to the channel. It also prevents the Channel Function Register from being changed while a channel operation is in progress.

2. Channel Data Register (CDR)

   This 17-bit (16 data bits plus one exec bit) register represents the interface between the data channel and memory. For output, the Channel Data Register can be loaded with a half-word from memory to be transferred to a device. During input, this register holds an assembled half-word which is to be stored in memory.

3. Channel Buffer Register (CBR)

   This 8-bit register is the character buffer register to which the selected device is connected. With either 4-bit or 8-bit data transfers, 8-bit bytes are transferred between the Channel Data Register and the Channel Buffer Register. Assembly of 4-bit bytes into 8-bit bytes, or disassembly of 8-bit bytes into 4-bit bytes is performed in the Channel Buffer Register. Parity checking and code conversion is also done in conjunction with the Channel Buffer Register. With 16-bit data transfers, data is transferred directly between the Channel Data Register and the selected device, bypassing the Channel Buffer Register. No parity checking or code conversion is provided for 16-bit transfers.

4. Control indicators as follows:

   Channel Ready (CDRY) indicator is true whenever the Channel Data Register is ready to transfer a half-word to memory on input, or accept a half-word from memory on output.

   Channel Automatic (CHA) indicator is true whenever the channel is under control of the Automatic Data Channel Processor. Details of this indicator are discussed with the ADCP.

   Channel Parity (CHP) indicator is set when either the channel or a selected device detects a parity failure during data transfer; the indicator is reset by a TSL instruction, or when a new channel operation is initiated.

Channel Signal (CHS) indicator can be set by the selected device on the channel when certain conditions on the device exist. The specific conditions that set CHS are different for each device, and are described in the section pertaining to devices.

When the *Channel Ready Interrupt* (CHRI) indicator is true, a channel interrupt will be generated whenever the CHRY indicator becomes true. The CHRI indicator, which enables the interrupt, can be set by an SFL instruction, and is reset by an SFL instruction.

When the *Channel Signal Interrupt* (CHSI) indicator is true, a channel interrupt will be generated whenever the CHS indicator becomes true. The CHSI indicator, which enables the interrupt, can be set by an SFL instruction, and is reset by an SFL instruction.

Channel Disconnect (CHD) control is used to implement the conditional disconnect action. The CHD control indicator is set by an SFL instruction and reset when a new channel operation is initiated. Disconnect procedures are discussed with the SFL instructions.

A block diagram of a data channel is shown in Figure 4-2.

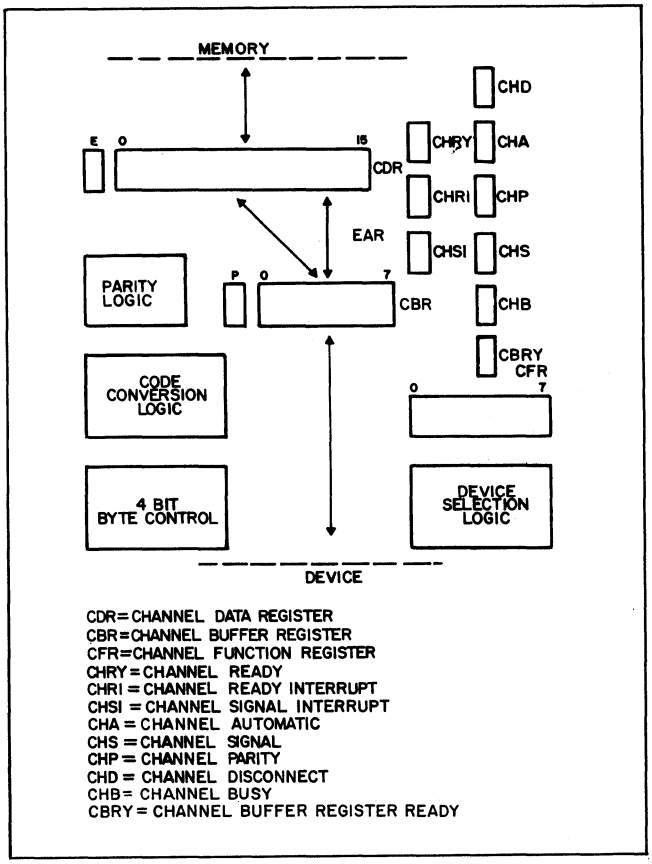


Figure 4-2. The Elements of a Data Channel

The Exchange Module Central Controller (EMCC) is shared by all the data channels in the Exchange Module. The EMCC is available to only one channel at a time. A scan mechanism searches for activity on the channels. When a request for EMCC action is detected, the scan locks on the particular channel. When the required transfers are complete, the EMCC is released and the scan continues. Included in the EMCC is an Exchange Assembly Register (EAR) which serves the following purposes:

1. All accesses to the Channel Data Registers are made through the Exchange Assembly Register. The Exchange Assembly Register acts as an intermediate buffer on all transfers between memory and the Channel Data Register.

2. With 4 or 8-bit byte operations, the assembly or disassembly of 8-bit bytes is performed in the Exchange Assembly Register. The Exchange Assembly Register, therefore, also acts like an intermediate buffer on all transfers between the Channel Data Register and the Channel Buffer Register.

3. During 16-bit operations, the Exchange Assembly Register also is a buffer between the Channel Data Register and the selected device. On output, when the device is ready, data is transferred from the Channel Data Register through the Exchange Assembly Register to the device. Similarly, on input, data from the device goes through the Exchange Assembly Register to the Channel Data Register.

4.2.3 Instructions

Instructions required to use the data channels are described below:

SFL	=M, , 1	Set a Function Line in bank 1 as defined by the effective address.
TSL	=M, , 1	Test a Sense Line in bank 1 as defined by the effective address.
Options:	*	
Flags	B for SFL instructions Z for TSL instructions	

There are four banks associated with TSL and SFL instructions. The data channels are considered bank 1, and any SFL or TSL instructions which refer to the data channels or devices connected to the channels should use the bank 1 designation.

Immediate addressing always must be used with all SFL and TSL instructions. Indirect addressing and indexing may also be used if desired.

The B flag is set following an SFL instruction if the specified function could not be performed; the B flag will be reset if the specified function was performed. The Z flag is set following a TSL instruction if the specified sense line or indicator was true; the Z flag will be reset if the specified sense line is not true.

The SFL instructions related to the data channels are as shown on Figure 4-3.

The SFL instruction for initialize channel/connect device will be executed only if the channel is idle, and no device is already connected to the channel. If a device is already connected, the SFL will be rejected and the B flag set. When the channel is not busy, the SFL performs three functions:

1. Device D is logically connected to the channel, and is initialized for data transfer.

2. All channel registers and indicators (except for CHRI and CHSI) are reset to initial conditions.

3. The Channel Function Register (CFR) is loaded with the 8 least significant bits of SFL address which specify the operation to be performed. These bits in the SFL address have the following meaning:

Bits	Value	Operation
E	1	Exec bit transfer
	0	No exec bit transfer
B	1	Binary transfer without code conversion
	0	BCD transfer with code conversion
L	1	Left half for half-word transfers to or from memory
	0	Right half for half-word transfers to or from memory
A	1	The L bit of the CFR is to be complemented after each half-word transfer to or from memory is completed
	0	The L bit is not to be complemented

Bits	Value	Operation
I	1	Input operation
	0	Output operation
N	0	Transfer E bits only
	1	Transfer 8-bit bytes, one per half-word
	2	Transfer 8-bit bytes, two per half-word
	3	Transfer 4-bit bytes, four per half-word
	4	Transfer 4-bit bytes, four per half-word
	5	Transfer 4-bit bytes, one per half-word
	6	Transfer 4-bit bytes, two per half-word
	7	Transfer 4-bit bytes, three per half-word

Details of the byte assembly/disassembly are discussed later. None of the remaining SFL instructions are ever rejected.

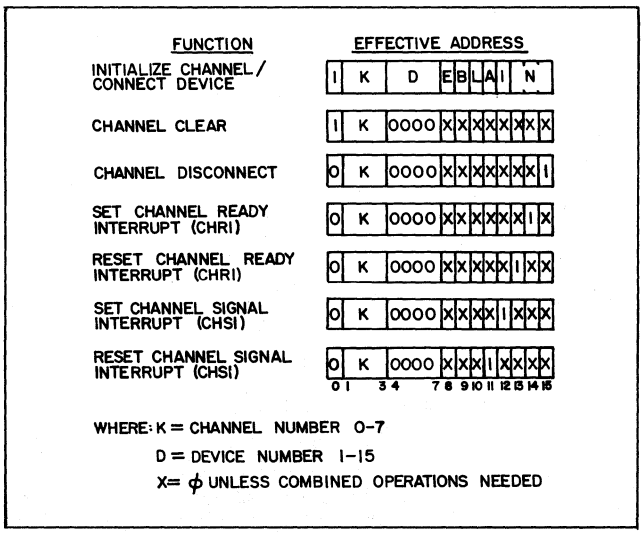


Figure 4-3. Data Channel SFL Instructions

The SFL Channel Clear instruction is an unconditional command; this instruction will disconnect the selected device and terminate any current operation, regardless of the state of the current operation. All channel indicators will be reset, and all indicators on devices connected to the channel will be reset, whether or not the device was selected when the SFL was executed. This instruction is similar to the console initialize control, but it affects only one data channel, rather than all channels. If conditional disconnect command is properly used, the unconditional disconnect need not be used except in error routines. Care should be exercised in the use of this instruction to avoid interfering with valid channel operations.

The SFL Disconnect instruction sets the CHD control indicator in the channel. The CHD indicator being set causes a disconnect action which is conditional on the state of the channel as follows:

1. If an output operation is in progress (I = O), then the device will be disconnected when the channel is ready (CHRY = 1). This feature permits the channel to complete the transfer of its current half-word before disconnecting the device.

2. If the ADCP is not in control (CHA = 0) and an input operation is in progress (I = 1), then the device will be disconnected immediately when CHD gets set.

3. If the ADCP is in control (CHA = 1), and an input operation is in progress (I = 1), then the device will be disconnected immediately after the next transfer into memory. This feature permits the use of SFL disconnect to terminate an ADCP operation in the middle of a block transfer without losing a completely assembled half-word. The data transfer operation can be resumed and completed later, provided that the channel control word is saved (using an STCC instruction).

The TSL instructions related to the data channels are shown on Figure 4-4.

The Z flag is set following a TSL instruction if the specified indicator is true, and reset if the indicator is not true.

The instructions pertaining to the Channel Data Register are the following:

LDCD

	M, , K	Load the Channel Data Register in channel K with a half-word at the effective address. The L bit in the Channel Function Register in the channel determines whether the right half or left half of the word at the effective address will be used.
STCD	M, , K	Store the contents of the Channel Data Register in channel K into the half-word at the effective address. The L bit in the Channel Function Register determines whether to store into the right half or left half.

Channel: K = channel number 0-7

Options: *

Flags: None

For both LDCD and STCD instructions, the computer waits for CDRY to be true before executing the transfer. If a given Data Channel is under ADCP control (CHA is true), no instruction to *that* Data Channel will be executed.

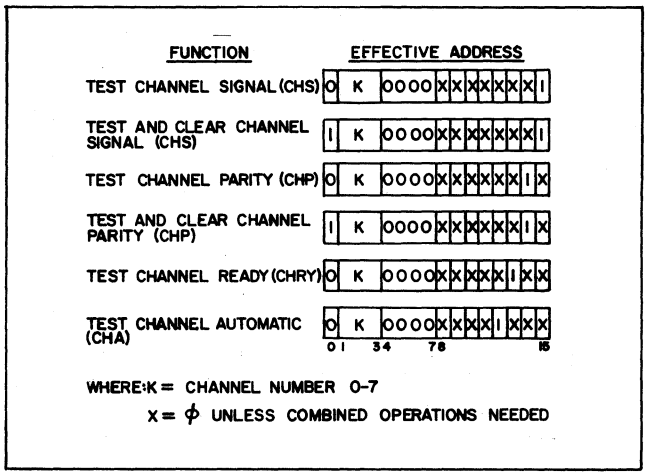


Figure 4-4. Data Channel TSL Instructions

4.2.4 Programming

The various modes of data transfer which can be achieved with the data channel are as follows:

1. Program controlled data transfer without interrupts.
2. Program controlled data transfer using interrupts.
3. Automatic data channel transfers.
4. Auto load or auto dump operations.

Auto load and auto dump operations are discussed under console operations. Automatic data channel operations are discussed in Paragraph 4. 3.

In program controlled operations, data is transferred between memory and a data channel by executing LDCD or STCD instructions.

The general sequence of instructions required for program controlled transfers without interrupts is the following:

1. Initialize the channel and connect a device with an appropriate SFL instruction.
2. Test the B flag to make sure the channel command was not rejected.

3. Transfer half-words to or from the data channel using LDCD or STCD instructions.

4. Test channel or device conditions using appropriate TSL instructions.

5. Terminate the operation and disconnect the device with a SFL disconnect instruction.

Since LDCD and STCD instructions are not executed until the channel is ready (CHRY = 1), no special timing considerations are required to transfer the data. The transfer instructions will be executed at a rate determined by the speed of the selected device. For relatively slow peripheral devices, this method of data transfer can be inefficient.

The use of the channel interrupt capability frees the processor for other tasks during the time that the data channel is not ready for a transfer to or from memory. The general sequence of instructions required to achieve program-controlled output with interrupts is the following:

1. Initialize the channel and connect a device with an appropriate SFL instruction; test the B flag to assure that the SFL was accepted.

2. Set CHRI to enable a channel ready interrupt and proceed with processing task. Note steps 1 and 2 may be inter-changed if desired.

3. When the channel interrupt occurs due to the channel becoming ready, transfer a half-word to or from the channel with an LDCD or STCD instruction. If more transfers are needed, return to the processing task.

4. Repeat the interrupt procedure until all data is transferred; then execute an SFL channel disconnect to terminate the process.

The basic programming sequences are illustrated in Figure 4-5.

4.2.5 Byte Assembly/Disassembly

The Exchange Assembly Register (EAR) handles the assembly of 8-bit bytes into half-words on input, and the disassembly of half-words into 8-bit bytes on output. When the exec bit is transferred with a half-word, it is treated like an additional 8-bit byte during assembly or disassembly. The Channel Buffer Register handles the assembly of 4-bit bytes into 8-bit bytes on input, and the disassembly of 8-bit bytes into 4-bit bytes on output.

All possible variations for byte size/byte count are shown in Figure 4-6. Note that bytes are right justified and left precedent within the half-word. That is, the left most byte to be transferred will always be transferred first. The numbers in the figure refer to the order of bytes transferred to or from the device.

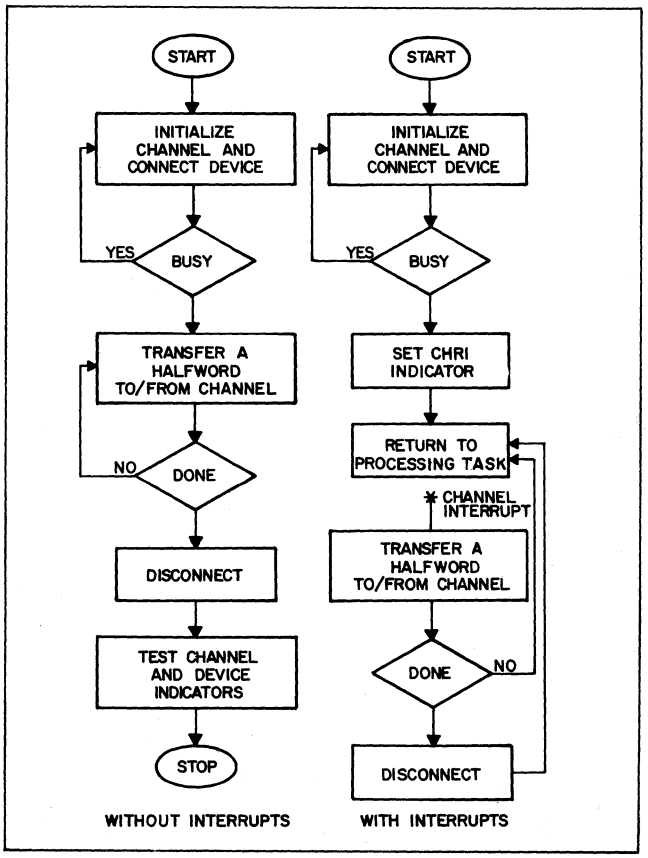


Figure 4-5. Program-Controlled Data Transfer

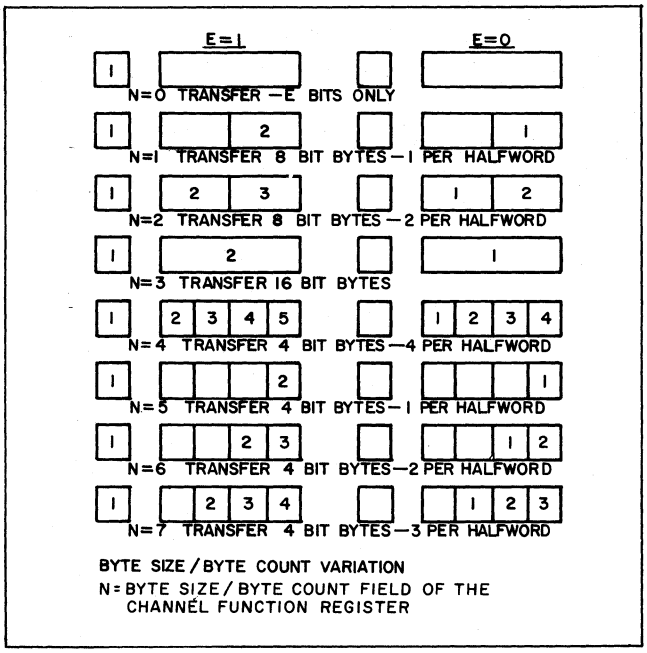


Figure 4-6. Byte Size/Byte Count Variation

All bits to the left of the first byte transferred will be set to zero on input, and ignored on output.

4.2.6 Code Conversion

The code conversion controlled by the B bit in the Channel Function Register, converts binary to BCD on output, and BCD to binary on input. The characters of the 8400 character set and the corresponding binary and BCD codes are shown in the Appendixes.

4.3 AUTOMATIC DATA CHANNEL PROCESSOR

4.3.1 Function

The Automatic Data Channel Processor (ADCP) is an optional expansion to the Exchange Module. This feature permits automatic transfer of data blocks, direct access to memory from the Exchange Module, as well as simultaneous program execution and data transfer.

The ADCP, using its own set of control words, permits data transfer, independent of the processor, between memory and an external device. Once an automatic data channel operation is initiated, the data transfer continues autonomously until the operation is completed or intervention by the processor.

When referring to separate memory banks, the ADCP and processor operate at full speed without interaction from one another. Concurrent requests from the ADCP and the processor to the same memory bank are handled on a cycle stealing basis, with the ADCP having priority. The presence of the *ADCP* option does not preclude program controlled operations on a data channel.

4.3.2 Structure

The ADCP involves the following elements:

1. One 32-bit Channel Control Register (CCR) for each channel. These registers are arranged in a high speed integrated circuit.

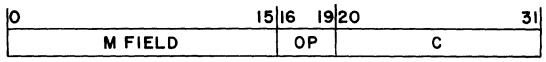
2. A 32-bit Exchange Control Register (ECR) which holds the control word for the operation currently in progress. All accesses to the CCR stack are made through ECR.

3. Direct data and control busses between the Exchange Module and memory.

4. One Channel Automatic Indicator (CHA) for each channel. The CHA is set whenever the CCR is loaded with a new command and reset whenever the channel is cleared or disconnected.

4.3.3 Control Words

The contents of the Channel Control Register (CCR) specifies the type of data transfer operation to be performed. In general, data is transferred to or from consecutive memory locations starting at some specified location M. The length of the block to be transferred is controlled either by a count decrementing to zero, or by the receipt of a signal in the data. The format of the Channel Control word is shown in Figure 4-7.



where

M = address of the first memory location involved in the transfer

OP = operation code which specifies how to transfer the data, and how to terminate the operation

C = count specifying the number of memory locations involved in the transfer.

Figure 4-7. Channel Control Word Format

For symbolic assembly purposes, the 32-bit control word for the Channel Control Register is expressed as:

OP M,,C

The OP codes and their meaning are listed below:

Table 4-1. OP Codes

Mnemonic OP-Code	Binary OP-Code	Function
TCD	1101	Transfer until count is zero, then disconnect and interrupt.
SCD	0101	Skip until count is zero, then disconnect and interrupt.
TCI	1001	Transfer until count is zero, then interrupt.
SCI	0001	Skip until count is zero, then interrupt.
TSD	1110	Transfer until signal is received, then disconnect and interrupt.

Table 4-1. OP Codes (Cont)

Mnemonic OP-Code	Binary OP-Code	Function
SSD	0110	Skip until signal is received, then disconnect and interrupt.
TSI	1010	Transfer until signal is received, then interrupt.
SSI	0010	Skip until signal is received and then interrupt.
TED	1111	Transfer until *either* count is zero, or signal is received, then disconnect and interrupt.
SED	0111	Skip until *either* count is zero, or signal is received, then disconnect and interrupt.
TEI	1011	Transfer until *either* count is zero, or signal is received, then interrupt.
SEI	0011	Skip until *either* count is zero, or signal is received, then interrupt.

Note that an interrupt is generated following all ADCP operations.

The skip operation is useful for passing over a block of data on input without transferring any information.

The count refers to the number of memory locations involved in the operation. If an *alternate* mode is used, then two half-word transfers constitute a count of one. If *non-alternate* mode is used, then each half-word transfer corresponds to a count of one.

The signal refers to those conditions appropriate to the selected device that would set the signal flip flop (CHS) in the channel.

The Channel Control Register is loaded with a control word using the instruction

LDCC M, , K

where

K = channel number 0-7

M = location in core memory of the 32-bit control word to be loaded into the CCR.

The contents of the Channel Control Register can be stored using the instruction

STCC M, , K

where

K = channel number 0-7

M = location in core memory in which the 32 bits of the CCR should be stored.

4.3.4 Operation

Initiating an ADCP operation on a data channel involves two steps:

1. Initializing the channel and connecting a device

2. Loading the Channel Control Register with a control word.

Channel initialization is performed by the channel SFL instruction which specifies the following:

channel number
device number
exec bits/no exec bits
code conversion/no code conversion
left half first/right half first
alternate/no alternate
to memory/from memory
byte size
byte count

The Channel Control Register is loaded with an LDCC instruction which also sets the Channel Automatic Indicator. Once the channel is initialized and Channel Automatic Indicator is set, the channel operation is under ADCP control and the processor is no longer required.

The LDCC instruction can either precede or follow the SFL instruction for channel initialization. Before initiating a new ADCP operation it is important to check that the channel is not busy on a previous ADCP request. An appropriate instruction sequence is as follows:

```
SFL   =CFCODE, , 1        INIT CHANNEL
JB    =*-1                TEST IF CHANNEL
                            BUSY
LDCC  =CCWORD, , K        LOAD CCR
```

An alternate sequence is as follows:

```
TSL   =CHA, , 1           TEST AUTOMATIC
JZ     *-1                WAIT IF CHA SET
LDCC   CCWORD, , K        LOAD CCR
SFL   =CFCODE, , 1        INIT CHANNEL
```

Once the Channel Automatic Indicator is set and the channel is initialized, all transfers to/from memory are under ADCP control. The instructions LDCD and STCD cannot be executed by the processor when the Channel Automatic Indicator is set.

If the transfer is in *non-alternate* mode, the M field is incremented and the C field decremented after every transfer to/from memory. If the transfer is in *alternate* mode, the M field is incremented and the C field decremented after every second transfer to/from memory, independent of the initial L/R bit in the Channel Function Register.

The Channel Control Register is updated immediately following the transfer to or from memory. The sequence of operations is summarized in Figure 4-8.

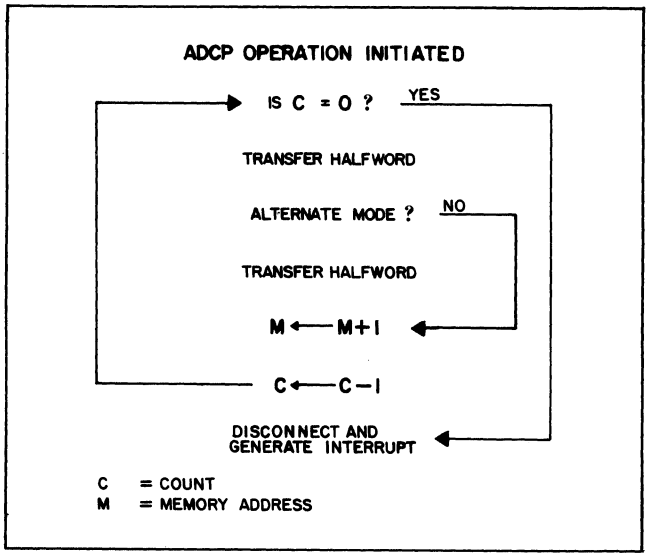


Figure 4-8. ADCP Action for a TCD Operation

Note that if a device is selected to read or write, and an ADCP count operation is started with the count zero, a record of information could be skipped on the external device. Care should be exercised in initiating operations with the count zero.

An ADCP operation can be terminated in several ways:

1. The count in the Channel Control Register reaching zero while a count operation is in progress.
2. A signal being detected in the data while a signal operation is in progress.
3. The processor executing an SFL instruction for channel disconnect.
4. The processor executing an SFL instruction for channel clear.
5. Depressing INITIALIZE pushbutton on console.
6. Depressing AUTO LOAD pushbutton on the console.
7. Depressing AUTO DUMP pushbutton on the console.

When an operation is terminated, the Channel Control Register is not changed; it remains as it was when the operation was completed. The STCC instruction can be used following an ADCP operation to examine the contents of Channel Control Register. Note that when reading, there is a half-word uncertainty--as measured by the remaining count in Channel Control Register--as to how many words were transferred to memory.

On termination, if disconnect was specified, the Channel Disconnect (CHD) flip-flop is set. The data channel and device are disconnected after the current half-word transfer is complete. For details on the disconnect action, refer to the section on operation of the Exchange Module.

If an SFL for conditional disconnect is executed during an ADCP input operation, the operation is stopped after the next complete half-word transfer. If an SFL for channel clear is executed, the ADCP operation is terminated immediately and all channel indicators are reset.

Note, that after reading data with a TCI and TCD operation, the SFL conditional disconnect waits for the *next* transfer before disconnecting the device. If no further data is to be transferred to the channel, an SFL channel clear must be used to disconnect.

4.4 SYSTEM INTERFACE

4.4.1 Function

The System Interface is designed for general communication to or from external devices where the function of the data channel is not required nor appropriate. Means are provided for direct data transfer between the processor and the registers of external devices. The capability for testing signals

and setting conditions on external devices is also provided. This interface facilitates direct computer control with a variety of system configurations.

4.4.2 Structure

The elements of the system interface are the following:

1. A 17-bit data output buss
2. A 17-bit data input buss
3. A 4-bit address field R
4. Up to 256 external interrupt lines
5. Command and control lines
6. The outputs from four system control buttons (SC1-SC4) on the operator's console.

The 17 bits (16 data plus 1 exec) of the output buss can be directed to any of 16 external registers as defined by the address field R. The output buss also transfers the address field of SFL and TSL instructions to external devices.

The input buss can receive data from any of 16 external registers as specified by the address field R.

Details of the external interrupts are discussed with the interrupt system in Chapter 3.

The outputs from the system control buttons are provided to enable operator control of external devices from the console. The system control buttons are latching switches. When the button is depressed, the output line is high (+5 volts); when the button is released, the output line is low (ground).

Provision is also made for four internal interrupt lines on the system interface. For those machines with no more than four data channels, the internal interrupt lines corresponding to channels 4, 5, 6, and 7 are accessible through the interface. For machines with more than four data channels, these interrupt lines are not available for use with the system interface.

All operations with the system interface operate on an asynchronous request/response/release basis with the external device. The general sequence of events during one operation is as follows:

1. The processor generates a request signal (input or output command) signal to the selected device, as defined by the address steering field "R".
2. The external device addressed by the request raises a ready signal when it is prepared to transfer or receive data.
3. A transfer complete signal is generated by the computer when the data is ready to transmit on output, or after the data has been received on input.
4. The device resets the ready signal when it has completed the transfer, and releases the computer from the current operation.

The output control lines associated with the system interface are listed below. The signals on these lines, generated by the computer, inform the external devices about the operation in progress.

Table 4-2. System Interface Output Control Lines

Output Line	Meaning
LDOB	Load output buss operation
STIB	Store input buss operation
SFL2	SFL operation in bank 2
SFL3	SFL operation in bank 3
TSL2	TSL operation in bank 2
TSL3	TSL operation in bank 3
TCS	Transfer complete strobe. (This signal is set when data is ready for output, or has been received on input.)

The input control lines are listed below. The signals on these lines, generated by external devices, are in response to the output control signals.

Table 4-3. System Interface Input Control Lines

Input Line	Meaning
SBY2	B flag response for bank 2 SFL's
SBY3	B flag response for bank 3 SFL's
SZE2	Z flag response for bank 2 TSL's
SZE3	Z flag response for bank 3 TSL's
SRDY	System Ready. (This signal is set when the device is ready to transmit or receive information, and reset when the operation is complete.)

The SRDY signal must be reset by the external device before the computer will be released and allowed to proceed. The execution time, therefore, of all instructions pertaining to the system interface depends on the speed of the external device addressed by the instruction.

4.4.3 SFL/TSL Instructions

SFL = M, , B Set Function Line in bank B. The effective address is transferred over the 16 data lines of the output buss, and appropriate control signals are set to indicate an SFL instruction in Bank B.

Banks: B = 2 or 3 for the system interface

Options: *

Flags: The B flag is set if the selected line (or lines) is already set, or if some conditions prevent the setting of the selected line (or lines).

TSL = M, , B Test Sense Line in bank B. The effective address is transferred over the 16 data lines of the output buss and appropriate control signals are set to indicate a TSL instruction in Bank B.

Banks: B = 2 or 3 for the system interface

Options: *

Flags: The Z flag is set if the selected line is set (binary 1). If multiple lines are addressed, the Z flag is set if any of the selected lines are high.

There are no restrictions on the use of address fields for either SFL or TSL instructions in Bank 2 or 3. External decoding logic can be added to the system interface to permit selection of up to 2^{16} line for each bank. Note, however, that EAI standard System Interface Expansion codes have been allocated to ensure satisfactory field expansion of a system and programming compatibility.

LDOB M, , R Load external register R on the Output Buss with the contents of memory location M.

STIB M, , R Store the contents of external register R on the Input Buss into memory location M.

Registers: R = 0, 1, 2 ....., 15

Options: *, =, /

Flags: None

If more than 16 external registers are to be used, an external address buffer can be added to the interface. LDOB and STIB instructions could then be preceded by an SFL instruction to set up the external address buffer. Standard EAI modules in this area (such as I/O Buss Controllers) are available.

4.5 PERIPHERAL DEVICES

4.5.1 Typewriter

The 8400 desk typewriter is a 132 column IBM Selectric. The typewriter can be connected to the data channel as an input or an output device, or it can be used for entering data directly into the computer registers. Details of the latter capability are discussed with console operations, Chapter 5. Parity is checked on both input and output. The maximum data rate is 15 characters per second.

A Typewriter Ready indicator on the console is lit when the typewriter is connected to the channel and waiting for input.

The typewriter keyboard and the corresponding character octal codes are shown in Figure 4-9.

4.5.1.1 *Data Format.* The typewriter transmits and receives two types of information: data characters and control characters. The data characters are the 64 members of the EAI 8400 character set. Control characters on the typewriter are the following:

carriage return

tab

backspace

upper case shift

lower case shift

index

Eight lines are used to transfer information to or from the typewriter--6 data lines, 1 control line, and 1 parity line. The parity bit is used by the data channel, and this bit never appears in core memory. Refer to Figure 4-10. The control line is high for all control characters, and low for all data characters. The position of a typewriter character in an 8-bit byte in core memory is shown in Figure 4-11.

The typewriter transmits and receives data characters in *BCD* mode. The data channel makes the required code conversion from BCD code to internal 8400 code on input, and vice versa on output. Details of the code conversion are also discussed in Appendix 5.

If byte size 4 mode is used, only the 4 least significant bits per character are transferred. In this mode, however, parity may not be correct.

4.5.1.2 *Programming.* The general sequence of instructions required to transfer data to or from the typewriter is the following:

1. Initialize the channel and connect the device with a channel SFL instruction. The SFL code should specify *BCD* mode to achieve proper code conversion.

2. Test busy to assure that the channel instruction was accepted by the Exchange Module.

3. Select ribbon color with a device SFL instruction.

4. Transfer data.

5. Disconnect the typewriter with a channel SFL instruction.

The ribbon color can be selected by an SFL instruction with the address field shown in Figure 4-12.

The SFL instruction for ribbon control can be issued any time, where the device is active or not; these instructions are never rejected. Black is considered the normal color. The ribbon color will be set to black when any of the following occur:

1. Channel Clear SFL is executed
2. Console Initialize
3. Auto Load
4. Auto Dump

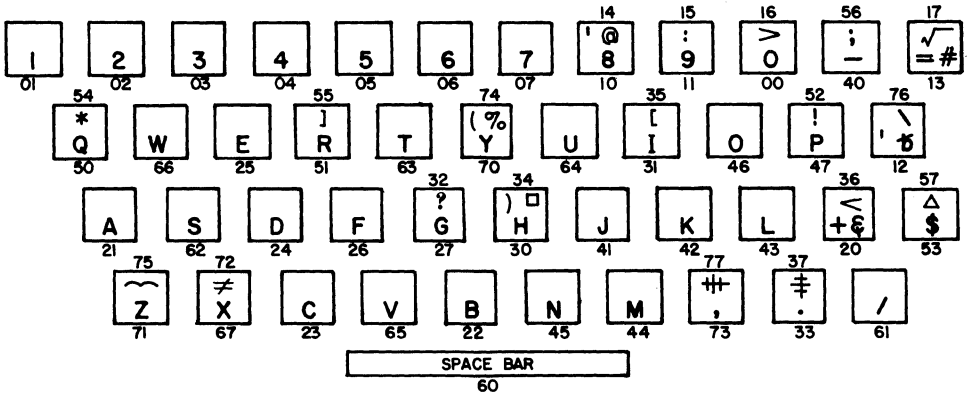


Figure 4-9. Typewriter Keyboard

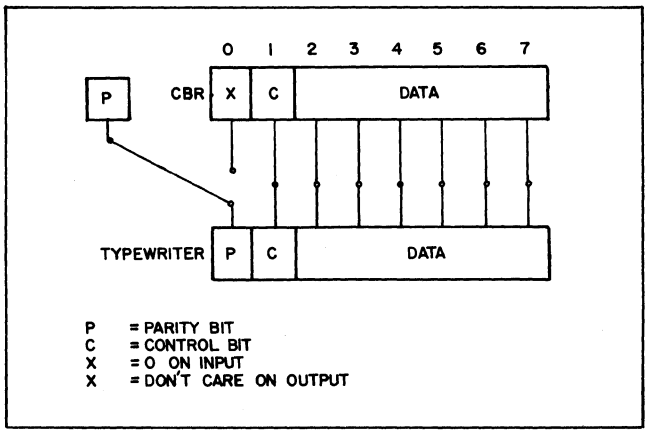


Figure 4-10. Connection of Typewriter to the Channel Buffer Register

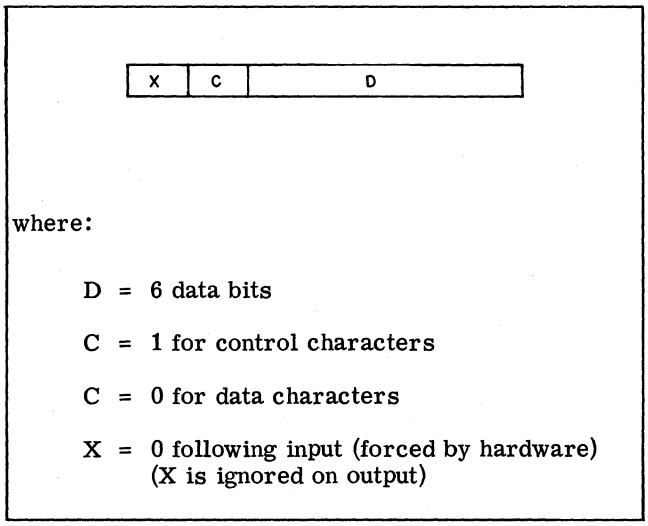


Figure 4-11. Typewriter Character Position in Memory

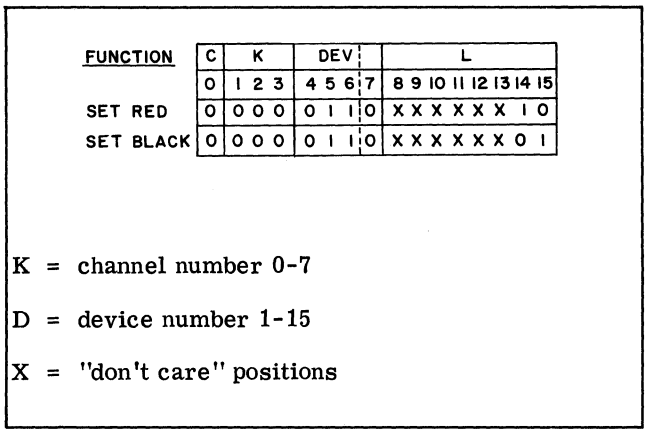


Figure 4-12. Typewriter SFL Codes

No TSL instructions are associated with the typewriter. Parity can be tested using the channel parity indicator. The channel signal indicator will be set in two situations:

1. During input when a carriage return is typed. The occurrence of channel signal on input will always terminate the current word assembly permitting transfer of the (possibly incomplete) half-word into memory.

2. During output if the typewriter printing mechanism fails to respond to an output character within a preset amount of time (approximately 110 msec).

4.5.2 Card Reader (Models 8452, 8453, and 8454)

The card reader is an input device which reads punched cards. All models can handle either 51 or 80 column cards. In addition, some units can handle 60 and 66 column cards. The Model 8452 Card Reader reads 400 cards per minute (cpm), the Model 8453 reads 800 cpm, and the Model 8454 reads 1400 cpm. All models provide read-check circuits and validity-checking apparatus; the models are interchangeable and can be programmed and operated in the same way.

4.5.2.1 *Operator Controls and Indicators.* The following switches and indicators are located on the control panel of the card reader:

Power On	This switch when pressed, applies power to the card reader.
Power Off	This switch, when pressed turns off the card reader.
Not Ready	This indicator is lit whenever the reader is not ready. The ready condition is defined under program indicators.
Feed Check	This indicator is lit whenever a jam occurs in the card feeding mechanism.
Read Check	This indicator is lit whenever a fault in the read circuitry is detected.
Validity On	This latching switch, when depressed, enables the validity-checking circuit, and lights the indicator. Releasing the switch inhibits the validity checking.
Validity Off	This indicator is lit when the validity-checking circuits detect an invalid character.
Reset	This switch, when pressed, clears the error indicators.
Start	This switch, when pressed, resets the Not Ready indicator and permits the reader to be put on line for data transfer.
Stop	This switch, when pressed, places the reader in a Not Ready condition.

4.5.2.2 *Data Format.* In general, cards have 80 columns and 12 rows. The card reader is capable of reading two types of cards: Hollerith cards and binary cards. Hollerith cards have one alphanumeric character per column, and each character is expressed in a 12-bit Hollerith code. The Hollerith code for the EAI 8400 character set is shown in Figure 4-13. When reading Hollerith cards, the card reader translates the 12-bit card code into a 6-bit BCD code. By using the *BCD* mode in the data channel, the BCD code is automatically converted to internal 8400 binary codes. Cards will be read as Hollerith cards when the data channel is selected in *BCD* mode. The validity checking in the reader pertains to Hollerith cards only; the reader checks that the 12-bit character punched on the card is a legal Hollerith character.

Binary cards are read as two 6-bit binary characters per column. The card reader strobes each column twice, reading the top 6-bits of a column first, and then the lower 6-bits of the same column. Each card contains 160 total characters. No decoding or validity checks are performed for binary cards. The correspondence between a binary card character and its image in core memory is shown in Figure 4-14. Cards are always selected as binary cards when the data channel is selected in *binary* mode.

Provision is made for reading mixed decks of binary and Hollerith cards as follows: if the data channel is selected in *BCD* mode, and the first column of a card has the 7 and 9 holes punched, the mode will be automatically switched to binary, and that card will

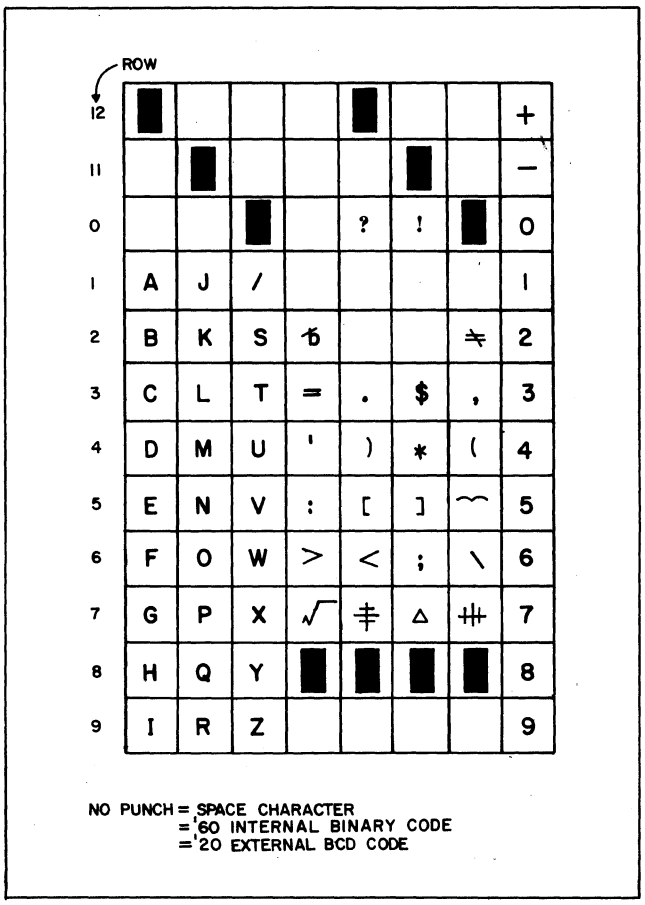


Figure 4-13. Hollerith-BCD Code on a Card

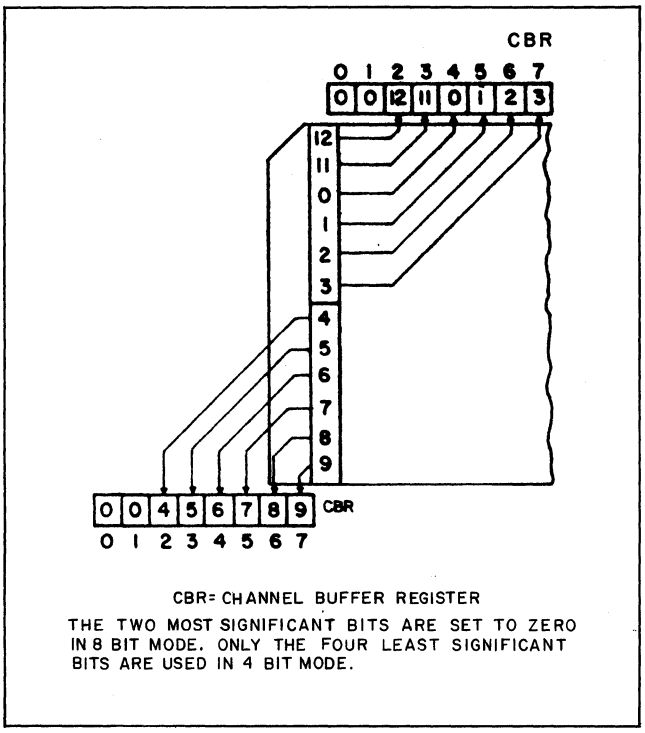


Figure 4-14. Position of Binary Card Characters in an 8-bit byte

be read as a binary card. The first column will also be strobed twice in this situation, and 160 binary characters will result from a binary card in mixed mode. At the end of the card, the mode is switched back to the BCD state.

The *8-bit* mode in the data channel normally will be used with the card reader. Since 6-bit characters are generated by the reader, the 2 most significant bits per 8-bit byte are set to zero by the data channel prior to transfer into memory. The *4-bit* mode in the data channel can be used in *Binary mode only;* in this case, only the 4 least significant bits of each 6-bit character generated by the reader are transferred to the data channel.

4.5.2.3 *Program Controls and Indicators.* The controls and indicators accessible by program are the following:

Reader Ready — This indicator is true when the following conditions exist on the reader:

1. Power on
2. Hopper not empty
3. Stacker not full
4. Start button depressed
5. No feed check error
6. No read check error
7. No validity check error
8. All covers in place

Binary Status This indicator is set when a 7-9 punch is detected in the first column of card and is reset by any of the following:

1. SFL instruction for channel clear
2. Initiating a new card cycle, whether by SFL instruction, or by continuous card feeding
3. Manual reset control on the reader

Card Cycle In Progress This indicator is set when a new card cycle is initiated—either by SFL instruction, or by continuous card feed—and reset when all 80 columns have been read.

Reader Error This indicator is *set* when one of the following:

1. A read-circuit malfunction is detected
2. An invalid character is detected while reading in *Hollerith* mode with the Validity switch on

This indicator is *reset* by any of the following:

1. SFL instruction for channel clear
2. SFL instruction which connects the reader to the channel
3. Manual reset control on the reader

Overflow This indicator is *set* when the data channel fails to accept a character from the reader before another character is read, and *reset* by any of the following:

1. TSL instruction for test and reset
2. SFL instruction for channel clear
3. SFL instruction which connects the reader to the channel

Start Card Cycle This command starts a card on the way to the read station, and sets the Card Cycle in Progress indicator.

Device Interrupt Enable (DINE) This program-controlled switch, when set, enables a channel interrupt to occur whenever the Reader Ready indicator is high. This switch can be set only when no device is currently connected to the channel, and is automatically reset whenever a device is connected to the channel, or a channel clear instruction is executed.

The TSL instructions associated with the card reader use the address field shown in Figure 4.15.

0	1	3	4	7	8	9	10	11	12	13	14	15	
0	K		D		X	X	X	1	X	X	X	X	CARD READER ERROR
0	K		D		X	1	X	X	X	X	X	X	CARD READER CONTINUE LEVEL
0	K		D		X	X	X	X	1	X	X	X	END OF FILE
0	K		D		X	X	X	X	X	1	X	X	CARD CYCLE LEVEL
0	K		D		X	X	X	X	X	X	1	X	CARD READY LEVEL
0	K		D		X	X	X	X	X	X	X	1	BINARY MODE
1	K		D		X	X	1	X	X	X	X	X	OVERFLOW

Where

K = channel number 0-7

D = device number 1-15

X = don't care

Figure 4-15. Card Reader TSL Codes

The Z flag is set in response to a TSL instruction, if the tested indicator is true; the flag is reset if the tested indicator is false.

The SFL instructions associated with the card reader use the address field shown in Figure 4.16.

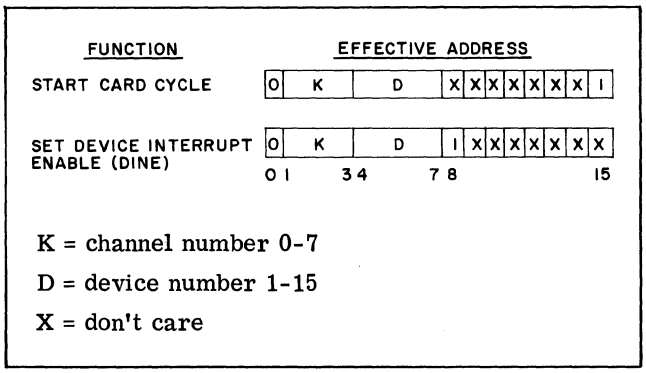


Figure 4-16. Card Reader SFL Codes

The SFL for Start Card Cycle is rejected, and the B flag set if a card cycle is in progress, or if the reader is not ready.

If another device is connected to the data channel, the SFL to set the DINE switch is rejected, and the B flag set.

4.5.2.4 *Programming*. The general sequence of instructions required to use the card reader is the following:

1. Test for end of previous card cycle with a TSL instruction.
2. Initiate a new card cycle with a device SFL instruction.
3. Test busy to make sure the SFL instruction was accepted, and take delay action if the reader is busy.
4. Initialize the channel and connect the reader with a channel SFL command.
5. Test busy to make sure the channel command was accepted.
6. Transfer data.
7. Disconnect the transfer.
8. Test error conditions using channel and device TSL instructions.

A new card cycle can be initiated either with the SFL for Start Card Cycle, or the SFL instruction which connects the reader to the channel. Once a card cycle is started, the card moves at a fixed rate through the read station, and data is generated at a rate determined by the card reader. If the SFL Start Card Cycle command is used, the reader must then be connected to the channel within 10 milliseconds, or the SFL channel command will be rejected and one card may be skipped.

In general, the channel SFL instruction which connects the reader to the channel will be rejected if either the reader is not ready, or if more than 10 milliseconds have elapsed since a new card cycle was started.

When the card reader is connected to the channel, the channel signal indicator will be set at the end of a card cycle (i.e., after all 80 columns are read). At the same time the channel signal indicator is set, a 100 microsecond timing signal will be triggered. At the end of the 100 microsecond period, if the reader has not been disconnected from the channel, a new card cycle will be started automatically. Therefore, once the reader is connected to the channel, cards are read continuously until the reader is disconnected from the channel.

No parity checking is performed by the data channel with the card reader. The channel parity indicator will be set when the reader is connected to the channel and any of the following occur:

1. Reader overflow
2. Read-Check error
3. Validity-Check error

A channel interrupt will result from the card reader in the following two cases:

1. When the reader is connected to the channel, and the reader becomes not ready.

2. When no device is connected to the channel, the DINE switch on the reader is set, and the reader becomes ready.

If the reader becomes not ready while connected to the channel, the data continues to be transferred, and the card will continue to move until the present cycle is complete. Once disconnected, however, the reader cannot again be connected until the not ready condition is reset (i.e., the cause of the not ready condition is removed).

4.5.3 Paper Tape Reader

The paper tape reader is an EAI model which can read 5, 7, or 8 channel tapes. The device reads 500 characters per second in either forward or reverse direction. Fanfold tape containers for tape supply and take-up are provided.

Desk controls and indicators pertaining to the reader are the following:

Power On/Off	This switch controls the power to the reader transport and electronics.
Run/Load	This switch must be in load position to insert or remove tapes from the reader. The switch must be in run position for the tape to move.
Reader Ready	This indicator is lit when the reader has been connected to the data channel.

4.5.3.1 *Data Format.* Paper Tapes can be read in either forward or reverse direction using either binary or BCD channel options. In all cases, the most significant bit per character on the tape is considered a parity bit. The data channel checks the lateral parity of each character, using odd parity in binary mode and even parity in *BCD* mode. The channel parity indicator is set when bad parity is detected.

Either a *4* or *8-bit* mode can be used, with or without exec bits. In the *8-bit* mode, the 7 least significant bits per character are transferred into the 7 least significant bits per byte, and the most significant bit is set to zero. In the *4-bit* mode, only the 4 least significant bits per character plus the parity bit are transferred to the data channel. The connection of the reader to the channel buffer register is similar to that of the typewriter as shown in Figure 4.10.

Blank tape is always skipped automatically in both binary and *BCD* mode.

4.5.3.2 *Programming.* The general sequence of instructions required to use the paper tape reader is the following:

1. Set forward or reverse direction with a device SFL instruction.

2. Test busy flag to see if the direction command was accepted.

3. Initialize the channel and connect the reader with a channel SFL instruction.

4. Test busy flag to ensure that the channel command was accepted.

5. Transfer data.

6. Disconnect the reader with a channel SFL instruction.

The reader direction can be selected by an SFL instruction with the address field shown in Figure 4-17.

	C	K			D				L							
M FIELD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FWD	0	0	0	0	0	0	1	0	X	X	X	X	X	X	0	1
REV	0	0	0	0	0	0	1	0	X	X	X	X	X	X	1	0

K = channel number 0-7

D = device number 1-15

X = don't care

Figure 4-17. Paper Tape Reader SFL Instructions

The direction control instructions will be rejected if the tape is moving in a direction contrary to that of the SFL command. This interlock prevents damage to the reader due to sudden reversal of drive power. The forward direction is considered normal. The Forward Direction is set when any of the following occur:

1. SFL channel clear instruction is executed
2. Console initialize
3. Auto Load
4. Auto Dump

When reading in *reverse* mode, the data channel assembled half-word has the same form as when the tape is read in the forward direction. The half-word transfers into memory, however, must be programmed differently in *forward* and *reverse* mode to achieve identical full-word formats in memory.

The SFL instruction which connects the reader to the channel starts the tape in motion. The channel SFL command will be rejected and the B flag set if the reader power is off, or if the reader is in a load (not run) condition. Tape motion will be automatically inhibited if either the channel buffer register is not ready to accept a character from the reader, or if the channel data register is not ready to accept a character from the buffer register. The channel buffer is not ready when it contains a completely assembled 8-bit byte, and is waiting to transfer its contents through the assembly register into the channel data register. The channel data register is not ready when it contains a completely assembled half-word, and is waiting to transfer its contents to core memory.

The stopping time for the tape reader (when running at 500 characters per second) is less than 500 microseconds. When the tape motion is inhibited by the channel, the tape stops on the character just transferred to the channel.

There are no TSL instructions associated with the tape reader. Parity can be tested using the channel parity indicator.

The channel signal indicator will be set whenever a stop code (octal 100) is detected. The detection of a stop code will also terminate the current word assembly, permitting transfer of the (possibly incomplete) half-word into memory. Stop code detection is enabled even in *4-bit* mode. Direction of a stop code during an auto load operation will terminate the input and disconnect the reader.

4.5.4 Paper Tape Punch

The paper tape punch is an EAI Model which handles 5 to 8 channel paper tapes. Ten characters per inch are punched at 110 characters per second.

Desk controls and indicators pertaining to the punch are the following:

Power On/Off	This switch, when on, enables program control over the punch power. When this switch is off, the punch power remains off unconditionally.
Tape Feed	This switch, if pushed when the punch power is on, causes blank tape to be punched as long as the switch is depressed.
Tape Low	This indicator is lit when only one foot or less of tape remains to be punched.
Punch Ready	This indicator is lit whenever the punch has been connected to the data channel for data transfer.

4.5.4.1 *Data Format.* Both *binary* and *BCD* mode may be used for transfer of information to the punch. A parity bit is generated in the data channel--odd parity for *binary* mode, and even parity for *BCD* mode. The parity bit is punched as the most significant bit of each character. The connection of the punch to the Channel Buffer Register is similar to that of the typewriter.

Either *4* or *8-bit* mode can be used, with or without exec bits. In the *8-bit* mode, the 7 least significant bits per byte are transferred to the punch, and the most significant bit is ignored. If exec bits are transferred, they are always punched first on the tape.

Blank tape can be produced by program by punching two characters (octal 12) in *BCD* mode, transferring 8 bits per byte, and no exec bits.

No parity checking is performed during output to the punch.

The stop code, a control character peculiar to paper tape, is defined as octal 100.

4.5.4.2 *Programming.* The general sequence of instructions required to use the punch is the following:

1. Turn on punch power with a device SFL instruction.
2. Initialize the channel and connect the punch with a channel SFL instruction.
3. Test the busy flag to assure that the channel initialization instruction was accepted.
4. Output the data.
5. Disconnect the punch with a channel SFL instruction.
6. Turn off the punch power (if desired) with a device SFL instruction.

The punch power is controlled by SFL instructions with the address field shown in Figure 4-18.

	C	K	D	L
M FIELD	0	1 2 3	4 5 6 7	8 9 10 11 12 13 14 15
POWER ON	0	0 0 0	0 1 0 0	X X X X X X 1 0
POWER OFF	0	0 0 0	0 1 0 0	X X X X X X 0 1

K = channel number 0-7

D = device number 1-15

X = don't care

Figure 4-18. Paper Tape Punch SFL Instructions

The SFL instructions for punch power can be issued anytime, whether the device is active or not; these instructions are never rejected.

The SFL instruction for channel initialization will be rejected, and a busy response generated if:

1. Punch power is not turned on
2. Tape low signal is true
3. Data channel is unavailable

No TSL instructions are associated with the punch.

The channel signal indicator will be set if the tape low indicator becomes true while the punch is connected to the channel. Punching can continue in this situation.

4.5.5 Line Printer (Models 8461, 8462, and 8463)

The line printer is an output device which prints one line of information at a time. Printers are available with the following specifications:

300, 600, or 1000 lines per minute
136 characters per line
10 characters per inch
6 or 8 lines per inch
65 character set
100 kcps data transfer rate

Model 8461, 300 LPM; model 8462, 600 LPM; model 8463, 1000 LPM, are all programmed as one printer.

The printer is internally buffered, and can hold one complete line of information at a time.

4.5.5.1 *Operator Controls and Indicators.* The controls and indicators located on the control of the printer are as follows:

Start	This switch is used to put the printer on-line, and permit the printer to accept data from the computer.
Stop	This switch is used to put the printer off-line, and make it unable to accept data from the computer.
Top-of-Form	This switch, if pushed when the printer is off-line, advances the paper to the first printing position of the next form.
Test	This switch, if pushed when the printer is off-line, initiates printing of a test pattern. Printing continues to the end of the current form after the release of the switch.
Yoke Open	This indicator is lit when the yoke is open.
No Paper	This indicator is lit when the printer is out of paper.
6 Line/ 8 Line	This switch/indicator selects the number of lines per inch that will be printed.
Alarm Status	This indicator is lit when any condition exists which keeps the printer from being ready.

4.5.5.2 *Data Format.* The line printer accepts characters in internal 8400 Binary Code. The *binary* mode in the data channel must be used, and all data transfers to the printer should use odd parity. The printer checks parity on all characters, and sets the channel parity indicator if a parity failure is detected. Any characters transferred with faulty parity are not printed.

All transfers must be in the *8-bit* mode. The seven least significant bits per byte are transferred to the printer. With one exception, the first bit (the most significant bit transferred) is ignored by the printer, and the six least significant bits determine the character to be printed. The end-of-line character which initiates the printing of a line is an octal 155, corresponding to a carriage return on the typewriter. This is the only control character recognized by the printer. In all other cases, a bit in the most significant position of a character is ignored.

4.5.5.3 *Vertical Format.* The general sequence of events in the printing of one line is the following:

1. Space paper
2. Send data followed by end-of-line character
3. Print line
4. Wait for next line

Paper spacing can be initiated in two ways:

1. By SFL command
2. By the first character in a line

When the first character in a new line is received by the printer, it is interpreted as vertical format specification and the character is not placed in the printer buffer. Receipt of the first character, the paper spacing, and the transfer of the remaining data overlaps with the paper advance operation. After

the paper spacing is completed, the line of characters is then printed.

If paper spacing is initiated by SFL instruction, the first character in the next line to be transferred to the printer will be ignored, and the character will not be printed. There is also an SFL instruction to inhibit paper spacing; the first character in the next line is also ignored following this SFL instruction.

If a parity failure is detected during printing of data by the printer, after printing the line, it is possible to inhibit paper advance via an SFL, and overprint the line by re-transmitting the information.

Two types of vertical formatting are available:

1. The count mode in which the number of lines is specified.

2. The tape mode in which the paper is to be advanced until a space in a specified channel on an external format tape is detected.

The external format tape is an 8 channel tape that can be punched in any desired pattern. The format tape can be easily changed by the operator.

The vertical format codes for use in first character paper control are as shown on Figure 4-19

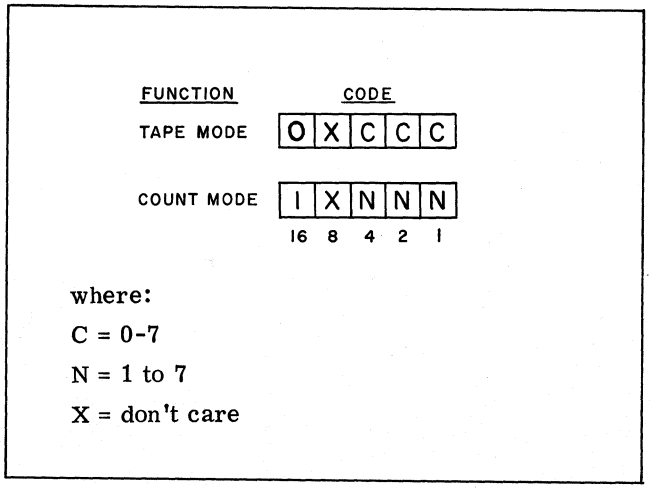


Figure 4-19. Vertical Format Codes

4.5.5.4 *Program Controls and Indicators.*

The control and indicators accessible by program are the following:

Printer Ready — This indicator is true when the power is on, the character drum or yoke is in place, the paper is in position, and the start switch has been depressed. When this indicator is low, the printer cannot be connected to the channel, or accept data from the channel.

Next Line Request (NLR) — This indicator is true when the printing of a previous line is complete, and the printer is ready for the next line. NLR is set following Channel Clear if the printer is ready.

Next Character Request — This indication will be true when the printer is ready to accept another character from the data channel. During printing, this indicator will be false.

Buffer Overflow — This indicator is set when more than 136 characters have been transmitted to the printer. This indicator is optional, and is not found on all printers.

Paper Advancing — This indicator is true whenever the paper is advancing.

Device Interrupt Enable — This program-controlled switch, when set, enables

a channel interrupt to occur whenever the Next Line Request indicator becomes true. This switch can be set by program only when no device is currently connected to the channel. This switch is automatically reset whenever a device is connected to the channel, or a channel clear instruction is executed.

The TSL instructions associated with the line printer use the address field shown in Figure 4-20.

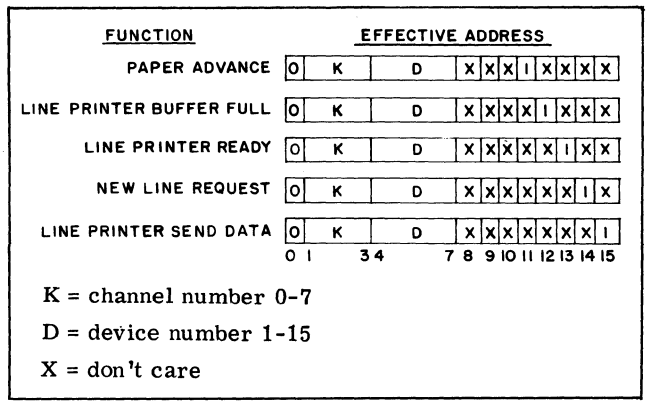


Figure 4-20. Line Printer TSL Instructions

The Z flag is set, in response to a TSL instruction, if the tested indicator is true, and the flag is reset if the tested indicator is false.

The SFL instructions associated with the line printer are shown on Figure 4-21.

SFL instructions for paper spacing are rejected if either a paper spacing or a printing cycle is in progress. The SFL to set the device-interrupt-enable-switch is rejected if any device is currently connected to the channel.

4.5.5.5 *Programming.* The general sequence of instruction required to use the line printer is as follows:

1. Test conditions on the printer, if desired, with device TSL instructions.
2. Initiate delay procedures, if required, if conditions prevent immediate transfer of data.
3. Specify paper advance with appropriate SFL instruction if it is desired to override or ignore the first character format specification.
4. Test the busy flag to make sure the paper spacing command was accepted.
5. Initialize the channel and connect the printer with a channel SFL command, specifying 8-bit bytes and *binary* mode.
6. Test the busy flag to make sure the channel command was accepted.
7. Transfer data followed by an end-of-line character.
8. Disconnect the printer.
9. Test error conditions using channel and device TSL instructions.

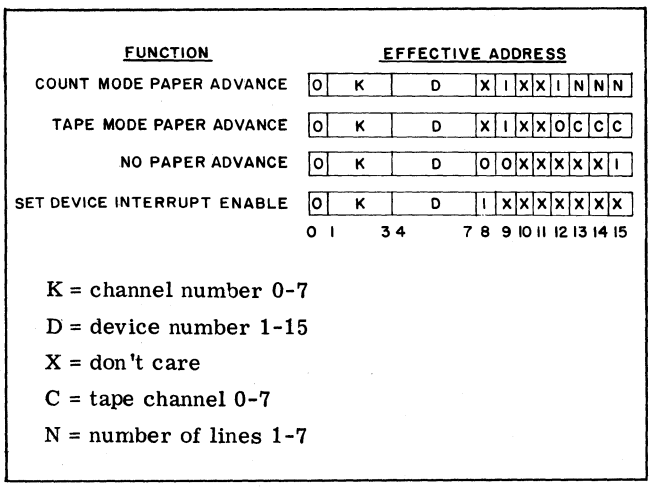


Figure 4-21. Line Printer SFL Instructions

The channel SFL command to set up the channel and connect the printer will be rejected only if the printer is in a not ready condition. It is possible to connect the printer when it is spacing paper or printing a previous line, but data transfer cannot commence until the Next Line Request signal comes high.

4.5.6 Card Punch (Models 8455 and 8456)

Type 8455 is an 80 column, row-oriented card punch capable of feeding, punching and checking cards at rates of 100 and 300 cards-per-minute respectively. Both types are equipped with individual controllers and storage buffers permitting independent card punching operation.

Transmission of data between the buffer and the punch is checked for parity to insure data integrity. A read after punch feature is provided to give a hole-count accuracy check on the data.

All units contain readily identified console buttons to facilitate operator control. Model 8455 has a hopper and stacker capacity of 800 cards. Model 8456 has a hopper capacity of 3500 cards, a primary stacker capacity of 3000 cards, an auxiliary stacker capacity of 850 cards and an error stacker capacity of 750 cards. Both types may be loaded and unloaded while operating.

4.5.6.1 *Operator Controls and Indicators.* The switches and indicators listed below are located on the control panel of the card punch.

Power On	Pressing this switch-indicator applies power to the card punch. The indicator is illuminated when the power is on.
Power Off	Pressing this switch turns off the card punch.
Not Ready	Indicator is illuminated whenever the punch is not ready. This will not occur if the following conditions exist: 1. Start switch depressed; 2. Cards in hopper; 3. Punch die in place; 4. Card stacker not full; 5. Card at ready station; 6. Covers in place; 7. No feed check error; and 8. No punch error.
Punch Check On	Depressing this latching switch enables post punch checking and illuminates the indicator. Releasing the switch inhibits the punch checking.
Punch Check	Indicator is illuminated whenever a punch error is detected.
Feed Check	Indicator is illuminated whenever a jam occurs in the card-feeding mechanism.
Start Switch	Pressing this switch moves a card to the ready station. It will reset the NOT READY indicator provided the punch was ready except for the absence of a card at the ready station.
Reset Switch	Pressing this switch clears all error indicators on the punch.

Stop	Pressing this switch stops the card feed and makes the punch not ready. If a card cycle is in progress, it will be completed.
Runout	When this switch is depressed, cards are run through the unit without punching. This switch is effective only when the punch is in a not ready condition.

4.5.6.2 *Programming.* The following instruction sequence is required to operate the card punch when the IOCS package (supplied by EAI) is not used:

1. Generate the binary card image of the data to be punched;
2. Initiate a punch cycle with the proper SFL instruction;
3. Initiate the channel and connect the punch with a channel SFL command;
4. Test busy to make sure the channel command was accepted;
5. Transfer data — the odd numbered binary cards are transferred six times, and then the even numbered characters are transferred six times;
6. Disconnect the punch; and,
7. Test error conditions using channel and device TSL instructions.

The Channel Signal Indicator (CHS) will be set, if the punch is connected to the channel and becomes ready to accept data for the next card.

The Channel Parity Indicator (CHP) will be set, if the punch is connected to the channel and the punch-check circuits detect a punch error.

A channel interrupt from the card punch will occur in the following cases:

1. When the punch is connected to the channel, and the punch becomes not ready;
2. When there is no device connected to the channel, the DINE switch on the punch is set, and the punch becomes ready to accept data for a new card.

4.5.6.3 *Card Punch Operation.* The general sequence of operation for punching a card consists of the four cycles listed below.

Feed Cycle	When the power is on and the start switch has been depressed, a card will move to the ready station.
Punch Cycle	A card is moved from the ready station under the line of 80 punches. Cards are fed sideways, and information is punched one row at a time starting at the top of the card.
Check Cycle	During this cycle, the card is fed through the post punch check brushes and checked for errors. Checking is performed by reading the punched row, comparing this row of information with that stored in the punch buffer. The comparison is accomplished by a hold count technique.
Stack Cycle	The card is placed on the output stack.

The cards follow one another through each cycle. The action is continuous; when one card is being punched one is being picked from the hopper, one is being read and another is being stacked.

4.5.6.4 *Program Controls and Indicators.* The controls and indicators that are accessible to a program by SFL instructions are listed below.

Start Card Punch Cycle	Setting this control starts a card toward the punch station without connecting the Card Punch Controller to the Data Channel.
Reject Card	When set, this control directs the card being punched into the auxiliary stacker or offsets it by 1/2" in the output hopper.
Eject Card	The card being punched is ejected to the output hopper. The next card is brought to the Registration Station and waits for the next START CARD PUNCH CYCLE command.
Set DINE (Device Interrupt Enable)	Setting this control enables a channel interrupt whenever the punch is ready and the punch cycle indicator is low. The switch can be set only when a device isn't currently connected to the channel. It becomes reset automatically whenever a device is connected to the channel or a channel clear is executed.
Start Card Punch Cycle and Connect Punch to Data Channel	Setting this control initiates a card punch cycle and connects the card punch controller to the Data Channel.

The indicators accessible to the program are listed below.

Test "Punch Ready" Status	Indicator is set when the cards are in the hopper, the die is in place, the card is in a position to be punched, the stacker is not full, the power is on, and there are no jams or punch errors.
Test "Ready to Punch Next Row" Status	Indicator is set when the unit is ready to punch the next row on the card.
Test "Punch Cycle" Status	Indicator is set when the card is in the punch cycle and remains set during the punching of the card.
Test "Punch Error" Status	Indicator is set when an invalid Hollerith character is punched.

4.5.7 Magnetic Tape Systems (Models 8472, 8474, 8476 and 8478)

System Type	8472	8474	8476	8478
Controller Operating Speed	45 IPS	75 IPS	120 IPS	150 IPS
Transport Type*	8473	8475	8477	8479
Recording Width	←——7 or 9-track——→			
Density	←——556 and 800 BPI——→			
*Standard system includes one tape transport.				

The SFL Instruction codes for these commands are shown below.

	C	K	D	L
Start Card Punch Cycle	0	K	D	X X X X X X X 1
Reject Card	0	K	D	X X X X X X 1 X
Eject Card	0	K	D	X X X X 1 X X X
Set DINE	0	K	D	1 X X X X X X X
Start Card Punch Cycle and Connect Punch To Data Channel	1	K	D	X X X X X X X X
	0	1 3	4 7	8 15

C = 0 Used to set channel/device control conditions, SFL (C)
1 Initialize channel/connect device, clear channel, SFL (F)
K Data channel number, 0 to 7
D Device number, 1 to 15

Figure 4-22. Card Punch SFL Instructions

The TSL Instruction codes for the above commands are shown below.

	C	K	D	L
Test "Punch Ready" Status	C	K	D	X X X X X X X 1
Test "Punch Cycle" Status	C	K	D	X X X X X X 1 X
Test "Ready To Punch Next Row" Status	C	K	D	X X X 1 X X X X
Test "Punch Error" Status	C	K	D	X X X X 1 X X X
	0	1 3	4 7	8 15

C = 0 Test status level only, TSLC
1 Test status and then reset status flip-flop, TSLF
F Data channel number, 0 to 7
D Device number, 1 to 15

Figure 4-23. Card Punch TSL Instructions

4.5.7.1 *Description.* Four types of magnetic tape systems are available, Types 8472, 8474, 8476 and 8478. Each system is provided with a single tape transport, a controller that can handle up to four transports, and protection facilities.

Transports are available for either of two tape recording widths, 7-track (compatible with IBM 729) and 9-track (compatible with IBM 360). If desired, the standard, 7-track transports may be updated in-the-field to provide them with the 9-track recording capability. The standard, 7-track unit provides six channels for data and one for parity, with both *binary* and *BCD* modes available. The optional, 9-track unit provides eight channels for data and one for parity with all data transfers in the odd-parity, *binary* mode.

Controllers for the four systems are of two types to handle the two tape recording widths. The controller used with 9-track tapes can also handle 7-track tapes. The 9-track tapes are handled by the 9-track system controller alone. Each controller can accommodate up to four tape transports, controlling one at a time except during a rewind operation. Having started the rewinding of one tape transport, the controller is immediately available to handle the control of another. This control is implemented using the following control facilities:

1. Power On/Off — switch and indicator
2. Remote — switch and indicator
3. Local — switch and indicator
4. Reset — switch
5. Forward — switch
6. Reverse — switch
7. Rewind — switch
8. Medium/High/Low Density — switch and indicator
9. File Protect — indicator
10. Unit Address — switch and indicator

4.5.7.2 *Operation and Format.* The recording of information in the *binary* mode involves the transfer of odd parity data, without code conversion, to and from the magnetic tapes. In the *BCD* mode, there is an output data conversion from the 8400 computer's internal code to the IBM compatible 5 code, and the data is transferred with even parity. Input data in the even-parity, IBM compatible 5 code, is converted to data expressed in the 8400's internal code.

Records on tape are separated by an end-of-record (EOR) gap (0.75 inch of blank tape). A record ends with three consecutive blank characters followed by a longitudinal parity character (LPC). These characters together constitute the end-of-record (EOR) mark. Files on tape are separated by an end-of-file (EOF) gap (3.75 inches of blank characters on tape) followed by an end-of-file (EOF) mark. The EOF

mark consists of a single character record with the LPC and EOR gap.

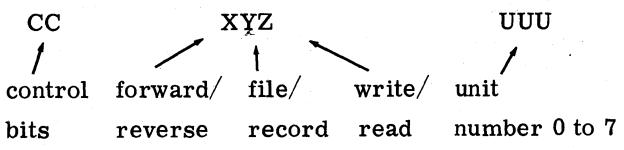
A reflective marker on one side of the tape located about ten feet from the starting end denotes beginning-of-tape (BOT). When this mark is encountered, the tape stops, and the BOT indicator is set to prevent tape reversal. The next forward command will reset this indicator. When a write command is given with the tape in the BOT position, a 3.75 inch gap in the tape results before writing begins.

A second reflective marker located on the other side of the tape about twelve feet from the other end denotes end-of-tape (EOT). When this mark is encountered, the EOT indicator is set but operations in the forward direction are not inhibited. The indicator will be reset when the first rewind command is given.

Parity checking is included with each data transfer operation. If either lateral or longitudinal parity failure is detected after writing, the error flag is set. The tape can then be back-spaced for a rewrite operation. This operation can be repeated up to ten times if necessary.

Information in a tape file can be protected in two ways. The operator can arrange the file security by placing on a tape reel a write enable ring to prevent writing on that reel. Another way involves the use of monitor-controlled flags in the unit control block. These flags can be used to prohibit placing the tape in motion and to prevent reading and writing operations.

4.5.7.3 *Instructions.* Each tape controller has an 8-bit control register whose configuration is described as,



This control register is loaded by one of the 8400's Set Function Line (SFL) instructions. The contents of the control register are placed in the 8-bit, L-Field of the SFL instruction address. The format of this instruction address appears below with a list of SFL instructions.

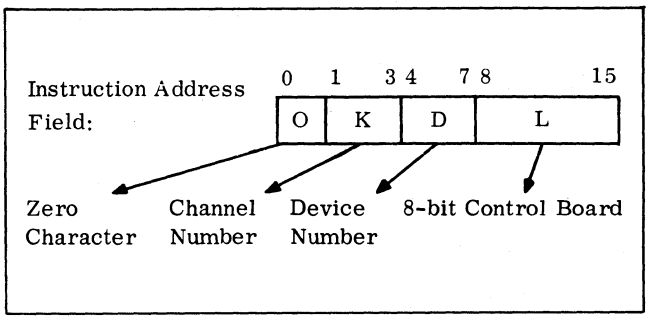


Figure 4-24. Magnetic Tape SFL Instruction

Table 4-4. SFL Instruction List

No.	Function	L-Field	Remarks
1.	Read record(s) forward	01 000 0-7	Move tape forward to read data. If channel initialized, transfer data until channel disconnected, otherwise skip one record.
2.	Read record(s) reverse (optional)	01 100 0-7	Same as 1 but in reverse direction.
3.	Write record forward only	01 001 0-7	Move tape forward to write data. If channel initialized, write data until data no longer presented. Then write LPC and interrecord gap. If channel not initialized, write .4 inch blank tape and stop.

Table 4-4. SFL Instruction List (Cont)

No.	Function	L-Field	Remarks
4.	Write blank tape forward only	01 011 0-7	Write 3-3/4 inch blank (all zero) tape.
5.	Search EOF forward	01 010 0-7	Move tape forward until a file mark is detected.
6.	Search EOF reverse	01 110 0-7	Same as 5 but in reverse direction.
7.	Write EOF	11 011 0-7	Write 3-3/4 inch of blank tape, and an EOF mark (octal 17) with its LPC.
8.	Rewind to load point	00 110 0-7	Rewind tape until the BOT mark is detected.
9.	Rewind and unlock	00 111 0-7	Same as 8, but tape switched to local mode when rewind is complete.
10.	Set unit field	00 000 0-7	Set up the unit field of control register but do not initiate any tape operation.
11.	Device interrupt enable	10XXX XXX	Set the device interrupt enable (DINE) switch if no device is connected to the channel.

SFL instructions (except 10 and 11) are rejected and the B busy monitor-controlled flag is returned if any of the following conditions exist:

1. the selected unit is not ready
2. the select unit is rewinding
3. any tape unit is in motion
4. The SFL instruction to set the unit field (number 10) is rejected only if the controller is not able to accept a new command due to some tape unit being in motion. This instruction is useful for determining if the controller is available, and for preceding certain TSL instructions which have no unit field.
5. The SFL instruction to set the DINE switch (number 11) is rejected only if a device is connected to the channel.

The indicators used in each tape system to signify various conditions include the following: EOR - end of record; EOF - end of file; ERR - error; BOT - beginning of tape; and EOT - end of tape. These indicators and other signals in the system can be tested by the 8400 computer's Test Status Line (TSL) instructions. The format of the instruction address field appears below with a list of TSL instructions.

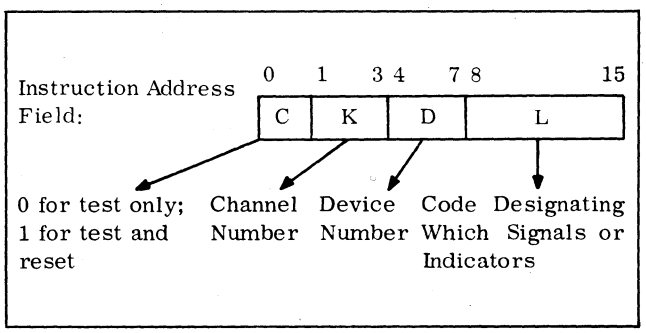


Figure 4-25. Magnetic Tape TSL Instruction

Table 4-5. TSL Instruction List

No.	Function	L-Field	Remarks
1.	Test and clear EOR	10XXXXX1	Test end of record for the selected tape, and then reset the indicator.
2.	Test and clear EOF	10XXXX1X	Test end of file for the selected tape, and then reset the indicator.
3.	Test and clear ERR	10XXX1XX	Test and clear the error indicator for the selected tape.
4.	Test tape unit ready	10XX1XXX	Test if selected unit is ready: power on, reel in place, automatic mode, etc.
5.	Test BOT	10X1XXXX	Test if selected unit is at beginning of tape.
6.	Test EOT	101XXXXX	Test if selected unit has detected the end of tape mark.
7.	Test automatic mode	11XXXXX1	Test if selected unit is in automatic (not local) mode.
8.	Test file protect	11XXXX1X	Test if selected unit is loaded with a reel with a file-protect ring.
9.	Test high density	11XX1XXX	Test if selected unit is set for high density (800 character per inch).
10.	Test medium density	11X1XXXX	Test if selected unit is set for medium density (556) character per inch.
11.	Test tape in motion	111XXXXX	Test if selected unit has its tape in motion.
12.	Test if rewinding	01XXX 0-7	Test if the unit addressed is in the rewinding state.
13.	Test 7-track mode	11XXX1XX	Test if selected unit is set for 7-track (not 9-track) mode.

In response to all TSL instruction, the Z (end) monitor-controlled flag is set if the specified signal or indicator is high, and the Z flag is reset if the specified signal or indicator is low. Note that most TSL instructions do not have a unit field in the L code. This means that if the controller is in use with one of the units, the TSL's refer only to that unit. If the controller is idle, an SFL instruction can be used to set the unit field, permitting TSL instructions to refer to any of the tapes as needed.

CHAPTER 5

COMPUTER CONSOLE OPERATIONS

5.1 INTRODUCTION

The system console, shown in Figure 5.1, provides complete and efficient organization of all display and control elements for the 8400. All operating controls are within arms reach of a seated operator. The control buttons are combination indicators and push-buttons for ease of observation of system status (Figure 5.2).

A system display panel and console typewriter to the immediate left of the operating controls is shown in Figure 5.3. The system display panel accommodates 7 registers and one counter. The operator may choose alternative registers using controls to the immediate right of the display panel. The typewriter is a standard electric model with the 8400 character set.

The optional paper tape reader is housed to the right of the operating controls immediately above the

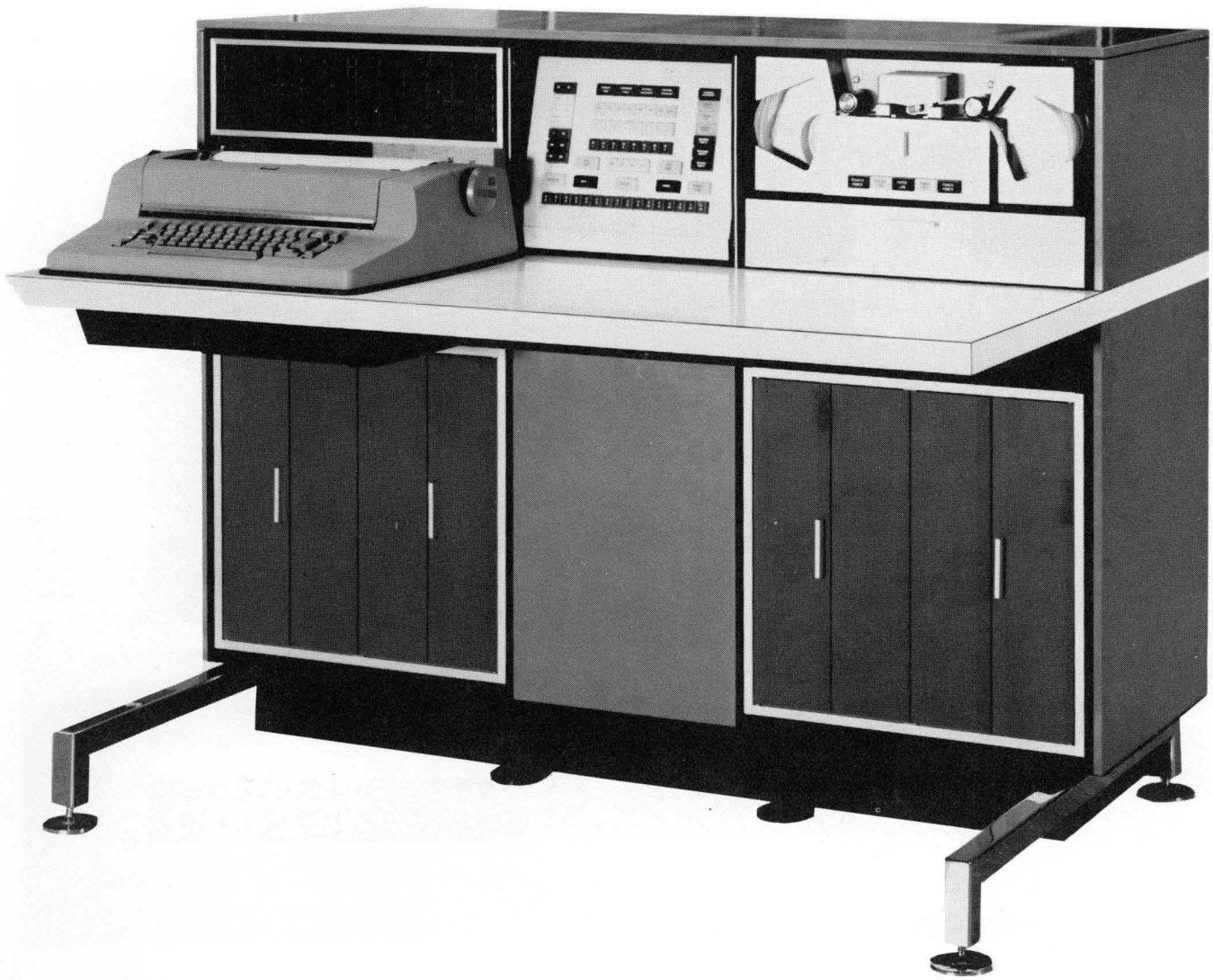


Figure 5.1 . Control Console

system maintenance panel, as shown in Figure 5.4. The paper tape reader is a 500 character per second photoelectric device. The 110 character per second paper tape punch is also housed in the system console.

5.2 CONTROLS AND INDICATORS

Figure 5.5, depicting the system control panel, is overlayed with a circled number key to the corresponding description in this section.

5.2.1 Register Controls

AF | AE ① These two pushbuttons determine which data is displayed on the ACCUMULATOR portion of display panel. Depressing the AF button causes bits 0 through 15 of A, and the AF portions of the Accumulator, to be displayed (32 bits). In this

A	AF

mode the display is used for monitoring floating-point operations. Depressing the AE button causes bits 0 through 15 of A, and the AE portions of the Accumulator to be displayed (32 bits).

A	AE

This configuration is used for monitoring extended fixed-point operations.

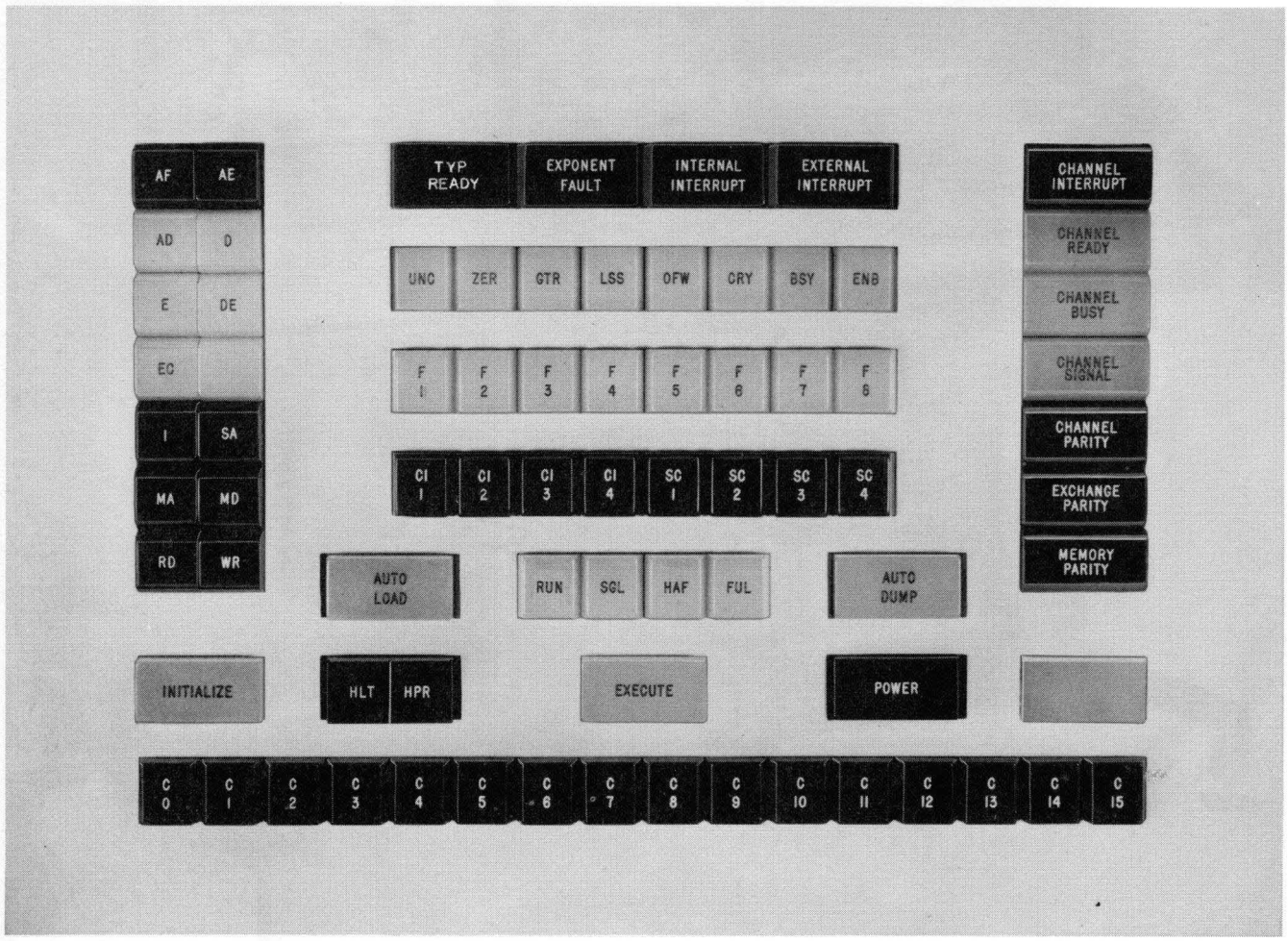


Figure 5.2. Control Panel

AD	D
E	DE
EC	

(2) These five interlocking pushbutton indicators determine which data is displayed in the DISPLAY REGISTER portion of the Display Panel. Only the first is of real interest to a programmer.

AD - The AD portion (bits 0 through 23), of the Accumulator is displayed together with the A and AF registers to show a 56 bit floating point operand.

DE - displays bits 0 through 15 of the D Register in the left half of the DISPLAY REGISTER and bits 0 through 15 of the E Register in the right half.

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.

EC - displays bits 0 through 31 of the Exchange Control Register. The Exchange Control Register controls the sequence of operations within the Exchange Module when using the Automatic Data Channel Processor.

5.2.2 Typewriter Input Controls

I	SA
MA	MD
RD	WR

(3) These six interlocking pushbutton indicators determine where typewritten inputs are routed within the computer. These controls are needed in a manual system for loading and may be

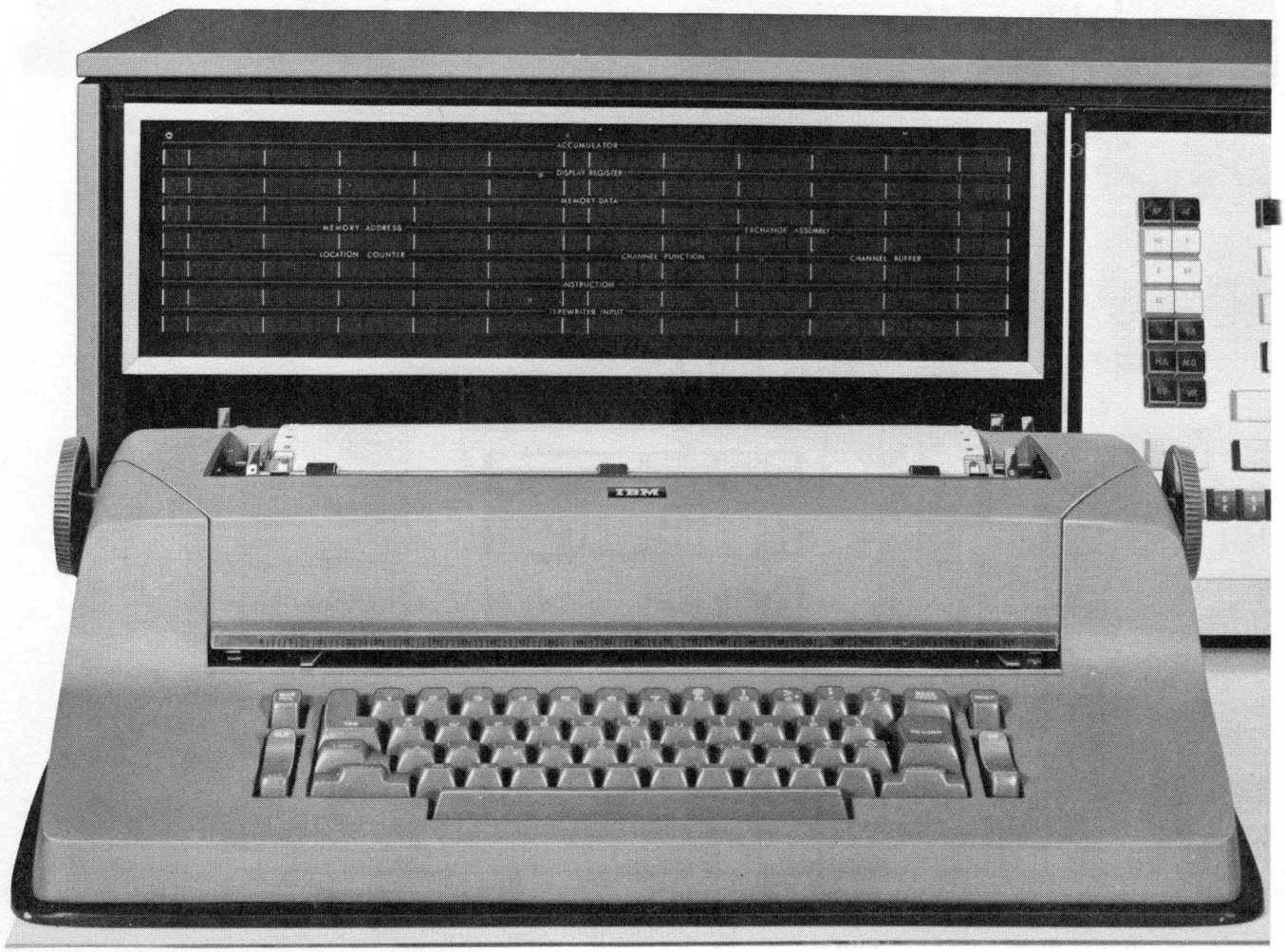


Figure 5.3. Register Display/Input-Output Typewriter

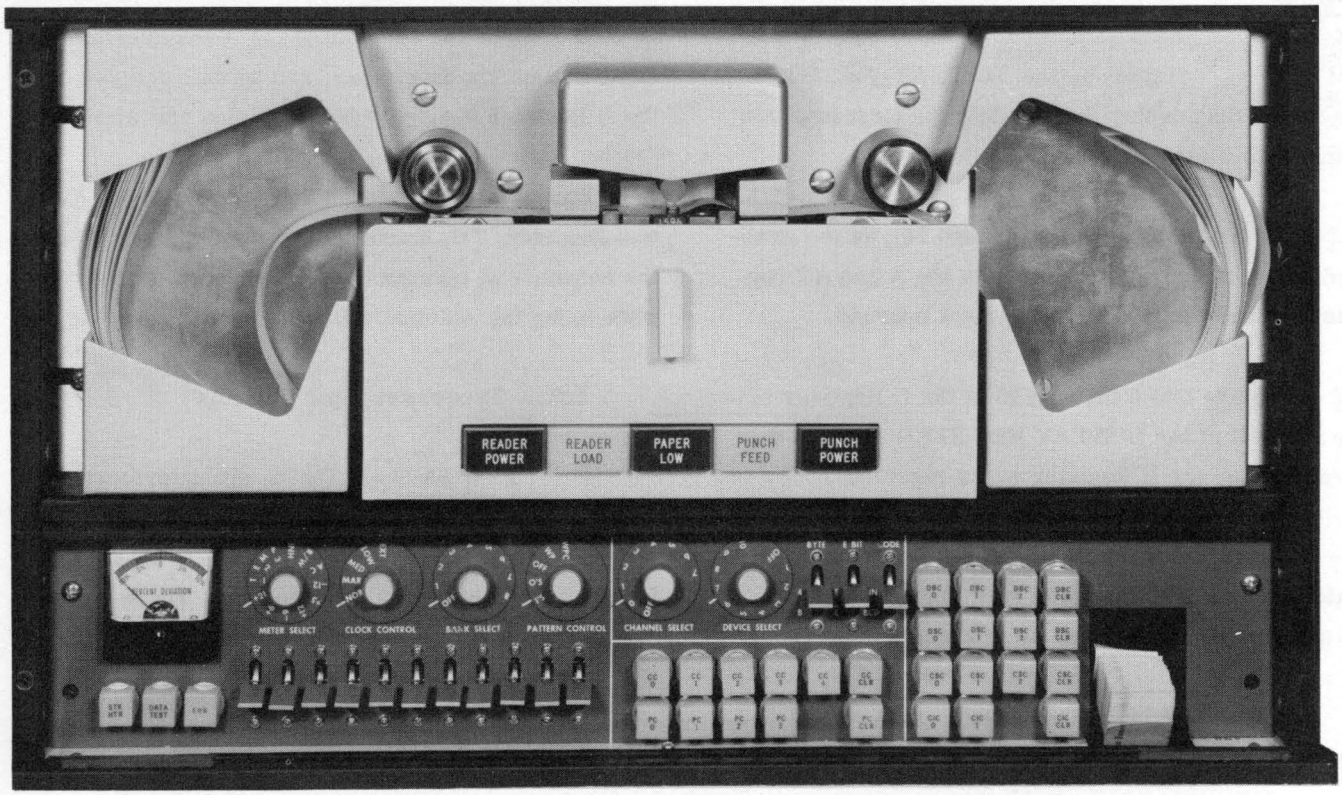


Figure 5.4. Paper Tape Reader and Maintenance Panel

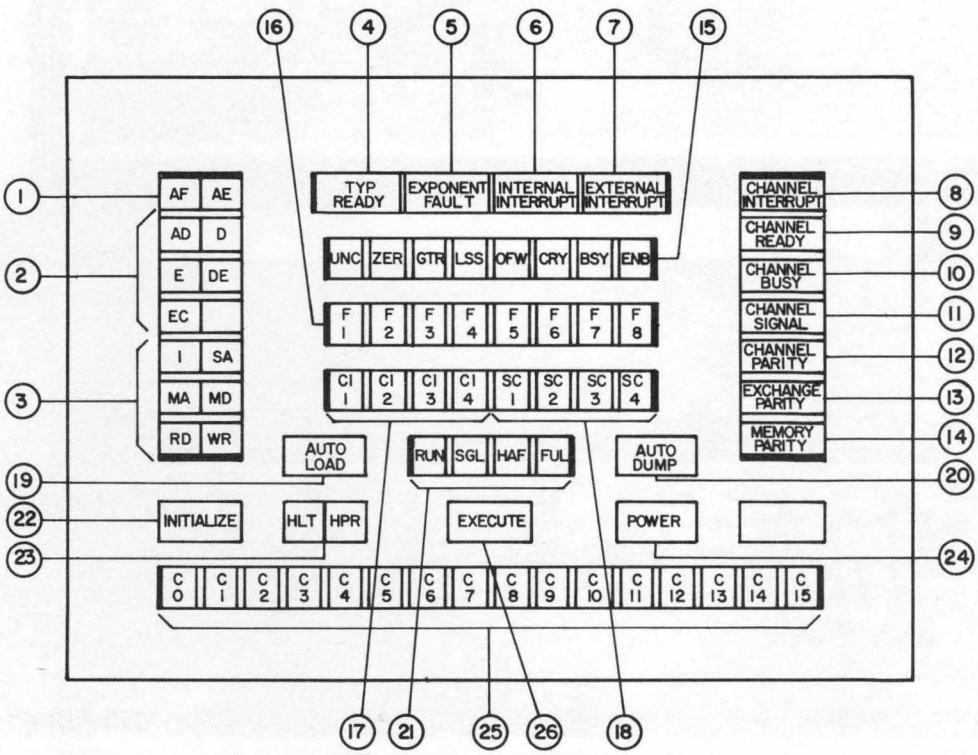


Figure 5.5. System Control Panel

used for memory readout and alteration. Data typed on the console typewriter are stored in the typewriter input register (W).

I (Instruction) - The W Register contents (bits 0-31) are transferred to the Instruction Register when the CR key is depressed.

SA (Starting Address) - The W Register contents (bits 0-15) are transferred to the Location Counter and the m field of the Instruction Register when the CR key is depressed.

MA (Memory Address) - The W Register contents (bits 0-15) are transferred to the m field of the Instruction Register when the CR key is depressed.

RD (Read from Memory) - Information is read from memory using the contents of MA as the location, and is displayed on the Memory Data Register (bits 0-35).

MD (Memory Data) - Typewriter information is transferred from the W Register to the Memory Data Register (bits 0-33) when the CR key is depressed.

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

TYP RDY (Typewriter Ready) (4)

This indicator when illuminated indicates that the typewriter has been selected as an input device.

5.2.3 Exponent Fault (5)

This indicator when illuminated indicates an exponent overflow or underflow within the Arithmetic Module. It remains on until reset by the program.

5.2.4 Interrupt Indicators

INTERNAL INTERRUPT (6)

This indicator, when illuminated, indicates an unserviced internal interrupt. (Chapter 3.)

EXTERNAL INTERRUPT (7)

This indicator, when illuminated, indicates an unserviced external interrupt.

CHANNEL INTERRUPT (8)

When lit this indicator 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.

5.2.5 Channel Condition Indicators

These three indicators indicate the 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 located on the Maintenance Panel. 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 BUSY (10)

This indicator when illuminated indicates that a data channel has been initialized and a device has been selected for data transfer.

CHANNEL SIGNAL (11)

This indicator when illuminated indicates that a peripheral device has sent a status signal, (e.g., gap on magnetic tape; STOP code on paper tape reader, low paper on paper tape punch, etc.) to the Exchange Module.

5.2.6 Parity Indicators

CHANNEL PARITY (12)

This indicator, when illuminated, indicates that an error has occurred during transfer from or to the peripheral selected device.

EXCHANGE PARITY (13)

This indicator, when illuminated, indicates than an error has occurred within the Exchange Module, which implies a data channel device error.

MEMORY PARITY (14)

This indicator when illuminated indicates that a memory parity error has occurred.

5.2.7 System Flag Indicators

These eight indicators display machine conditions occur during the course of a program. (See Chapter 1.)

UNC (Unconditional) - Illuminated when the rounding flip-flop of the Accumulator is in the 1 state.

ZERO (Z Flag) - Illuminated when the contents of the Accumulator are zero, as a true result of TSL'S and Boolean Connect Instruction, and when the results of a Compare are equal.

GTR (G Flag) - Illuminated when the contents of the Accumulator is greater than zero, or as the result of a Compare instruction.

LSS (L Flag) - Illuminated when the contents of the Accumulator is less than zero, or as the result of a Compare instruction.

OFW (V Flag) - Illuminated when there is an overflow condition in the Accumulator.

CRY (C Flag) - Illuminated when there has been a carry out of the most significant bit of the Accumulator (A1, not A0).

BSY (B Flag) - Illuminated when an addressed function line or data channel is busy.

ENB (E Flag) - Illuminated when the interrupt system is enabled.

5.2.8 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 logic ONE.

5.2.9 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 pushbutton is depressed a second time. Paragraph 3.4 describes the console interrupts in detail.

5.2.10 Configuration Switches

SC1 *Through* SC4 (18) - These four pushbutton indicators are control switches for special 8400 configurations. The outputs of these switches are accessible through the System Interface.

5.2.11 AUTO LOAD and AUTO DUMP

AUTO LOAD (19) and AUTO DUMP (20) - These indicator pushbuttons are used to load information

from a single peripheral device into memory, or dump onto a single peripheral device during manual operations. Each is illuminated during the operation.

5.2.12 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 (Single-Step) - 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 (Half) - 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.

5.2.13 System Controls and Indicators

INITIALIZE (22) - This indicator pushbutton is used to put the machine into an initial state. All data channels are cleared, all interrupts are reset, all flags are reset, and the machine is placed in a halt condition. Memory is not cleared.

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

HLT - This indicator pushbutton is used to 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 Jump 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.

EXECUTE (26) - This pushbutton indicator is used to start execution of a program. The location of the first instruction to be executed is specified by the contents of the location counter.

5.2.14 Console Register

C0 *Through* C15 (25) - The pushbutton indicators represent bits 0 through 15 of the Console Register. The individual bits may be set and reset manually or by programming. Depressing a pushbutton complements the bit.

5.3 CONSOLE DISPLAY

The system display panel immediately above the console typewriter is described below. (Refer to Figure 5.6.)

5.3.1 Accumulator (1)

This area displays bits 0 through 15 of the accumulator (A segment) and bits 0 through 15 of either the AE or AF segment dependent on which button on the control panel is depressed.

5.3.2 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 AD segment of the Accumulator.

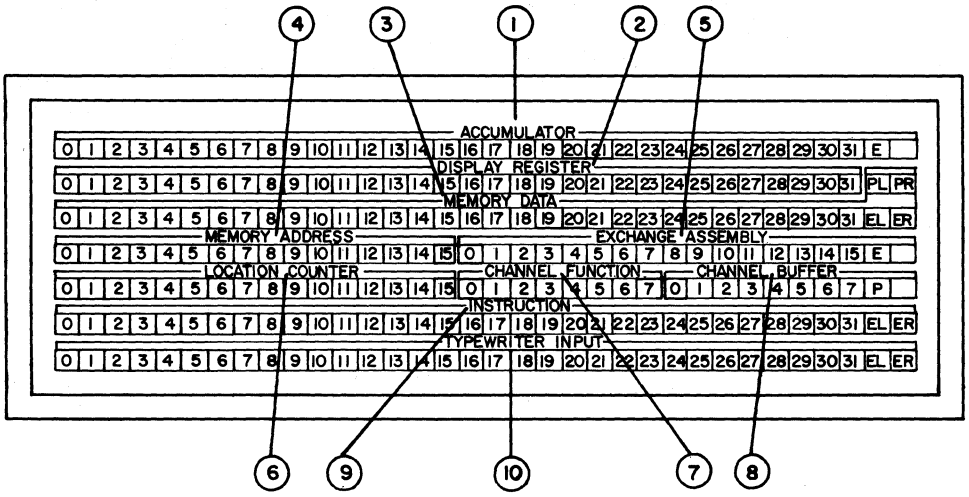


Figure 5.6. System Display Panel

Pushbutton	Data Display
D	Bits 0 through 31 of the D Register in the Arithmetic Module.
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.

5.3.3 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 the BANK SELECT switch is in AUTO position unless memory is being requested by control.

5.3.4 Memory Address (4)

This area displays the address of the memory location being accessed by the Control Module.

5.3.5 Exchange Assembly (5)

This area displays the contents of the Exchange Assembly Register in the Exchange Module.

5.3.6 Location Counter (6)

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

5.3.7 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, the bits are interpreted 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 bit does not accompany data.

Indicator	Function
Bit 1	This indicator when illuminated specifies that data is being transferred in binary code without code conversion. When extinguished data is in BCD form, using the hardware code conversion in the data channel.
Bit 2	The indicator specifies which half of the memory word is being addressed.
Bit 3	When illuminated this indicator specifies that alternate left and right half words are being accessed in memory during data transfer between memory and the Exchange Module.
Bit 4	This indicator specifies the direction of data transmission. When illuminated it indicates transmission to memory from the Exchange Module.
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

5.3.8 Channel Buffer (8)

This area displays the 8 bits of the Channel Buffer Register which is located in the Exchange Module.

5.3.9 Instruction (9)

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

5.3.10 Typewriter Input (10)

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

5.4 Maintenance Panel

The Maintenance Panel contains controls and indicators which are used for both test and normal operating purposes. The circled numbers in the following descriptions are keyed to Figure 5.7.

5.4.1 Lamp Test

ON/OFF (1) - 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. This switch is depressed only momentarily.

5.4.2 Keyboard

UNLCK/LCK (2) - This two position toggle switch controls the keyboard lock on the typewriter. In the "UNLCK" position the typewriter keyboard is unconditionally unlocked. In the "LCK" position the keyboard is under program control and cannot be used unless unlocked by the program.

5.4.3 Clock Control

NOR/MAR/MED/LOW/EXT (3) - This 5 position rotary switch determines the clock frequency of the system.

NOR - Normal clock frequency.

MAR - Higher marginal clock frequency used for system testing and maintenance.

MED - Medium clock frequency of approximately 1.2 kilocycles.

LOW - Low clock frequency approximately of 1.5 cycles per second.

EXT - System clock supplied by an external device.

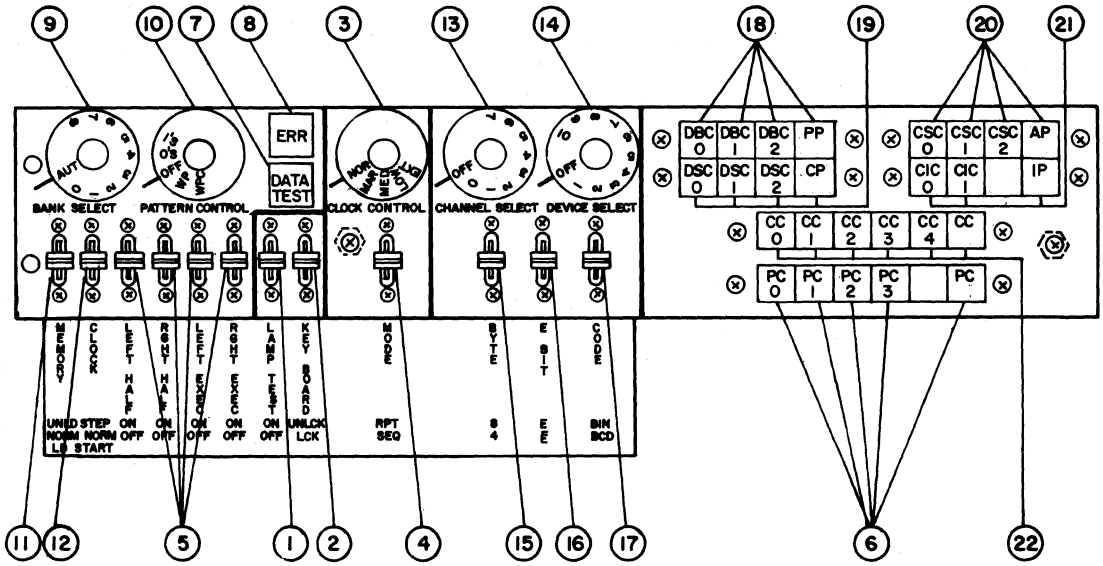


Figure 5.7. Maintenance Control Panel

5.4.4 Mode

SEQ/RPT (4) - 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 defines the location of the next instruction in a normal manner. 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.

5.4.5 Left Half, Right Half, Left Exec, and Right Exec (5)

These four, two position toggle switches determine the memory word format during manual data entry from the typewriter. 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.

5.4.6 PC0, PC1, PC2, and PC3 (6)

These four indicators show the state of the Control Module Phase Counter flip-flops. The binary weighing of each stage is as follows:

PC0 - 8
PC1 - 4
PC2 - 2
PC3 - 1

5.4.7 Data Test (7)

When generating a memory test pattern, the DATA TEST indicator is lit when "1"s are being loaded into, or unloaded from, the addressed memory location.

5.4.8 ERR (Error) (8)

This indicator is illuminated if a memory error is detected during unloading of the memory test pattern.

5.4.9 Bank Select (9)

Each memory bank has a separate Memory Data Register and Memory Address Register. The position of the BANK SELECT switch determines which bank is displayed on the System Display Panel MEMORY DATA and MEMORY ADDRESS indicators. The AUT (automatic) position is used to display the address and data specified by the normal program.

5.4.10 Pattern Control (10)

The PATTERN CONTROL switch is a five position rotary switch used to select the memory test pattern during memory self test. The patterns are as follows:

1's	All ones are written into each location of the memory bank under self test.
0's	All zeros are written into each location of the memory bank under self test.
OFF	The memory self test function is disabled.
WP	Logic ONES and ZEROS are alternately written into the memory bank, creating a checkerboard pattern.
WPC	The complement of WP is written into the memory bank.

5.4.11 Memory - LD/NORM/UNLD (11)

This switch is used during memory self test to load or unload the test pattern to/from memory.

5.4.12 Clock - STEP/NORM/START (12)

During the memory self test procedure this switch controls the memory clock generator. Depressing the switch momentarily to the START position starts the 0.5 megacycle memory clock. Depressing the switch momentarily to the STEP position stops the clock.

To manually increment the memory address register, the switch is moved from the NORM to the STEP position. The address is incremented by one each time the switch is depressed to the STEP position.

5.4.13 Channel Select (13)

This nine position switch performs the following functions:

1. Selects the Exchange Module Data Channel to be used during an Auto Load or Auto Dump operation.
2. Connects the Channel Function Register to the selected channel.
3. Connects the Channel Buffer Register to the selected channel.
4. Connects the following indicators on the System Control Panel to the selected Data Channel:

   CHANNEL INTERRUPT
   CHANNEL READY
   CHANNEL BUSY
   CHANNEL SIGNAL
   CHANNEL PARITY

5.4.14 Device Select (14)

This 12 position switch selects the device to be used during Auto Load or Auto Dump operations.

5.4.15 Byte 4/8 (15)

During Auto Load or Auto Dump this switch determines the byte size for data assembly or disassembly.

5.4.16 E BIT E/E (16)

The E BIT switch is a two position switch that determines whether or not an EXEC bit is associated with each data word during an Auto Load or Auto Dump operation.

5.4.17 Code-BIN/BCD (17)

This two position switch, when in the BIN position, specifies that data is transferred without code conversion and that odd parity generation and checking is performed. The BCD position specifies collating to binary code conversion and even parity generation and checking. This switch is used during an Auto Load or Auto Dump operation.

5.4.18 DBC0, DBC1, DBC2 (18)

These three indicators monitor the state of the Device Buffer Counter in the Exchange Module. The indicators are illuminated when the corresponding bit is in the "1" state. Table 5.1 illustrates the coding of these indicators.

Table 5.1. Exchange Module Counter Coding

Counter Status	DBC0 DSC0 CSC0	DBC1 DSC1 CSC1 C1C0	DBC2 DSC2 CSC2 C1C1
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0

5.4.19 DSC0, DSC1, DSC2 (19)

The Data Stack Counter in the Exchange Module is monitored by these three indicators. The indicators are illuminated when the corresponding bit position is in the "1" state. Table 5.1 illustrates the coding of these indicators.

5.4.20 CSC0, CSC1, CSC2 (20)

These three indicators monitor the status of the Control Stack Counter in the Exchange Module, if the system is equipped with an Automatic Data Channel Processor. The indicator is illuminated when the corresponding bit is in the "1" state. Table 5.1 illustrates the coding of these indicators.

5.4.21 C1C0, C1C1 (21)

The status of the Control Interface Counter in the Exchange Module is monitored by these two indicators. The indicator is illuminated when the corresponding bit is in the "1" state. Table 5.1 illustrates the coding of these indicators.

5.4.22 CC0 Through CC4 (22)

These five indicators monitor the status of the Cycle Counter in the Arithmetic Module. The Cycle Counter is used to control the sequence of operations within the Arithmetic Module. The indicators are illuminated when the corresponding bit position is in the "1" state. The indicators are coded as shown in Table 5.2.

Table 5.2 Arithmetic Module Cycle Counter Coding

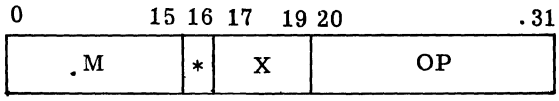
CC0	CC1	CC2	CC1	CC0
16	8	4	2	1

APPENDIX 1

WORD FORMATS

1. INSTRUCTIONS

Instructions are of the form:



INSTRUCTION

Where:

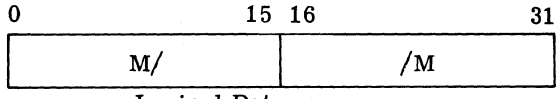
M	represents a 16 bit address
*	represents a 1 bit indirect indicator
X	represents a 3 bit indexing modifier
OP	represents a 12 bit operation code

A generalized representation of an instruction written in assembly language is shown below, where the modifiers are optional.

OP* M, X

2. LOGICAL DATA

Logical data is of the form:



Logical Data

Where:

M/	represents 16 binary bits of information - left-halfword
/M	represents 16 binary bits of information - right-halfword

Logical data is normally written using a "slash convention" such that a full word is "LEFT/RIGHT" and halfwords are "LEFT/" or "/RIGHT".

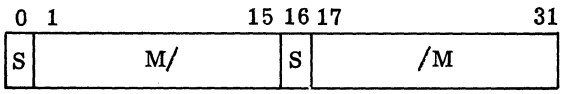
A logical instruction accesses only halfword data. It is written OP* M, X for left half and OP* /M, X for right halfword.

A logical instruction can access a left halfword in the *immediate* mode. The data is the left halfword of the instruction itself and is written OP* = M, X where the = character is used to designate this *immediate* mode.

If both indirect and immediate options are called for in an instruction, indirect chaining occurs first; the final effective address is modified and is subject to immediate interpretation.

3. FIXED POINT FRACTIONS

Fixed point fractions are defined as signed, scaled 15 bit single or paired quantities of the form:



Fixed Point Data

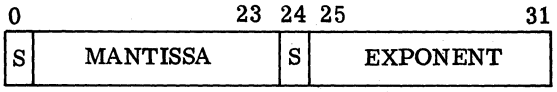
There are arithmetic instructions which access halfword data in a manner similar to logical instructions.

There are also extended arithmetic instructions which treat a 32 bit word as a single fixed point quantity. The left halfword contains the most significant 15 bits and the right halfword the least significant 15 bits of a 30 bit fraction. Sign bit 0 determines sign bit 16 in this case.

Note that in halfword arithmetic, the partial words can be treated independently, whereas in the extended operations whole words must be used.

4. FLOATING POINT NUMBERS

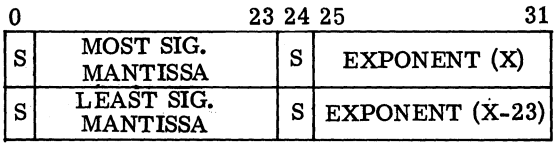
Floating point numbers are single or paired quantities of the form:



Floating-Point, One Word

Note that a single 32 bit word consists of a signed 23 bit fractional part and a signed 7 bit exponential part.

There are arithmetic instructions which also handle paired (double-precision) words of this format where the fractional part is continued in a succeeding memory word and the exponential parts differ by 2^{23} This latter form is called double precision floating point.



Floating-Point, Two Words

A floating point quantity is said to be *normalized* if absolute value of the mantissa lies between one-half and one; otherwise, it is *unnormalized.*

5. INTEGERS

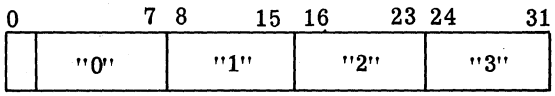
There are instructions called *integer* instructions which can handle halfwords in floating point operations. *In the integer mode, halfword fixed point signed 15 bit numbers are automatically changed into floating point format scaled as integers as part of an operation.*

Conversely, a floating point number can be automatically *integerized* and stored as halfword information. Since a 16 bit address can be the data in these integer-floating point operations, immediate addressing is very useful. Normalized and unnormalized modes are also possible.

6. ALPHANUMERIC DATA

Alphanumeric data is stored in 8-bit bytes within memory. Usually word boundaries are not useful in manipulating alphanumeric or byte size data - that is, data whose size is less than 16 bits.

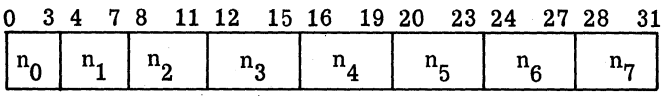
An alphanumeric word (8-bits/byte) would normally take the form:



Alphanumeric

Where each " " above is a binary *coded* character.

A binary-coded decimal word (4-bits/byte) would take the form:



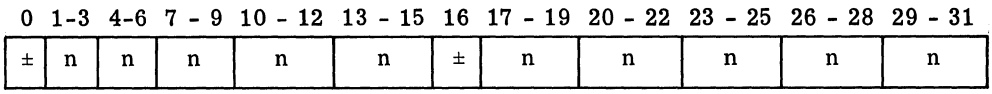
Binary-Coded Decimal

Where each n is a binary-coded decimal number.

In manipulating byte data, the logical instructions can be written to access 1, 2, 4, 8 of 16 bits at a time within the halfword. Logical operations comprise the 16 boolean connectives for two variables including *and, or not, equivalence, nor,* and *nand* among others.

7. GENERALIZED DATA

Generalized data may be thought of as being stored in *any* arbitrary form. One such format is "signed octal" which is used extensively in EAI 8400 Documentation. Each *halfword* is written as a sign (+ or -) and five octal digits (15 bits):



Octal

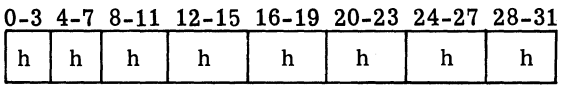
The same information is written in assembly language as

'± nnnnn/'±nnnnn

using the slash convention to separate halfwords and the *apostrophe to denote signed octal representation.*

Another arbitrary format is hexadecimal. Here, 4-bit bytes are encoded from the character set

0, 1, --- 9, A, B, C, D, E, F



Hexadecimal

Hexadecimal words may be written in assembly language

"hhhh/"hhhh

where the slash convention is used to separate halfwords and the double apostrophe is used to indicate hexadecimal representation.

A user may develop other arbitrary formats as his needs demand, using the VFD (Variable Field Data) pseudo-operation.

APPENDIX 2

Flag Test (0000-0377)

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	X			0	0	0	0	FL		OP					

OCTAL	C	CONDITION (1)
000a		UNCONDITIONALLY TRUE
001a	N	UNCONDITIONALLY FALSE
002a	Z	RESULT = 0 AFTER ARITH. OR BOOL. INSTR.
003a	NZ	RESULT ≠ 0 AFTER ARITH. OR BOOL. INSTR.
004a	C	RESULT > 0 AFTER ARITH. INSTR.
005a	NC	RESULT NOT > 0 AFTER ARITH. INSTR.
006a	L	RESULT < 0 AFTER ARITH. INSTR.
007a	NL	RESULT NOT < 0 AFTER ARITH. INSTR.
010a	V	OVERFLOW SINCE FLAG RESET
011a	NV	NO OVERFLOW SINCE FLAG RESET
0112	C	CARRY IN LAST ARITH. INSTR.
013a	NC	NO CARRY IN LAST ARITH. INSTR.
014a	B	LAST I-O INSTR. NOT EXECUTED (BUSY)
015a	NB	LAST I-O INSTR. WAS EXECUTED
016a	E	INTERRUPT SYSTEM ENABLED
017a	NE	INTERRUPT SYSTEM NOT ENABLED
020a	1	FLAG 1 TRUE
021a	N1	FLAG 1 FALSE
022a	2	FLAG 2 TRUE
023a	N2	FLAG 2 FALSE
024a	3	FLAG 3 TRUE
025a	N3	FLAG 3 FALSE
026a	4	FLAG 4 TRUE
027a	N4	FLAG 4 FALSE
030a	5	FLAG 5 TRUE
031a	N5	FLAG 5 FALSE
032a	6	FLAG 6 TRUE
033a	N6	FLAG 6 FALSE
034a	7	FLAG 7 TRUE
035a	N7	FLAG 7 FALSE
036a	8	FLAG 8 TRUE
037a	N8	FLAG 8 FALSE

TRUE = SET
FALSE = RESET
TRIGGER = COMPLEMENT

a	MNEMONIC FLAGS	OPERATION
0	HJc m,X	HALT AND JUMP TO EFFECTIVE ADDRESS (EA)
1	EXc m,X	EXECUTE THE INSTR AT EA
2	Lc m,X	LINK TO THE SUBR. AT EA
3	LRc m,X	LINK TO EA AND RESET FLAG
4	JTc m,X	JUMP TO EA AND TRIGGER FLAG (2)
5	JSc m,X	JUMP TO EA AND SET FLAG
6	JRc m,X	JUMP TO EA AND RESET FLAG
7	Jc m,X	JUMP TO EA

(1) HALT, EXECUTE, LINKS AND JUMPS ARE CONDITIONAL UPON THE STATE OF THE FLAG TESTED WHEREAS SET, RESET AND TRIGGER ARE NOT CONDITIONAL

(2) PERMITS LOW PRIORITY INTERRUPTS AFTER A HIGHER PRIORITY INTERRUPT HAS OCCURRED.

Input-Output (0400-0577)

All in this group are privileged instructions

OCTAL	MNEMONIC	FLAGS	INSTRUCTION
0420 + R	LDOB = m,,R		LOAD OUTPUT BUSS R WITH E.A. $0 \leq R \leq 17_8$
0440 + R	STIB m,,R		STORE (INPUT BUSS R) AT E.A.
0460 + R	LDOB m,,R		LOAD OUTPUT BUSS R WITH (E.A.)
0500 + R	STIB /m,,R		STORE (INPUT BUSS R) AT /E.A.
0520 + R	LDOB /m,,R		LOAD OUTPUT BUSS R WITH (/E.A.)
054k	STCC m,,k		STORE CHANNEL k's CONTROL WORD AT E.A. $0 \leq k_8 \leq 7$
055k	STCD m,,k		STORE CHANNEL k's DATA WORD AT E.A.(1)
056k	LDCC m,,k		LOAD CHANNEL k's CONTROL WORD WITH (E.A.)
057k	LDCD m,,k		LOAD CHANNEL k's DATA WORD WITH (E.A.)(1)

Registers, TSL's and SFL's-Exec Bits (0600-0777)

SFL, TSL, LOT, LDM, LDE and LDC are privileged instructions

IMMEDIATE

0607	REX = m,X		RESET L.H. EXEC. BIT AT INSTRUCTION ADDRESS
0617	TEX = m,X	Z	TEST L.H. EXEC. BIT AT INSTRUCTION ADDRESS
062k	LDs = m,X		LOAD REGISTER s WITH E.A.
0627	SEX = m,X		SET L.H. EXEC. BIT AT INSTRUCTION ADDRESS
0630 + b	TSL = m,,b	Z	TEST SENSE LINE(S) IN BASE o SPECIFIED BY E.A. $0 \leq b_8 \leq 3$
0634 + b	SFL = m,,b	B	SET FUNCTION LINE(S) IN BANK b SPECIFIED BY E.A.

LEFT HALF

064s	STs m,X		STORE (REGISTER s) AT E.A.
0647	REX m,X		RESET L.H. EXEC. BIT AT E.A.
0657	TEX m,X	Z	TEST L.H. EXEC. BIT AT E.A.
066s	LDs m,X		LOAD REGISTER s WITH (E.A.)
0667	SEX m,X		SET L.H. EXEC. BIT AT E.A.

RIGHT HALF

070s	STs /m,X		STORE (REGISTER s) AT /E.A.
0707	REX /m,X		RESET R.H. EXEC. BIT AT E.A.
0717	TEX /m,X	Z	TEST R.H. EXEC. BIT AT E.A.
072s	LDs /m,X		LOAD REGISTER s WITH (/E.A.)
0727	SEX /m,X		SET R.H. EXEC. BIT AT E. A.

(α) IS READ AS "THE CONTENTS OF α"

SPECIAL REGISTER S

0	AE	EXTENDED FIXED POINT ACCUMULATION
1	F	FLAG REGISTER
2	L	LOCATION COUNTER
3	T	TIMER REGISTER
4	M	INTERNAL INTERRUPT MASK REGISTER
5	E	EXTERNAL INTERRUPT MASK REGISTER
6	C	CONSOLE REGISTER

(1) IN STCD AND LDCD THE FORMAT OF STORAGE DEPENDS ON THE CHANNEL FUNCTION REGISTER WHICH IS SET UP BY SPL TO DEVICE O ON THE CHANNEL.

Index Jump Test (1000-1777)

16	17 18 19	20	21	22	23	24	25 26 27 28 29 30 31
*	X	0	0	1	T	±	C

OCTAL	MNEMONIC	FLAGS	OPERATIONS
1000	XJ m, X ± C	ZGL	INCREMENT INDEX X BY C AND JUMP TO M
1400	XIT m, X ± C	ZGL	INCREMENT INDEX X BY C AND JUMP TO M IF THE RESULT IN X HAS OPPOSITE SIGN OF C. $-128 \leq C \leq 127$

Arithmetic (2000-3777)

16	17 18 19	20	21	22	23 24 25	26	27	28	29 30 31
*	X	0	1		Mode		$		OP

2000 + C	NORMALIZED 32 BIT FLOATING POINT (FL PT)
2100 + C	NORMALIZED 56 BIT FL PT
2200 + C	UNNORMALIZED 32 BIT FL PT
2300 + C	UNNORMALIZED 56 BIT FL PT
2400 + C	INTEGER EXECUTED AS FL PT IMMEDIATE
2400 + C	INTEGER EXECUTED AS FL PT LEFT HALF
2440 + C	INTEGER EXECUTED AS FL PT RIGHT HALF
2600 + C	UNNORMALIZED INT EXECUTED AS FL PT IMM.
2640 + C	UNNORMALIZED INT EXECUTED AS FL PT L.H.
2700 + C	UNNORMALIZED INT EXECUTED AS FL PT R.H.
3400 + C	INDEX ARITHMETIC IMMEDIATE
3440 + C	INDEX ARITHMETIC LEFT HALF
3500 + C	INDEX ARITHMETIC RIGHT HALF
3540 + C	EXTENDED FIXED POINT
3600 + C	FIXED POINT IMMEDIATE
3640 + C	FIXED POINT LEFT HALF
3700 + C	FIXED POINT RIGHT HALF

C	
00	SB
01	CS
02	CA
03	AD
04	CP
05	CP
06	ST
07	SR
10	DV
11	CD
20	$SB
21	$CS
22	$CA
23	$AD
24	$CP
25	$MP
26	$ST
27	$SR
30	$DV
31	SCD

Shifts (3000-3377)

N - NORMALIZED
E - EXTENDED
L - LOGICAL

16	17 18 19	20	21	22	23	24	25	26	27	28 29 30 31
*	X	0	1	1	0	N		L	$	- UNUSED

3000	ASH ± m,	X	ZGLV	ARITHMETIC SHIFT
3020	$ASH ± m,	X	ZGLV	
3040	ROT ± m,	X		LOGICAL ROTATE
3060	$ROT ± m,	X		
3100	EASH ± m,	X	ZGLV	EXTENDED ARITHMETIC SHIFT
3120	$EASH ± m,	X	ZGLV	
3140	EROT ± m,	X		EXTENDED LOGICAL ROTATE
3160	$EROT ± m,	X		
3200	NRM ,	X		NORMALIZE AND LOAD SHIFT COUNT IN X
3220	$NRM ,	X		
3300	ENRM ,	X		EXTENDED NORMALIZE AND LD S.C. IN X
3320	$ENRM ,	X		

IMMEDIATE MODE IS ASSUMED
IF EA IS POSITIVE, SHIFT OR ROTATE RIGHT
IF EA IS NEGATIVE, SHIFT OR ROTATE LEFT

Boolean Connectives (4000-7777)

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
*	X			1	OP					b					

OCTAL	MNEMONIC	FLAGS	OPERATION
4000 + b	RA	Z	A←ZEROES
4100 + b	BLA	Z	A←$\underline{A}$ NOR m
4200 + b	CBHA	Z	A←$\underline{A}$ AND m
4300 + b	ALA	Z	A←$\underline{A}$
4400 + b	CBLA	Z	A←$\underline{A}$ NOR m
4500 + b	MLA	Z	A←$\overline{m}$
4600 + b	BDA	Z	A←A EXCL. OR m
4700 + b	ELA	Z	A←A NAND m
5000 + b	BHA	Z	A←A AND m
5100 + b	BSA	Z	A←A EQUIV. m
5200 + b	MHA	Z	A←$\underline{m}$
5300 + b	CEHA	Z	A←$\underline{A}$ OR m
5400 + b	BEQT	Z	A←$\underline{A}$; Z SET IF A = m
5500 + b	CELA	Z	A←$\underline{A}$ NAND m
5600 + b	EHA	Z	A←A OR m
5700 + b	SA	Z	A←ONES

FORMAT OPNn m, x, BYTE

n , 1, 2, 4, 8 BYTE SIZE: UNSPECIFIED = 16 BIT BYTE
BYTE = 0, 1, . . . , 15 BYTE POSITION

OPTIONS: * ALL BYTE SIZE/BYTE POSITIONS
= IS BIT BYTE TO ACCUM. ONLY

ALL BOOLEAN CONNECTIVES ONLY EFFECT A 1, 2, 4, 8, OR 16 BIT BYTE OF THE ACCUMULATOR WHICH IS SPECIFIED BY THE b FIELD OF THE INSTRUCTION. THE VALUE OF b MAY BE DETERMINED FROM THE TABLE OPPOSITE

IS READ
"IS REPLACED BY"

Boolean Connectives (4000-7777)

NAME		OCTAL	MNEMONIC	FLAGS	OPERATION
RESET		6000 + b	RM	Z	m - ZEROES
BOTH LOW		6100 + b	BLM	Z	m - $\underline{A}$ NOR m
BOTH HIGH	USING $\overline{A}$	6200 + b	CBHM	Z	m - $\underline{A}$ AND m
ACCUMULATION	LOW	6300 + b	ALM	Z	m - $\underline{A}$
BOTH LOW	USING $\overline{A}$	6400 + b	CBLM	Z	m - $\underline{A}$ NOR m
MEMORY LOW		6500 + b	MLM	Z	m - $\overline{m}$
BOTH	DIFFERENT	6600 + b	BDM	Z	m - A EXCL. OR m
EITHER LOW		6700 + b	ELM	Z	m - A NAND m
BOTH HIGH		7000 + b	BHM	Z	m - A AND m
BOTH SAME		7100 + b	BSM	Z	m - A EQUIV. m
MEMORY HIGH		7200 + b	MHM	Z	m - $\underline{m}$
EITHER HIGH	USING $\overline{A}$	7300 + b	CEHM	Z	m - $\underline{A}$ OR m
BYTE EQUAL.	TEST	7400 + b	AHM	Z	m - $\underline{A}$
EITHER LOW	USING $\overline{A}$	7500 + b	CELM	Z	m - $\underline{A}$ NAND m
EITHER HIGH		7600 + b	EHM	Z	m - A OR m
SET		7700 + b	SM	Z	m - ONES

TABLE FOR COMPUTING THE VALUE OF b

BYTE SIZE	1		2		4		8		16		
OPERAND	/M		/M		/M		/M		=	/M	
BYTE # 0	01	41	02	42	04	44	10	50	00	20	60
1	03	43	06	46	14	54	30	70			
2	05	45	12	52	24	64					
3	07	47	16	56	34	74					
4	11	51	22	62							
5	13	53	26	66							
6	15	55	32	72							
7	17	57	36	76							
8	21	61									
9	23	63									
10	25	65									
11	27	67									
12	31	71									
13	33	73									
14	35	75									
15	37	77									

Double Precision D

FORMAT OPN m,X (1)

OPTIONS	FLAGS	OPERATIONS	TITLE
* U $ =	ZGL C	A:AF:AD - m:m + 1 A:AF:AD	SUBTRACT
* U $ =	ZGL	-m: m + 1 A:AF:AD (5)	CLEAR & SUBTRACT
* U $ =	ZGL	m: m + 1 A:AF:AD (5) (7)	CLEAR & ADD
* U $ =	ZGL C	A:AF:AD + m: m + 1 A:AF:AD	ADD
* $ =	ZGL	(2)	COMPARE
* U $ =	ZGL	(2)	MULTIPLY
* $		A:AF:AD → m: m + 1 (3) (6) (7)	STORE
* $	Vc	(2) (4)	STORE ROUNDED
* $	ZGL V	(2)	DIVIDE
* $ =	ZGL V	(2)	CLEAR & DIVIDE

(1)

USE OF EQUAL SIGN (=) OPTION CAUSES DATA GIVEN (56) BITS TO BE PLACED IN LITERAL POOL

(2) SUBROUTINE

(3) DST FUNCTIONALLY EQUIVALENT TO DSTU

(4) DSTU FUNCTIONALLY EQUIVALENT TO DSRU

(5) DCAU, DCSU LOAD ACCUMULATOR EXEC BITS

(6) DSTU STORES ACCUMULATOR EXEC BITS

(7) DCAU, DSTU USED TO MOVE DOUBLE PREC. DATA

OPTIONS

* INDIRECT
/ HALF WORD
= IMMEDIATE
$ SAVE
U UNNORMALIZED.

THE $ SYMBOL IS THE ONLY OPTION THAT IS A PREFIX

INTEGER 1

FORMAT OPN /m,X (1)

MNEMONIC	OCTAL	OPTIONS	FLAGS	OPERATIONS
SB	00	* / U $ =	ZGL C	A:AF - mf → A:AF
CS	01	* / U $ =	ZGL	-mf → A:AF
CA	02	* / U $ =	ZGL	mf → A:AF
AD	03	* / U $ =	ZGL C	A:AF = mf → A:AF
CP	04	* / U $ =	ZGL	Z SET IF A:AF = m A:AF UNCHANGED G SET IF A:AF > m L SET IF A:AF < m
MP	05	* / U $ =	ZGL	A:AF m → A:AF:AD
ST	06	* / U $	V	A:AF → mi A:AF UNCHANGED (2)
SR	07	* / $	VC	A:AF (ROUNDED) → m, A:AF UNCHANGED (3)
DV	10	* / U $ =	ZGL V	A:AF:AD mf → QUOTIENT A:AF
CD	11	* U / $ =	ZGL V	CLEAR AD THEN INTEGER DIVIDE

(1)

USE OF THE SLASH (1) OPTION IS REQUIRED FOR RIGHT HALF WORD ADDRESSING, IF OMITTED LEFT HALFWORD IS ASSUMED

mf 16 BIT MEMORY OPERAND CONVERTED TO FLOATING POINT (MANTISSA FILLED WITH TRAILING ZEROS, EXPONENT + 15 ATTACHED)

mi 32 BIT FLOATING POINT NUMBER SHIFTED UNTIL EXPONENT IS +15. LEADING 16 BITS OF MANTISSA IS STORED AS INTEGER

FLAGS

Z = ACCUM. ZERO
G = ACCUM. ZERO
L = ACCUM. ZERO
V = OVERFLOW
C = CARRY

(2) 1ST FUNCTIONALLY EQUIVALENT TO ISTU

(3) ISR FUNCTIONALLY EQUIVALENT TO ISRU

32 Bit Floating Point F

FORMAT OPN m,X (1)

OPTIONS	FLAGS	OPERATIONS	TITLE
* U $ =	ZGL C	A:AF -m → A:AF	SUBTRACT
* U $ =	ZGL	-m → A:AF (3)	CLEAR & SUBTRACT
* U $ =	ZGL	m → A:AF (3) (5)	CLEAR & ADD
* U $ =	ZGL C	A:AF + m → A:AF	ADD
* U $ =	ZGL	Z SET IF A:AF = m A:AF UNCHANGED G SET IF A:AF > m L SET IF A:AF < m	COMPARE
* U $ =	ZGL	A:AF m → A:AF:AD	MULTIPLY
* $		A:AF → m (1) (4) (5)	STORE
* U $	C	A:AF (ROUNDED) → m, A:AF UNCHANGED (2)	STORE ROUNDED
* U $ =	ZGL V	A:AF:AD ÷ m QUOTIENT A:AF	DIVIDE
* U $ =	ZGL V	CLEAR AD THEN FLOATING DIVIDE	CLEAR & DIVIDE

USE OF THE EQUAL SIGN (=)
OPTION CAUSES DATA (32 BITS)
TO BE PLACED IN THE LITERAL
POOL.

(1) FST FUNCTIONALLY EQUIVALENT TO FSTN

(2) FSR DIFFERS FROM FSRU IF ROUNDING RESULTS IN AN UNNORMALIZED NUMBER

(3) FCAU, FCSU LOAD ACCUMULATOR EXEC BITS

(4) FSTU STORES ACCUMULATOR EXEC BITS

(5) FCAU, FSTU USED TO MOVE WHOLE WORDS & EXEC BITS

OPTIONS

* INDIRECT

/ HALF WORD

= IMMEDIATE

S SAVE

U UNNORMALIZED

16 Bit Fixed Point

FORMAT OPN /m,X (1)

MNEMONIC	OCTAL	OPTIONS	FLAGS	OPERATIONS
SB	00	* / $ =	ZGL VC	A -m → A
CS	01	* / $ =	ZGL V	-m → A (2) (3)
CA	02	* / $ =	ZGL	m → A (2) (3)
AD	03	* / $ =	ZGL VC	A + m → A
CP	04	* / $ =	ZGL	Z SET IF A = m A UNCHANGED G SET IF A > m L SET IF A < m
MP	05	* / $ =	ZGL V	A m → A:AE
ST	06	* / $		A → m
SR	07	* / $	VC	A (ROUNDED) → M
DV	10	* / $ =	ZGL V	A:AE ÷ m QUOTIENT → A REMAINDER → AE
CD	11	* / $ =	ZGL V	CLEAR AE THEN DIVIDE

(1)

USE OF THE SLASH (/)
OPTION IS REQUIRED FOR
RIGHT HALF-WORD ADDRESSING
IF THE SLASH IS OMITTED
LEFT HALF-WORD ADDRESSING
IS ASSUMED.

FLAGS

Z = ACCUM. ZERO
G = ACCUM. > ZERO
L = ACCUM. < ZERO
V = OVERFLOW
C = CARRY

(2) 0 → AF
0 → AD (1-8)

(3) CA, CS LOAD LEFT ACCUMULATOR EXEC BIT

(4) ST STORES LEFT ACCUMULATOR EXEC BIT

Index X

FORMAT OPN/m, X (1, 2, 4)

OPTIONS	FLAGS	OPERATIONS	TITLE
* $ / = U	ZGL VC	X -m → X	SUBTRACT
* $ / = U	ZGL V	-m → X	CLEAR & SUBTRACT
* $ / = U	ZGL	m → X	CLEAR & ADD
* $ / = U	ZGL VC	X + m → X	ADD
* $ / = U	ZGL	Z SET IF X → m X UNCHANGED G SET IF X > m L SET IF X < m	COMPARE
* $ / = U		(3)	MULTIPLY
* $ / = U		X → m	STORE
* $ / = U	C	(3)	STORE ROUNDED
* $ / = U	AGL V	(3)	DIVIDE
* $ / = U	ZGL V	(3)	CLEAR & DIVIDE

(1)

USE OF THE SLASH (/) OPTION IS REQUIRED FOR RIGHT HALF-WORD ADDRESSING IF THE SLASH IS OMITTED LEFT HALFWORD ADDRESSING IS ASSUMED

(2)

AN INDEX REGISTER MUST BE SPECIFIED

(3)

SUBROUTINE

(4)

ADDRESS m IS NON-INDEXABLE

OPTIONS

* INDIRECT
/ HALFWORD
= IMMEDIATE
$ BLANK
U UNNORMALIZED

Extended Precision

FORMAT OPN m, X

MNEMONIC	OCTAL	OPTIONS	FLAGS	OPERATIONS
SB	00	* $ =	ZGL VC	A:AE -m → A:AE
CS	01	* $ =	ZGL V	-m → A:AE (3)
CA	02	* $ =	ZGL	m → A:AE (3)
AD	03	* $	ZGL VC	A:AE + m → A:AE
CP	04	* $ =	ZGL	(1)
MP	05	* $ =	ZGL	(1)
ST	06	* $		A:AE → m
SR	07	* $	C	(1)
DV	10	* $ =	ZGL V	(1)
CD	11	* $ =	ZGL V	(1)

(1)

SUBROUTINE

(2)

USE OF THE EQUAL SIGN (=) OPTION CAUSES DATA (32 BITS) TO BE PLACED IN THE LITERAL POOL

(3)

0 → AF
0 → AD (1-8)

FLAGS

Z = ACCUM ZERO
G = ACCUM > ZERO
L = ACCUM < ZERO
V = OVERFLOW
C = CARRY

APPENDIX 3

TABLE OF INTERRUPT ADDRESS CODES

Interrupt Name	Priority	Octal Address	Mask (M Internal Mask E External Mask)
P/P	1	'40	Cannot be masked
Data Exec	2	'41	M '40000
Priv. Inst.	3	'42	M '20000
Inst. Exec	4	'43	M '10000
Exp. Fault	5	'44	M '4000
Mem. Protect	6	'45	M '2000
Timer	7	'46	M '1000
Console	8	'47	M '400
Data Channel 0	9	'50	M '200
Data Channel 1	10	'51	M '100
Data Channel 2	11	'52	M '40
Data Channel 3	12	'53	M '20
Data Channel 4	13	'54	M '10
Data Channel 5	14	'55	M '4
Data Channel 6	15	'56	M '2
Data Channel 7	16	'57	M '1
External Interrupt (1, 0)*	17	'60	E '-0
External Interrupt (1, 1)	18	'61	E '40000
External Interrupt (1, 2)	19	'62	E '20000
External Interrupt (1, 3)	20	'63	E '10000
External Interrupt (1, 4)	21	'64	E '4000
External Interrupt (1, 5)	22	'65	E '2000
External Interrupt (1, 6)	23	'66	E '1000
External Interrupt (1, 7)	24	'67	E '400
External Interrupt (1, 8)	25	'70	E '200
External Interrupt (1, 9)	26	'71	E '100
External Interrupt (1, 10)	27	'72	E '40
External Interrupt (1, 11)	28	'73	E '20
External Interrupt (1, 12)	29	'74	E '10
External Interrupt (1, 13)	30	'75	E '4
External Interrupt (1, 14)	31	'76	E '2
External Interrupt (1, 15)	32	'77	E '1
External Interrupt (2, 0)	33	'100	Set by SFL = '60, 0 Reset by SFL = '61, , 0
↓	↓	↓	↑ This enables or disables all interrupts in banks 2-16
External Interrupt (16, 15)	271	'457	

More than one interrupt is enabled by forming a composite mark code that is the arithmetic sum of the individual code; e.g., for External Interrupts 4, 5, 6, mark = E '7400.

**(b, n) Where b=Bank number*

n=Number of the interrupt in the bank.

HYBRID OPERATIONS

Operation Codes for High-Speed Conversions

1. ANALOG-TO-DIGITAL CONVERSIONS

SFL = '-47400+a,,2	Sequential Conversion
SFL = '+47400+a,,2	Random Conversion
SFL = '+17400+a,,2	Individual T/S : Store
SFL = '+07400+a,,2	Individual T/S : Track
SFL = '+37400+a,,2	Block of T/S : Store
SFL = '+27400+a,,2	Block of T/S : Track
SFL = '+47600,,2	ADC Control Readout
TSL = '+07400,,2	ADC Test

Digital-to-Analog Conversions

SFL = '-07000+a,,2	Sequential Load DAC
SFL = '-47000+a,,2	Sequential Jam DAC
SFL = '+07000+a,,2	Random Load DAC
SFL = '+47000+a,,2	Random Jam DAC
SFL = '+27000+a,,2	Individual DAC Channel: Transfer
SFL = '+17000+a,,2	Block of DAC Channels: Transfer
SFL = '+47200,,2	Clear all DAC Registers
TSL = '+07000,,2	DAC Test

a = conversion channel address or block address.

2. OPERATION CODES FOR ANALOG MONITOR/CONTROL

DVM Conversion	LDOB = '16+c,,12
DVM Readout	LDOB = '01+c,,12
Analog Address Selection	LDOB = '04+c,,12
Analog Address Readout	LDOB = '04+c,,12
Analog Address Step	LDOB = '17+c,,12
Analog Value Selection	LDOB = '05+c,,12
Potentiometer Setting	LDOB = '15+c,,12
Single Print	LDOB = '14+c,,12
Analog Mode Selection	LDOB = '07+c+m,,12
Analog Time Constant Selection, 1000:1	LDOB = '10+c+t1,,12
10:1	LDOB = '11+c+t2,,12
Logic Mode Selection	LDOB = '12+c+d,,12
Logic Word Input	LDOB = '02+c,,12
Logic 16 Bit Word Output	LDOB = '02+c,,12
Logic 8 Bit Word Output	LDOB = '13+c,,12
Mask Register Loading	LDOB i,,12
Status Word Readout	LDOB = '06+c,,12
Fault Word Readout	LDOB = '03+c,,12
Monitor Word Readout	STIB = M,,12

where *c* = analog console number times 2^5 in octal

for example: c = 40 for console 1

c = 100 for console 2

and *m* follows the table below:

OD	'0400
RT	'1000
ST	'1400
OP	'2000
H	'2400
IC	'3000
PS	'3400

and *t1* follows the table below:

MSEC	'000
SEC	'400

and *t2* follows the table below:

FAST	'0400
MED	'1400
SLO	'1000

and *d* will be determined by the following table:

RUN	'0400
STOP	'1000
CLR	'1400

finally *i* corresponds to a mask placed into the most significant two octal digits of the address, thus varying between '00000 and '77000 changing only the leftmost two digits.

3. HYBRID SENSELINES AND FUNCTION LINES

TSL = '6000 + C + S ,, 2
SFL = '6000 + C + S ,, 2

where c is console # as defined above and
s is the address of the sense or control line in octal.

APPENDIX 4

TABLE OF SFL/TSL CODES

1. PROCESSOR INTERRUPT SFL

SFL	Function	Flag Indication
21	Reset Privileged Instruction Interrupt Occurred Indicator	B flag if set
23	Reset Memory Parity Error Indicator	B flag is set
25	Reset Console Interrupt 1 Indicator	B flag if set
27	Reset Console Interrupt 2 Indicator	B flag if set
31	Reset Console Interrupt 3 Indicator	B flag if set
33	Reset Console Interrupt 4 Indicator	B flag if set
40	Set Memory Protect Bank 1 Indicator	B flag if set
41	Reset Memory Protect Bank 1 Indicator	B flag if set
42	Set Memory Protect Bank 2 Indicator	B flag if set
43	Reset Memory Protect Bank 2 Indicator	B flag if set
44	Set Memory Protect Bank 3 Indicator	B flag if set
45	Reset Memory Protect Bank 3 Indicator	B flag if set
46	Set Memory Protect Bank 4 Indicator	B flag if set
47	Reset Memory Protect Bank 4 Indicator	B flag if set
60	Set External Interrupt Enable Indicator	B flag if set
61	Reset External Interrupt Enable Indicator	B flag if set
62	Set Internal Timer On-Off Control	B flag if set
63	Reset Internal Timer On-Off Control	B flag if set
65	Reset Monitor Mode Indicator	B flag if set
67	Reset Memory Protect Interrupt Occurred Indicator	B flag if set

2. PROCESSOR INTERRUPT TSL

TSL	Function	Flag Indication
21	Tested Privileged Instruction Interrupt Occurred Indicator	Z flag if set
23	Tested Memory Parity Error Indicator	Z flag if set
25	Tested Console Interrupt 1 Indicator	Z flag if set
27	Tested Console Interrupt 2 Indicator	Z flag if set
31	Tested Console Interrupt 3 Indicator	Z flag if set
33	Tested Console Interrupt 4 Indicator	Z flag if set
40	Tested Memory Protect Bank 1 Indicator	Z flag if set
42	Tested Memory Protect Bank 2 Indicator	Z flag if set
44	Tested Memory Protect Bank 3 Indicator	Z flag if set
46	Tested Memory Protect Bank 4 Indicator	Z flag if set
60	Tested External Interrupt Enable Indicator	Z flag if set
62	Tested Internal Timer On-Off Control Indicator	Z flag if set
65	Tested Monitor Mode Indicator	Z flag if set
67	Tested Memory Protect Interrupt Occurred Indicator	Z flag if set

3. EXCHANGE INTERRUPT SFL

SFL	Function	Flag Indication
-00000+K*	Unconditional Channel Clean	B
00001+K	Disconnect Channel	B
00002+K	Enable Channel Ready Interrupt	B
00004+K	Disable Channel Ready Interrupt	B
00010+K	Enable Channel Signal Interrupt	B
00020+K	Disable Channel Signal Interrupt	B

4. EXCHANGE INTERRUPT TSL

TSL	Function	Flag Indication
00001+K	Test Channel Signal	Z
-00001+K	Test Channel Signal and Clear	Z
00002+K	Test Channel Parity	Z
-00002+K	Test Channel Parity and Clear	Z
00004+K	Test Channel Ready	Z

**K= 00000 for Channel 0* *= 40000 for Channel 4*
= 10000 for Channel 1 *= 50000 for Channel 5*
= 20000 for Channel 2 *= 60000 for Channel 6*
= 30000 for Channel 3 *= 70000 for Channel 7*

5. HYBRID SFL's

Digital-to-Analog Conversion

SFL	Function
0700+a**	Sequential/Normal Conversion
47000+a	Sequential/Jam Conversion
-07000+a	Random/Normal Conversion
-47000+a	Random/Jam Conversion
27000+a	Individual Channel Transfer
17000+a	Block Transfer
07200	Clear all DAC Registers

Analog-to-Digital Conversion

47400+a	Sequential Conversion
-47400+a	Random Conversion
17400+a	Individual T/S Store
07400+a	Individual T/S Track
37400+a	Block T/S Store
27400+a	Block T/S Track
27600	Control Readout

***a = DAC Channel Address or Block Address.*

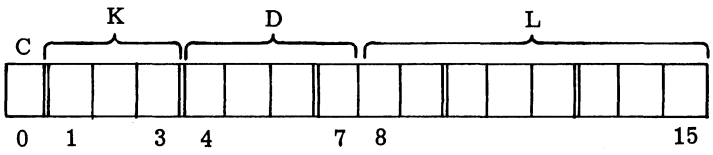
SFL AND TSL CODES FOR 8400 STANDARD PERIPHERAL EQUIPMENTS

GENERAL FORMAT

(Applicable to *all* Data Channels and Device Controllers attached to Data Channels.)

SFL Instructions

"M" Field



C = 0→SFLC. Used for general Channel/Device control conditions.

C = 1→SFLF. Used to Initialize Channel and Connect Device.
Also, for Unconditional Channel/Device Disconnect if D = 0.

K = Data Channel Designator, 0 to 7.

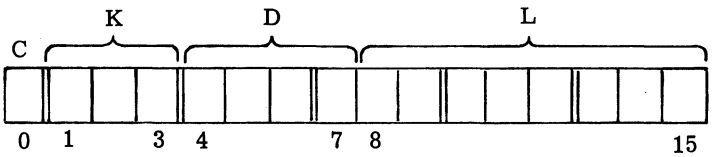
D = Peripheral Device Designator, 1 to 15.

L = M field allocated for:

1. Setting Channel Function Register (CFR), if C = 1 and D = 0.
2. Device Control, if C = 0 and D ≠ 0, (i.e., Device Control Word).
3. Channel Control, if C = 0 and D = 0, (i.e., CHRI and CHSI).

TSL Instructions

"M" Field



C = 0→TSLC. Intended for general Channel/Device status testing.

C = 1→TSLF. Intended for general Channel/Device status testing, followed by resetting the addressed status line(s) to zero on the same instruction.

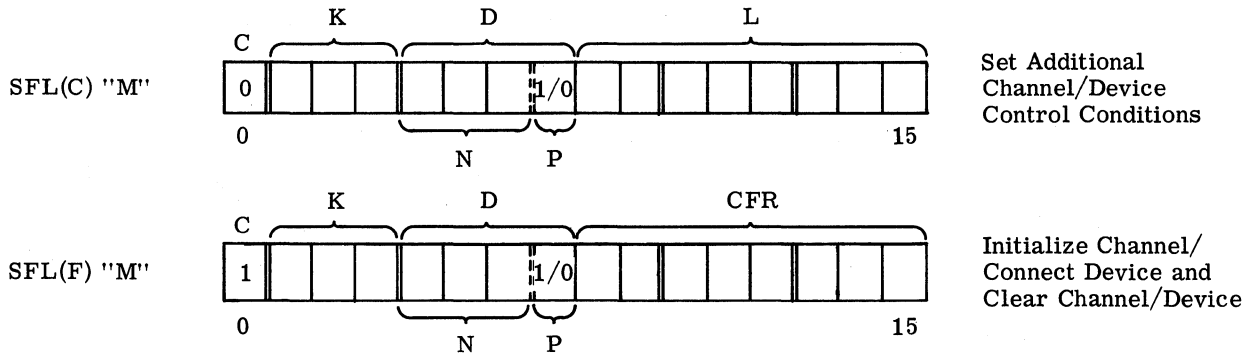
K = Data Channel Designator, 0 to 7.

D = Peripheral Device Designator, 1 to 15.

L = M field allocated for:

1. Testing Channel conditions, if D = 0.
2. Testing Peripheral Device conditions, if D ≠ 0.

SFL/TSL INSTRUCTIONS FOR TYPEWRITER OPERATIONS



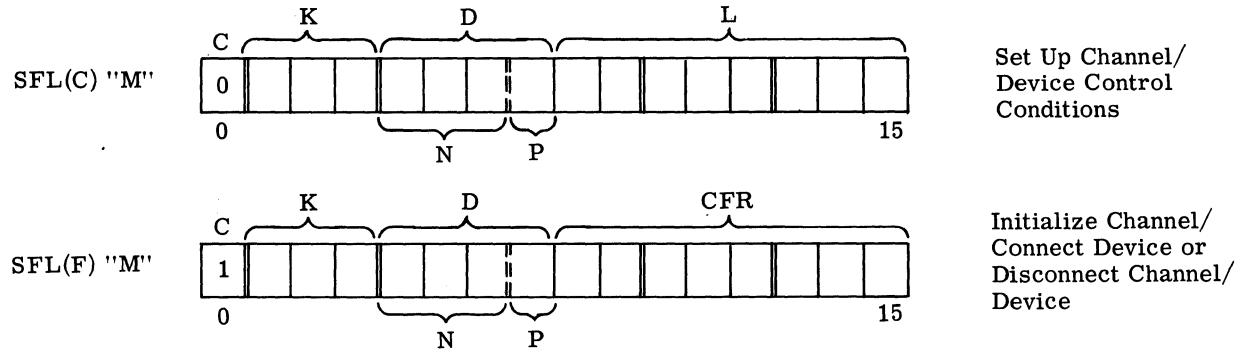
K = 0 to 7
D = (N)p = (3)o

Function	"L" Code/CFR								Comments	Indicators Affected
Set Ribbon Black SFL(C)	X	X	X	X	X	X	X	1	Normal state when Data Channel is cleared or disconnected.	None
Set Ribbon Red SFL(C)	X	X	X	X	X	X	1	X		None
Initialize Channel/ Connect Device	CFR (Channel transfer conditions, data format.)								Permits input or output to Typewriter. Unlocks keyboard for input.	Channel Signal Channel Parity Channel Interrupt

TSL Instructions

There are no TSL instructions associated directly with Typewriter status signals. Parity error and certain signals (carriage return, illegal control character) are tested via the Channel Parity and Channel Signal indicators, respectively.

SFL/TSL INSTRUCTIONS FOR PAPER TAPE READER OPERATIONS



K = 0 to 7
D = (N)p = (1)o

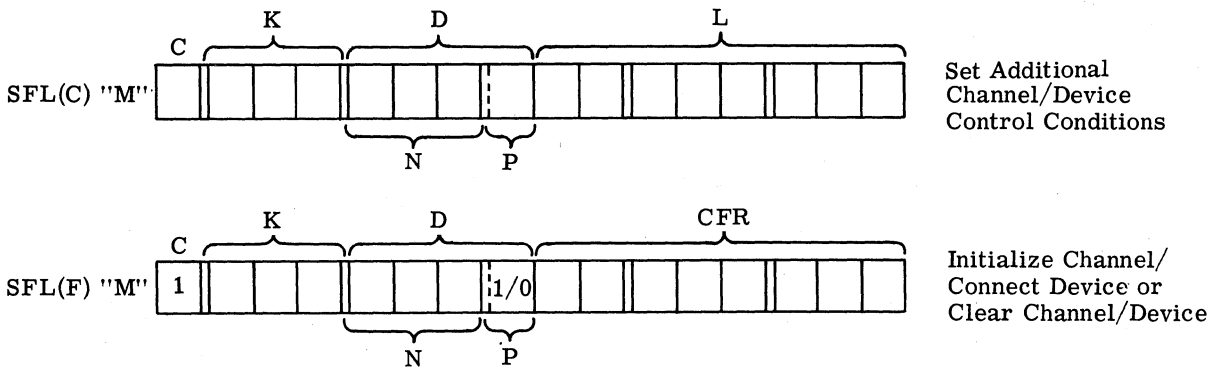
X - Don't Care Bits

Function	"L" Code/CFR								Comments	Indicators Affected
Set Reader Forward SFL(C)	X	X	X	X	X	X	X	1	Set direction of tape motion to Forward.	Device "Busy" response sets "Busy" flag if tape in motion.
Set Reader Reverse SFL(C)	X	X	X	X	X	X	1	X	Set direction of tape motion to Reverse.	As Above
Initialize Channel/ Connect Device SFL(F)	CFR								Connects Reader to Data Channel and starts tape motion.	Channel Parity Channel Signal Channel Interrupt Stop Code Device "Busy" if Power off or Reader in "Load" state.

TSL Instructions

No status lines are tested via TSL instructions directly.

SFL/TSL INSTRUCTIONS FOR PAPER TAPE PUNCH OPERATIONS



K = 0 to 7

D = (N)p = (2)o

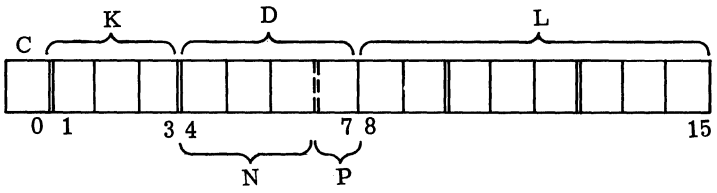
X - Don't Care Bits

Function	"L" Code/CFR	Comments	Indicators Affected
Power On SFL(C)	X X X X X X 1 X		None
Power Off SFL(C)	X X X X X X X 1		None
Initialize Channel/ Connect Device SFL(F)	CFR (Channel data transfer conditions, format, etc.)	Set up Channel transfer conditions, connect Punch and start data transfer if Channel and Punch are ready.	"Busy" flag in Flag Register if: a) Power off and b) Tape is low. Channel Parity Channel Signal Channel Interrupt

TSL Instructions

There are no status levels tested by TSL instructions.

SFL INSTRUCTION LIST (FUNCTIONS AND CODES)
SERIAL (COLUMN-BY-COLUMN) CARD PUNCH

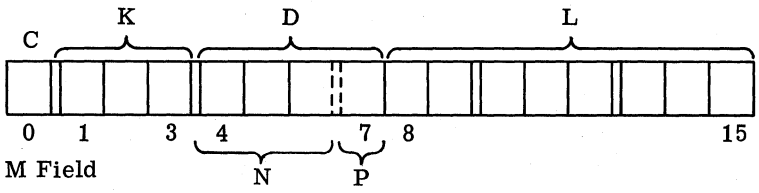


M field
C = 0→SFLC - Used to Set Channel/Device Control Conditions
C = 1→SFLF - Used to Initialize Channel, Connect Device, Unconditional Channel/Device Disconnect (D = 0)
K = Data Channel Number, 0 to 7
D = Device Number; Card Punch Device No. = (N)p = (2)1
L = M Field Allocated for Setting of Data Channel Function Register, CFR (C = 1) or Device Control Conditions (D, C = 0)
X = Don't Care

Function	SFLC or SFLF	CFR (for SFLF or L (for SFLC) 8	9	10	11	12	13	14	15	Comments	Program Indicators Affected
Start Card Punch Cycle	SFLC	X	X	X	X	X	X	X	1	Initiate "Card Punch Cycle" w/o connecting CP Controller to Data Channel.	"Busy" Flag. See other sections for conditions.
Start Card Punch Cycle *and* Connect Punch (Cont) to Data Channel	SFLF	CFR: X	X	X	X	X	X	X	X	Initiate "Card Punch Cycle" and connect CP controller to Data Channel.	See other sections for conditions.
Eject Card* (Terminate Card Cycle)	SFLC	X	X	X	X	1	X	X	X	Card ejected to output hopper after punching previous column. Next card brought to Reg. Station -- waiting for next "Start Card Punch Cycle" command.	None Unconditional Command
Reject (Offset) Card	SFLC	X	X	X	X	X	X	1	X	Ejected mispunched card appears offset by 1/2 inch output hopper.	None Unconditional Command
Set DINE Flip-Flop (DINE = Dev. Interrupt Enable)	SFLC	1	X	X	X	X	X	X	X	Set Dev. Int. Enable flip-flop to allow Dev. Int. when Dev. is NOT selected, DINE flip-flop set and Card Punch is ready for next Card Punch Cycle.	"Busy" Flag, if DINE flip-flop could not be set.

**A card is also ejected in response to "Carriage Return" character, $(155)_8$. This character is not punched.*

TSL INSTRUCTION LIST (FUNCTIONS AND CODES)
SERIAL (COLUMN-BY-COLUMN) CARD PUNCH



C = 0, Test Only, TSLC

C = 1, Test and Reset, TSLF

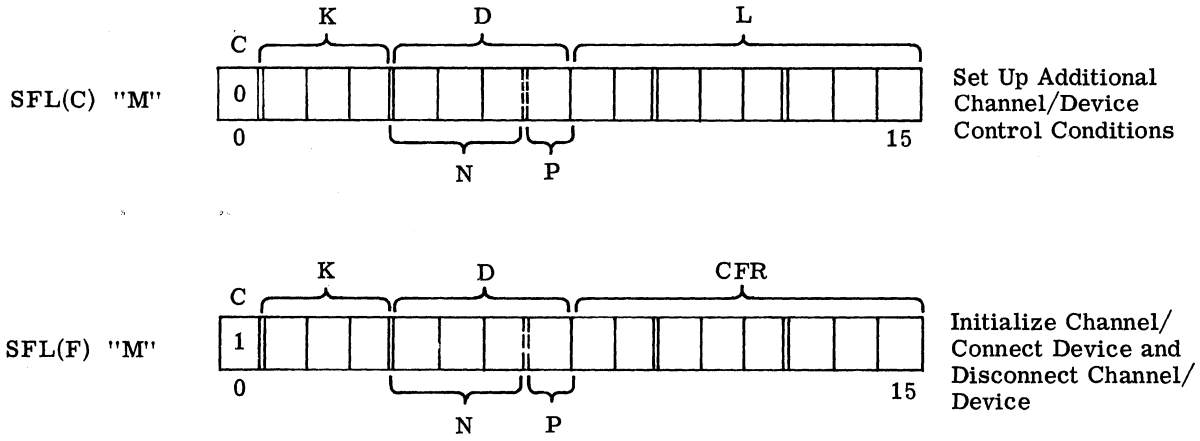
K = Data Channel Number, 0 to 7

D = Device Number; Card Punch Controller Code = (N)p = (2)1

L = Channel/Device Control Field (Channel Field if D = 0)

Function	TSLC or TSLF	L Field 8	9	10	11	12	13	14	15	Comments
Test "Card Punch Operable" Status	TSLC	X	X	X	X	X	X	X	1	See other section for definition when signal is true.
Test "Card Punch Cycle" Status	TSLC	X	X	X	X	X	X	1	X	This line becomes true on start of new card cycle (in response to SFL command) and is reset when card is ejected.
Test "Binary Mode" Status	TSLC	X	X	1	X	X	X	X	X	This line is "true" when the card is being punched in "Binary Mode" and "false" when card is being punched in "Hollerith Mode."
Test "End of Card" Status	TSLC	X	X	X	X	X	1	X	X	This line becomes true on ejection of card and remains true until *next* card is in Register Station. Start "Card Punch Cycle" SFL commands will be rejected during this period.
Test "Punch Error" Status	TSLF	X	X	X	X	1	X	X	X	This line becomes true when an error (["Echo-check"] error or "overflow") is detected on punching a particular column. The Punch Controller will eject the card in response to the Punch "next column data request." Data Channel will be disconnected and an interrupt generated.

SFL INSTRUCTIONS FOR CARD READER OPERATIONS

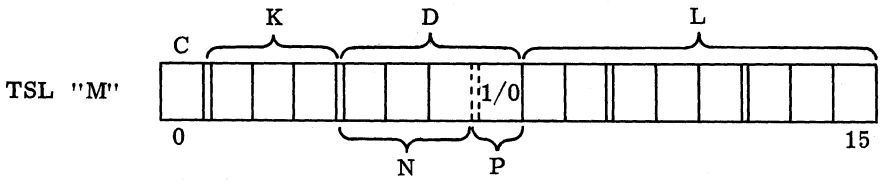


D = (N)p =(1)1 X = Don't Care Bit Positions,
K = 0 to 7 Combined Operations Possible

Function	"L" Code/CFR	Comments	Indicators
Start Card Cycle SFL(C)	X X X X X X X 1	A card is started on its way to Read Station. An SFL(F) must follow within a period of time to enable reading.	Device "Busy" setting "Busy" Flag if Card Cycle in Progress or Card Reader not Ready.
Initiate Channel/Connect Device SFL(F) Instruction	CFR (Data Channel Options)	Connects Card Reader to Data Channel initiates "Card Cycle" if not already started. Cards are read continuously until Data Channel is disconnected.	Channel Signal, Channel Interrupt, Channel Parity
Disconnect Power from Card Reader SFL(C)	X X X X X 1 0 X	Disconnect Power when Reader not expected to be used for some time.	No Device "Busy" Response
Set "Device Interrupt Enable" (DINE) Flip-Flop	1 X X X X X X X	Enables Channel Interrupt if Reader is Ready for next card cycle, (i. e., Card Reader Ready = "1").	"Busy" response if any device already selected on this Data Channel.

NOTE: The two different instructions to start a card cycle are available to permit the use of Data Channel with other devices on the channel while a card is being moved relatively slowly into "read station".

TSL INSTRUCTIONS FOR LINE PRINTER OPERATION



D = (N)p = (3)1

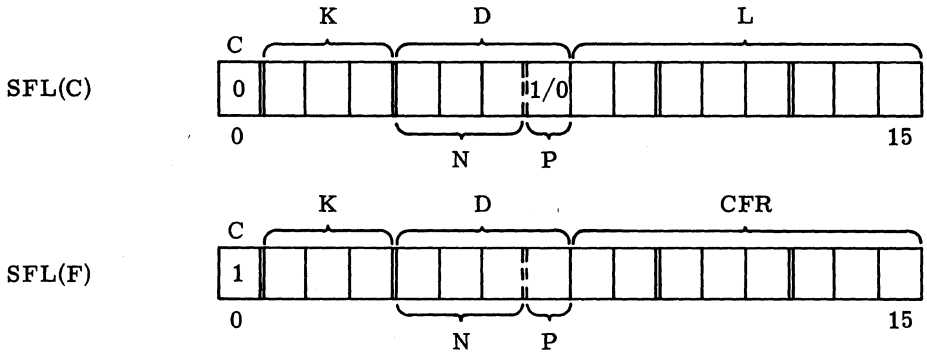
C = 0, Test Status Line Only

C = 1, Test Status Line, Then Clear Status Indicator

X = 0, Unless Combined Test Specified

Function	"L" Field								Status Definition and/or Comments
Test Printer Ready	X	X	X	X	X	1	X	X	Printer in operable condition. Power turned on. "Operate-Standby" switch on Operator Panel in "operate" position, down gate is closed. Paper is in printing position.
Next Character Request (=Send Data)	X	X	X	X	X	X	X	1	Printer available to accept next character. The line will be false during printing or paper spacing operations.
Next Line Request	X	X	X	X	X	X	1	X	Printing of previous line is complete. The Printer is ready for next line.
Printer Buffer Full	X	X	X	X	1	X	X	X	132 characters transmitted to Printer Buffer without print (end of message) command.
Paper Advancing	X	X	X	1	X	X	X	X	Paper advancing not complete.

SFL INSTRUCTIONS FOR PARALLEL (ROW-BY-ROW)
CARD PUNCH OPERATIONS



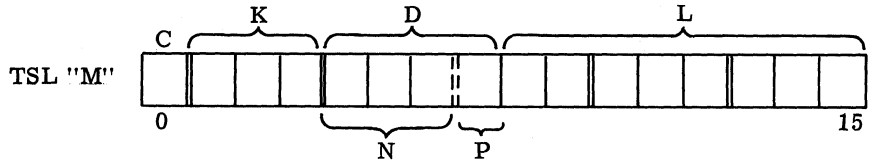
K = 0 to 7

D = (N)p = (2)1
= Device #

X = Don't Care Unless Combined Operation Desired

Function	"L" Code/CFR								Comments	Indicators Affected
Start Card Punch Cycle	X	X	X	X	X	X	X	1	Start card on its way to Punch Station. Stop after one card moved unless Punch connected to Data Channel.	Device "Busy" response if a) Punch not Ready b) Power to Punch off.
Reject Card to AUX Stacker	X	X	X	X	X	X	1	X	The card being punched to be ejected to AUX Stacker.	None
Initialize Channel/ Connect Device	CFR								Permits data transfer and punch operation. Allows bringing next card into punch station if Channel/ Device not discon- nected.	Device "Busy" response if instruction is too late to punch complete card. Channel Signal, Channel Parity, Channel Interrupt.
Set "Device Interrupt Enable" (DINE) Flip-Flop	1	X	X	X	X	X	X	X	Permits an Interrupt when device becomes operable again - following some fail- ure.	Channel Interrupt "Busy" response if at least one device already connected to channel.

TSL, INSTRUCTIONS FOR CARD READ OPERATION

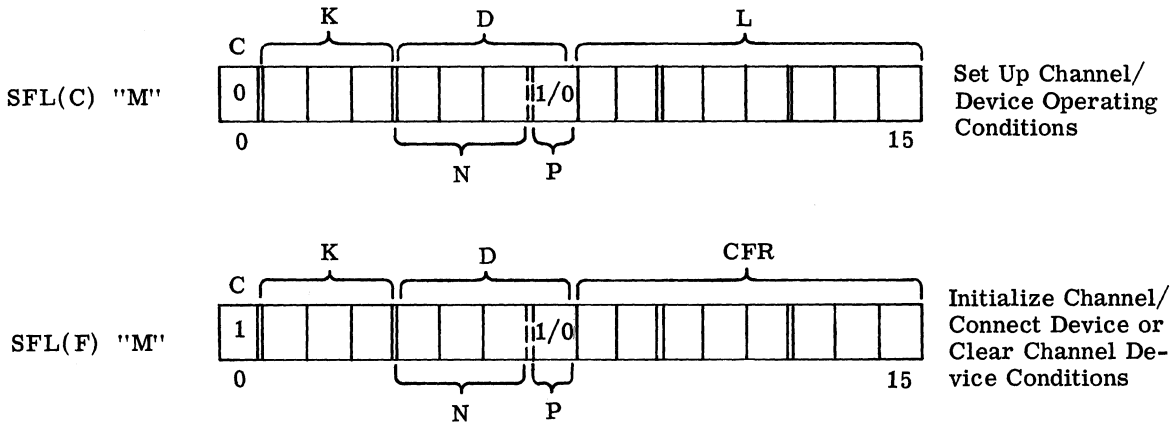


C = 0, Test Only
C = 1, Test and Clear
D = (N)p, K = 0 to 7
= (1)1

X = Don't Care Conditions Unless Combined Tests Requested

Function	"L" Code								Status Definition and/or Comments
Test "Card Reader Ready" Status	X	X	X	X	X	X	1	X	No jams. Stacker not full. Covers in place. Power on. Start button depressed. Feeder ready (Model 1). Read Circuits OK. Card line mechanism locked (Model 1). Hopper not empty. No character validity error.
Test "Read Cycle in Progress" Status	X	X	X	X	X	1	X	X	Response to "Start Card Cycle" command. "True" until all 80 columns are read. New card cycle initiated, after some delay when this signal goes "False" provided Data Channel remains connected.
Test "Hopper Empty" Status	X	X	X	X	1	X	X	X	True when hopper is empty *and* End of File button has been depressed.
Test "Reader Error" Status	X	X	X	1	X	X	X	X	A photo cell malfunctioning or invalid character detected (Hollerith Mode) - if validity switch is on.
Test "Overflow" Status	X	X	1	X	X	X	X	X	A character has been missed. Once a card cycle is initiated characters are available at fixed intervals.
Test "Binary Card" Status	X	X	X	X	X	X	X	1	The card being read is a "Binary Card".
Test "Card Reader Continue" Status	X	1	X	X	X	X	X	X	Card Reader has been put on-line again and the operator has depressed "Card Reader Continue" button.

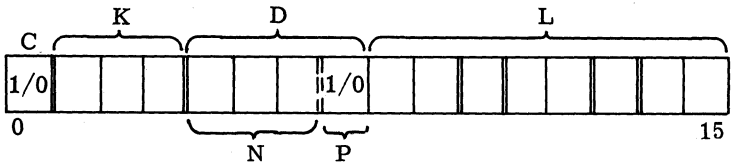
SFL INSTRUCTIONS FOR LINE PRINTER OPERATION



K = 0 to 7
D = (N)p = (3)1
X = Don't Care Bit Positions

Function	"L" Code/CFR								Comments	Indicators Affected
Initialize Channel/ Connect Device SFL(F)	CFR L								Connects Printer to Data Channel and makes the latter responsive to signals from the Printer buffer.	Device "Busy" response if Printer is not ready.
Set "Device Interrupt Enable" (DINE) Flip-Flop	1	0	X	X	X	X	X	X	Enables Channel Interrupt Device Busy if DES="1".	
Advance Carriage to Control Tape Hole on Channel 1	X	1	X	X	0	0	0	0	Top of Form	
2	X	1	X	X	0	0	0	1	Different tapes available to suit particular format requirements.	Device "Busy" response if a) printing cycle not complete b) previous space operations not complete.
3	X	1	X	X	0	0	1	0		
4	X	1	X	X	0	0	1	1		
5	X	1	X	X	0	1	0	0		
6	X	1	X	X	0	1	0	1		
7	X	1	X	X	0	1	1	0		
8	X	1	X	X	0	1	1	1		
Adv. Carr. by 1 Line	X	1	X	X	1	0	0	X	Permit direct paper spacing control without reference to tape loop.	Same as in previous case.
2	X	1	X	X	1	0	1	0		
3	X	1	X	X	1	0	1	1		
4	X	1	X	X	1	1	0	0		
5	X	1	X	X	1	1	0	1		
6	X	1	X	X	1	1	1	0		
7	X	1	X	X	1	1	1	1		
"1" in this Position Specifies Paper Space Instructions									The same codes are used to obtain paper spacing by 1st character in a line.	
Disable Auto Carriage Advance on 1st Character of a Line	0	0	X	X	X	X	X	1	To permit overprinting of a line if desired.	Device "Busy" response if paper spacing taking place.

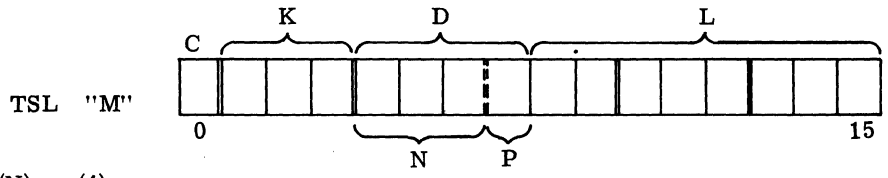
TSL INSTRUCTIONS FOR PARALLEL (ROW-BY-ROW)
CARD PUNCH OPERATION



K = 0 to 7
D = (2)1 = (N)p = Device Number
C = 0 Test Status Level Only
C = 1 Test Status and Then Reset Status Flip-Flop
X = Don't Care Bits, Unless Combined Tests Desired

Function	"L" Code								Status Definition and Comments
Test "Punch Ready" Status	X	X	X	X	X	X	X	1	Cards in hopper. Die in place. Card line mechanism locked up. Card in position to be punched. Stacker not full. Power on. No jam condition. Covers are in place. No punch error.
Test "Ready to Punch Next Row" Status	X	X	X	X	X	1	X	X	Punch is ready to punch next row on card.
Test "Punch Cycle" Status	X	X	X	X	X	X	1	X	Card is in punch cycle. Status remains true for the duration of punching a card.
Test "Punch Error" Status	X	X	X	X	1	X	X	X	An invalid Hollerith character punched. Inhibited in "BIN" mode.

TSL INSTRUCTION LIST AND CODES FOR MAGNETIC TAPE OPERATIONS

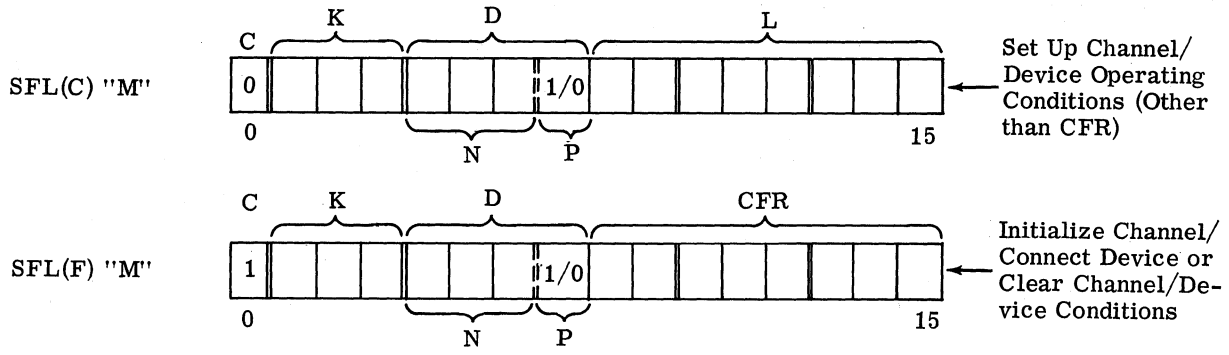


D = (N)p = (4)o
K = 0 to 7
X = 0, Unless "Combined" Tests Preferred

If C = 0 Test Only
If C = 1 Test and Clear Status Indicator

Function	"L" Field								Status Definition and/or Comments
Test and Clear End of Record Indicator	1	0	X	X	X	X	X	1	This indicator is needed only to enable programmer to test whether EOR has been reached when Data Channel is not connected to tape unit.
Test and Clear End of File Indicator	1	0	X	X	X	X	1	X	Set up by EOF marker in Read or Write (by Check head) operations.
Test and Clear Overflow Indicator	1	0	X	X	X	1	X	X	Set up when Character is missed in Read operation or is late (Rate check) in Write operation.
Test if Tape Unit Ready	1	0	X	X	1	X	X	X	Tape Unit is ready to accept a new instruction.
Test if Tape is at Load Point (BOT)	1	0	X	1	X	X	X	X	Load point tab is at photo sense head.
Test if End of Tape (EOT) has been Reached	1	0	1	X	X	X	X	X	End of tape tab has passed the photo sense head.
Test if File-Protect is On	1	1	X	X	X	X	1	X	Tape unit is loaded with reel equipped with a file-protect ring.
Test for High Density	1	1	X	X	1	X	X	X	Tape unit selected for High Density (800 bpi) recording.
Test for Medium Density	1	1	X	1	X	X	X	X	Tape unit selected for Medium Density (556 bpi) recording.
Test if Tape Unit is Rewinding	0	1	X	X	X	Unit 0 to 7			The unit addressed is in the Rewinding state. A number of tape units can be in the Rewinding status at the same time.
Test if selected tape unit is set for 7-track mode.	1	1	X	X	X	1	X	X	This test is relevant when both the 7-track and the 9-track features are available in a particular Magnetic Tape System.

SFL INSTRUCTION LIST AND CODES FOR MAGNETIC TAPE OPERATIONS



FWD/REV = Forward/Reverse Motion
F/R = File/Record
W/R = Write/Read
D = (N)p = (4)o

X = 0, Unless "Combined" Operations Required

Function	"L" Code or CFR						Comments	Indicators (Enabled)
	Control		Fwd/Rev	F/R	W/R	Unit #		
1. Read Record(s) FWD	0	1	0	0	0	0 to 7	Read record and enable data transfer if Channel Initialized and Device connected. Otherwise skip one record.	Channel Parity, Channel Signal, Channel Interrupt, EOR, EOT, Overflow.
2. Read Record(s) REV	0	1	1	0	0	0 to 7	As 1 but tape moves in REV direction.	As in 1 but BOT instead of EOT.
3. Search End of File FWD	0	1	0	1	0	0 to 7	As in 1 but Channel Initialize/Device Connect Instr. omitted. Stop after one record. Search for File Mark instead of End of Record.	EOT, EOF
4. Search End of File REV	0	1	1	1	0	0 to 7	As in 3 but tape moves in REV direction.	BOT, BOF
5. Write Record(s) (FWD Only)	0	1	0	0	1	0 to 7	Write a record after Channel Initialize/ Device Connect Instr. and write LPC at end of record (word count = 0).	Channel Parity, Channel Signal, Channel Interrupt, EOR, EOT, Overflow
6. Write Blank Tape (=Erase) FWD	0	1	0	1	1	0 to 7	Write as in 5 but record "all zero" characters. Stop tape after 3-3/4 inches.	Channel Parity (if tape not blank), EOT
7. Set Unit Field	0	0	0	0	0	0 to 7	Set up the unit field of control register, but do not initiate any tape operation to allow testing of sel. unit status.	
8. Write End of File	1	1	0	1	1	0 to 7	Write EOF mark and its LPC, (17)8 in both cases, under control of Tape Controller.	EOF, EOT
9. Rewind to Load Point	0	0	1	1	0	0 to 7	Initiate Rewind operation via a trigger. Transfer to other tape units not inhibited while rewinding.	BOT
10. Rewind and Unlock	0	0	1	1	1	0 to 7	As in 9 but tape unit switched to "LOCAL" mode.	BOT
11. Initialize Channel Connect Device	CFR						This instruction enables data transfer and makes channel responsive to Device signals.	"Busy" flag if data transfer not possible at that time.
12. Set "Device Interrupt Enable" (DINE) Flip-Flop	1	0	X	X	X	X	Allow channel interrupt when MT becomes operable again and this flip-flop is set.	"Busy" response if any device on this Data Channel already connected for data transfer (DES flip-flop set).

APPENDIX 5

CHARACTER CODE EQUIVALENCE TABLE

Char.	Card Punch (Hollerith)	Octal **	Octal *	Hex	Char.	Card Punch (Hollerith)	Octal **	Octal *	Hex
0	0	00	00	00	-	11	200	40	20
1	1	04	01	01	J	11-1	204	41	21
2	2	10	02	02	K	11-2	210	42	22
3	3	14	03	03	L	11-3	214	43	23
4	4	20	04	04	M	11-4	220	44	24
5	5	24	05	05	N	11-5	224	45	25
6	6	30	06	06	O	11-6	230	46	26
7	7	34	07	07	P	11-7	234	47	27
8	8	40	10	08	Q	11-8	240	50	28
9	9	44	11	09	R	11-9	244	51	29
↑	2-8	50	12	0A	!	11-0-8	250	52	2A
=	3-8	54	13	0B	$	11-3-8	254	53	2B
'	4-8	60	14	0C	*	11-4-8	260	54	2C
:	5-8	64	15	0D	]	11-5-8	264	55	2D
>	6-8	70	16	0E	;	11-6-8	270	56	2E
✓	7-8	74	17	0F	Δ	11-7-8	274	57	2F
+	12	100	20	10	Blank	No Punch	300	60	30
A	12-1	104	21	11	/	0-1	304	61	31
B	12-2	110	22	12	S	0-2	310	62	32
C	12-3	114	23	13	T	0-3	314	63	33
D	12-4	120	24	14	U	0-4	320	64	34
E	12-5	124	25	15	V	0-5	324	65	35
F	12-6	130	26	16	W	0-6	330	66	36
G	12-7	134	27	17	X	0-7	334	67	37
H	12-8	140	30	18	Y	0-8	340	70	38
I	12-9	144	31	19	Z	0-9	344	71	39
?	12-0-8	150	32	1A	≠	0-2-8	350	72	3A
.	12-3-8	154	33	1B	,	0-3-8	354	73	3B
)	12-4-8	160	34	1C	(	0-4-8	360	74	3C
[	12-5-8	164	35	1D	n	0-5-8	364	75	3D
<	12-6-8	170	36	1E	\	0-6-8	370	76	3E
≠	12-7-8	174	37	1F	⊣⊢	0-7-8	374	77	3F

CARRIAGE RETURN	664	155
TAB	764	175
BACK SPACE	670	156
UPPER CASE	470	116
LOWER CASE	770	136
INDEX	764	135
STOP CODE	400	100

* = Right 8 Bit Byte in Each Half-Word

** = Left 8 Bit Byte in Each Half-Word

APPENDIX 6

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

TABLE POWERS OF TWO

OCTAL-DECIMAL INTEGER CONVERSION

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

Octal-Decimal Integer Conversion Table (Sheet 1 of 7)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

Octal-Decimal Integer Conversion Table (Sheet 2 of 7)

4000 (Octal)	2048 (Decimal)
to	to
4777 (Octal)	2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 (Octal)	2560 (Decimal)
to	to
5777 (Octal)	3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

Octal-Decimal Integer Conversion Table (Sheet 3 of 7)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

Octal-Decimal Integer Conversion Table (Sheet 4 of 7)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

Octal-Decimal Integer Conversion Table (Sheet 5 of 7)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

Octal-Decimal Integer Conversion Table (Sheet 6 of 7)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

Octal-Decimal Integer Conversion Table (Sheet 7 of 7)

APPENDIX 8

HOLLERITH CARD CODES

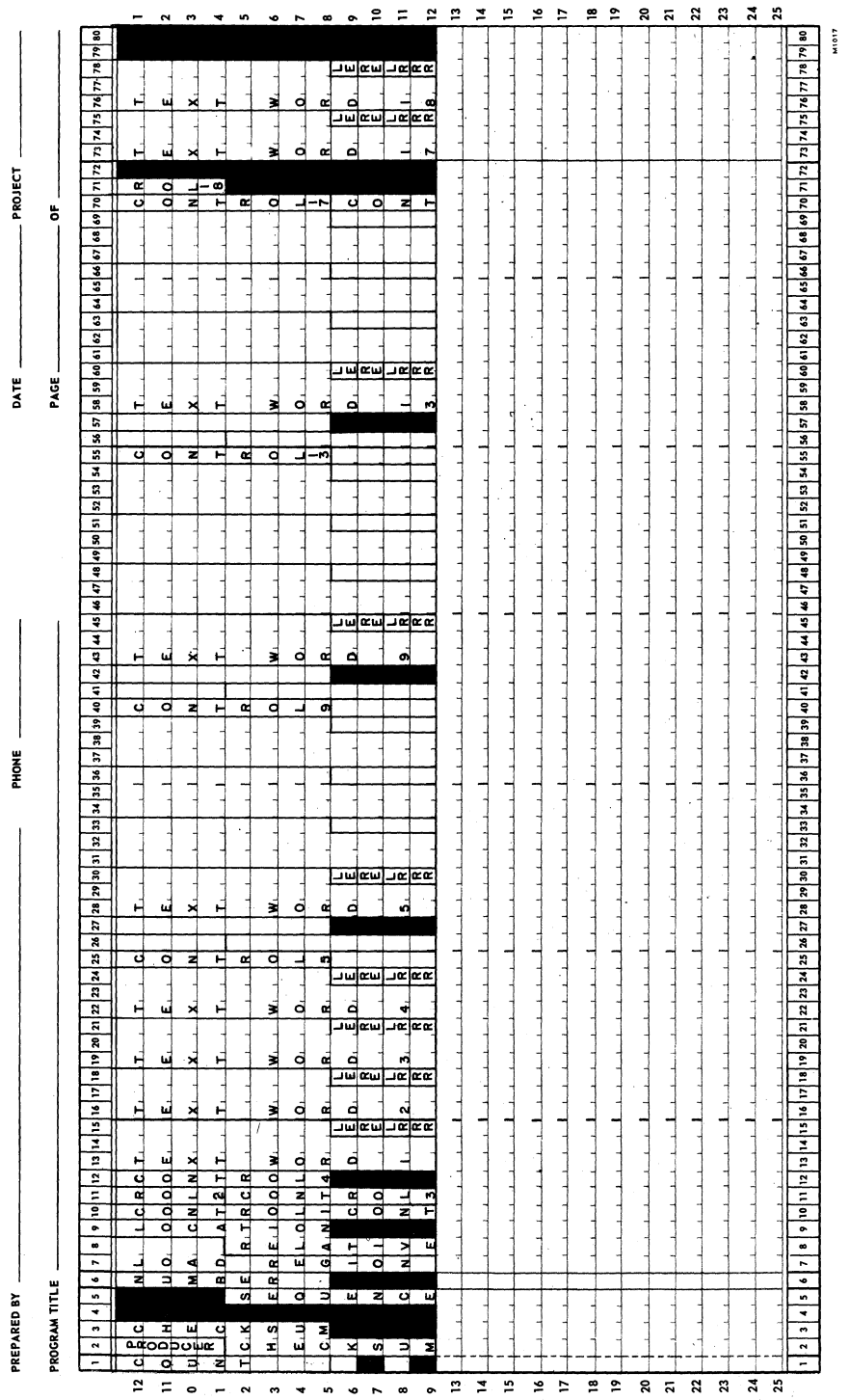
8400 Graphics	Standard Hollerith Card Codes	Internal Binary Code	8400 Graphics	Standard Hollerith Card Codes	Internal Binary Code
ø (ZERO)	0	00	-	11	40
1	1	01	J	11-1	41
2	2	02	K	11-2	42
3	3	03	L	11-3	43
4	4	04	M	11-4	44
5	5	05	N	11-5	45
6	6	06	O	11-6	46
7	7	07	P	11-7	47
8	8	10	Q	11-8	50
9	9	11	R	11-9	51
BLANK	8-2	12	!	11-0	52
=	8-3	13	$	11-8-3	53
'	8-4	14	*	11-8-4	54
:	8-5	15	]	11-8-5	55
>	8-6	16	;	11-8-6	56
✓	8-7	17	Δ	11-8-7	57
+	12	20	BLANK	NO PUNCH	60
A	12-1	21	/	0-1	61
B	12-2	22	S	0-2	62
C	12-3	23	T	0-3	63
D	12-4	24	U	0-4	64
E	12-5	25	V	0-5	65
F	12-6	26	W	0-6	66
G	12-7	27	X	0-7	67
H	12-8	30	Y	0-8	70
I	12-9	31	Z	0-9	71
?	12-0	32	≠	0-8-2	72
.	12-8-3	33	,	0-8-3	73
)	12-8-4	34	(	0-8-4	74
[	12-8-5	35	^	0-8-5	75
<	12-8-6	36	\	0-8-6	76
≠	12-8-7	37	≠	0-8-7	77

NOTES:

1. *In the binary mode data from a card reader is transferred without conversion. Each Card column is divided into two characters. Either 4- or 6-bit characters will be transferred depending upon the format option specified.*
2. *In the Hollerith mode data from the card reader (assumed to be in BCD codes) is automatically converted to collating codes.*
3. *Data to a card punch is presented in collating code - character by character - in the same manner as to the line printer.*
4. *No parity bit is presented with data to/from card equipment. Error and code validity checks are performed at the respective card device.*

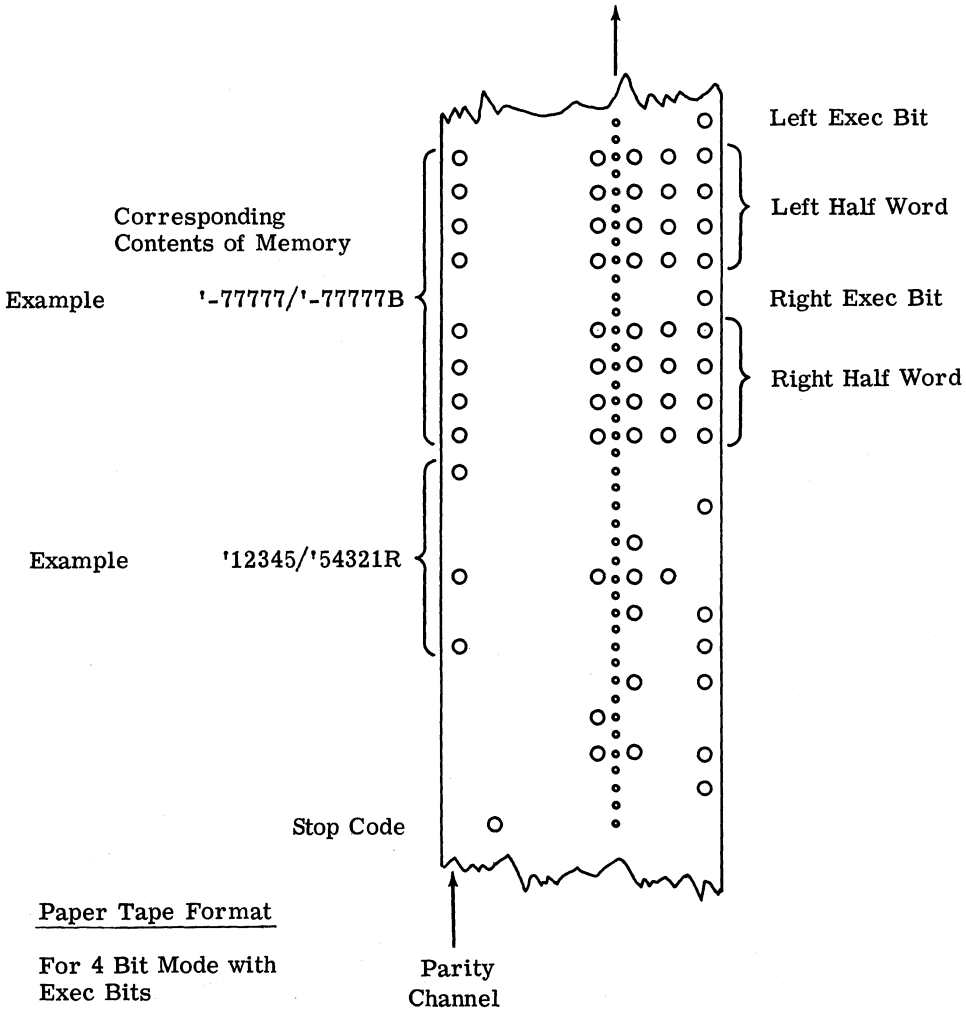
APPENDIX 9

LINKING LOADER TEXT BINARY CARD FORMAT



APPENDIX 10

PAPER TAPE FORMAT



Paper Tape Format

For 4 Bit Mode with
Exec Bits

APPENDIX 11

TWO'S COMPLEMENT ARITHMETIC

1. THE TWO'S COMPLEMENTS SYSTEM

In the sign-magnitude system, the sign bit has a value of -1 or +1. The sign is *multiplied* by the value represented by the magnitude bits to form the implied number.

In two's complements, the sign bit has a variable negative weight, depending on the position of the binary point. The weight of the sign is *added* to the value represented by the magnitude bits to form the implied number.

The Representation of a 2's Complements Number

If b_s is the content of the sign bit, and b_j the content of any other bit in position j, the number N represented by the bit configuration is given by

$$N = -b_s \cdot 2^n + \sum_{j=1}^{m} b_j 2^{n-j}$$

Note that the summation in the equation is *always* positive.

2. RANGE OF NUMBERS

Let the binary point be immediately to the right of the sign bit (n = o). Then m bits can represent $-1 \leqq N \leqq 1-2^{-m}$. Note that -1 is inside the range ($b_s = 1$, $b_j = 0$), but +1 is not.

3. TRUNCATION AND ROUND-OFF

In the natural, real number system, a given number X can assume any value on the *continuous* range $-\infty \leqq x \leqq$. Natural numbers have *infinite precision* (i.e., their representation requires an infinite number of digits). The numbers that the digital computer must deal with are *finite-precision, quantized* numbers. We shall refer to these as "digital" or "synthetic" numbers. Digital numbers are evidently a function of the continuous natural argument X. Two such consistent functions are:

$y = \lfloor x \rfloor$ ("least integer in x", L.I.)

$y = \lceil x \rceil$ ("greatest integer in x", G.I.)

These functions (Figure 1) are defined as follows: In L.I., if n is an integer, $y = n-1$ for $n-1 \leqq x < n$. In G.I., $y = n$ for $n-1 \leqq x < n$. A third, less consistent scheme, also shown in Figure 2, is defined by $y = n-1$ for $n-1 \leqq x < n$, $x > o$, but $y = n$ for $n-1 < x < n$, $x < o$.

In the discussion above and in the following, we assume y to have integer values only (binary point to the right of least significant bit of A register, say). This does not detract from the generality of the results. To convert the y values to fractions, simply multiply by 2^{-15}.

In the 8400, the "lease integer" functions, $y = \lfloor x \rfloor$, is used, as this is most compatible with two's complement notation. The "a symetry" in the range of numbers, discussed in Section 2.1 can now be seen as a direct consequence of the L.I. function.

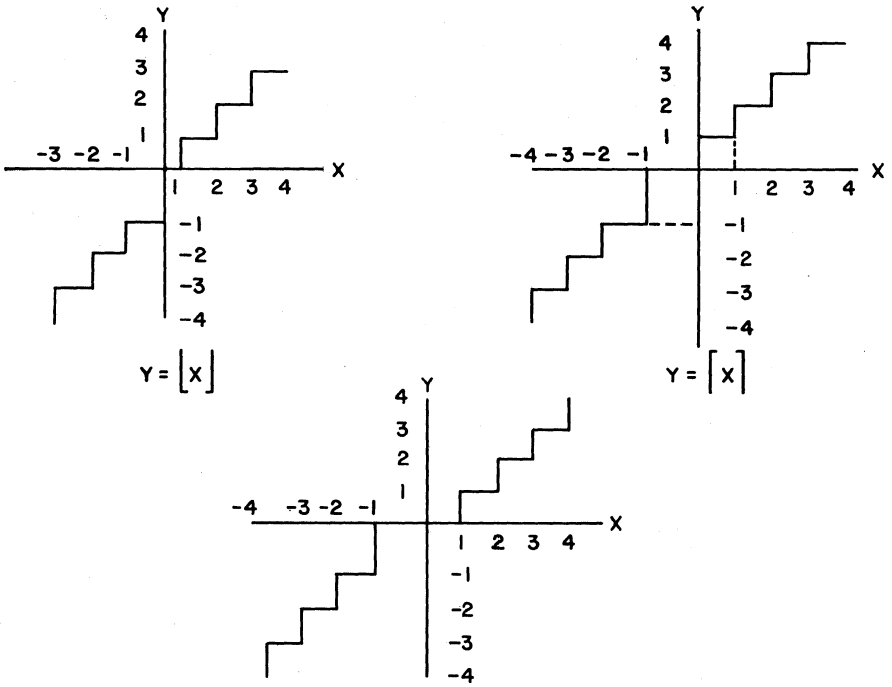


Figure A11.1. Digital Numbers as Functions of Continuous Natural Numbers

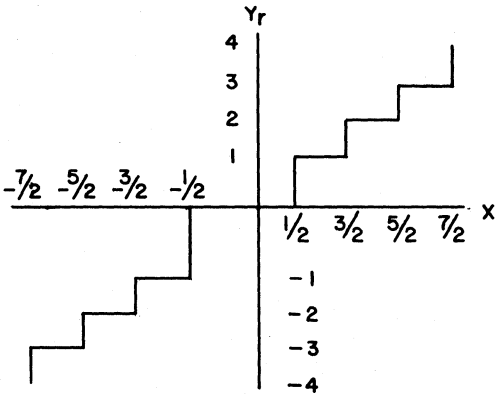


Figure A11.2. Rounded Digital Numbers as a Function of Natural Numbers

Truncation

The implication of the $y = \lfloor x \rfloor$ function is that when an 8400 number is *truncated,* it automatically assumes its "least integer" value. Thus dividing 1 (bit 15 = 1, all others zero) by 2 (ASH 1) results in the natural number 0.5., which, truncated, becomes zero. On the other hand, the number -1 ($b_s = b_j = 1$ in the first example, vis all bits high) when divided by 2 results in the natural number -0.5, which the 8400 truncates to -1.

Round-Off

To *round* a number, one can use the SR instruction. For example, doing

```
EASH    k
SR      0
```

will divide the number in the A register by 2^k and round it. The rounded function y_r appears as in Figure 2.

The function y_r is defined as $y_r = \lfloor |x| - 1/2 \rfloor$. Thus 0.5 yields 0 for the truncated result, but 1 for the rounded result and -0.5 yields -1 truncated and 0 rounded. Note that the error committed in round-off is $\epsilon_r \leqq 0.5$.

Programming for G.I. and "Sign-Magnitude"

If the programmer needs $y = \lceil x \rceil$, he can simply do:

(operation resulting in A:AE)

```
AD = 1
ST = GI
```

It is also quite easy to get the function (sgn x) $\lfloor |x| \rfloor$ (Figure 2) by doing:

(operation resulting in A:AE)

```
        EXL     ADD
        ST      SAVE
        =
ADD AD          = 1
```

Note, in particular, that the IST instruction will truncate (i.e., get the "least integer in x") a floating-point number prior to storing. The schemes just discussed of obtaining other types of truncation are particularly useful here. For example, a floating point number can be "greatest-integer" truncated by

```
        FCA     NUMBER
        IAD     = 1
        IST
```

or "sign-magnitude" truncated by

```
        FCA     NUMBER
        EXL     ADD
        IST
        =
ADD IAD         = 1
```

4. SHIFTS

Left shifts must have zero-fill for low order bits. Right shifts must have sign-fill for high order bits. The need for sign fill in right shifts is that a right shift of k is a division by 2^k of the magnitude bits if they are zero filled in the vacated high order bits. As the sign bit is an additive value in two's complement it must also

be divided by two and added to the shifted mantissa. Note that $-2^n/2^k = -2^{n-k}$ which is represented in two's complement as a one in the sign bit position followed by k one's. Adding this value in gives the appearance of a sign fill in vacated high order bits.

5. OVERFLOWS

An overflow indicates that the result of an arithmetic operation exceeds the permissible range of the computer. In two's complement, the overflow V is defined as the exclusive -OR of the carry C and the drop-off D:

$$V = C\bar{D} + \bar{C}D$$

where D is a carry *from bit position S*. That this definition is correct can be quickly verified by noting that, if a number has $b_s = b_1 = 1$, it cannot be more positive then -0.5, so it is well within the permissible range *and will remain so* when shifted left once (multiplied by 2). Thus, a carry accompanied by a drop-off is *not* an overflow, and, of course, neither is C = D = 0. A carry without a drop-off signifies a positive overflow, while a drop-off without a carry indicates a negative overflow.

6. MULTIPLE PRECISION

An extended precision number X can be regarded as a single number. In fact, all legitimate extended (E) and double-floating (D) instructions of the 8400 regard numbers in this way; that is, the computer effectively *ignores* the sign of the AE (AD) registers in all legitimate operations (including properly-programmed combat subroutines).

MP and ASH, EASH, *reset* the sign of AE, ECA, ECS, DCA, DCAU, DCS, and DCSU set the sign bit of the AE(AD) as the corresponding memory bit was set. EST and DST set to corresponding sign bit in memory to the value of the sign bit of AE(AD). All other arithmetic operations ignore this bit.

When X is to be treated as *two separate single precision* numbers, their *sum* must clearly add up to X. That is,

$$X = (A) + (AE) \cdot 2^{-15}$$

It should be clear that, unless *the sign of AE is zero,* the equation is not satisfied.

Similar considerations show that the sign of the AD register must be zero if the content of that register is to be operated on by F-type instructions.

Note that a single-precision number which is to become the least-significant portion of an extended-precision number is treated as follows

```
CA      LO
EASH    15
```

To convert a single precision number to extended precision you do

```
CA      HO
LDAE    = 0
```

Converting a single precision floating point number to double can be done by clearing the AD register:

```
        FCAU    NUMBER
        FMPU    = '40000/1
        DST
```

or

```
        $DCAU   = 0
         FCAU   $
         DST
```

Operation Time is in Microseconds, and Includes Instruction Fetch (1) Arithmetic	Operand Address		32 Bit Flt. Pt. (Prefix F)	56 Bit Flt. Pt. (Prefix D)	16 32 Bit Integer (Prefix I)	16 Bit Fix. Pt. (Prefix [Blank])	32 Bit Fix Pt. (Prefix E)	16 Bit Index (Prefix X)
CA, AD CS, SB	HIGH SPEED REGISTERS (2)	♦	3.89	NA	3.89	3.06	NA	3.06
		=	4.17	4.72	4.17	3.33	3.89	3.33
		$	4.17	4.72	4.17	3.33	3.89	3.33
	MEMORY		5.28	8.06✱*	5.28	4.45	5.28*	4.45
MP	HIGH SPEED REGISTERS	♦	6.67	By Sub	6.67	5.28	By Sub	By Sub
		=	6.94	By Sub	6.94	5.56	By Sub	By Sub
		$	6.94	By Sub	6.94	5.56	By Sub	By Sub
	MEMORY		8.06	By Sub	8.06	6.67	By Sub	By Sub
CD, DV	HIGH SPEED REGISTERS	♦	10.00	By Sub	10.00	7.78	By Sub	By Sub
		=	10.28	By Sub	10.28	8.06	By Sub	By Sub
		$	10.28	By Sub	10.28	8.06	By Sub	By Sub
	MEMORY		11.39	By Sub	11.39	9.17	By Sub	By Sub
CP	HIGH SPEED REGISTERS	♦	3.89	By Sub	3.89	3.33	By Sub	3.33
		=	4.17	By Sub	4.17	3.61	By Sub	3.61
		$	4.17	By Sub	4.17	3.61	By Sub	3.61
	MEMORY		5.28	By Sub	5.28	4.72	By Sub	4.72
ST	MEMORY		4.17	6.11	5.00	4.17	4.17	4.17
SR	HIGH SPEED REGISTERS	♦	3.06	By Sub	3.33	2.78	By Sub	By Sub
	MEMORY		4.72	By Sub	5.00	4.45	By Sub	By Sub

MODIFIERS

Indexing OP M, (X)	Indirect Addressing OP * M	Save $ OP M
For each level of indexing Add .56 if H.S. Register Addressed Add .28 if MEM addressed except ✱ Add .56	For each level of indirect addressing Add 2.00	For a prefixed Save *no additional time required. Add 0 Add .28 except if Save also addressed ($ OP$) Add .56
Add .56	Add 2.00	Add .56

NOTES:

1. Time shown for arithmetic operations is minimum execution time and does not include pre-alignment or post normalization. For each pre-alignment or post normalization-add 0.28 μ sec.

EXAMPLES:

$DST M_{ov}, X = 5.84

DST M_{ov}, X	= 4.72
$	= .56
X	= .56
	5.84

$CA = 3.X = 3.90

CA = 3	= 3.06
$	= .28
X	= .56
	3.90

$FMP $, X = 8.06

FMP $	= 6.94
$ (OP $)	= .56
X	= .56
	8.06

2. High Speed Registers Are:
   - ♦ Self addressing accumulator
   - = Immediate addressing (16 Bit Operand field of instruction Reg.)
   - $ Save Register