File No. 1130-01
Form A26-5881-1

Systems Reference Library

IBM 1130 Functional Characteristics

This manual provides basic programming and operating information for the 1130 Computing System. The functional aspects of the System are explained in detail, and the operational characteristics are described in terms of program instructions, input/output operations, and Central Processing Unit console displays and functions. Intended as a reference manual, the material presented assumes some prior knowledge of stored program computers.

PREFACE

Intended primarily as a reference tool, this manual presents information on a level that requires a minimum of prior knowledge of stored-program computers. Some of the terms used in the following pages, however, may be unfamiliar to the inexperienced. To avoid lessening the value of the book as a reference tool, explanations of terms are confined to the context of their use.

The IBM publication, Introduction to IBM Data Processing Systems (Form F22-6517) provides an excellent introduction to the stored-program computer.

This publication replaces and makes obsolete the previous edition, IBM 1130 Computing System Principles of Operation (Form A26-5881-0). Significant changes have been made throughout the manual.

Copies of this and other IBM publications can be obtained through IBM Branch Offices. Comments concerning the contents of this publication may be addressed to: IBM, Product Publications Department, San Jose, Calif. 95114

CONTENTS

INTRODUCTION

The IBM 1130 Computing System provides the capacity and versatility to accomplish the engineering and scientific computations that formerly were possible only with large computer systems. The 1130 fulfills the "general purpose" requirements of these areas with computing power well above previous systems in the same cost range. The 1130 Computing System can also handle supporting commercial data processing applications.

The design of the 1130 System is oriented to the operator. Only a minimum of training and experience with computing systems is necessary to make the 1130 usable by engineering and research personnel for solutions to problems in individual projects. In addition, programs and programming systems, supplied by IBM, relieve the user of detailed programming and provide for the statement of problems in familiar language.

The compact, easily-operated 1130 System features the IBM 1131 Central Processing Unit (CPU) with core storage capacity of 4096 or 8192 16-bit words. An additional 512,000 words of storage is available on-line with the disk storage feature of the 1131 Model 2. Disk storage provides random or sequential access to data. The interchangeable disk cartridge places the required information at the disposal of the system and allows virtually unlimited off-line storage capacity. The CPU also includes a console with data displays and switches for operator control, a keyboard for data entry, and a console printer.

The basic 1130 System consists of the CPU and either the IBM 1442 Card Read-Punch or the IBM 1134 Paper Tape Reader and the IBM 1055 Paper Tape Punch. To either of these configurations can be added: disk storage, additional (4096 words) core storage (total 8192), an IBM 1627 Plotter, and the IBM 1132 Printer. Paper tape or card units not already a part of the basic system can also be added.

The IBM 1130 Computing System applications are varied and include some aspect of every industrial, financial, and governmental operation. In the aerospace, construction, engineering, and fabrication and assembly industries, the 1130 can be used for complex mathematical problems, operation analysis and scheduling, estimating, equipment and machine design, simulation, and job cost analysis.

In the processing industries, blending formulas, material balance, material evaluation, forecasting, and unit operations are a few of the applications suitable for the 1130.

Also, in many areas of the transportation, marketing, financial, insurance, utilities, and distribution fields, the 1130 System provides capability not previously available in a system of its size.

FUNCTIONAL CHARACTERISTICS

The ability of the IBM 1131 Central Processing Unit to ask for and accept input data, perform the calculations required, and produce the output results desired, is due to the many functional elements of the machine. Each of these elements is explained in this section, and from these descriptions the CPU emerges as the sum of its parts — the nerve center of the computing system.

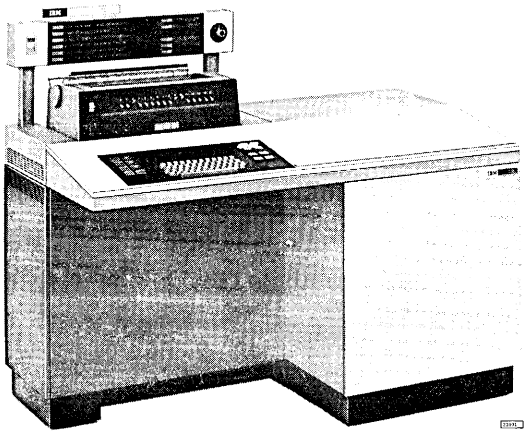


Figure 1. IBM 1131 Central Processing Unit *

The descriptions that follow concern the storage of data and program instructions, the formats in which data and instructions are stored and used, functions of CPU registers, the fundamental arithmetic operations and how they are performed, and the aspects of addressing core storage and attached input/ output (I/O) units.

CORE STORAGE

The 1131 main storage uses magnetic cores for data and program instruction storage. Core storage capacity is 4096 16-bit words; an additional 4096 words are available as a special feature. Each 16-bit word has 2 additional bits, called Parity bits, which are used for internal data checking only.

The main storage memory cycle (the time required to place a word in core storage or retrieve it from core storage) is 3.6 μsec.

Addressing

Each 16-bit word in core storage is locatable through an address that specifies the position of the word. Addresses range from 0000 to 4095 or 8191, according to the capacity installed. The high-order address is contiguous with the low-order address, which provides for "wrap-around" addressing. This means that in sequential processing of addresses, 4095 or 8191 is followed by 0000 without further specification by the CPU.

Reserved Storage Locations

The following are core storage decimal addresses reserved for specific purposes and not available for general data storage.

Tag Bits	Core Storage Address	Description
00	--	Displacement
01	0001	Index Register 1
10	0002	Index Register 2
11	0003	Index Register 3
--	0008 - 0013	Interrupt Addresses
--	0032 - 0039	Printer Scan Field

20246

The use of each of the foregoing addresses is described in the appropriate subsequent section of this manual.

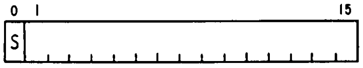
DATA FORMAT

Data in the 1131 CPU is in fixed-point binary form. Each number is treated as a signed integer; positive numbers are in true binary with a sign of 0, and negative numbers must be stored and operated upon in 2's complement form with a sign bit of 1. Complementing is done by inverting each bit of the number (including the sign bit) and adding 1 to the low-order bit. The following example illustrates this.

Positive number	0001101001001100
Inverted	1110010110110011
Add 1	1
Resulting negative number	1110010110110100

Data is stored as either a single precision word or a double precision word. A single precision data word comprises 16 bits; bit positions are numbered 0 to 15 from left to right. The high-order bit (0) is the sign position.

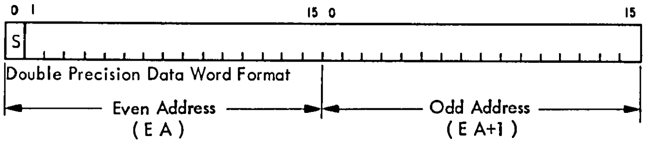
*NOTE: The illustrations in this manual have a code number in the lower corner. This is a publishing control number and is not related to the subject matter.



Single Precision Data Word Format

20070

The largest base-10 (decimal) values of single precision words are +32,767 and —32,768. A double precision data word contains 32 bits, and is composed of two sequential single precision words. The high-order bit (0) is the sign position.



20071A

A double precision data word is addressed by the leftmost word, which must have an even address.

The highest base-10 values of double precision data words are +2,147,483,647 and —2,147,483,648. The largest positive number (2^{31}—1) is one less than the largest negative number (2^{31}) because the sign (0 for plus, 1 for minus) is, arithmetically, part of the number.

All CPU storage is in binary form, and internal addressing and console displays are in 16-bit binary notation. Because of the ease of operation with 16-bit words in the hexadecimal number system (base—16), all programming systems for the IBM 1130 Computing System use this notation. Figures 2 and 3 show comparable values of decimal and hexadecimal number systems. Space obviously does not permit a complete listing; therefore, the value of each power of 2 through 2^{15}—1 and 2^{31}—1 are shown. The binary and hexadecimal number systems are described in Appendix A. Tables for conversion of decimal and hexadecimal numbers are in Appendix F. Hexadecimal numbers greater than the range of the table can be converted to decimal numbers by multiplying their factors. For example, 2 and 10 are factors of 20. As shown in the conversion table, the decimal equivalents of these two hexadecimal factors are 2 and 16. Their product is, of course, 32 which is the decimal equivalent of hexadecimal 20.

INSTRUCTION FORMATS

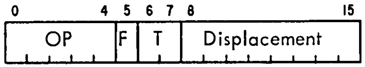
Program instructions in the 1130 System are in either short or long format.

Positive Binary Values	Powers of 2	Absolute Values		Negative Binary Values
Bit Positions 11 1111 0123 4567 8901 2345		Decimal Notation Base-10	Hexa-decimal Notation Base-16	Bit Positions 11 1111 0123 4567 8901 2345
0000 0000 0000 0000	-	0	0	No negative zero
0000 0000 0000 0001	0	1	1	1111 1111 1111 1111
0000 0000 0000 0010	1	2	2	1111 1111 1111 1110
0000 0000 0000 0100	2	4	4	1111 1111 1111 1100
0000 0000 0000 1000	3	8	8	1111 1111 1111 1000
0000 0000 0001 0000	4	16	10	1111 1111 1111 0000
0000 0000 0010 0000	5	32	20	1111 1111 1110 0000
0000 0000 0100 0000	6	64	40	1111 1111 1100 0000
0000 0000 1000 0000	7	128	80	1111 1111 1000 0000
0000 0001 0000 0000	8	256	100	1111 1111 0000 0000
0000 0010 0000 0000	9	512	200	1111 1110 0000 0000
0000 0100 0000 0000	10	1,024	400	1111 1100 0000 0000
0000 1000 0000 0000	11	2,048	800	1111 1000 0000 0000
0001 0000 0000 0000	12	4,096	1,000	1111 0000 0000 0000
0010 0000 0000 0000	13	8,192	2,000	1110 0000 0000 0000
0100 0000 0000 0000	14	16,384	4,000	1100 0000 0000 0000
0111 1111 1111 1111	-	32,767	7,FFF	1000 0000 0000 0001
No positive equivalent	15	32,768	8,000	1000 0000 0000 0000

20072A

Figure 2. Value Ranges, Single Precision Word

Short Instruction Format



Short Instruction Format

20074

OP (Operation) Code. These five bits specify the operation performed. Specific operations are described in the OPERATION section of this manual.

F (Format). The F bit controls the instruction format. A zero (0) indicates a short instruction format, a one (1) designates a long instruction format.

T (Tag). These two bits specify the instruction counter or index register (XR) to be used for Effective Address generation.

Displacement. The data contained in these eight bits is added to the data in the instruction counter or index register specified by the tag bits to form the Effective Address (EA). (The EFFECTIVE ADDRESS GENERATION section of this manual describes this and other aspects of address modification.) If the displacement amount is negative it is in 2's complement, and the sign, in bit position 8, is maintained in the resulting high-order position when the Displacement is expanded to 16 bits for address modification.

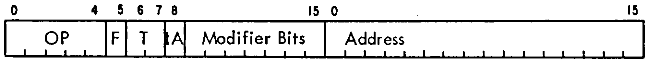
NOTE: Displacement bits have other uses; for example, bits 8 and 9 are used as shift modifiers, etc.

Positive Binary Values	Powers of 2	Absolute Values		Negative Binary Values
Bit Positions 11 1111 1111 2222 2222 2233 0123 4567 8901 2345 6789 0123 4567 8901		Decimal Notation Base - 10	Hexidecimal Notation Base - 16	Bit Positions 11 1111 1111 2222 2222 2233 0123 4567 8901 2345 6789 0123 4567 8901
0000 0000 0000 0000 0000 0000 0000 0000	-	0	0	No negative zero
0000 0000 0000 0000 0000 0000 0000 0001	0	1	1	1111 1111 1111 1111 1111 1111 1111 1111
0000 0000 0000 0000 0000 0000 0000 0010	1	2	2	1111 1111 1111 1111 1111 1111 1111 1110
0000 0000 0000 0000 0000 0000 0000 0100	2	4	4	1111 1111 1111 1111 1111 1111 1111 1100
0000 0000 0000 0000 0000 0000 0000 1000	3	8	8	1111 1111 1111 1111 1111 1111 1111 1000
0000 0000 0000 0000 0000 0000 0001 0000	4	16	10	1111 1111 1111 1111 1111 1111 1111 0000
0000 0000 0000 0000 0000 0000 0010 0000	5	32	20	1111 1111 1111 1111 1111 1111 1110 0000
0000 0000 0000 0000 0000 0000 0100 0000	6	64	40	1111 1111 1111 1111 1111 1111 1100 0000
0000 0000 0000 0000 0000 0000 1000 0000	7	128	80	1111 1111 1111 1111 1111 1111 1000 0000
0000 0000 0000 0000 0000 0001 0000 0000	8	256	100	1111 1111 1111 1111 1111 1111 0000 0000
0000 0000 0000 0000 0000 0010 0000 0000	9	512	200	1111 1111 1111 1111 1111 1110 0000 0000
0000 0000 0000 0000 0000 0100 0000 0000	10	1,024	400	1111 1111 1111 1111 1111 1100 0000 0000
0000 0000 0000 0000 0000 1000 0000 0000	11	2,048	800	1111 1111 1111 1111 1111 1000 0000 0000
0000 0000 0000 0000 0001 0000 0000 0000	12	4,096	1,000	1111 1111 1111 1111 1111 0000 0000 0000
0000 0000 0000 0000 0010 0000 0000 0000	13	8,192	2,000	1111 1111 1111 1111 1110 0000 0000 0000
0000 0000 0000 0000 0100 0000 0000 0000	14	16,384	4,000	1111 1111 1111 1111 1100 0000 0000 0000
0000 0000 0000 0000 1000 0000 0000 0000	15	32,768	8,000	1111 1111 1111 1111 1000 0000 0000 0000
0000 0000 0000 0001 0000 0000 0000 0000	16	65,536	10,000	1111 1111 1111 1111 0000 0000 0000 0000
0000 0000 0000 0010 0000 0000 0000 0000	17	131,072	20,000	1111 1111 1111 1110 0000 0000 0000 0000
0000 0000 0000 0100 0000 0000 0000 0000	18	262,144	40,000	1111 1111 1111 1100 0000 0000 0000 0000
0000 0000 0000 1000 0000 0000 0000 0000	19	524,288	80,000	1111 1111 1111 1000 0000 0000 0000 0000
0000 0000 0001 0000 0000 0000 0000 0000	20	1,048,576	100,000	1111 1111 1111 0000 00000000 0000 0000
0000 0000 0010 0000 0000 0000 0000 0000	21	2,097,152	200,000	1111 1111 1110 0000 0000 0000 0000 0000
0000 0000 0100 0000 0000 0000 0000 0000	22	4,194,304	400,000	1111 1111 1100 0000 0000 0000 0000 0000
0000 0000 1000 0000 0000 0000 0000 0000	23	8,388,608	800,000	1111 1111 1000 0000 0000 0000 0000 0000
0000 0001 0000 0000 0000 0000 0000 0000	24	16,777,216	1,000,000	1111 1111 0000 0000 0000 0000 0000 0000
0000 0010 0000 0000 0000 0000 0000 0000	25	33,554,432	2,000,000	1111 1110 0000 0000 0000 0000 0000 0000
0000 0100 0000 0000 0000 0000 0000 0000	26	67,108,864	4,000,000	1111 1100 0000 0000 0000 0000 0000 0000
0000 1000 0000 0000 0000 0000 0000 0000	27	134,217,728	8,000,000	1111 1000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000	28	268,435,456	10,000,000	1111 0000 0000 0000 0000 0000 0000 0000
0010 0000 0000 0000 0000 0000 0000 0000	29	536,870,912	20,000,000	1110 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000	30	1,073,741,824	40,000,000	1100 0000 0000 0000 0000 0000 0000 0000
0111 1111 1111 1111 1111 1111 1111 1111	-	2,147,483,647	7F,FFF,FFF	1000 0000 0000 0000 0000 0000 0000 0001
No positive equivalent	31	2, 147,483,648	80,000,000	1000 0000 0000 0000 0000 0000 0000 0000

20073A

Figure 3. Value Ranges, Double Precision Word

Long Instruction Format



Long Instruction Format

20075A

The first eight bit positions of the long instruction are the same as the short format. The remaining bit positions of this double precision word are used as follows.

IA (Indirect Address). A zero indicates a direct address (contained in the second word). A 1-bit in this position designates an indirect address, which is described in the EFFECTIVE ADDRESS GENERATION section.

Modifier Bits. Bit positions 9 through 15 have various uses as modifiers and are described under the applicable instructions.

Address. These 16 bits contain the address which may be used in its current form or modified by indirect addressing and/or EA modification.

REGISTERS

The CPU has auxiliary storage areas, called registers, that are used to store data during the performance of operations directed by the stored program. Each register has a distinct purpose and is concerned with a specific type of data. Closely interrelated, they provide the CPU with the necessary functions to provide the results required.

Index Registers

Index registers are located in core storage and are used to contain data added to an instruction to provide an effective address. In a short instruction, the amount in the displacement field of the instruction is added to the amount in the index register specified by the tag bits (6 and 7). The result becomes the effective address used by the instruction in the operation specified by the OP code. These operations and the functions of the EA are explained more fully in the OPERATION section.

Register Number	Instruction Code in Bits 6 and 7	Core Storage Location
1	01	0001
2	10	0002
3	11	0003

Machine Registers

The ten registers in the CPU are basic to the system and are functional elements of the CPU. Each register operates as necessary to enable the CPU to provide the results specified by the program. The abbreviation for each register name is the designation by which it is usually identified.

ACC (Accumulator). This 16-bit register contains the result of an arithmetic operation. It can be loaded from or stored in core storage, shifted right or left, and otherwise manipulated by specific arithmetic and logical instructions.

EXT (Accumulator Extension). This 16-bit register is the low-order extension of the ACC. It is used during multiply and divide operations, shifting of the ACC and EXT, and double-word arithmetic.

TAR (Temporary Accumulator). This 16-bit register is the image of the ACC and is used to store the contents of the ACC during effective address computation.

AFR (Arithmetic Factor Register). This 16-bit register holds one operand during arithmetic and logical operations. (The other operand is provided by the ACC.)

SBR (Storage Buffer Register). This 16-bit register is the buffer between the CPU and core storage, and every word of data transferred into or out of core storage passes through the SBR.

SAR (Storage Address Register). This 14-bit register contains the address pertaining to each reference to a core storage word.

IAR (Instruction Address Register). This 14-bit register holds the address of the next sequential instruction.

OP (Operation Register). This 5-bit register holds the OP code of the instruction being performed.

TAG (Operation Tag Register). This 3-bit register contains the F and T bits of the instruction. It controls the instruction length and selects the index register.

CCC (Cycle Control Counter). This 6-bit register is used primarily to count CPU cycles and control shift operations.

ARITHMETIC FUNCTIONS

The arithmetic functions of the 1131 CPU are addition, subtraction, multiplication, and division. Results of arithmetic operations are algebraic. The correct sign is maintained as part of each operation.

Data is stored in the CPU in binary form; positive quantities have a plus sign (0) in the high-order position of the word and negative quantities have a minus sign (1) in the high-order position. Negative numbers are stored and operated upon in 2's complement form.

Addition and subtraction can be done in either single or double precision. Multiplication operates with single precision multiplier and multiplicand and provides a double precision product. In division, the dividend is double precision and the divisor and quotient are single precision.

Each arithmetic operation is described in detail under the specific program instruction in the OPERATION section.

Indicators

The two indicators associated with the ACC are the Overflow and Carry indicators. Each can be turned on irrespective of the other. The conditions of the indicators are explained under each instruction.

Carry Indicator. The Carry indicator is ON if the last position shifted out of the high-order position of the ACC contains a 1-bit. This indicator is reset for each add, subtract, or shift-left operation; it facilitates multiple precision (beyond double word) arithmetic.

Overflow Indicator. This indicator is turned on by an add, subtract, or divide operation when the result exceeds the capacity of the ACC or by a Load Status instruction in which the word at the EA has a one in bit position 15. The Overflow indicator can be turned off only by program test, a Store Status instruction, or a Load Status instruction in which the word at the EA has a zero in bit position 15.

EFFECTIVE ADDRESS GENERATION

As has been noted previously, the location of a 16-bit single precision word or a 32-bit double precision word is denoted by a binary address. The range of addresses, expressed in decimal numbers, is 0000 through 4095 or 8191. Most of the program instructions, which are explained in the OPERATION section, instruct the CPU to obtain the data at a specified location and perform a certain operation on it. For example, an Add instruction could say, in effect, add the amount stored at location 1904 to the amount in the accumulator and leave the result in the accumulator.

The location 1904 is the effective address of the data referred to by the instruction.

It is part of the versatility of the 1131 CPU that the address in the instruction being executed can be modified as a specific occasion requires. As the result of a particular computation, for example, one of several courses may be indicated. Computation of the effective address of the location of the next instruction or of the next data worked on allows the CPU to proceed according to the predetermined course of action. The factors involved in computing the effective address are described in the following paragraphs.

Short Instruction

The short instruction displacement field contains the amount that is added to a specified figure to achieve the EA (effective address). The Tag bits of the instruction designate the other factor. See Figure 4.

IAR. Tag bits of 00 indicate that the displacement is added to the IAR to form the effective address. The IAR contains the address of the next or immediately following instruction.

	F = 0 (Direct Addressing)	F = 1, IA = 0 (Direct Addressing)	F = 1, IA = 1 (Indirect Addressing)
T = 00	EA = Disp + IAR	EA = Add	EA = C/Add
T = 01	EA = Disp + XR1	EA = Add + XR1	EA = C/Add + XR1
T = 10	EA = Disp + XR2	EA = Add + XR2	EA = C/Add + XR2
T = 11	EA = Disp + XR3	EA = Add + XR3	EA = C/Add + XR3

Disp = Contents of Displacement field of instuction.
Add = Contents of Address field of instruction.
C = Contents of Location specified by Add or Add + XR.

20111B

Figure 4. Determination of Effective Address

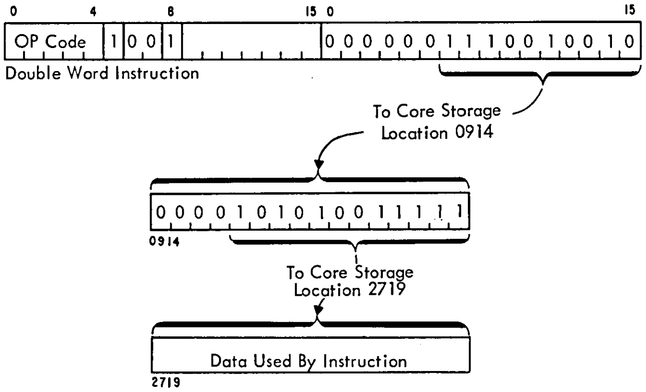
Index Registers. The three Index registers can also be used to modify the displacement to form the effective address. Tag bits of 01, 10, or 11 designate Registers 1, 2, or 3. Again, the contents of the specified register, added to the displacement, form the effective address.

Long Instruction

Long instructions are modified in much the same way as short instructions with the added versatility of indirect addressing. See Figure 4.

Direct Addressing. In the long instruction, a direct address is indicated by a 0 in the IA field. The effective address is governed by the contents of the Tag field. Tag bits of 00 indicate that the Address field of the instruction contains the effective address, which requires no modification. Tag bits of 01, 10, or 11 specify that the contents of the Address field are added to Index Register 1, 2, or 3, respectively, to form the EA.

Indirect Addressing. A 1-bit in the IA field of the instruction signifies that addressing is indirect, i.e., the Address field of the instruction contains the address of the location in memory that contains the EA significant to the accomplishment of the instruction. The indirect address can be the contents of the Address field of the instruction (T = 00), or it can be modified by being added to an index register (T = 01, 10, or 11). As an example (Figure 5), the indirect address is 0914. The CPU goes to that address and finds the contents of the location to be 2719. The EA, then, is 2719. This provides one more level of modification of a given address and provides more versatility in programming for variations to the main line program.



20114A

Figure 5. Indirect Addressing

The IBM 1130 instruction set (Figure 6) comprises 24 individual instructions divided into five classes. Modifications of these instructions enable additional operations. In the descriptions that follow, the name of the instruction is followed by the mnemonic symbols, the binary representation of the operation code, and execution times.

LOAD AND STORE INSTRUCTIONS

Load ACC (LD-11000)

The contents of the memory location specified by the EA replace the contents of the ACC. The contents of the memory location are unchanged.

The Carry and Overflow indicators are not affected.

Load Double (LDD-11001)

The contents of the memory locations specified by the EA and EA + 1 are loaded into the ACC and EXT, respectively. This instruction provides a double-word load for use with double-word arithmetic. The EA must be even for correct operation. If the EA is odd, the contents of that location are loaded into both the ACC and EXT. The contents of the memory location are not changed.

Carry and Overflow indicators are not affected.

Store Accumulator (STO-11010)

The contents of the ACC replace the contents of the memory location specified by the EA. The contents of the ACC are not changed.

The Carry and Overflow indicators are not affected.

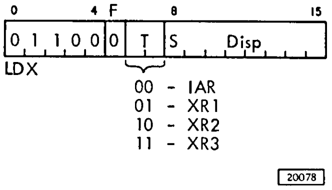
Store Double (STD-11011)

The contents of the ACC and EXT replace the contents of the memory locations specified by the EA and EA + 1. This instruction provides a double-word store for use with double-word arithmetic. The EA must be even for correct operation. If the EA is odd, the contents of the ACC are stored at the EA and the contents of the EXT are not stored. The contents of the ACC and EXT are not changed.

The Carry and Overflow indicators are not affected.

Load Index (LDX-01100)

The contents of the register specified by the Tag bits of the instruction are replaced by the data specified. In the short instruction (F = 0), the register is loaded with the Displacement. In the long instruction (F = 1) the register is loaded with the Address (IA = 0) or the contents of the memory location specified by the Address (IA = 1).



A Tag of 00 results in an unconditional branch to the address loaded into the IAR.

The Carry and Overflow indicators are not affected.

Store Index (STX-01101)

The contents of the register specified by the Tag bits are stored in the memory location specified by the EA. (See the table under Load Index for Tag bit codes.) The contents of the register are not changed.

The Carry and Overflow indicators are not affected.

Store Status (STS-00101)

The status of the Carry and Overflow indicators are stored in bits 14 and 15, respectively, of the word at the EA. Bits 0-7 of the storage word remain unchanged; bits 8-13 are reset to zeros. The status of each indicator is reflected by storing a 1-bit if the indicator is ON and a 0 if the indicator is OFF.

The Carry and Overflow indicators are reset as a result of the operation.

NOTE: The word in memory in which the status of the indicators is stored is normally the next Load Status instruction, the description of which follows.

Load Status (LDS-00100)

This instruction is always in the short format (F=O). The Carry and Overflow indicators are set to the

Instruction	Mnemonic	Binary OP Code	Execution Times (in microseconds)							
			Single Word (F = 0)				Double Word (F = 1)			
			T = 00		T = 01, 10, or 11		T = 00		T = 01, 10, or 11	
			Avg.	Max.	Avg.	Max.	Avg.[1]	Max.[1]	Avg.[1]	Max.[1]
Load and Store										
Load ACC	LD	11000	7.6	-	11.2	-	10.8	-	14.8	-
Load Double	LDD	11001	11.2	-	14.9	-	14.4	-	18.0	-
Store ACC	STO	11010	7.6	-	11.2	-	10.8	-	14.8	-
Store Double	STD	11011	11.2	-	14.9	-	14.4	-	18.0	-
Load Index	LDX	01100	4.5	-	7.2	-	7.2	-	11.8	-
Store Index	STX	01101	7.6	-	11.2	-	11.8	-	15.4	-
Load Status*	LDS [7]	00100	3.6	-	3.6	-	-	-	-	-
Store Status	STS	00101	7.6	-	11.2	-	10.8	-	14.8	-
Arithmetic										
Add	A	10000	8.0	13.0	11.7	16.6	11.2	16.2	15.3	20.3
Add Double	AD	10001	12.2	22.0	15.8	25.6	15.3	25.2	19.3	29.5
Subtract	S	10010	8.0	13.0	11.7	16.6	11.2	16.2	15.3	20.3
Subtract Double	SD	10011	12.2	22.0	15.8	25.6	15.3	25.2	19.3	29.5
Multiply	M	10100	25.7	40.0	29.3	43.6	29.3	43.6	32.9	47.2
Divide	D	10101	76.0	150.8	79.6	154.4	79.6	154.4	83.2	150.0
And	AND	11100	7.6	-	11.2	-	10.8	-	14.8	-
Or	OR	11101	7.6	-	11.2	-	10.8	-	14.8	-
Exclusive Or	EOR	11110	7.6	-	11.2	-	10.8	-	14.8	-
Shift Left* Modifier Bits 8 & 9:										
Shift Left ACC 00	SLA [7]	00010								
Shift Left ACC and EXT 10	SLT [7]	00010								
Shift Left and Count ACC 01	[8] SLCA [7]	00010								
Shift Left and Count ACC and EXT 11	[8] SLC [7]	00010	[3]	-	[4]	-	-	-	-	-
Shift Right* Modifier Bits 8 & 9:										
Shift Right ACC 00 or 01	SRA [7]	00011								
Shift Right ACC and EXT 10	SRT [7]	00011								
Rotate Right 11	RTE [7]	00011	[5]		[6]					
Branch										
Branch and Store IAR	BSI	01000	7.6	-	11.2	-	10.8[2]	-	14.8	-
Branch or Skip on Condition	BSC	01001	3.6	-	3.6	-	7.2[2]	-	11.2	-
Modify Index and Skip	MDX	01110	4.5	9.9	11.2	16.2	18.5	23.4	18.5	23.4
Wait*	WAIT [7]	00110 [9]	3.6	-	3.6	-	-	-	-	-
Input/Output										
Execute I/O	XIO [10]	00001	11.2	-	14.8	-	14.8	-	18.4	-

* Valid in short format only

NOTES:

1 Indirect addressing, where applicable, adds 3.6 μsec to execution time

2 If branch is taken

3 3.6 + .45(N-4)

4 7.2 + .45(N-4)

5 N >16: 3.6 + .45(N-19)
N < 16: 3.6 + .45(N-4)

6 N >16: 7.2 + .45(N-19)
N < 16: 7.2 + .45(N-4)
where N=number of position shifted

7 Indirect addressing not allowed

8 If T=00, functions as SLA or SLT

9 All unassigned OP codes are defined as Wait operations

10 If XIO Read or Write, add 3.6 μsec

Figure 6. 1130 Instruction Set and Execution Times conditions indicated by bits 14 and 15, respectively, of the instruction. A 1 sets the indicator to the ON condition; a 0 sets it to the OFF condition.

The Carry and Overflow indicators are set according to the bits in positions 14 and 15.

NOTE: The Load Status instruction is the word in memory in which the status of the indicators is stored by the previous (Store Status) instruction.

ARITHMETIC INSTRUCTIONS

Add (A-10000)

The contents of the memory location specified by the instruction are added algebraically to the contents of the ACC. Negative data is in 2's complement form. The sum replaces the contents of the ACC. The contents of the memory location remain unchanged.

The Overflow indicator is turned on if the sum is greater than the capacity of the ACC, $2^{15}-1$ or -2^{15}. If the indicator is ON when the overflow occurs, it is not changed.

The Carry indicator is set by a carry out of the high-order bit position of the ACC. The Carry indicator is dynamic and is conditioned for each ADD instruction.

Add Double (AD-10001)

The contents of the memory locations at EA and EA + 1 are added algebraically to the contents of the ACC and EXT. Negative data is in 2's complement form. This instruction provides double-word addition in which the ACC and EXT are considered as one 32-bit accumulator. The sum replaces the contents of the ACC and EXT; the contents of the memory locations are not changed. The EA must be even for correct operation. If the EA is odd, the contents of that location are added to both the ACC and EXT.

The Carry and Overflow indicators are affected as in the Add instruction (for the two word result, or course).

Subtract (S-10010)

The contents of the memory location specified by the instruction are directly subtracted from the contents of the ACC. The result replaces the contents of the ACC. The contents of the memory location are not changed.

The Carry and Overflow indicators are affected as in the Add instruction. The Carry indicator, if on, reflects a borrow condition.

Subtract Double (SD-10011)

The contents of the memory locations at EA and EA + 1 are subtracted from the contents of the ACC and EXT. This instruction provides double-word subtraction in which the ACC and EXT are considered one 32-bit accumulator. The difference replaces the contents of the ACC and EXT; the contents of the memory location are not changed. The EA must be even for correct operation. If the EA is odd, the contents of that location are subtracted from both the ACC and EXT.

The Carry and Overflow indicators are affected in the same way as in the Subtract instruction.

Multiply (M-10100)

The contents of the memory location specified by the instruction (the multiplicand) is multiplied algebraically by the contents of the ACC (multiplier). The 32-bit product replaces the contents of the ACC and EXT. Bit 15 of the EXT is the low-order bit, and bit 0 of the ACC is the high-order bit. Contents of the memory location are unchanged.

The Carry and Overflow indicators are not affected.

NOTE: The largest product that can be developed is 2^{30}, which results from multiplier and multiplicand of -2^{15}.

Divide (D-10101)

The contents of the ACC and EXT are considered a 32-bit, double word dividend, divided by the contents of the memory location specified by the instruction. The quotient replaces the contents of the ACC and the remainder, which carries the sign of the dividend, is placed in the EXT.

The Overflow indicator is turned ON by an attempt to divide by zero or by a quotient overflow, which occurs when the quotient exceeds the range of -2^{15} to $2^{15}-1$. An overflow causes the ACC and EXT to be left in an undefined state.

Logical AND (AND-11100)

The contents of the memory location specified by the instruction are ANDed, bit-by-bit, with the contents of the ACC; the results replace the contents of the ACC. The contents of the memory location remain unchanged.

The AND operation compares each bit position of two words (fields) and places a 1-bit in the result field (ACC) position if both fields contain a 1-bit in that position. The table that follows illustrates the four possible bit combinations in the same bit position of two ANDed words.

Memory	ACC	Result
0	0	0
0	1	0
1	0	0
1	1	1

The Carry and Overflow indicators are not affected.

Logical OR (OR-11101)

The contents of the memory location specified by instruction are ORed, bit-by-bit, with the contents of the ACC. The results replace the contents of the ACC; the contents of the memory location are unchanged.

The OR operation compares each bit position of two words (fields) and places a 1-bit in that position of the result field (ACC) if either field contains a 1-bit in that position. The table that follows illustrates the four possible bit combinations in the same bit position of the two ORed fields.

Memory	ACC	Result
0	0	0
0	1	1
1	0	1
1	1	1

The Carry and Overflow indicators are not affected.

Logical Exclusive OR (EOR-11110)

The contents of the memory location specified by the instruction are exclusive-ORed, bit-by-bit, with the contents of the ACC. The result replaces the contents of the ACC; the contents of the memory location are unchanged.

The exclusive-OR operation compares each bit position of two words (fields) and places a 1-bit in that position of the result field (ACC) if either field, but not both, contains a 1-bit in that position. The table that follows illustrates the four possible bit combinations in the same bit position of the two exclusive ORed fields.

Memory	ACC	Result
0	0	0
0	1	1
1	0	1
1	1	0

The Carry and Overflow indicators are not affected.

SHIFT INSTRUCTIONS

All shift operations are in the short format (F=0) only. Each of the three Shift Right and four Shift Left instructions is defined by bits 8 and 9 of the basic Shift Right and Shift Left instructions. Except for the Shift Left and Count instructions, the number of positions shifted is controlled by the field specified by the Tag bits, as shown by the table that follows. (XR is the abbreviation for index register.)

Tag Bits	Shift Controlled By Low-Order Six Bits
00	Displacement
01	XR1
10	XR2
11	XR3

If the shift count is zero in the control field addressed, the instruction performs as a No-OP, and the Carry indicator is not affected.

Shift Left ACC (SLA-00010)

Bits 8 & 9 = 00

The ACC is shifted left the number of positions specified by the shift count, and vacated (low-order) bit positions are set to 0. The EXT is not affected.

The condition of the Carry indicator is determined by the contents of the last bit position shifted out of the ACC. The Carry indicator is turned ON by a 1-bit in the last position shifted out of the high-order position of the ACC; it is turned OFF by a 0. The Overflow indicator is not affected.

Shift Left ACC and EXT (SLT-00010)

Bits 8 & 9 = 10

The ACC and EXT are shifted left (as a 32-bit double word) the number of positions specified by the shift count, and vacated bit positions are set to 0.

The Carry and Overflow indicators are affected as in the Shift Left ACC instruction.

Shift Left and Count ACC (SLCA-00010)

Bits 8 & 9 = 01

Tag bits of 00 cause this instruction to be executed the same as a Shift Left ACC instruction. Tag bits of 01, 10, or 11 cause the six low-order bits of the designated register to be transferred to the CCC (Cycle Control Counter) as a shift count. The count is decremented by one for each position the ACC is shifted to the left. The shift is terminated by a 1-bit being shifted into the high-order position of the ACC or the CCC being decremented to zero. The decremented count is loaded into the six low-order positions of the index register. Bit positions 0-7 of the index register are not affected.

The Carry indicator is turned ON if the shift is terminated by a 1-bit in the high-order of the ACC. It is turned OFF if the shift is terminated by the CCC being decremented to zero. If a 1-bit in the high-order position of the ACC coincides with the CCC being decremented to zero, the Carry indicator is turned OFF.

The Overflow indicator is not affected.

NOTE: If the count (n) is decremented to zero, a shift left n positions has occurred. If the count is initially zero or if the sign bit is initially a 1 bit, the instruction performs as a No-op.

Shift Left and Count ACC and EXT (SLC-00010)

Bits 8 & 9 = 11

This instruction is the same as the Shift Left and Count ACC instruction, except that both the ACC and EXT are shifted. The high-order bits of the EXT are shifted into the low-order positions of the ACC, and the vacated low-order positions of the EXT are set to zero.

The Carry and Overflow indicators are the same as for the Shift Left and Count ACC instruction.

Shift Right ACC (SRA-00011)

Bits 8 & 9 = 00 or 01

The ACC is shifted right logically the number of positions specified by the shift count. Low-order bits shifted out are lost; high-order positions vacated are set to zeros. The EXT is not affected.

The Carry and Overflow indicators are not affected.

Shift Right ACC and EXT (SRT-00011)

Bits 8 & 9 = 10

The ACC and EXT are shifted right arithmetically (as a 32-bit double word) the number of positions specified by the shift count. The value of the sign-bit (position 0 of the ACC) is entered in all vacated positions. Low-order bits of the EXT are shifted out and lost.

The Carry and Overflow indicators are not affected.

Rotate Right ACC and EXT (RTE-00011)

Bits 8 & 9 = 11

The ACC and EXT are shifted right (as a 32-bit double word) the number of positions specified by the shift count. In effect, a continuous loop is formed, so that the high-order positions of the ACC pick up the bits shifted out of the low-order positions of the EXT. For example, if the shift count is three, all positions of the ACC and EXT shift three positions to the right, and the values of EXT bit positions 13, 14, and 15 are put in ACC bit positions 0, 1, and 2.

The Carry and Overflow indicators are not affected.

BRANCH INSTRUCTIONS

Branch instructions provide the means for departing from a sequential series of instructions, by testing to determine if a stated condition or combination of conditions exists, and returning to the point from which the departure was made.

Branch or Skip on Condition (BSC-01001)

Six separate conditions of the ACC can be tested by placing a 1-bit in the appropriate bit position of the instruction. The bit positions and corresponding conditions tested for are contained in the table that follows.

Bit Position	Condition
15	Overflow indicator OFF
14	Carry indicator OFF

Bit Position	Condition
13	ACC contents even
12	ACC positive, not zero
11	ACC negative
10	ACC zero

The contents of the ACC are not changed by testing.

Short Instruction Format (F = 0)

If any one of the conditions specified by the instruction is true, the program skips over the next word in memory and goes to the second word in sequence. This means that a BSC instruction in the short format must always be followed by a short-format instruction. If an instruction in the long format were to follow, a skip would send the program to the second word of the instruction, and a programming error would result.

If none of the conditions is true, the next sequential instruction is executed.

If bit positions 10 through 15 contain zeros (no condition tested), the instruction performs as a No-Op.

Long Instruction Format (F = 1)

When none of the conditions specified is true, the program branches to the EA. If any one of the conditions is true, the next sequential instruction is executed.

If no condition is specified, the program branches to the EA. This allows the long format of the BSC instruction to be used as an unconditional branch. An explanation of the computation of the EA is contained in the section, Effective Address Generation. When this instruction specifies an indirect address (IA = 1), it enables a return to the program routine from which the CPU departed to accomplish a subroutine. This is accomplished by making the indirect address in this instruction identical to the EA of the Branch and Store IAR instruction that effected the branch.

Programming Note. When an interrupt request has been detected by a priority level, the program is directed to service the request by interrupting. During the servicing, all interrupt requests of equal or lower status are effectively constrained from interrupting while the servicing of the higher priority is in progress. However, if a request is detected for a higher priority level than is presently in progress, the program is immediately interrupted again. This is frequently called Nesting of Interrupts.

At the completion of servicing any level of interrupt, it is necessary to signal the priority hardware to reset the priority-status of the highest level that is on. This reset permits lower priority requests, including those that may have been temporarily constrained but recorded, to be accepted once again by the CPU. This is effected by making Bit 9 = 1 in this instruction. This programmed recognition of waiting interrupts should not be confused with a normal subroutine linkage back to a mainline program in which Bit 9 should be set to zero.

The BSC is a conditional instruction, and when Bit 9 = 1, the interrupt level is reset only when the branch or skip occurs.

Indicators

The Overflow indicator is reset when tested by the BSC instruction; the Carry indicator is not reset by testing.

Branch and Store IAR (BSI-01000)

The BSI instruction can be used in either the short or long format.

Short Instruction Format (F = 0)

The contents of the IAR (the location of the next sequential instruction) are stored at the EA of the BSI instruction. The IAR is then set to the EA + 1, which becomes the location of the next instruction executed. For example, assume that the BSI instruction is at memory location 0500 and that the EA generated is 0600. Execution of the BSI instruction stores 0501 at memory location 0600 and branches to 0601, which is the location of the next instruction.

Long Instruction Format (F = 1)

In the long format, the BSI instruction branches conditionally under the same circumstances as the BSC instruction. The conditions to be tested are designated by bit placement in Bits 10-15, as shown by the table in the description of the BSC instruction. If none of the conditions is true, the contents of the IAR are stored at the EA and execution of the instruction proceeds as described for the short format. If one or more of the conditions is true, the next sequential instruction is executed.

Indicators

In the short format, the Carry and Overflow indicators are not affected; in the long format, the Overflow indicator is reset when tested. The Carry indicator is not reset.

Modify Index and Skip (MDX-01110)

This instruction can be used to modify an index register, the IAR, or the contents of a word in memory. Except as noted, a skip occurs only if the index register or memory word being modified changes sign or is zero after modification.

A skip causes the program to skip over the next word in memory and go to the second word in sequence. This means that an MDX instruction must always be followed by an instruction in the short format. If a long-format instruction were to follow, a skip would send the program to the second word of the instruction, and a programming error would result.

Short Instruction Format (F = 0)

The expanded Displacement is added to the register specified by the Tag bits of the instruction, according to the table that follows. The displacement is expanded to 16 bits by duplicating the sign bit 8 positions to the left in the resulting high-order position.

Tag Bits	Operation
00	Displacement added to IAR
01	Displacement added to XR1
10	Displacement added to XR2
11	Displacement added to XR3

When the Tag 'bits of the instruction are 00, the MDX instruction becomes a branch, a skip, or a No-Op. Since the IAR contains the address of the next instruction, a Displacement value of zero merely sends the CPU to the next instruction; a positive value of one results in a skip; and any other value results in a branch to the modified address in the IAR. (The Displacement can also be negative.)

Long Instruction Format (F = 1)

Modification is accomplished according to the contents of the Tag and IA fields of the instruction. If the Tag is 00, independent of the IA bit, the expanded Displacement (bits 8 through 15 of the first word of the instruction) is added to the contents of the memory location specified by the Address field of the instruction. The displacement is expanded to 16 bits by duplicating the sign bit 8 positions to the left in the resulting high-order position. If the Tag bits are not 00, the IA bit becomes the controlling factor, as shown below.

IA Bit = 0. The contents of the Address field of the instruction are added to the index register (XR) specified by the Tag bits:

T = 01	XR1
T = 10	XR2
T = 11	XR3

IA Bit = 1. The contents of the memory location specified by the Address are added to the designated index register, according to the Tag bit values noted above.

Indicators

The Carry and Overflow indicators are not affected.

Wait (WAIT-00110 and Undefined OP Codes)

This instruction is in the short format only. The operation of the CPU stops in a wait condition and can be restarted manually or by the detection of an interrupt. A manual restart causes resumption of the program with the next sequential instruction; an interrupt causes resumption at a point determined by the interrupt branch operation. Cycle sharing operations continue in the wait condition.

The Carry and Overflow indicators are not affected.

INPUT/OUTPUT OPERATIONS

The IBM 1130 Computing System offers a variety of I/O devices. The Keyboard for input and the Console Printer for output are standard on the IBM 1131 CPU (Central Processing Unit), Models 1 and 2. In addition, the 1131 Model 2 provides the large capacity and random-access availability of data inherent in disk storage. The following attached units offer a wide diversity of I/O media:

IBM 1134 Paper Tape Reader
IBM 1055 Paper Tape Punch
IBM 1132 Printer
IBM 1442 Card Read-Punch
IBM 1627 Plotter

The programmed operation of each of these units is described in succeeding sections. The operating procedures that have to do with the mechanical functioning of the devices are covered in the publication, IBM 1130 Computing System, Input/Output Units (Form A26-5890).

EXECUTE I/O (XIO-00001)

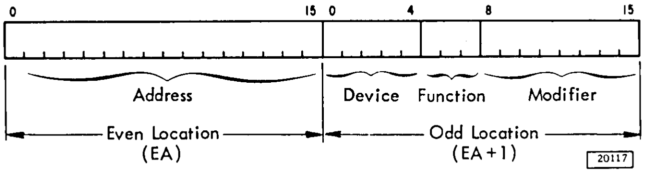
This instruction can be in either the short or long format, and operation is the same, except for the inherent differences in the manner in which the EA is generated, and the fact that the long format can have either a direct or indirect address.

The effective address is the memory location of the first word of the I/O Control Command (IOCC); EA + 1 is the location of the second word of the IOCC.

The contents of the ACC, if significant, must be stored prior to execution of the XIO instruction because the ACC is used in the analysis of the IOCC.

Input/Output Control Command

The format of the IOCC follows.



Address

The use of this 16-bit field depends on the function and the device specified.

Device

This 5-bit field identifies the I/O device.

00010	1442 Card Read-Punch
00110	1132 Printer
00100	Disk storage
00101	1627 Plotter
00011	1134 Paper Tape Reader, 1055 Paper Tape Punch
00001	Console Keyboard, Console Printer
00111	Console Entry Switches

Function

The primary I/O functions are specified by the 3-bit function code:

000 – Not used

001 – Write
This code is used to transfer a single word from storage to an I/O unit. The address of the storage location is provided by the Address field of the I/O Control Command.

010 – Read
This code is used to transfer a single word from an I/O unit to storage. The address of the storage location is provided by the Address field of the I/O Control Command.

011 – Sense Interrupt
This code is used only on Level 4 and causes the ACC to be loaded with the Interrupt Level Status Word (ILSW) for Level 4.

100 – Control
This code causes the selected device to interpret the Modifier field as a specific control action.

101 – Initiate Write
This code provides the ability to initiate a WRITE operation on a device or unit which will subsequently make data transfers from storage via a Data Channel.

110 – Initiate Read
This code provides the ability to initiate a READ operation from a device or unit which will subsequently make data transfers to storage via a Data Channel.

111 – Sense Device
This code loads the ACC with the DSW (Device Status Word) for the device specified in the IOCC. The status indicators are reset by specifying modifier bits as follows: Bit 15 for the highest level to which the device is connected, Bit 14 for the next highest level, and so on.

Modifier

This portion of the command provides additional information for the device and function specified.

INTERRUPT

The Interrupt facility provides an automatic branch from the normal program sequence based upon an external condition. A maximum of six interrupt levels are available with the 1130 Computing System. They are assigned as follows:

Level	Device
0	1442 Card Read-Punch (column read, punch)
1	1132 Printer
2	Disk Storage
3	1627 Plotter
4	1442 (operation complete); Keyboard/Console Printer; 1134 Paper Tape Reader; 1055 Paper Tape Punch
5	Console (Program Stop switch)

Interrupt Philosophy

Because of the number of types of interrupt requests, it is not always possible to cause a branch to a unique address for each interrupt condition. For the same reason, it is frequently not desirable to cause one branch for all interrupt requests and require the program to determine the individual request(s) requiring service. Therefore, it is expedient to group the many individual request lines into a lesser number of priority levels. This accomplishes two very important functions: First, it allows all interrupt requests common to a specific device to have the privilege of interrupting immediately if the only requests waiting or being serviced are of a lower priority level. Service is returned to the initial request only after all higher level requests have been serviced. Second, since a unique branch can be defined for each interrupt priority level, it is possible to combine many interrupt requests on a common priority level and therefore use a common interrupt subroutine to service many requests.

There are two important operating characteristics of the 1130 interrupt system: (1) when more than one request line is connected to any priority level, it is necessary, by programming means, to identify the individual request(s) causing the priority level to be energized; (2) the first request that causes an interrupt prevents future requests on the same or lower priority levels from interrupting until the completion of servicing the first interrupt is signaled by a "branch out" operation. (See Branch or Skip on Condition-BSC.)

Interrupts that occur on the same level for which an interrupt is being serviced can be detected and acknowledged before the branch out operation is executed.

Program Operation

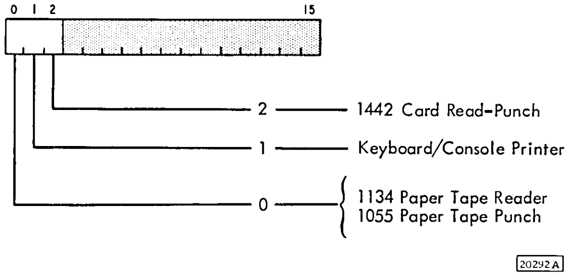
An interrupt may be recognized by the CPU at the completion of any program instruction. It is initiated by the basic interrupt control, which forces execution of a CPU-generated Branch and Store IAR (BSI) instruction. The indirect address of the generated BSI instruction is in location 8-13, corresponding to the level of interrupt. This location should contain the address of the location in the interrupt routine where the IAR is to be stored.

As defined by the BSI instruction description, the IAR is stored at the EA (effective address) and program execution is resumed with the Branch to the EA + 1. It is the responsibility of the interrupt subroutine to store all data and/or index registers that are used by the routine, and to restore the same registers prior to departing from the subroutine. (See the description of BSC.)

Several devices can request an interrupt on level 4. It thus becomes necessary for the program to determine the requesting device. This is accomplished by issuing an XIO instruction with a Function of Sense Interrupt.

The Sense Interrupt function is decoded and sent to all I/O devices, along with the current interrupt level being serviced. Each device that is requesting service on level 4 will have a bit appear in the ILSW that is loaded into the Accumulator, provided that level 4 is being serviced. The Sense Interrupt command will therefore produce meaningful results only if executed in a program sequence that is a result of interrupt level 4, and before a Branch Out command is executed in this routine.

Interrupt Level Status Word



Although a 16-bit ILSW could exist for each priority level, only level 4 uses the ILSW in the 1130 System.

Each device with an interrupt request signal assigned to priority level 4 is given a particular bit position in its ILSW to indicate its interrupt request status, a 1-bit if ON and 0 if OFF. The status indicator(s) in the device(s) is not affected by the sensing of the ILSW. It is possible for a device to contain several conditions which may cause an interrupt on the same interrupt level. When this condition exists, the interrupt conditions are logically ORed to become a single interrupt. The identification of the interrupting condition within the device is accomplished by sensing the Device Status Word (DSW) as discussed in subsequent paragraphs.

Interrupt Identification

Following loading of ILSW 4 in the ACC (accomplished by an XIO-Sense Interrupt instruction), the Shift Left and Count instruction is used to facilitate examination of the ILSW. First, an index register is loaded with a quantity which corresponds to the number of request signals connected to interrupt priority level 4, followed by the Shift Left and Count instruction (SLC). The resulting count in the Index register is unique and corresponds to the first non-zero bit of the ILSW in the Accumulator. (It is also possible to execute a Shift Left and Count of both the ACC and EXT. Refer to the SLC Instruction Description.) The SLC is followed by a Branch or Skip on Condition instruction (BSC) utilizing the F = 1 format with IA = 1, indexed with the result of the SLC. This provides, in conjunction with a branch table, a unique branch for each non-zero bit of the ILSW.

After the device causing an interrupt has been identified from data in the ILSW, it is necessary to determine the indicator(s) within the particular device causing the interrupt. This is accomplished by issuing a subsequent XIO Sense Device instruction with an area assignment corresponding to that of the device being interrogated. The Status indicators are reset after the information has been loaded in the ACC, if a bit is present in position 15 of the modifier. If a device can initiate interrupts on more than one interrupt level, the indicators are reset by specifying modifier bits as follows: Bit 15 for the highest level to which the device is connected, Bit 14 for the next highest level, and so on.

The data in the ACC is now referred to as the DSW (Device Status Word).

Device Status Word

The DSW contains one bit of information for each indicator within the device. These usually fall into three categories, (1) error or exception interrupt conditions, (2) normal data or service-required interrupts, and (3) routine status conditions.

Programming Note. When an interrupt request has been detected by a priority level, the program is directed to service the request by interrupting. During the servicing, all interrupt requests of equal or lower status are effectively constrained from interrupting while the servicing of the higher priority is in progress. However, if a request is detected for a higher priority level than is presently in progress, the program is immediately interrupted again. This is frequently called Nesting of Interrupts.

At the completion of servicing any level of interrupt, it is necessary to signal the priority hardware to reset the priority-status of the highest level that is on. This reset permits lower priority requests (including those that may have been temporarily constrained, but recorded) to be accepted once again by the CPU. The reset is accomplished by a BSC instruction with Bit 9 = 1. This programmed recognition of waiting interrupts should not be confused with a normal subroutine linkage back to a mainline program in which Bit 9 should be set to zero.

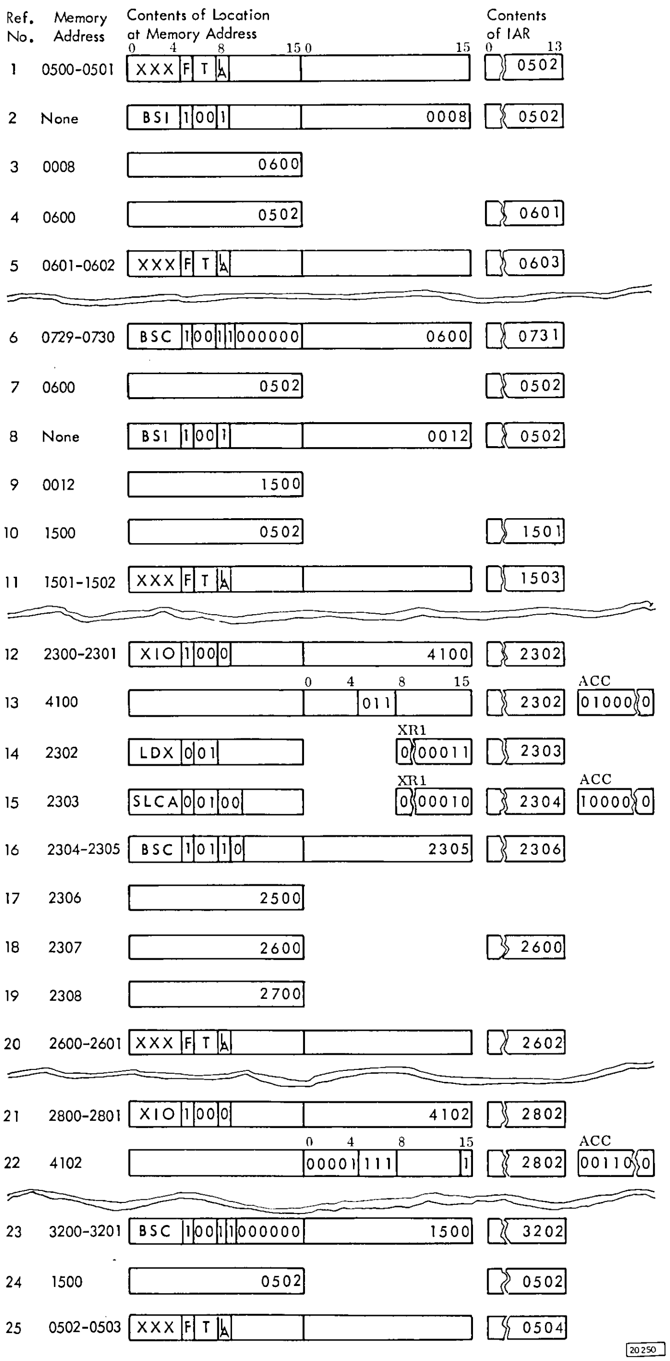
The BSC is a conditional instruction, and when Bit 9 = 1, the interrupt level is reset only when the branch or skip occurs.

Figure 7 shows a typical procedure for recognizing interrupts.

DISK STORAGE

Disk storage provides the IBM 1130 Computing System with low-cost random or sequential access data storage. On-line data capacity is 512,000 words; off-line capacity is virtually unlimited because the interchangeable disk cartridge is easily removed and replaced with another. Thus, the large storage capacity, comparable to that of magnetic tape, coupled with the unique advantage of random access, affords the 1130 Computing System great flexibility in the handling of engineering, scientific, industrial, and commercial programs.

Sample Interrupt Recognition Procedure



Ref. No.	Memory Address	Contents of Location at Memory Address (0–15)	(0–15)	Contents of IAR (0–13)	XR1	ACC
1	0500-0501	XXX F T IA		0502		
2	None	BSI 1 00 1	0008	0502		
3	0008	0600				
4	0600	0502		0601		
5	0601-0602	XXX F T IA		0603		
6	0729-0730	BSC 1 00 1 1 000000	0600	0731		
7	0600	0502		0502		
8	None	BSI 1 00 1	0012	0502		
9	0012	1500				
10	1500	0502		1501		
11	1501-1502	XXX F T IA		1503		
12	2300-2301	XIO 1 00 0	4100	2302		
13	4100		011	2302		01000 0
14	2302	LDX 0 01		2303	0 00011	
15	2303	SLCA 0 01 00		2304	0 00010	10000 0
16	2304-2305	BSC 1 01 1 0	2305	2306		
17	2306	2500				
18	2307	2600		2600		
19	2308	2700				
20	2600-2601	XXX F T IA		2602		
21	2800-2801	XIO 1 00 0	4102	2802		
22	4102		00001 111 1	2802		00110 0
23	3200-3201	BSC 1 00 1 1 000000	1500	3202		
24	1500	0502		0502		
25	0502-0503	XXX F T IA		0504		

20250

Figure 7. Sample Interrupt Recognition Procedure, Part I

The notes that follow are numbered to correspond to the reference numbers in the procedure. Each reference number cited in text is circled, e.g., (1), to avoid confusion with numbers necessary to the procedure, such as memory addresses.

In the registers, instructions, and data words, only the necessary 0-bits and 1-bits are shown. Op codes are shown in alphabetic symbols, and decimal numbers are used to identify memory locations. Binary notation, where used, is obvious. IAR, ACC, and XR1 are shown only where needed for understanding of the operation.

1. Mainline program instruction. During execution of this instruction, the 1442 initiates a card read interrupt.

2. At the conclusionof the (1) instruction, the CPU blocks the next program instruction and interposes a CPU-generated BSI to start the Level 0 Interrupt procedure.

3. IA of the Level 0 BSI is 0008; the EA at 0008 is 0600.

4. The IAR (0502) is stored at the EA (0600); the IAR is then loaded with the EA + 1 (0601).

5. First instruction of the Level 0 interrupt subroutine. The subroutine must store the status of each register, all data, etc., that could be altered by execution of the subroutine. Before leaving the subroutine, the program must restore all registers, data, etc., to the condition that existed when the interrupt occurred.

6. The last instruction of the interrupt subroutine is a BSC instruction with a Bit-9 value of 1, which resets the priority status so that interrupts of equal or lower priority can be recognized. If no interrupt is waiting, return to the interrupted program is effected by the IA(0600) of the BSC being equal to the EA of the CPU-generated BSI that initiated the interrupt routine, (2). The BSC is shown as an unconditional branch (Bits 10-15 = 0); the branch could have been conditional, i.e., the branch executed only if conditions specified by Bits 10-15 were true.

NOTE: The term interrupted program is used to designate either the main program being executed or an interrupt subroutine of lower priority. For example, the (1) instruction could be in a routine to service a console printer-keyboard, Level 4 interrupt. Thus, the mainline program can be thought of as a routine with no priority, to which the CPU returns when no interrupts are waiting.

7. The EA (0502) is the location of the next mainline program instruction and is loaded into the IAR.

8. To illustrate an interrupt with a low priority occurring while a higher priority interrupt is being serviced, we assume that the console printer-keyboard initiated a Level 4 interrupt while the card read interrupt was being serviced. We assume that no Level 0, 1, 2, or 3 interrupts are waiting and that the Level 4, CPU-generated BSI can now be interposed, as in ② for Level 0.

9. The IA (0012) of the BSI is the memory location assigned to Level 4 interrupts and contains the EA (1500).

10. The IAR is stored at the EA (1500) and then loaded with EA+1 (1501).

11. First instruction of Level 4 subroutine. See ⑤ . Last housekeeping instruction takes subroutine to 12 .

12. The XIO instruction EA (4100) is the memory location of the IOCC. The IAR contents remain at 2302 because the IOCC controls an I/O device and is not a sequential program instruction.

13. The IOCC function code of 011 (Sense Interrupt) causes the ILSW for Level 4 to be loaded into the ACC.

14. XR1 is loaded with a quantity equal to the number of response signals connected to the ILSW.

15. SLCA instruction is terminated when the 1-bit associated with the console printer-keyboard interrupt is shifted into the high order position of the ACC. XR1 is reduced by one.

16. BSC instruction address is modified by XR1 (+2) to form the IA (2307). A bit in the 0-position of the ILSW (paper tape reader and paper tape punch) results in an XR1 of 3 and an IA of 2308. A bit in the 2-bit position (card read-punch) results in an XR1 of 1 and an IA of 2306.

17. An IA of 2306 has the EA 2500, which is the memory location of the first instruction of the Card Read-Punch interrupt subroutine.

18. An IA of 2307 has the EA 2600, which is the memory location of the first instruction of the Console Printer-Keyboard interrupt subroutine.

19. An IA of 2308 has the EA 2700, which is the memory location of the first instruction of the Paper Tape Reader and Paper Tape Punch interrupt subroutine.

20. First instruction of housekeeping sequence for Console Printer-Keyboard subroutine.

21. The XIO instruction EA (4102) is the memory location of the console printer-keyboard IOCC.

22. The IOCC Sense Device function code (111) causes the DSW of the console printer-keyboard (00001) to be loaded into the ACC. The 1-bit in position 15 causes the response to be reset. The example shows 1-bits in positions 2 and 3 of the ACC (DSW), which indicate that the operator initiated an interrupt request on the keyboard and that the console entry switches are to be read. A programmed subroutine determines the cause of the interrupt (Bit 2) and the interrupting device (Bit 3). A routine then follows that reads the data into memory, accomplishes any housekeeping required, and releases the CPU, as shown in ㉓ .

23. Procedure is the same in ⑥ and ⑦ . The IA (1500) of the BSC instruction is equal to the EA of the CPU-generated BSI instruction that initiated the interrupt; see ⑧ , ⑨ , and ⑩ .

24. The EA (0502), located at 1500, is loaded into IAR.

25. The instruction at 0502 is the next one to be executed in the mainline program. See ① for previous mainline instruction.

Figure 7. Sample Interrupt Recognition Procedure, Part II

System programs, object programs, subroutines, and often-used table data can be stored in the same removable disk cartridge and a specific computation accomplished by mounting the cartridge and feeding the variable data into the 1131 CPU. This simplifies problem solving and increases the throughput of the System.

DESCRIPTION

Disk storage for the 1130 System is contained in the CPU cabinet and is connected to the CPU by a high-speed data channel. It is composed of two components: the disk and drive assembly and the access mechanism.

Disk Assembly

The disk assembly (Figure 8) is a single disk drive, completely enclosed in a protective housing, or cartridge. The recording medium is an oxide-coated disk that provides two surfaces for the magnetic recording of data. When mounted in the CPU enclosure, the disk drive rotates at the rate of 1500 revolutions per minute.

Access Mechanism

The disk storage access mechanism has two horizontal arms. Each arm has a magnetic read/write head, and each head is positioned to read or write on the corresponding disk surface as the access arms straddle the disk in the manner of a large tuning fork. The entire assembly moves horizontally forward and backward, so that the heads have access to the entire recording area.

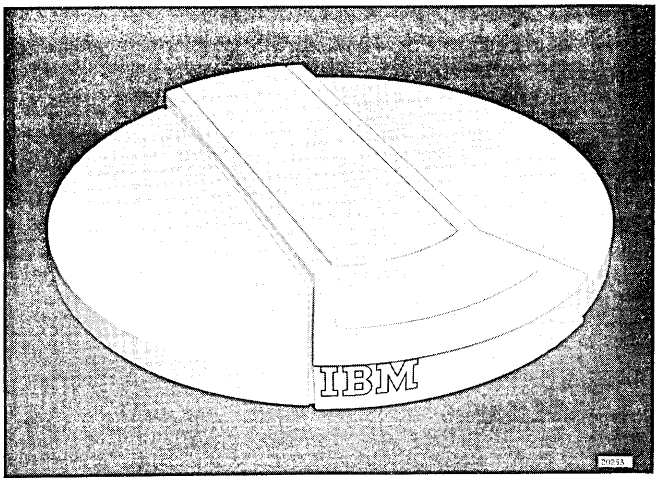


Figure 8. Disk Assembly Cartridge

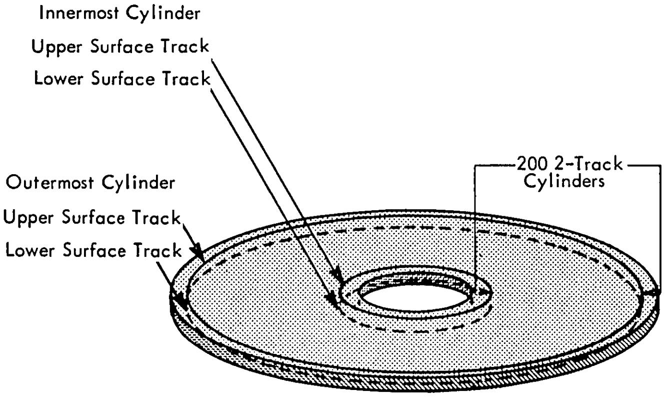
The access mechanism is positioned automatically, at the home position (outside cylinder) when the disk cartridge is inserted.

Disk Organization and Capacity

The access mechanism is moved back and forth by program instructions and can be placed in any one of 200 positions, from a point near the periphery of the disk to a point near the center of the disk. At each position, the heads can read or write in a circular pattern on both surfaces of the disk, as it revolves. These circular patterns of data are called tracks. The track on the upper surface of the disk and the corresponding track on the lower surface, both of which can be read or written while the access mechanism is in the same position, are called a cylinder. Figure 9 shows the innermost and outermost cylinders of two tracks each. To complete the picture, the 198 intermediate cylinders, or pairs of tracks, should be visualized; they were omitted for the sake of clarity of the diagram.

For convenience in transferring data between the CPU core storage and disk storage, each track is divided into four equal segments called sectors. Sectors are numbered by the cylinder, from 0 through 7, as shown in Figure 10. Sectors 0 - 3 divide the upper surface track, and Sectors 4 - 7, the lower. A sector contains 321 data words and is the largest segment of data that can be read or written with a single instruction.

In the programs and programming systems provided by IBM, e.g., the Monitor System and its programs, the first word of a 321-word sector is used for cylinder sector number.



NOTE: The thickness of the disk has been greatly exaggerated in order to show the relative positions of the upper and lower surface tracks.

Figure 9. Disk Storage Cylinder Schematic

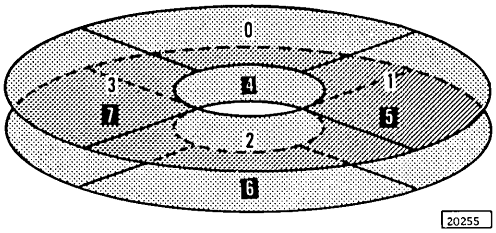


Figure 10. Disk Storage Sector Numbers

Therefore, the first word of the sector can not be used by the programmer if the Assembler program or other components of the Monitor System are to be used.

A disk storage word comprises 16 data bits and four check and space bits.

Table 1 shows the organizational components of disk storage. Note that capacities are based on the 320-word sector.

Timing

Timing considerations of disk storage operation involve three elements: access time, reading and writing data, and the time during which the CPU is tied up.

Access. The access mechanism moves in increments of two cylinders at the rate of 15 ms per increment. Thus, in the formula that follows, the number of cylinders (N) must be even. (The next higher even number is used if an odd number of cylinders is specified.) During the 20 ms stabilization period that follows the last incremental movement, a Read or Write instruction can be given and will be started at the end of the stabilization period.

$$\text{Access time (ms)} = 7.5(N) + 20$$

Table 1. Disk Storage Data Organization

No of. ↓ Per →	Word	Sector	Track	Cylinder	Disk
Bits	16	5,120	20,480	40,960	8,192,000
Data Words		320	1,280	2,560	512,000
Sectors			4	8	1,600
Tracks				2	400
Cylinders					200

Read/Write. Reading or writing of data in disk storage is at the rate of 27.8 μsec per word. Average rotational delay time is 20 ms, based on 1500 rpm, or 40 ms per revolution. Thus, a sector can be read or written in an average of 30 ms. Although there are no timing considerations for head switching, there are programming considerations in consecutive sector operations because there is an interval of over 420 μsec between sectors; the interval is increased by 27.8 μsec for each word less than 321 read or written.

A full cylinder of eight 321-word sectors can be read or written in 100 ms because the rotational delay is required for only the first sector.

CPU Time. An interrupt in a disk storage operation occurs only at the end of the Seek or Read/Write operation. This means that once the instruction is initiated, disk storage operation is virtually independent of the CPU. As data is being read or written, a cycle is literally "stolen" from the CPU operation in progress every 27.8 μsec for the transmission of the next word. Thus, except for the normal instruction times, the CPU is busy only 14 ms of the 100 ms required to read or write a full cylinder. The remaining 86 ms are available for other program operations.

Data Checking

Data is checked on each transmission between core storage and disk storage. The number of bits of each word is divided by four as the word leaves core storage, and the number of bits necessary to make the division even (Modulo 4) is added to the end of the word. The Modulo 4 test is performed again each time a word is written in disk storage or read from it. A word that is not Modulo 4 causes the Data Check bit to be set in the disk storage DSW.

Procedure for Changing the Disk Storage Cartridge

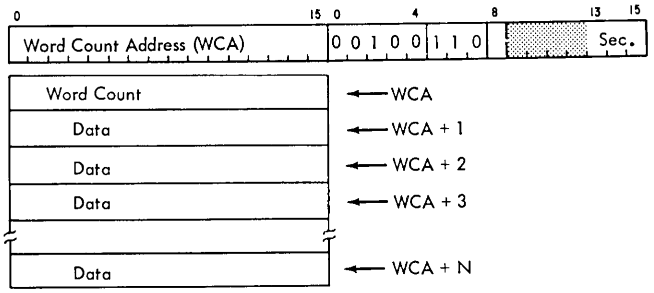
1. Turn off the disk drive motor with the power switch adjacent to the disk mechanism.
2. Open the hinged cover of the disk storage drive enclosure.
3. Pull down on the release/lock handle and remove the cartridge. An interlock prevents removal of the cartridge until the disk has stopped spinning.
4. Another cartridge can be installed by simply inserting it into the aperture. The access mechanism read/write heads are automatically positioned as the cartridge is inserted.

5. Raise the access release/lock handle to lock the cartridge in place.
6. Close the cover of the disk storage drive enclosure and start the disk drive motor. An interlock prevents the motor from starting unless the cartridge is correctly inserted and the disk is in place. The disk reaches ready status in approximately 90 seconds.

PROGRAMMING DISK STORAGE

I/O Control Command (IOCC)

Initiate Read (110)



20106A

This instruction causes the number of words specified by the Word Count to be read from the disk storage sector identified by Modifier bits 13-15. The Address word of the instruction contains the WCA (Word Count Address), and Modifier bit 8 determines whether the command is a Read instruction (0) or a Read-Check instruction (1).

A full sector, 321 words, is the maximum transmission with one instruction. Succeeding sectors, or parts of sectors, require an Initiate Read instruction for each one.

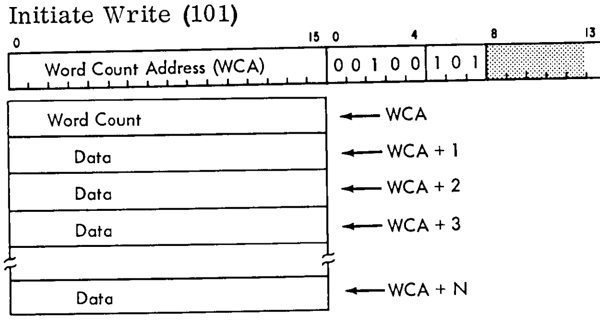
An Operation-Complete interrupt occurs when the number of words in the Word Count has been transmitted.

Read Instruction (Bit 8 = 0). Beginning with the first word of the indicated sector, data is read into core storage location WCA + 1 and ascending addresses. The Word Count, which is stored at the location specified by the WCA, controls the number of words transmitted and, consequently, the number of core storage locations occupied by the disk storage data. For example, assume that a Word Count of 152 is stored at WCA 1000. The 152 words read from disk storage would be stored at addresses 1001 through 1152.

The programmer must be aware of the core storage locations required for incoming disk storage data so that useful data is not written over and lost.

Read-Check Instruction (Bit 8 = 1). Data is read from disk storage, as in the Read instruction, and the number of bits of each word is checked for Modulo 4. If the division for any word is not even, the Data Check indicator bit is set in the disk storage DSW. Neither disk storage nor core storage is affected by the Read-Check instruction.

Initiate Write (101)

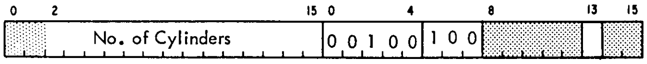


20107A

This instruction causes the number of words specified by the word Count to be written in disk storage, beginning at the first word of the sector indicated by Modifier bits 13-15. The Address word of the instruction contains the address of the Word Count (WCA). The data is transmitted from core storage location WCA + 1 and ascending addresses. A full sector, 321 words, is the maximum transmission with one instruction. Succeeding sectors, or parts of sectors, require an Initiate Write instruction for each one.

An Operation-Complete interrupt occurs when the number of words in the Word Count has been transmitted.

Control (100)



20108A

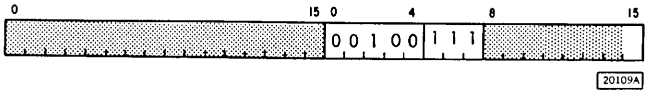
This instruction causes the access mechanism to move in increments of two cylinders for the number of cylinders specified by the Address word of the instruction. If the number of cylinders is odd, the first increment consists of one cylinder.

Modifier bit 13 controls the direction of movement: a 0 moves the access mechanism forward (toward the center of the disk); a 1 moves it backward.

When the access mechanism has moved the number of cylinders specified, an Operation-Complete interrupt occurs.

NOTE: Cylinders do not carry an identifying number. It is the responsibility of the program, therefore, to maintain the necessary information relative to the position of the access mechanism. A Control command which specifies an access motion of zero cylinders is treated as a No-op and does not result in an Operation Complete interrupt.

Sense Device (111)



This instruction causes the Device Status Word (Figure 11) of disk storage to be read into the ACC. All indicators are reset if Modifier bit 15 is a 1.

Interrupt

Operation Complete. This is the only interrupt associated with disk storage, and is turned on at the end of a Read, Read-Check, or Write operation. It is also turned on when the access mechanism has reached the designated cylinder during a Control instruction.

Indicators

Operation Complete. This indicator is turned on at the end of a Read, Read-Check, Write, or Control (access movement) operation. It is turned off by a Sense Device instruction with Modifier bit 15 set to one.

Disk Busy. This indicator is on during execution of a disk storage instruction. It is turned off when the operation is complete.

Error Check. This indicator is turned on when a parity (Modulo 4) error is detected during a Read, Read-Check, or Write instruction. It is turned off by a Sense Device instruction with Modifier bit 15 set to one.

Disk Not Ready. This indicator is on when disk storage is not ready to receive an instruction, which means that the disk is busy or has not reached ready status.

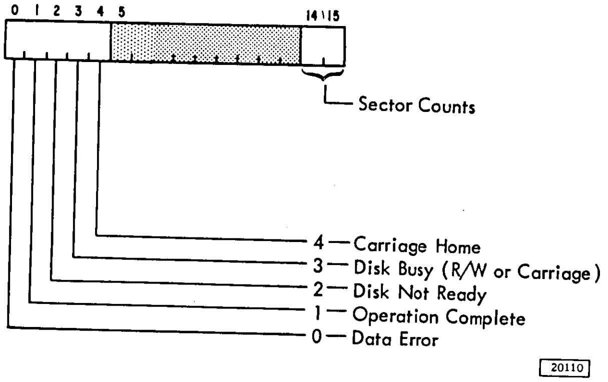


Figure 11. Disk Storage Device Status Word

Carriage Home. This indicator is on when the access mechanism is at the Home position (Cylinder 00).

Sector Count. These bits represent the sector number, to Modulo 4, of the next sector available for reading or writing. Thus, the quantity represented by the bits is the sector number on the upper track.

Programming Considerations

Disk Organization

It is important in planning a routine for loading disk storage that the cylinder concept be taken into consideration. Related data should be grouped in the same cylinder, when possible, to eliminate unnecessary seek operations. Therefore, when disk addresses are assigned to a group of related data, the disk locations made available should be limited to the number required, plus an expansion factor. The most frequently used data should be stored in the low-numbered cylinders to minimize seek time.

Customer Error Correction Routines

In the event that an error is detected by the CPU circuitry, it is recommended that the following procedure be executed:

1. Re-seek the cylinder upon which the error was detected.
2. Re-execute the operation in which the error occurred.

This procedure should be executed from three to ten times prior to establishing the occurrence of a disk error.

CONSOLE

The Console (Figure 12) is an integral part of the IBM 1131 Central Processing Unit and consists of the input keyboard, console printer, display panel, function switches and lights and Console Entry switches.

While the keyboard and console printer are usually considered as one unit, control of each of them by the operator and by the stored program is discrete. For this reason, the functional description and programmed operation of each unit is considered separately in the sections that follow.

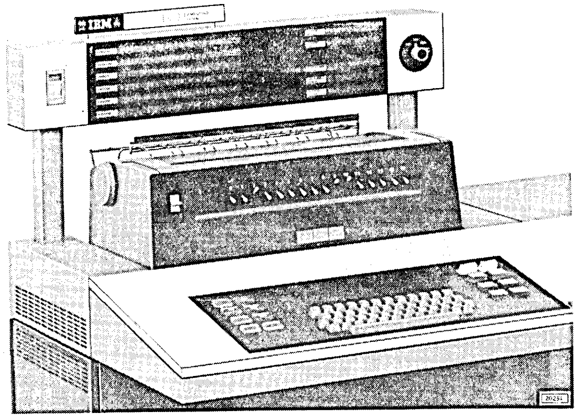


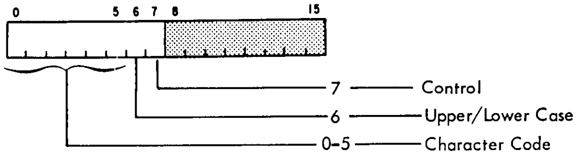
Figure 12. 1131 Console

CONSOLE PRINTER FUNCTIONAL DESCRIPTION

The console printer provides output at a maximum rate of 15.5 characters per second. Data to be printed is transferred from core storage to the console printer by direct program control.

Data characters and control characters (space, tabulate, etc.), are sent to the console printer by means of the Write command. Because control characters and data characters are sent in the same manner, the message to be printed contains a mixture of data characters and control characters in the sequence necessary to give the desired formatted output.

The character format within a core storage word to be transmitted to the console printer is:



15646A

Each word transmitted to the console printer contains one data character or one control character.

Data Coding

Data to be printed is coded by the program into the console printer code. Figure 13 shows the characters which can be printed by the standard print element.

Character Code Bits						U/L Case		Ctrl
B0	B1	B2	B3	B4	B5	B6=0 LC	B6=1 UC	B7
0	0	1	1	1	1	A	A	0
0	0	0	1	1	0	B	B	0
0	0	0	1	1	1	C	C	0
0	0	1	1	0	0	D	D	0
0	0	1	1	0	1	E	E	0
0	0	0	1	0	0	F	F	0
0	0	0	1	0	1	G	G	0
0	0	1	0	0	1	H	H	0
0	0	1	0	0	0	I	I	0
0	1	1	1	1	1	J	J	0
0	1	0	1	1	0	K	K	0
0	1	0	1	1	1	L	L	0
0	1	1	1	0	0	M	M	0
0	1	1	1	0	1	N	N	0
0	1	0	1	0	0	O	O	0
0	1	0	1	0	1	P	P	0
0	1	1	0	0	1	Q	Q	0
0	1	1	0	0	0	R	R	0
1	0	0	1	1	0	S	S	0
1	0	0	1	1	1	T	T	0
1	0	1	1	0	0	U	U	0
1	0	1	1	0	1	V	V	0
1	0	0	1	0	0	W	W	0
1	0	0	1	0	1	X	X	0
1	0	1	0	0	1	Y	Y	0
1	0	1	0	0	0	Z	Z	0
1	1	1	1	1	1	1	(	0
1	1	0	1	1	0	2	+	0
1	1	0	1	1	1	3	<	0
1	1	1	1	0	0	4	¬	0
1	1	1	1	0	1	5	)	0
1	1	0	1	0	0	6	;	0
1	1	0	1	0	1	7	*	0
1	1	1	0	0	1	8	'	0
1	1	1	0	0	0	9	"	0
1	1	0	0	0	1	0	|	0
1	1	0	0	0	0	#	=	0
1	0	1	1	1	1	/	–	0
1	0	0	0	0	1	-	?	0
1	0	0	0	0	0	,	:	0
0	1	0	0	0	1	&	>	0
0	1	0	0	0	0	$	!	0
0	0	0	0	0	1	@	%	0
0	0	0	0	0	0	.	¢	0

20085

Figure 13. Console Printer Character Coding

The data-character code also contains (in B6) the information as to whether the character is an upper-case (UC) shift or lower-case (LC) shift character. The printer shifts automatically as required for each data character.

A printer Write command is modified by the B7 position of the output character word. If B7 equals one, the Write command to the printer is interpreted as a control function. If B7 equals zero, the Write command is interpreted as a print function.

The codes for console printer Control functions are shown in Figure 14.

Function	0 1 2 3 4 5 6 7 8 15
Carrier Return	1 0 0 0 0 0 0 1
Tabulate	0 1 0 0 0 0 0 1
Space	0 0 1 0 0 0 0 1
Backspace	0 0 0 1 0 0 0 1
Shift to Red*	0 0 0 0 1 0 0 1
Shift to Black*	0 0 0 0 0 1 0 1
Line Feed	0 0 0 0 0 0 1 1

* May be done concurrently with any other function.

20090A

Figure 14. Console Printer Control Functions

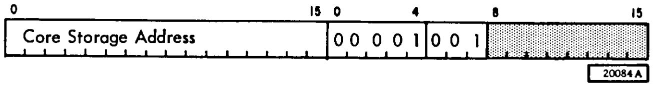
PRINTER PROGRAMMING

The console printer operates in the 1130 System under direct program control.

I/O Control Commands (IOCC)

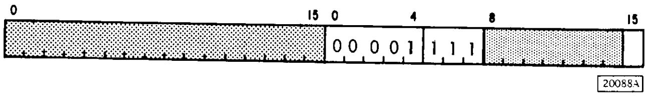
The console printer is addressed by a 5-bit device code in the IOCC, 00001.

Write (001)



This command causes bits 0-7 of the word at the core storage location specified by the Address to be sent to the printer for printing or control.

Sense Device (111)



This command causes the keyboard/console printer Device Status Word to be placed in the ACC. Figure 15 shows only those bits which relate to the console printer. Modifier bit 15 on specifies that all keyboard/console printer responses are to be reset.

Interrupts

There is only one interrupt associated with the console printer attachment.

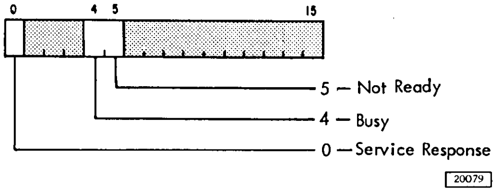


Figure 15. Console Printer Device Status Word

Service Response. This interrupt occurs each time the console printer has completed printing the data and/or the control operation required by the last word transmitted by the Write command.

Indicators

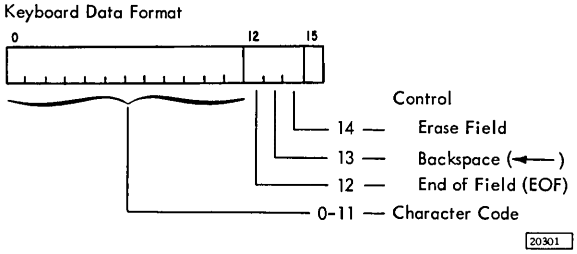
The following indicators are entered into the CPU by a Sense Device command.

Not Ready. When off, this indicates that the printer is properly loaded with forms, has dc power, and is not busy. It is necessary that the program always determine that the Not Ready indicator is off before a Write command is given. If a Write command is given while Not Ready is on, loss of information will probably occur. No indication is given of this loss.

Busy. When on, this indicates that the console printer is in the process of typing a character or executing a control and therefore should not be given a Write command. The Busy line is active from the time data is sent to the printer until the printer has completed the action required.

KEYBOARD FUNCTIONAL DESCRIPTION

The input speed of the keyboard (Figure 16) is limited only by the speed of the operator. Keyboard entries are not automatically printed unless the CPU is programmed to provide an output of the entry on the printer. The keyboard emits a coded character for each key struck by the operator. These characters are related to IBM card coding. (See Appendix E. Character Code Chart). Striking the A character key places bits in positions 0 and 3 of the CPU word; striking the I character key places bits in positions 0 and 11 of the word; striking a 9 character places a bit in position 11 of the word; etc.



The two-position Console/Keyboard switch indicates to the program the desired source of the console input data, either the keyboard or the console entry switches.

Keyboard Function Keys

Interrupt Request. This key initiates a keyboard restore and causes an interrupt in the CPU.

End of Field (EOF). When the CPU reads in response to this key, a word containing a 12 bit only is placed in memory. Analysis of this word allows the program to determine that no further characters are to be sent in this message.

Backspace (←). When the CPU reads in response to this key, a word containing a 13 bit only is placed in memory. Analysis of this word allows the program to determine that the last character received is to be replaced by the next character to be entered.

Erase Field. When the CPU reads in response to this key, a word containing a 14 bit only is placed in memory. Analysis of this word allows the program to determine that the message being entered is to be deleted and replaced by a corrected message.

Mode. There are two mode keys: Numeric (upper case shift) and Alphabetic (lower case shift). These keys place the keyboard in the indicated mode. The keyboard remains in the selected mode until changed. If the numbers or symbols which appear on the top portion of the keys are desired, the keyboard must be placed in Numeric mode.

Restore. This key allows the operator to restore the keys if they should become locked.

Keyboard Light

Keyboard Select. This light comes on when the CPU has performed a Control command. This light goes off when a Read command is performed.

Operating Procedure

The following procedure describes a typical use of the keyboard (manual start).

1. The operator presses the Interrupt Request key which initiates a Request interrupt and places the keyboard in a Restore status.
2. The CPU honors the Request interrupt and determines that the Keyboard is the device that caused the interrupt.
3. The CPU issues a Control command to select the keyboard. When the keyboard is selected, the Select light is turned on to signal the operator that a character can be entered.

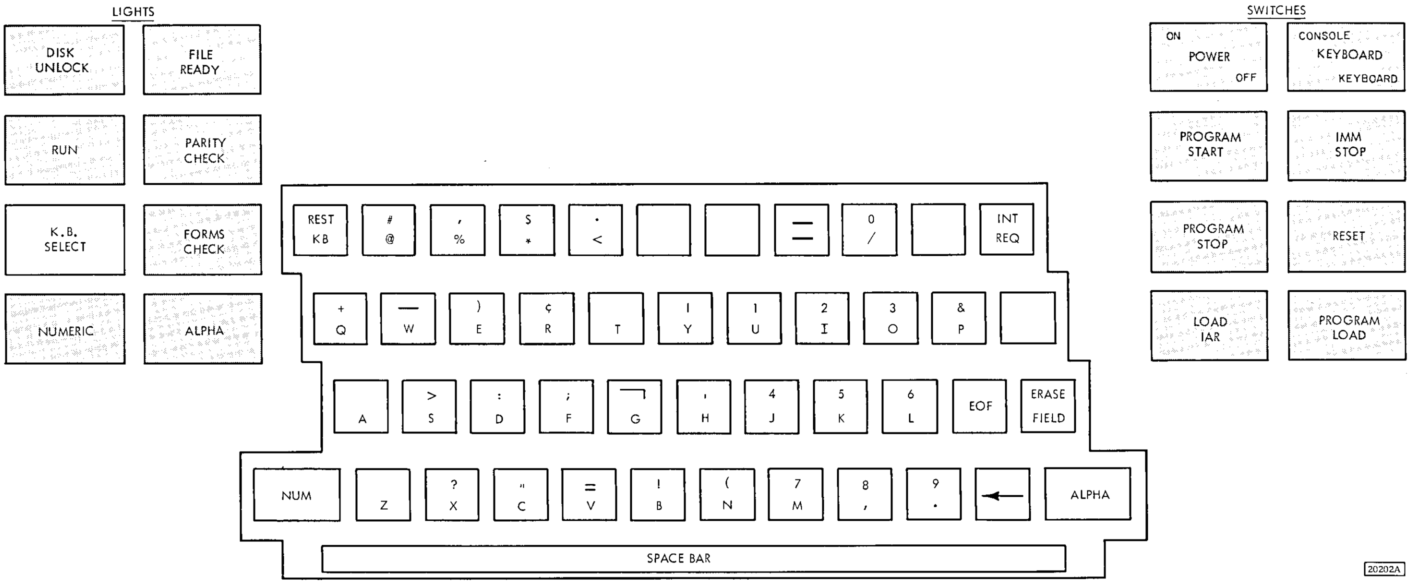


Figure 16. 1131 Console Keyboard

4. When a character key is pressed, the keyboard initiates a Service interrupt to the CPU.
5. In response to the Service interrupt, the CPU performs a Read command which enters the character into core storage and removes the keyboard from the selected status.
6. Before another character can be entered, the CPU stored program must issue another Control command to select the keyboard.

NOTE: When the request is initiated by the stored program, the operation is the same, beginning at Step 3.

If the CPU performs a Read command when the keyboard is not selected, no bits are entered.

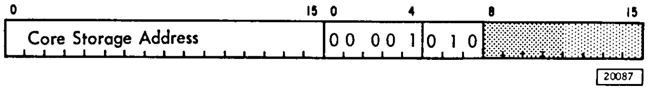
KEYBOARD PROGRAMMING

The keyboard operates under direct program control of the 1130 Computing System.

I/O Control Commands (IOCC)

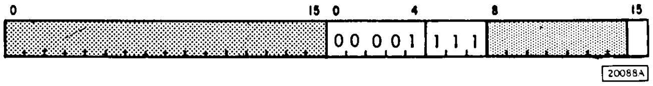
The keyboard is addressed by the same Device code used by the console printer, 00001.

Read (010)



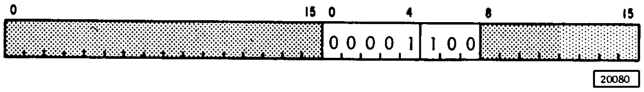
This command enters a single input character from the keyboard into the core storage location specified by the Address of the IOCC.

Sense Device (111)



This command reads the keyboard/console printer Device Status Word into the ACC. Figure 17 shows only those bits associated with the keyboard. Modifier bit 15 on specifies that all keyboard/console printer responses are to be reset.

Control (100)



This command places the keyboard in a Select status so that a character can be entered.

Interrupts

The two interrupts associated with the keyboard are assigned to the same level of priority.

Interrupt Request. This interrupt is initiated by the Request key located on the keyboard.

Keyboard Response. This interrupt signals that a character key has been pressed and that a character is ready to be entered into core storage.

CONSOLE DISPLAY PANEL

The contents of the registers within the computer are displayed on the console panel (Figure 18) by means of small incandescent lights. Each bit in each register position is represented by a light. The light is on when the bit which it represents is present in the word displayed. The Mode switch selects the operating mode of the system.

Indicator Displays

Instruction Address Register (IAR). The Instruction Address register is one row of 14 indicator lamps. Each lamp displays the status of one bit position in the IAR.

Storage Address Register (SAR). The Storage Address register is one row of 14 indicator lamps. Each lamp displays the status of one bit position in the SAR.

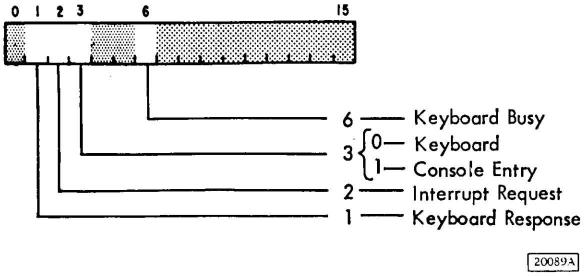


Figure 17. Keyboard Device Status Word

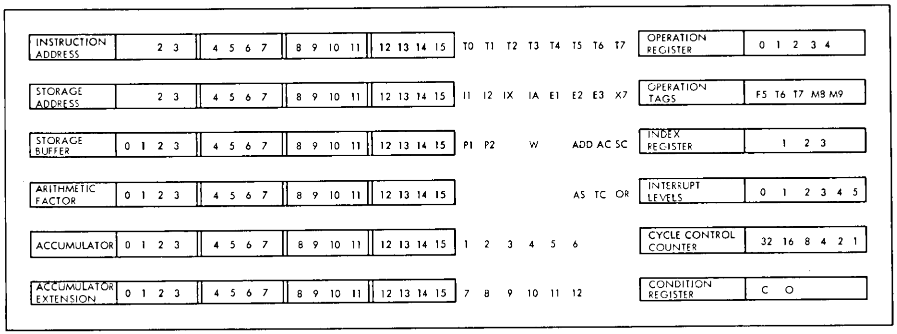

INSTRUCTION ADDRESS 2 3 4 5 6 7 8 9 10 11 12 13 14 15
T0 T1 T2 T3 T4 T5 T6 T7
OPERATION REGISTER 0 1 2 3 4
STORAGE ADDRESS 2 3 4 5 6 7 8 9 10 11 12 13 14 15
I1 I2 IX IA E1 E2 E3 X7
OPERATION TAGS F5 T6 T7 M8 M9
STORAGE BUFFER 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
P1 P2 W ADD AC SC
INDEX REGISTER 1 2 3
ARITHMETIC FACTOR 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
AS TC OR
INTERRUPT LEVELS 0 1 2 3 4 5
ACCUMULATOR 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6
CYCLE CONTROL COUNTER 32 16 8 4 2 1
ACCUMULATOR EXTENSION 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
7 8 9 10 11 12
CONDITION REGISTER C O

Figure 18. Console Panel

Storage Buffer Register (SBR). One row of 16 indicator lamps. Each lamp indicates the status of one bit position in the SBR. At the end of each machine cycle the SBR reflects the bits that were read into core storage on the cycle just completed.

Arithmetic Factor Register (AFR). One row of 16 indicator lamps. Each lamp indicates the status of one bit position in the AFR.

Accumulator Register (ACC). One row of 16 indicator lamps. Each lamp indicates the status of one bit position in the ACC.

Accumulator Extension Register (EXT). One row of 16 indicator lamps. Each lamp indicates the status of one bit position in the EXT.

Clock Timer (T). One row of eight indicator lamps. These lamps indicate the last clock step completed.

Machine Cycle. One row of seven indicator lamps. These lamps indicate the type of machine cycle in process when in Single Step mode. They indicate the machine cycle just completed when in any other mode.

Control Functions. Two rows of three indicator lamps. These lamps indicate the status of the following functions: Add, Arithmetic Control, Shift Control, Accumulator sign, Accumulator Carry, and Zero Remainder.

CE Lights. Two rows of six indicator lamps. Each lamp can be wired by a CE to give a visual indication of any status condition in the machine.

Operation (OP) Register. One row of five indicator lamps. These lamps indicate the operation in process when in Single Step mode or Single Machine Cycle mode. They indicate the operation just completed when in any other mode.

Operation Tags. One row of five indicator lamps. These lamps indicate the status of the Format, Tag, and Modifier bits of the instruction shown in the operation register.

Interrupt in Process. One row of six indicator lamps. These lamps indicate the Interrupt level being serviced.

Cycle Control Counter. One row of six indicator lamps. These lamps indicate the binary value contained in the shift counter.

Condition Register. One row of two indicator lamps. These lamps indicate the status of the Carry indicator and the Overflow indicator.

Parity (2). These two indicators reveal which half of the word contains the parity error when one occurs.

Wait OP. This indicator is on when the CPU is in a wait condition.

X7. This indicator is off when "cycle steal" is operating.

Mode Switch

The Mode switch (Figure 19) selects one of seven operating modes.

Single Step (SS). With the Mode switch set to SS, each depression and release of the Start key causes the 1131 clock to advance one step, e.g., from T1 to T2.

Single Memory Cycle (SMC). With the mode switch set to SMC, each depression of the Start key causes the 1131 to advance one machine cycle (for example, from I-1 to I-2).

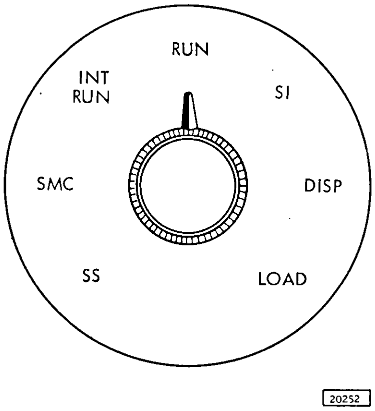


Figure 19. Console Mode Switch

Interrupt Run (INT RUN). With the mode switch set to INT RUN, a level 5 interrupt occurs after each mainline program instruction is completed. This is a convenient device for program trace routines.

Program Run (RUN). With the mode switch set to its normal position, RUN, pressing the Start key causes the 1131 to advance through its stored program.

Single Instruction (SI). With the mode switch set to SI, each depression of the Start key causes the 1131 to interpret and execute a single instruction.

Display Core Storage (DISP). With the mode switch set to DISP, pressing the Start key will display (in the SBR) the core storage word at the location specified by the address in the Instruction Address register (IAR), and advances the IAR.

Load Core Storage (LOAD). With the mode switch set to LOAD, pressing the Start key will load the data from the Console Entry switches into core storage at the location specified by the address in the IAR, and advances the IAR.

CONSOLE ENTRY SWITCHES

These 16 toggle switches are used to set up data or instructions to be entered into core storage. Each switch represents a bit position in a 16-bit word. The procedures that follow provide for entering the information from the CES (Console Entry switches) by means of manual control, keyboard interrupt, or XIO instruction.

Manual Entry. This procedure causes the bits set in the CES to be loaded into the word at the core storage address in the IAR.

1. Set the Mode switch to Load.
2. Set the CES to the core storage address where the data is to be stored.
3. Press the Load IAR switch.
4. Set the data word in the CES.
5. Press the Start key.

Keyboard Interrupt. This procedure requires an interrupt subroutine to service a Level 4 interrupt.

1. Set the Console/Keyboard switch to Console.
2. Press the keyboard Interrupt Request key.
3. A Level 4 interrupt occurs, and the Keyboard DSW is loaded into the ACC by a Sense Device instruction.
4. DSW bit 3 was set to 1 by the Console/Keyboard switch. This indicates to the interrupt subroutine that the CES should be read by the XIO Read instruction.
5. Return to the main line program is by the regular method of a BSC instruction with Modifier bit 9 set to one.

XIO Read Instruction. The settings of the CES can be read by an XIO Read instruction at any time during a stored program routine. The Device code of the instruction is set to 00111.

CONSOLE FUNCTION SWITCHES AND LIGHTS

These switches and lights are located on both sides of the keyboard (Figure 20).

Function Lights

Forms Check. This indicator is turned on when the last form has been detected by the Console Printer forms contact.

Keyboard Select. This indicator is turned on by a programmed instruction (XIO Control) that requests input data from the keyboard.

Parity Check. This indicator is turned on when a parity error (even number of bits) is detected in either half of the word read out of storage.

Alphabetic. This lamp indicates that the keyboard is in alphabetic (lower case) shift. The letters and

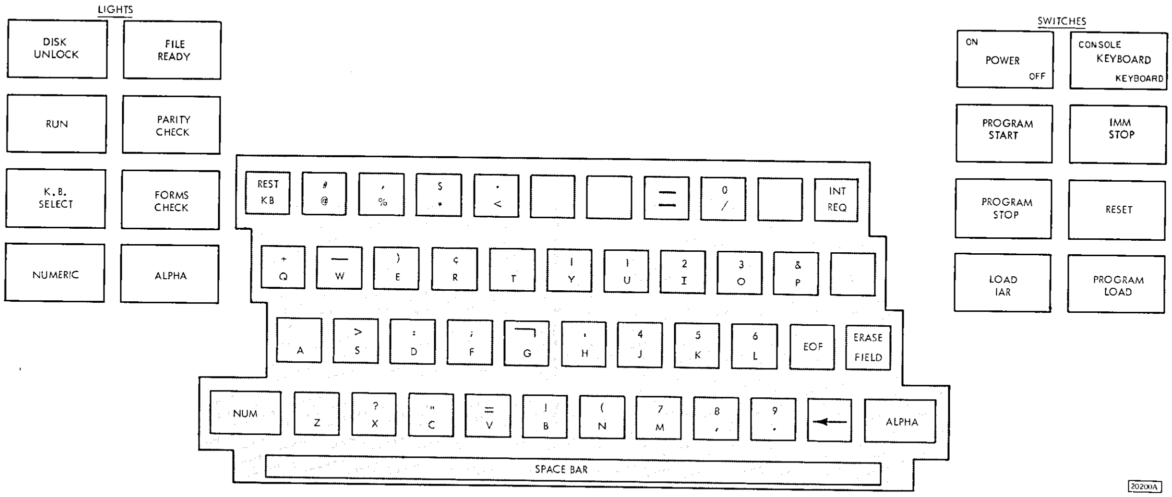


Figure 20. Console Function Lights and Switches

symbols which appear in the bottom portion of the keyboard character keys can be entered when the keyboard is in alphabetic shift.

Numeric. This lamp indicates that the keyboard is in numeric (upper case) shift. The letters and symbols which appear on the top portion of the keyboard character keys can be entered only when the keyboard is in numeric shift.

File Ready. This indicator is on when disk storage is available for reading or writing.

Disk Unlock. This indicator is on when the disk cartridge may be removed from the drive.

Run. This indicator is on when the CPU is in operation and the meter is running.

Function Switches

Parity Run/Stop. This two position toggle switch is set to STOP if the programmer wishes the program to stop when a parity error is detected. When this switch is set to RUN, the program will continue to run even if a parity error is detected.

Console/Keyboard. This two position toggle switch sets bit position 3 in the Keyboard DSW, which indicates to the program the desired source of the console input data (either the Keyboard or the Console Entry switches). See the preceding Console Entry switches and the DSW for the Sense Device (111) command (Figure 17).

Program Start. Pressing this pushbutton switch causes the machine to take one clock step or machine cycle and continue to take additional cycles if required by the setting of the mode switch.

IMM Stop. Pressing this pushbutton switch causes an immediate stop of the processor interrupt, although the I/O devices will finish their present cycle. Data from the 1132, the 1442, or disk storage will be lost if they are operating at the time the Stop key is pressed. A complete program restart is normally required.

Program Stop. Pressing this switch causes a level 5 interrupt. After the program has satisfied all interrupts for levels 0 through 4 (all I/O devices except the console) it enters a routine for level 5. Because the console is the only unit on level 5, the DSW for the console can be sensed without interrogating the interrupt level status word. The program stop bit in the DSW should be used to branch the program to a wait loop that causes the CPU and I/O units to cycle down to a stop and blocks all mainline operations until the console operator intervenes. This indicator is reset by pushing the program start key.

Reset. Pressing this pushbutton switch (effective only when out of Run mode or stopped) resets all I/O and machine registers, cycle and control triggers, and status indicators.

Load IAR. Pressing this pushbutton switch places the status of the 16 Console Entry switches in the IAR. The console Mode switch must be set to Load position.

Program Load. Pressing this pushbutton switch loads the first card or paper tape record into core storage, beginning at 0000.

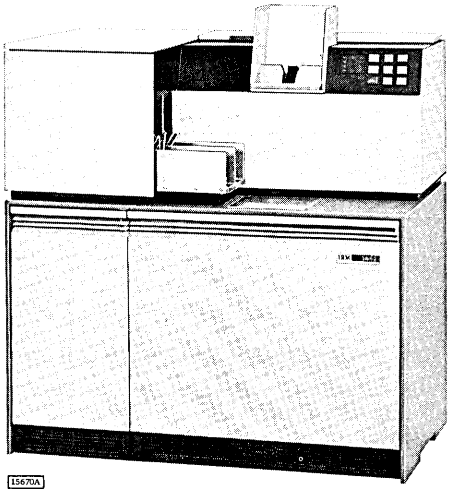


Figure 21. IBM 1442 Card Read-Punch

IBM 1442 CARD READ-PUNCH

The IBM 1442 Card Read-Punch (Figure 21), Model 6 or Model 7, provides card input/output for the IBM 1130 Computing System.

Card Read-Punch operations are under direct program control.

FUNCTIONAL DESCRIPTION

The IBM 1442 Card Read-Punch is a single unit that processes cards serially, column-by-column, from a single supply hopper. All cards first pass the read station, then the punch station. This permits each card to be read, punched, or read and punched. Reading and punching cannot occur simultaneously, however, because of the difference in operating speeds.

Maximum machine speeds are:

Card Reading:	Model 6, 300 cards per minute
	Model 7, 400 cards per minute
Card Punching:	Model 6, 80 columns per second
	Model 7, 160 columns per second

Maximum reading rates are attained only when successive Start-Read commands arrive early enough to re-energize the read clutch before the clutch latch point is reached. To accomplish this, successive Start-Read commands must arrive within 35 milliseconds (25 ms Model 7) after the End-Card interrupt is given by the Card Read-Punch. If a Start-Read command does not arrive within this time, the maximum reading rate becomes 285 cards per minute (cpm) for Model 6 and 375 cpm for Model 7.

Punching rates depend on the position of the card when the last column is punched. The punching speed ranges are:

Model 6 — 49 cpm to 255 cpm
Model 7 — 90 cpm to 340 cpm

The approximate time required to process a single card is:

Model 6 — 12.5 ms + 12.5 ms per card column spaced or punched.
Model 7 — 6.5 ms + 6.5 ms per card column spaced or punched.

Data Coding

The Card Read-Punch reads and punches IBM card image only. Any code translation required must be done by the stored program. As shown in Figure 22, the twelve rows (12-9) in a card column correspond to the 0-11 bits, respectively of a core storage word. A 1-bit represents a punched hole; a 0-bit represents a card position not punched. Thus, the word in Figure 22 contains 1-bits in bit-positions 0 and 3 to represent the "A" read from the card. For output, a 1-bit results in a hole punched in the related position of the card read column.

A special Load mode is initiated by pressing the Program Load key on the 1130 Console. In the Load mode, data is split (Figure 23), as it enters core storage, to form the load program.

OPERATING PROCEDURES

Before any operation can begin, the Card Read-Punch must be placed in the "Ready Condition." With power on and cards in the hopper, the Start key is pressed. This feeds the first card into position at the read station (Figure 24).

Card Feeding

Card reading or punching may begin after the initial feed cycle (run in).

A constant-speed drive moves the cards through the serial path during a feed cycle. A feed cycle is initiated by a Control command of Feed cycle or Start-Read or Punch (if no card is positioned at the punch station). The feed cycle does three things.

1. It moves a card from the punch station to the stacker.

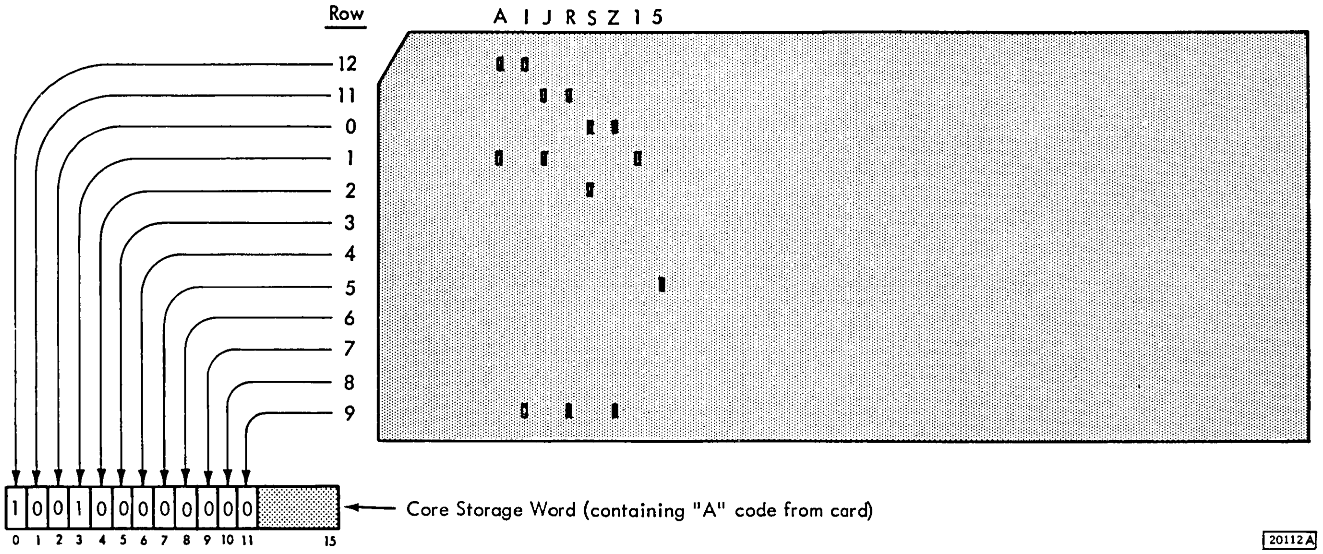


Figure 22. Normal Mode Read

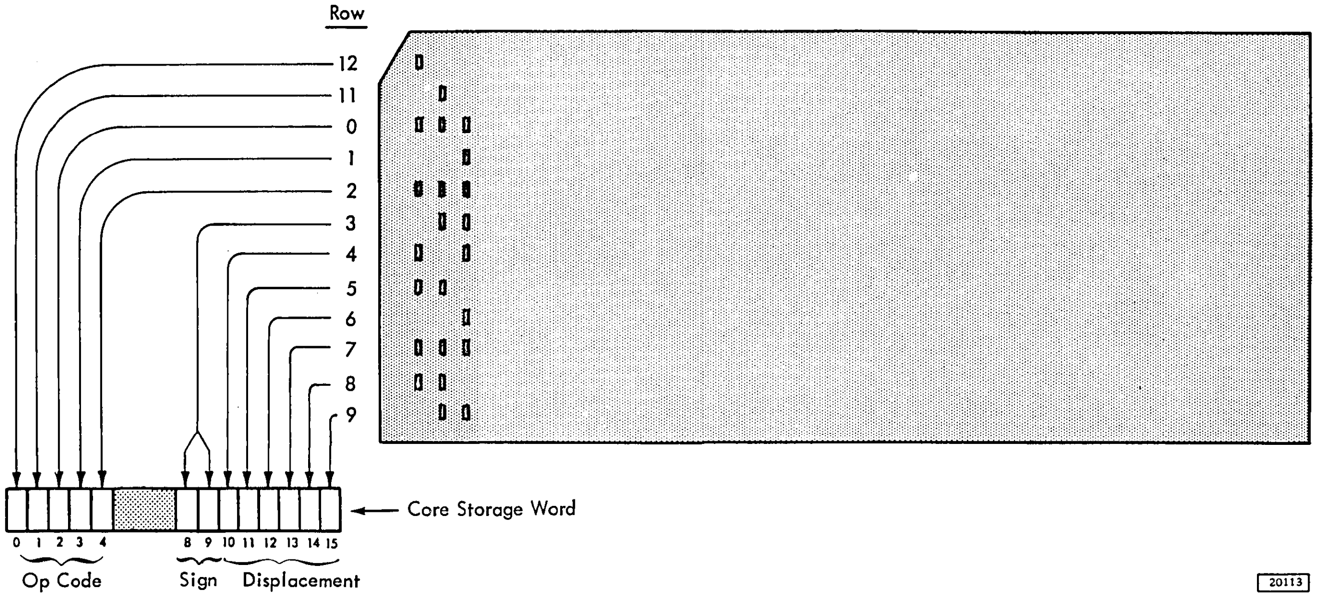


Figure 23. Load Mode Read

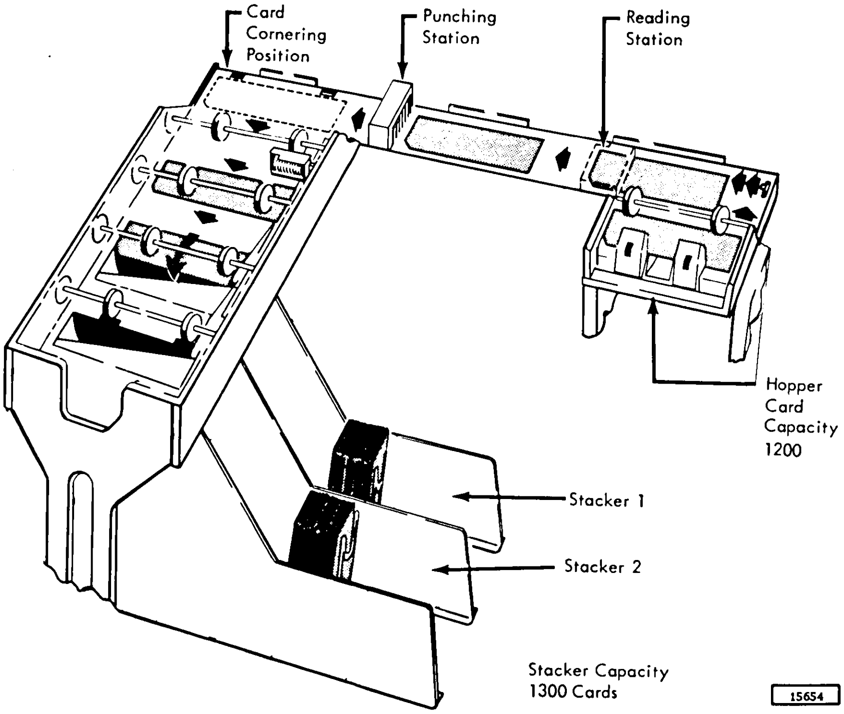


Figure 24. 1442 Card Path Schematic

2. It moves a card through the read station and places it in the punch station with column 1 under the punches.
3. It moves a card from the hopper to the read station.

An incremental drive moves the card through the punch station for punching.

When the hopper is emptied, the operator can either reload the hopper and continue operations or he can initiate a last-card sequence.

Program Load

Program load can be initiated by pressing the Program Load key on the 1130 Console after a system reset and the "run in" cycle of a load card. This "Load" mode causes the load-card data to be placed in 80 consecutive memory positions beginning at memory position 00000, then causes the CPU to go to memory position 00000 for its next instruction.

Card Reading

A Control (Start-Read) command initiates card reading. This command causes columns 1-80 of the card to be read in one continuous motion of the card. Each column of data is read, checked, and placed in a buffer register. A Read Response interrupt is given for each column read. Checking is accomplished automatically by reading each column twice and comparing the results bit-by-bit. This read-check-interrupt process continues until all 80 columns have been read. An Operation Complete interrupt is given after all 80 columns have been read. The Last Card indicator will be on if the card read is the last card in the deck.

Card Punching

A Control (Start-Punch) command initiates card punching. As each column passes the punch station a Punch Response interrupt is given.

Automatic checking is accomplished by comparing the punch check echo data with the single-character punch buffer, which contains the character from the CPU. Each column punched is checked at the same time that the Punch Response interrupt is given for the data of the next column to be punched.

The card motion and punching process continues until the punch data word contains a one in the 12-bit position (punch data is in bits 0-11). When this end-punch bit is detected, the Card Read-Punch punches that column, moves to the next column, and gives an Operation Complete interrupt. No more Punch Response interrupts are given. All punching on the card must be completed at one time.

A feed cycle is necessary to eject a punched card to the stacker, and can be initiated by a Control command.

Programming Note. A Control command specifying Start Punch results in a feed cycle if it has not been preceded by a Control command specifying Feed Cycle or Read Card.

Last Card Sequence

When the hopper becomes empty during a feed cycle, the Card Read-Punch is taken out of Ready status. The operator may continue processing cards by loading more cards into the hopper and pressing the Start key or he may initiate a last-card sequence by pressing the Start key without loading more cards in the hopper.

If the last-card sequence is to be entered, the program determines this by testing the Last Card indicator in the Device Status Word. This indicator is turned on when the last card passes the read station.

When the Start key is pressed without cards in the hopper, the 1442 is placed in the Ready condition and allows two more feed cycles to be taken.

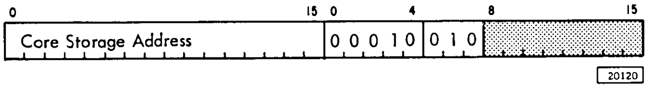
PROGRAMMING

The IBM 1442 Card Read-Punch operates under the direct program control of the CPU.

I/O Control Commands

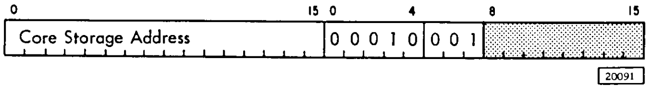
The Card Read-Punch is addressed by the 5-bit Device code, 00010

Read (010)



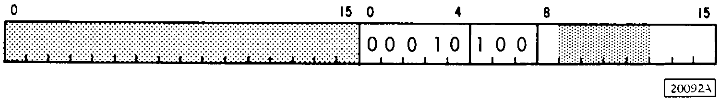
This command causes a card column image to be entered from the Card Read-Punch into the core storage location specified by the Address.

Write (001)



This command causes the data in the memory location specified by the Address of the IOCC to be transmitted and punched as a card column image in the card.

Control (100)



This command causes the Card Read-Punch to accomplish the function specified by the Modifier.

Modifier bits that have significance are:

Bit 14 Feed cycle - causes all cards in the feed path to advance one station. There are no Read Column Response interrupts.

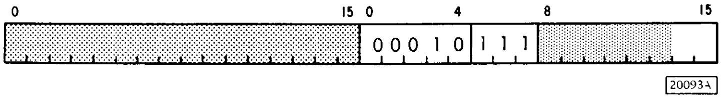
Bit 13 Start Read - causes the card to move through the read station. As each column is read and checked, the Card Read-Punch initiates a Read Column Response interrupt.

Bit 15 Start Punch - starts the punching operation and initiates a Punch Response interrupt. If a card is not at the punch station, a card will feed past the read station without data entering the system.

Bit 8 Stacker Select - causes the card leaving the punch area to enter the alternate stacker. This control applies only to the next card leaving the punch station.

Modifier bits B14, B13, and B15 of a Control command should not be used in combination with each other.

Sense Device (111)



This command directs the Card Read-Punch to place its Device Status Word (Figure 25) into the CPU ACC. Modifier bit 15 on resets responses for level 0; modifier bit 14 on for level 4.

Interrupts

The three interrupts associated with the Card Read-Punch are divided into two groups.

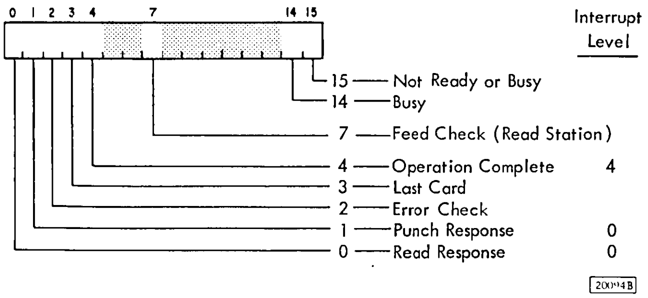


Figure 25. 1442 Device Status Word

Level 0 Interrupt

Read Response. This interrupt signals that a column of data is ready to be entered. This interrupt is on interrupt level zero, which guarantees service of the request within 800 μsec for the 1442 Model 6 and 700 μsec for the Model 7. Time from Initiate Read to first Read Column Request interrupt is 28.4 ms on Model 6 and 23.8 ms on Model 7.

Punch Response. This interrupt signals that a column of data must be transmitted from the CPU within 1000 μsec for the 1442 Model 6 and 500 μsec for the Model 7. Time from the Start Punch instruction to the first Punch Column Response interrupt varies from 1.22 ms to 12.5 ms on the Model 6 and 1.56 to 6.25 ms on the Model 7.

Level 4 Interrupt

Operation Complete. This interrupt occurs after a card has been read. It indicates that column 80 of the card has passed the read station. This interrupt occurs 20.6 ms after column 80 for the Model 6 and 15.4 ms after column 80 for the Model 7.

This interrupt also occurs after the last column to be punched has been punched and checked with the punch drive stopped. This will occur 12.5 ms after the terminating Write function for the Model 6 and 6.25 ms after the terminating Write function for the Model 7.

This interrupt is forced if a hopper check, feed check in the Punch Station, transport error, or feed clutch error occurs while the 1442 is busy. This interrupt is also forced if a read registration check or punch check occurs. No subsequent reading or punching can be done in the card which caused the error regardless of the column in which the error occurred.

There is no time limit on the request for service of this interrupt.

Indicators

Not Ready. This indicator shows that the 1442 is either busy or is not in a ready condition. When the 1442 is not ready, manual intervention is required to ensure that the following conditions are met.

1. Power On.
2. Card registered at Read Station (initially).
3. Cards are in hopper or last-card sequence is in progress.
4. Stacker not full.
5. Feed-Check light off (no card jam or feed failure conditions).
6. If the Stop key has been pressed, the Start key must have been subsequently pressed.
7. Chip box not full or removed.

Busy. The Busy indicator indicates that a command cannot be initiated because an operation is already in progress.

Last Card. This indicator shows that column 80 of the last card has passed the read station and the hopper is empty, and will be on when the Operation Complete Interrupt occurs.

Feed Check (Read Station). This indicator comes on as a result of a feed check detected at the Read Station.

Error Check. Indicates that any of seven error conditions exists in the 1442. Any of the seven error conditions remove the 1442 from the ready condition; the 1442 can be reset only by depressing the Non-Process Run Out key while the hopper is empty.

The error conditions are:

1. Read Registration Check. Indicates that a read error has occurred. This can result from incorrect registration of the card or failure of the first and second reading of the column to compare equal. When this error is detected, an Operation Complete interrupt is forced and column interrupts are terminated.
2. Punch Check. Indicates that a punching error has been detected. When this error is detected, an Operation Complete interrupt is forced and column interrupts are terminated.
3. Hopper. Indicates card failed to pass properly from the hopper to the read station. See Programming Note below.
4. Transport. Indicates a jam in the stacker. See Programming Note below.
5. Feed Check - Read Station. Indicates a card failed to eject from the read station.
6. Feed Check - Punch Station. Indicates a card improperly positioned at the punch station. See Programming Note below.
7. Feed Clutch. Indicates the 1442 took a feed cycle that was not called for. See Programming Note below.

Programming Note. Error indicator is not turned on until after the operation complete interrupt is given. An exception to this is an XIO Start Punch operation requiring an automatic feed cycle. If another operation is initiated before the error indicator is turned on, these errors force an operation complete interrupt although no reading or writing has taken place. An XIO Start Punch requiring an automatic feed cycle is treated as two operations: (1) feed cycle (2) punch operation.

IBM 1134 PAPER TAPE READER AND IBM 1055 PAPER TAPE PUNCH

The IBM 1134 Paper Tape Reader and the IBM 1055 Paper Tape Punch (Figures 26 and 27) provide paper tape input/output for the IBM 1130 Computing System.

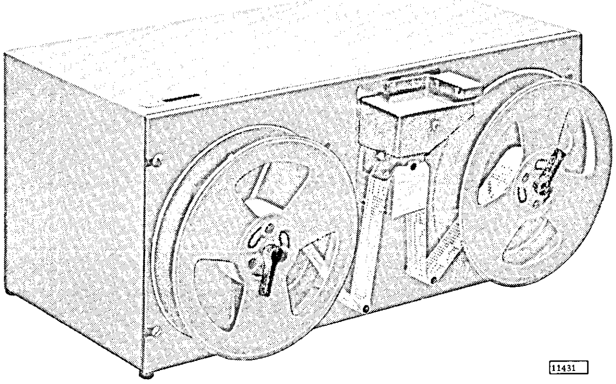


Figure 26. IBM 1134 Paper Tape Reader

FUNCTIONAL DESCRIPTION

The 1134 and 1055 operate under direct program control.

The 1134 Paper Tape Reader reads one-inch eight-track paper tape at a maximum rate of 60 columns per second (cps).

The 1055 Paper Tape Punch punches one-inch eight-track paper tape at a maximum punching rate of 14.8 cps.

Character Code

The 1134 Paper Tape Reader reads input data into the core storage as an image of the holes in the tape. One paper tape character is read into each addressed core storage location. Any code translation must be made by programming. (See Appendix E.)

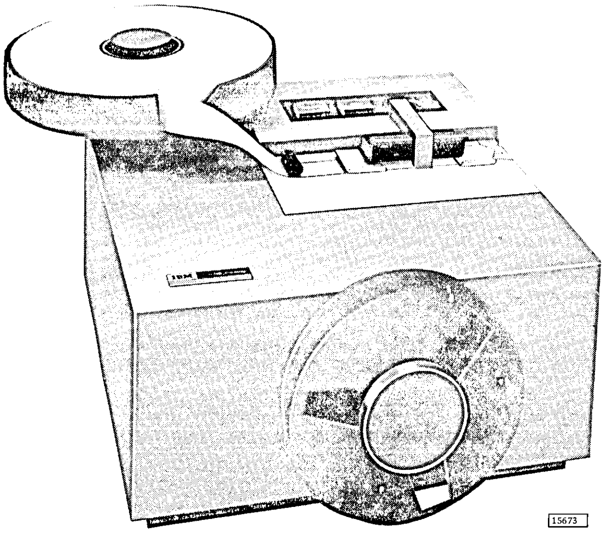


Figure 27. IBM 1055 Paper Tape Punch

Figure 28 indicates which bits of the word correspond to the respective holes in the paper tape read by the 1134.

The 1055 Paper Tape Punch punches data as an image of the data contained in the core storage word on a character-to-character basis as shown in Figure 28.

Special data-character and control-character (feed code, etc.) coding and recognition must be handled by the stored program.

Program Load

An 1130 System that does not have a 1442 Card Read-Punch will have the Program Load feature added to the paper tape reader. This feature operates by means of design logic rather than program control. Four-bit paper tape characters are automatically assembled into 4-character groups to form 16-bit data words. The Program Load feature then loads these words into core storage beginning at location 0000.

Only tape channels 4, 3, 2, and 1 are used. When a channel 5 punch is encountered, program loading stops; the IAR is reset to zero; and program control begins at 0000.

Characters ——→	First	Second	Third	Fourth
Tape Channels ——→	4321	4321	4321	4321
Bits ——→	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15

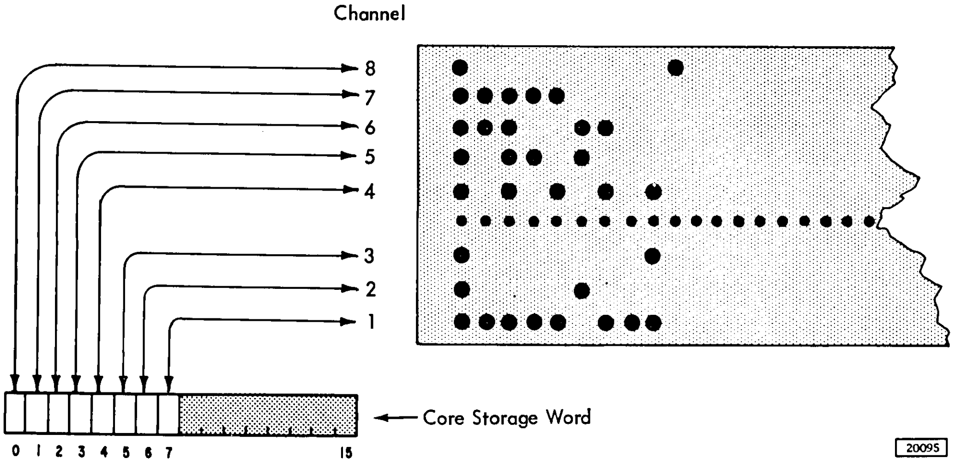


Figure 28. Paper Tape/Core Storage Word Format

DESCRIPTION OF OPERATION

Paper Tape Reader

A Control command initiates the reading of data from the 1134. This command moves the paper tape the character position and loads the character which was at the read station into an input buffer. At the time the buffer has been loaded with data, an interrupt is initiated signaling the program that information is available for reading into the core storage position specified by the Address word of a subsequent Read (Paper Tape) command.

The elapsed time from the execution of the Control (Read Paper Tape) command until the interrupt is initiated is approximately 500 μs. To maintain the 60 cps operating speed of the 1134, the Read command must be given with 15 ms after the interrupt so that another Control (Read Paper Tape) command can be executed to energize the reader clutch preparatory to reading the next character.

Paper Tape Punch

Execution of a command initiates the punching of data onto the 1055. The execution of the Write command starts the punch, and the data in the core storage position specified by the Address word is punched into the tape. Each core storage word contains one paper tape character to be punched in the tape.

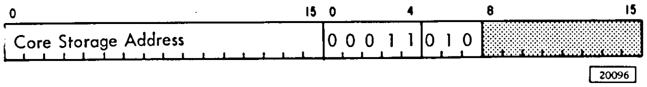
PROGRAMMING

The IBM 1134 Paper Tape Reader and the IBM 1055 Paper Tape Punch operate under direct program control with the exception of the paper tape Program Load feature.

I/O Control Commands (IOCC)

The 1134 and 1055 are addressed by the same 5-bit Device code, 00011.

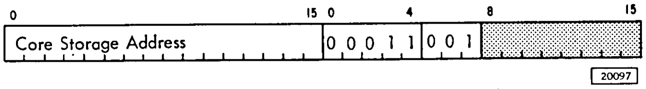
Read (010)



This command reads one character from paper tape into core storage.

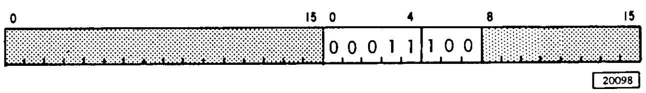
The Address word specifies the location in core storage where the tape character is to be stored.

Write (001)



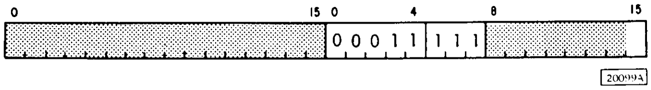
This command writes one character from core storage to the paper tape punch. The Address word specifies the location in core storage where the tape character is stored.

Control (100)



This command must be given prior to each character to be read from the 1134. Execution of this command causes: (1) one character to enter the paper tape reader buffer, and (2) the tape to be advanced one column. A Reader Service Response interrupt is initiated to indicate that a character from paper tape can be read into the core storage location specified by a subsequent Read (Paper Tape) command.

Sense Device (111)



This command is used to enter the Device Status Word (Figure 29) into the ACC. Modifier bit 15 on, indicates that the responses are to be reset.

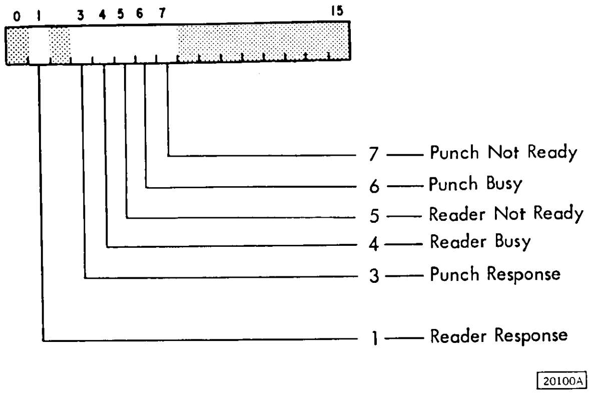


Figure 29. Paper Tape Device Status Word

Interrupts

Reader Response. This interrupt occurs when the reader has completed the execution of a Control command. This interrupt indicates to the CPU that a character is available to be entered into core storage.

Punch Response. This interrupt occurs when the punch has completed punching as directed by the execution of a Write command. It indicates that the punch can accept the next command.

Indicators

The following indicators can be entered into the ACC by a Sense Device command.

Punch Not Ready. This indicator is on when tape is not feeding freely from the tape spool, when the tape pressure roll holder is not down and holding the tape against the feed wheel, or when tape is not present. Manual intervention is required to clear these conditions. The indicator is also on if the punch is "busy." (See Punch Busy indicator.)

This indicator should always be tested by the program before a Write command is given. If a Write command is given while this indicator is on, loss of information will probably occur. No indication is given of this loss.

Reader Not Ready. This indicator is on when the tape tension switch is open. This condition exists when the paper tape is broken or not feeding freely. Manual intervention is required to clear these conditions. This indicator is also on if the reader is "busy." (See Reader Busy indicator.)

The program should test this indicator before a Read command is given. If a Read command is given while this indicator is on, erroneous data can be read into core storage. No valid indication can be given as to whether the data read is correct or incorrect.

Punch Busy. This indicator is on for the total time the punch is mechanically engaged and punching a character (68 ms). During this time the punch should not be sent another Write command.

Reader Busy. This indicator is on from the time a Control command (Start Paper Tape Reader) is given until data is available. A Reader Response interrupt signals that data is available.

IBM 1627 PLOTTER

The IBM 1627 Plotter (Figure 30) provides an exceptionally versatile, reliable, and easy-to-

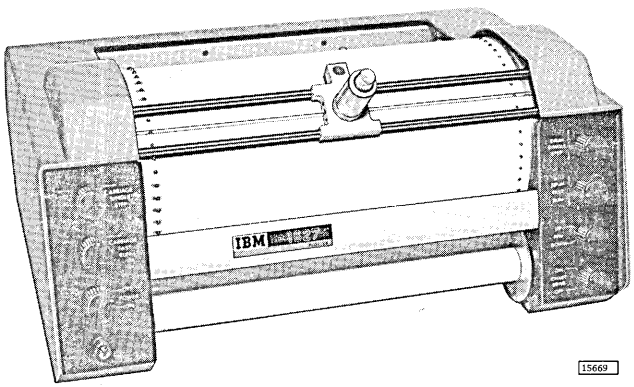


Figure 30. IBM 1627 Plotter

operate plotting system for the IBM 1130 Computing System. The plotter converts tabulated digital information into graphic form. Bar charts, flow charts, organization charts, engineering drawings, and maps are among the many graphic forms of data which can be plotted on the 1627 Plotter.

Two models of the 1627 are available and the major characteristics are as follows.

Model 1 – Plotting area: 11 inches by 120 feet, 1/100-inch incremental-step size, 18,000 steps/minute.

Model 2 – Plotting area: 29-1/2 inches by 120 feet, 1/100-inch incremental-step size, 12,000 steps/minute.

Figure 31 gives more information on both models.

		Model 1	Model 2
Speed	X, Y Increments	18,000 Steps/Min	12,000 Steps/Min
	Pen Status Change	600 Operations/Min	600 Operations/Min
Increment Size		1/100 Inch	1/100 Inch
Chart Paper	Width	12 Inches	31 Inches
	Plotting Width	11 Inches	29 1/2 Inches
	Length	120 Feet	120 Feet
	Sprocket Hole Dimensions	.130 Inch Dia on 3/8 Inch Centers	.188 Inch Dia on 1 Inch Centers

15667

Figure 31. 1627 Operating Characteristics

OPERATION

Data from core storage is transferred serially to the 1627 under direct program control, where it is translated into 1627 actuating signals. These signals are then converted into drawing movements by the 1627 Plotter.

The actual recording is produced by incremental movement of the pen on the paper surface (y-axis) and/or the paper under the pen (x-axis). The pen is mounted in a carriage that travels horizontally across the paper, as viewed from the front of the Plotter. The vertical plotting motion is achieved by rotation of the pin feed drum, which also acts as a platen (Figure 32).

The drum and the pen carriage are bidirectional; that is, the paper moves up or down, and the pen moves right or left. Control is also provided to raise or lower the pen from or to the paper surface. The pen remains in the "raised" or "lowered" position until directed to change to the opposite status.

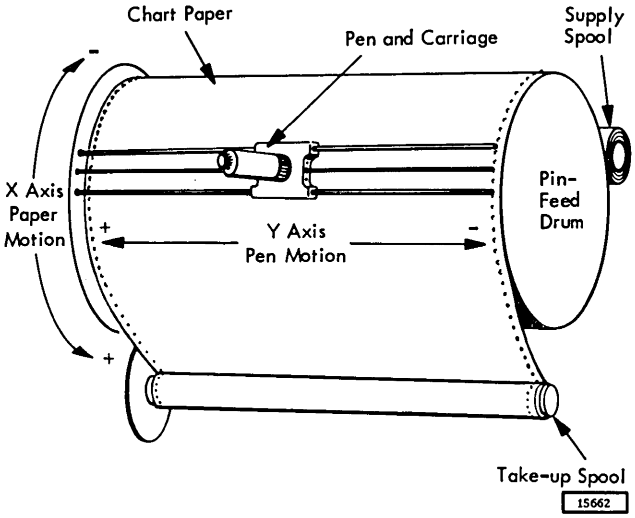


Figure 32. Plotter Paper and Pen Movements

The drum and pen-carriage movements and the pen status are controlled by digits transferred to the 1627. Each output word is decoded into a directional signal which causes a 1/100-inch incremental movement of the pen carriage (Figure 33) and/or paper, or a raise-pen or a lower-pen movement. The motion or action resulting from each word in the output record is shown in Figure 34.

The time required for execution of raise-pen and lower-pen commands is 100 ms. The time to plot a point is approximately 5 ms (3.3 ms for 300 steps/sec).

PROGRAMMING

The IBM 1627 Plotter operates under direct program control of the IBM 1130 Computing System.

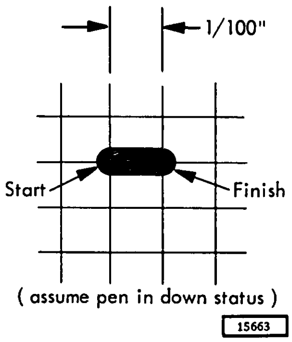


Figure 33. Result of One Horizontal (y-axis) Movement

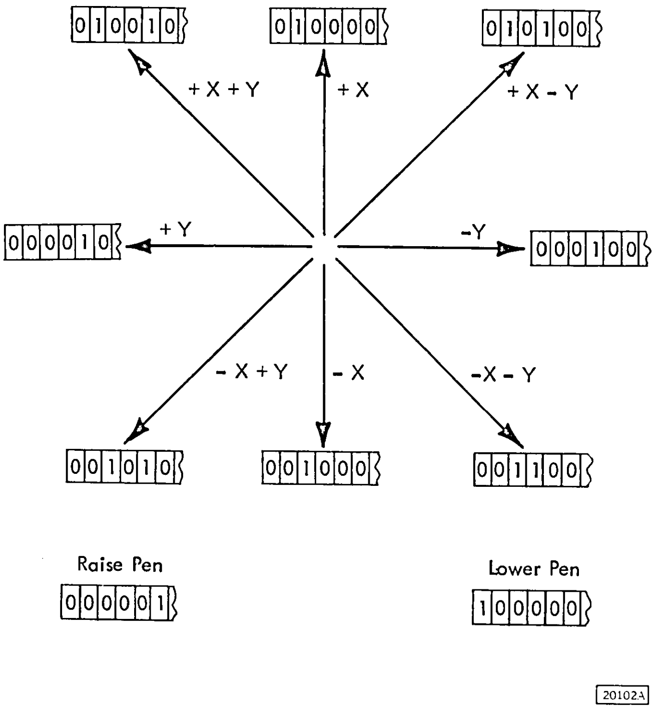
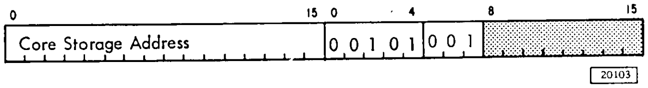


Figure 34. Plotter Control Codes

I/O Control Commands (IOCC)

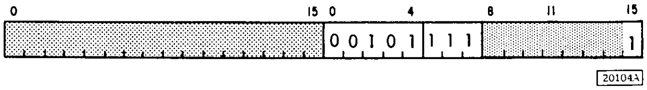
The 1627 is addressed by the 5-bit Device code of the IOCC.

Write (001)



This command causes bit positions 0 through 5 of the word in the core storage location specified by the Address to be sent to the 1627 to control the movement of the pen or drum.

Sense Device (111)



This command causes the 1627 Device Status Word (Figure 35) to be placed in the Accumulator. Modifier bit 15 on specifies reset for the plotter response.

Interrupt

There is only one interrupt associated with the 1627 attachment.

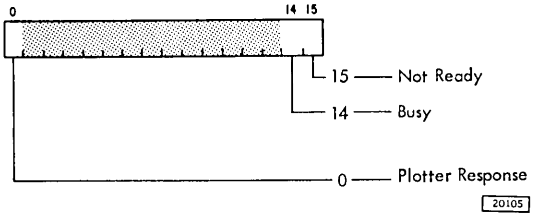


Figure 35. 1627 Device Status Word

Plotter Response. This interrupt occurs when the 1627 has completed the action specified by the last character transmitted by the Write command.

Indicators

Not Ready. When this indicator is off, it indicates that the 1627 has Power ON and can accept information.

Busy. This indicates that the 1627 is in a Busy status and cannot accept a character. After the first Write command the program should wait for succeeding plotter interrupts to initiate Write commands. If a Write command is given while Busy is on, loss of information will probably occur. No indication is given of this loss.

IBM 1132 PRINTER

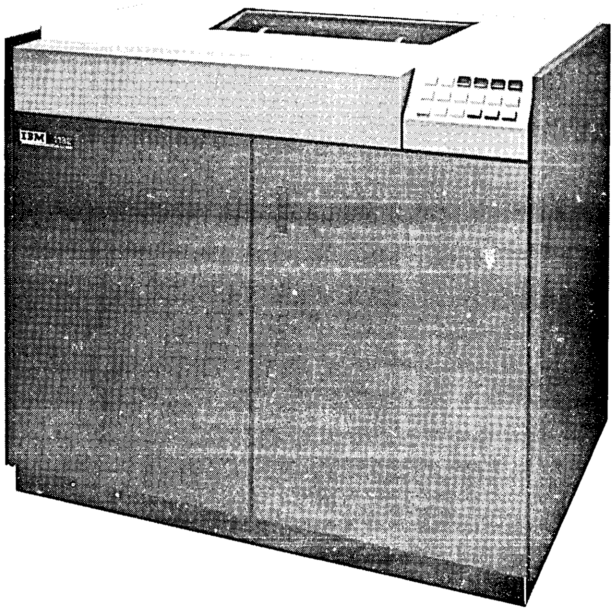
The 1132 Printer (Figure 36) provides printed output for the 1130 Computing System at maximum rates of 82 lpm (lines per minute) for alphameric printing and 110 lpm for numeric printing. The print line is 120 print positions long; horizontal spacing is 10 characters per inch. Vertical spacing of six or eight lines per inch can be selected by the operator.

FUNCTIONAL DESCRIPTION

The 1132 contains a printwheel with 48 alphabetic, numeric, and special characters for each of the 120 printing positions. Special (FORTRAN) characters are as follows:

& - / . $, * ( ) ' + =

Each wheel rotates continuously and moves forward to print when the data in the output record specifies that the character to be printed is in the print position. Thus, all similar characters for the entire line are printed on the same cycle. Forty-eight cycles are required to print the complete line.



23085

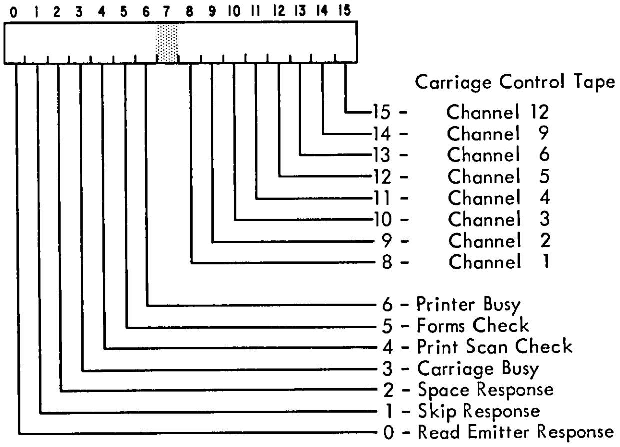
Figure 36. IBM 1132 Printer

Forms control is provided through a tape-controlled carriage that uses the standard IBM carriage tape. Channels 1 through 6, 9, and 12 are available to the stored program.

The 1132 uses interrupt circuitry and responds on level 1. The core storage address related to the interrupt level is 0009; the device code of the printer is 00110. When an interrupt occurs, the DSW (Figure 37) for the printer can be sensed directly because the 1132 is the only device on level 1.

Operation

The data to be printed is assembled in core storage in the same order, including spaces, as the line that is to be printed. During each of the 48 cycles necessary to print all 48 characters, the character next in position to print is read from the character emitter and is compared with each character of the output record. For each equal comparison, a 1-bit is put in the printer scan field in the position corresponding to the printwheel to be fired. The printer scans the field in a cycle-steal mode and fires each printwheel whose position contains a 1-bit. The printer scan field is located in core storage locations 0032 through 0039. The 16 bits of each of the first seven words and bits 0 through 7 of the eighth word represent the 120 printwheels.



20241

Figure 37. 1132 Device Status Word

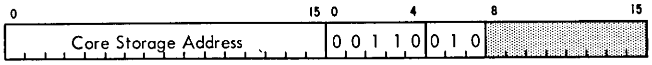
PROGRAMMING

The IBM 1132 Printer operates under direct program control of the CPU.

Printer Control Instructions

The 1132 Printer is addressed by the binary device code of 00110.

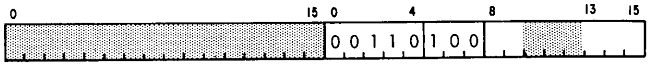
Read Emitter (010)



20239

This instruction causes the 8-bit code of the next character to be printed, emitted by the printer, to be read into the core storage location specified.

Control (100)



20240A

This command causes the execution of the function specified by the modifier bit. A 1-bit in the position indicated in parentheses after each instruction causes the operation described.

Start Printer (Bit 8). This causes the printer to start taking the printer scan field information. The printer continues to take print scan cycles at 11.2ms intervals until it receives a Stop Printer command. Each position that contains a 1-bit causes the corresponding printwheel to print the character in position on that cycle. After the field of eight words has been scanned, a 1-bit is placed in bit position 0 of the 1132 Device Status Word. (See Figure 37.) This causes an interrupt when level 1 is the highest level waiting.

Stop Printer (Bit 9). This instruction causes the printer to be put in a ready (not-busy) state and inhibits subsequent printer interrupts. The Stop Printer instruction should not be given until all of the following conditions are met:

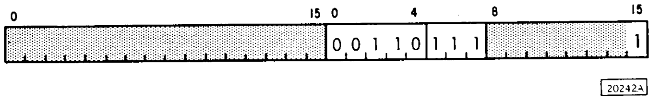
- 18 scan cycles have been completed after the command to print the last character.
- The carriage has stopped after a skip operation.
- The interrupt response from the last space command has occurred.

Start Carriage (Bit 13). This command initiates a skip operation, which is halted by a Stop Carriage instruction.

Stop Carriage (Bit 14). This command stops the carriage at the end of a Skip operation. A punch in carriage control tape channel 1, 2, 3, 4, 5, 6, 9, or 12 initiates an interrupt request, identified by Bit 1 of the DSW. When the desired tape channel bit in the DSW is on, Stop Carriage command should be given.

Space (Bit 15). This command is given to space the carriage one line after a line is printed. After the space operation, an interrupt is initiated and a 1-bit is put in bit position 2 of the DSW to indicate spacing is completed. Another space can now be initiated.

Sense Device (111)



This instruction causes the DSW of the 1132 Printer to be placed in the ACC. The functions of the bit positions of the DSW are shown in Figure 37.

Programming Notes

Prior to initiating a Start Printer instruction, the program should set the printer scan field (0032-0039) to zeros to ensure that no erroneous bits are sensed on the first scan.

Before the first line of a record is printed, the status of the 1132 indicators should be checked. This is done by bringing the printer DSW into the ACC with a Sense Device instruction. The modifier bits of the Sense Device instruction should be set to zeros to prevent reset of the DSW responses and indicators. Bits 3, 5, and 6 (see Figure 37) of the DSW are tested. If all three positions are set to zero, the printer is ready to print the first line.

After the code of the next character has been emitted by the printer and read into core storage by a read emitter instruction, 11.2 ms (milliseconds) are available to scan the output record and set up the 1-bits in the printer scan field. At the end of 11.2 ms, the printer begins its scan and fires each printwheel represented by a 1-bit. If the program has been interrupted for a considerable period by level 0, the programmed scan may not have been completed. To ensure that the program is aware of this condition, the first steps of the scan program for each character should clear the printer scan field to zeros and, upon completion of the programmed scan, place a bit in position 15 of the eighth word (0039). When the printer scans the field it checks this position. If it is zero, the scan incomplete indicator (bit 4 of the DSW) is turned on. The program can test this indicator and branch to an error routine that provides for 47 idle scan cycles and a resumption of programmed scanning with the character not completed.

After the final scan cycle for a line of printing, 16 idle scan cycles must be taken before spacing or skipping can be started. If the operation is a single or double space, the next scan cycle can be started two scan cycles after the space operation is initiated. If the spacing operation is for more than three spaces, scan cycles for the next print line can be started after the last space command is given.

After each printer scan cycle, a 1-bit is placed in bit position 0 of the 1132 DSW, causing an interrupt when level 1 is the highest pending level. During an idle scan cycle the printer scan field should be set to zeros, except for bit 15 of the eighth word. This prevents the incomplete scan indicator from being turned on.

APPENDIX A. NUMBER SYSTEMS

NUMBER CONCEPT

The concept of assigning a symbol to represent a value or a quantity has been important to man since the earliest attempts to communicate. As life became more complex, the need for symbols to represent more than one or two and be more precise than "many" became evident. Early counting methods, based on the ten fingers on both hands, evolved into the decimal system, which is in common use today.

The decimal system is built around the base ten and uses the 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 symbols. Combining these symbols and the place system for their arrangement, any number can be expressed, no matter how large or how small. The value of each symbol depends on its place in a row of symbols. For example, the symbol 1, by itself, has a place value of 1. Combined with another symbol, as in 21, the 1 symbol still has a place value of 1. Reverse the symbols, however, (12) and the 1 symbol now has a place value of 10.

This concept can be readily applied to any other number system. For example, imagine a number system containing only the symbols 0, 1, 2, 3, and 4. Since there are five symbols used, the system is called quinary or, more commonly, a base 5 system. To count in this system, the first symbol used is the 0. This is followed by the 1, 2, 3, and 4. At this point, all five symbols have been used. The next step is to assign the decimal value of five to the 1 symbol by placing it one position to the left and combining it with the 0 symbol (10). This combination is then followed by the 11, 12, 13, and 14 combinations. The third symbol in the system (2) is then assigned the decimal value of ten and is combined with the 0 giving the combination 20. This is followed by 21, 22, 23, 24, 30, and so forth.

The following table shows the arrangement of symbols used to represent the same values in each system of notation.

Decimal	Quinary	Decimal	Quinary
0	0	10	20
1	1	11	21
2	2	12	22
3	3	13	23
4	4	14	24
5	10	15	30
6	11	16	31
7	12	17	32
8	13	18	33
9	14	19	34

The main difficulty in using an unfamiliar number system is recognizing the new values assigned to familiar symbols. For example, to add the decimal symbols 3 and 4 and get a decimal result of 7 is simple for anyone acquainted with the decimal system. To add the quinary symbols 3 and 4 and get a quinary result of 12 is more difficult because of limited use of the quinary system.

Arithmetic Tables

The construction of arithmetic tables makes operations faster and easier. Table 2 shows sample add tables for both decimal and quinary systems.

To use these tables, the symbols being added (3 and 4 in the decimal table) are located, one on the top and the other on the left side of the table. Lines are then projected until they meet. The value at the intersection is the result of the addition. Using the quinary table: 4 + 3 = 12, 11 + 4 = 20, 2 + 4 = 11, and so forth. The results are expressed in quinary values.

The same principle may be applied to other arithmetic processes. Multiply tables for both systems are shown in Table 3. The use of these tables is the same as with the add tables; only the results differ. For example, 3 x 4 with the decimal table gives the result of 12, while 3 x 4 with the base 5 table gives the result 22; both results represent the same quantity.

Table 2. Add Tables

Decimal

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	6
2	2	3	4	5	6	7
3	3	4	5	6	7	8
4	4	5	6	7	8	9
5	5	6	7	8	9	10
6	6	7	8	9	10	11

Quinary

	0	1	2	3	4	10
0	0	1	2	3	4	10
1	1	2	3	4	10	11
2	2	3	4	10	11	12
3	3	4	10	11	12	13
4	4	10	11	12	13	14
10	10	11	12	13	14	20
11	11	12	13	14	20	21

Table 3. Multiply Tables

Decimal

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	6	8
3	0	3	6	9	12
4	0	4	8	12	16

Quinary

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

Binary Mode

Computers function in what is called a binary mode. This term simply means that the computer components can indicate only two possible states or conditions. Therefore, the binary mode system may also be called a base 2 system. For example, the ordinary light bulb operates in a binary mode; it is either on, producing light; or it is off, not producing light. The presence or absence of light indicates whether the bulb is on or off. Likewise, within the computer, transistors are either conducting or not conducting, magnetic materials are magnetized in one direction or in the opposite direction; and specific voltage potentials are present or absent (Figure 38). The binary modes of operation of the components are signals to the computer, as the presence or absence of light from an electric light is to a person.

Representing data within the computer is accomplished by assigning or associating a specific value to a binary indication or group of binary indications. For example, a device to represent values could be designed with four electric light bulbs and switches to turn each bulb on or off (Figure 39).

The bulbs are assigned arbitrary values of 1, 2, 4, and 8. When a light is on, it represents the value associated with it. When a light is off, the value is not considered. With such an arrangement, the single value represented by the four bulbs will be the numeric sum indicated by the lighted bulbs.

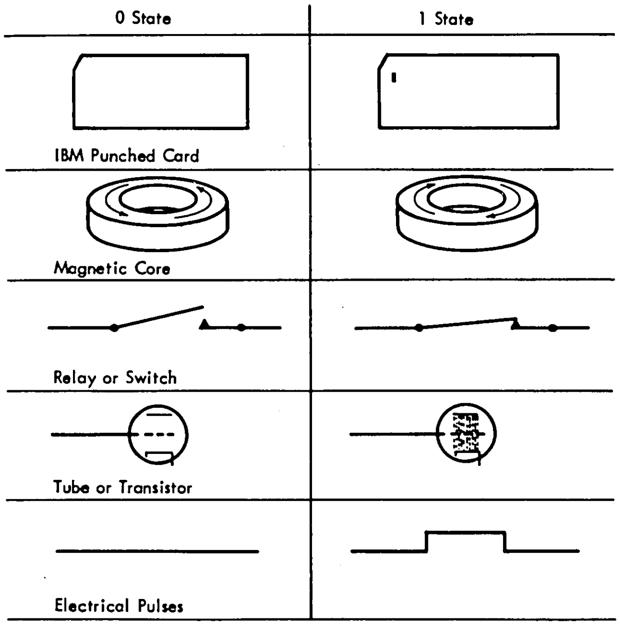


Figure 38. Binary Indicators

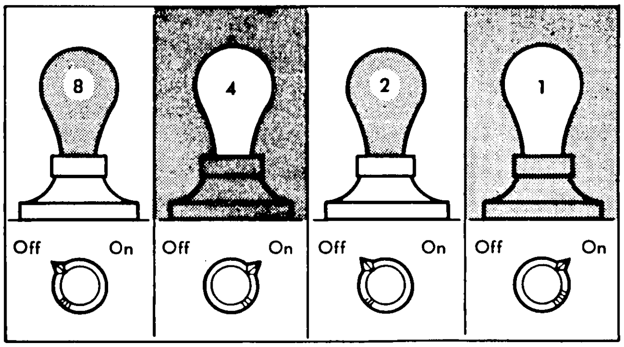


Figure 39. Decimal Data Representation

Values 0 through 15 can be represented. The value 0 is represented by all lights off; the value 15, by all lights on; 9, by having the 8 and 1 lights on and the 4 and 2 lights off; 5, with the 1 and 4 lights on and the 8 and 2 lights off; and so on.

The value assigned to each bulb or indicator in the example could have been something other than the values used. This change would involve assigning new values and determining a scheme of operation. In a computer, the values assigned to a specific number of binary indications become the code or language for representing data.

Because binary indications represent data within a computer, a binary method of notation is used to illustrate these indications. The binary system of notation uses only two symbols, zero (0) or one (1), to represent all quantities. In any single position of binary notation, the 0 represents the absence of a related or assigned value and the 1 represents the presence of a related or assigned value. Using the light bulbs in Figure 32, for example, the binary notation 0101 would represent a decimal 5.

The binary notations 0 and 1 are commonly called bits. The 0 bit is described as no bit and the 1 bit is described as a bit. Although 0 or 1 bits are necessary to illustrate the condition of a binary indication or a group of binary indications, the 1 bits are the bits generally referred to. For example, the binary notation 0101 of Figure 32 would be described as having a bit in the 1 and 4 bit positions. The assumption is that there are no bits (0 bits) in the 2 and 8 bit positions.

Binary Number System

In some computers, the values associated with the binary notation are related directly to the binary number system. This system is not used in all computers, but the method of representing values using this numbering system is useful in learning the general concept of data representation.

The common decimal number system uses ten symbols or digits to represent all quantities, and the place value of the digits signifies units, tens, hundreds, thousands, and so on. The binary or base 2 number system uses only two symbols or digits: 0 and 1. The position value of the bit symbols (0 or 1) is based on the progression of powers of 2; the units position of a binary number has the value of 1; the next position, a value of 2; the next, 4; the next, 8; the next, 16; and so on (Figure 40).

8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Figure 40. Place Value of Binary Numbers

In pure binary notation, the binary digits or bits indicate whether the corresponding power of 2 is absent or present in each position of the number. The 1 bit represents the presence of the value and the 0 bit represents the absence of the value. The place value of the digits does not signify units, tens, hundreds, or thousands, as in the decimal system; instead, the place value signifies units, twos, fours, eights, sixteens, and so on. Using this system, the quantity 12, for example, is expressed with the symbols 1100, meaning $(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0)$ or $(1 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1)$.

Figure 41 shows the binary representation of the decimal values 0 through 9. Note that the decimal digits 0 through 9 are expressed by four binary digits. The system of coding or expressing decimal digits in an equivalent binary value is called binary coded decimal (BCD). For example, the decimal digits 2, 6, 5, 4, 9, and 8 would appear in binary coded decimal form as shown in Figure 42

Decimal Value	Place Value			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Figure 41. Binary Representations

Although binary numbers, in general, have more terms than their decimal counterparts (about 3.3 times as many), computation in the binary system is quite simple.

For addition, it is only necessary to remember three rules:

1. Zero plus zero equals zero.
2. Zero plus one equals one.
3. One plus one equals zero with a carry of one to the next position on the left.

To see how the rules work, consider the addition of 15 + 7, with these numbers expressed in binary notation:

	Sixteens	Eights	Fours	Twos	Ones	
(Carries)	(1)	(1)	(1)	(1)		
	0	1	1	1	1	= 15
	0	0	1	1	1	= 7
	1	0	1	1	0	= 22

In the ones column, we have 1 + 1 for a sum of 0 and a 1 carried to the twos column. In the twos column, we have 1 + 1 for a sum of 0, but we must also add the carry from the ones column, making a final sum of 1 with a carry to the fours column. In the eights column, we have a 1 + 0 giving a sum of 1, but adding in the carry from the fours column makes the final sum 0 with a carry to the sixteens column. In this column, we have 0 + 0, giving a sum of 0 and to this we add the carry from the eights column, making a final sum of 1.

The resultant sum of the addition contains 1's in the sixteens, fours, and twos column, which is the binary representation of 22, the correct sum of 15 plus 7 (16 + 4 + 2 = 22).

The rules for subtraction of binary digits are equally simple:

1. Zero minus zero equals zero.
2. One minus one equals zero.

Decimal Digits	2				6				5				4				9				8			
Binary Value	0	0	1	0	0	1	1	0	0	1	0	1	0	1	0	0	1	0	0	1	1	0	0	0
Place Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1

Figure 42. Binary Coded Decimal

3. One minus zero equals one.
4. Zero minus one equals one, with one borrowed from the left.

Using the same numbers as we did in the addition, the subtraction is:

	SIXTEENS	EIGHTS	FOURS	TWOS	ONES	
(Borrows)	(0)	(0)	(0)	(0)	(0)	
	0	1	1	1	1	= 15
	− 0	0	1	1	1	= 7
	0	1	0	0	0	= 8

In the ones column we have 1 - 1 for a sum of 0 with no borrows. The same procedure occurs in the twos and fours columns. In the eights column, we have 1 - 0 for a sum of 1. In the sixteens column, we have 0 - 0 for a sum of 0. With the subtraction finished, we have 1's in the eights column only, signifying the answer to be 8.

For multiplication, only three rules are needed:

1. Zero times zero equals zero.
2. Zero times one equals zero; no carries are considered.
3. One times one equals one.

In the binary multiplication table, all that is necessary when multiplying one number (multiplicand) by another (multiplier) is to examine the multiplier digits one at a time and, each time a 1 is found, add the multiplicand into the result, and each time a 0 is found add nothing. The multiplicand must be shifted for each multiplier digit, but this is no different from the shifting done in the decimal system.

An example of binary multiplication is 26 x 19:

DECIMAL		BINARY
26 = 16 + 8 + 0 + 2 + 0	=	11010
× 19 = 16 + 0 + 0 + 2 + 1	=	10011
Using the rules, the product is		11010
arrived at by a series		11010
of adding the multiplicand		00000
and shifting whenever		00000
a 1 is in the		11010
multiplier.		111101110

Interpreting the binary result of the multiplication by using the ones, twos, fours,...etc. system, we find:

256 + 128 + 64 + 32 + 0 + 8 + 4 + 2 + 0

which equals 494, proving the problem.

Binary division is accomplished by applying similar concepts. From the examples of addition, subtraction, and multiplication, you can see that whatever operation the computer is working on is accomplished by repetitive addition.

The computer uses the binary system internally. However, it is able to convert one system to another by using a stored program. Thus, input/output data may be expressed in decimal (or any other) form when the programmer finds it convenient to do so.

Hexadecimal Number System

It has been noted that binary numbers require about three times as many positions as decimal numbers to express the equivalent number. This is not much of a problem to the computer; however, in talking and writing or in communicating with the computer, these binary numbers are bulky. A long string of 1's and 0's cannot be effectively transmitted from one individual to another. Some shorthand method is necessary.

The hexadecimal number system fills this need. Because of the simple relationship of hexadecimal to binary, numbers can be converted from one system to another by inspection. The base or radix of the hexadecimal system is 16. This means there are 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The letters A, B, C, D, E, and F represent the 10-base system values of 10, 11, 12, 13, 14, and 15, respectively.

Four binary positions are equvalent to one hexadecimal position. The following table shows the comparable values of the three number systems.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D

Decimal	Binary	Hexadecimal
14	1110	E
15	1111	F

At this point, all 16 symbols have been used, and a carry to the next higher position of the number is necessary.

16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
19	0001 0011	13
20	0001 0100	14
21	0001 0101	15

Remember that as far as the internal circuitry of the computer is concerned, it understands only binary. But an operator can look at a series of lights on the computer console showing binary 1's and 0's, for example, 0001 1110 0001 0011, and say that the lights represent the hexadecimal value 1E13, which is easier to state than the string of 1's and 0's.

NUMBER CONVERSIONS

Before converting numbers from one system to another, it is best to review what a number represents. In the decimal system, a number is represented or expressed by a sum of terms. Each individual term consists of a product of a power of ten and some integer from 0 to 9. For example, the number 123 means 100 plus 20 plus 3. This may also be expressed as:

$$(1 \times 10^2) + (2 \times 10^1) + (3 \times 10^0)$$

Ten is said to be the base or radix of this system. Radix is defined as an integer used in a system of notation whereby all numbers are expressed as powers of the integer. In the decimal system, the radix is 10; in the binary system, it is 2.

The base, or radix, of a number is expressed as a subscript. For example, 123_{10} is a decimal (base-10) number. Thus, 1101001_2 is a binary number, and 437_{16} is a hexadecimal number. The subscript is usually eliminated when the base is obvious.

If 2 is chosen as the base, numbers are said to be represented in the binary system. Consider the number 1 111 011. What do these zeros and ones represent? They represent the coefficients of the ascending powers of 2. Expressed in another way the number is:

$$(1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3)$$
$$+(0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

The places do not have the meaning of units, tens, hundreds, thousands, etc., as in the decimal system; instead they signify units, twos, fours, eights, sixteens, etc. In applying the above information, the decimal number 123 breaks down in both systems as:

```
DECIMAL                    BINARY

1 2 3                      1 1 1 1 0 1 1
| | └─3(units)             | | | | | | └─1(units)
| └──20(tens)              | | | | | └───2(twos)
└───100(hundreds)          | | | | └─────0(fours)
    123₁₀                  | | | └───────8(eights)
                           | | └────────16(sixteens)
                           | └──────────32(thirty-twos)
                           └──────────64(sixty-fours)
                                      123₁₀
```

In the hexadecimal system, a number is represented in the same manner, except that the base is 16. The digits of the number represent the coefficients of the ascending powers of 16. Consider the hexadecimal number:

$$75_{16} = 7\,(16^1) + 5\,(16^0)$$
$$= 112 + 5$$
$$= 117_{10}$$

Similarly:

```
Hexadecimal 75
            |└─5 (units)
            └─112 (sixteens)
              117₁₀
```

By remembering what a number represents in the binary or hexadecimal system, you can convert the number to its decimal equivalent by the method shown. As the numbers get bigger, this method becomes quite impractical. The following section provides detailed methods for converting from one system to another.

Integers

Appendix F is a table for conversion of decimal to hexadecimal and hexadecimal to decimal. The use of the table is explained in a later section. This and the following sections provide conversion methods if the table is not available or the number is beyond the range of the table.

Decimal to Hexadecimal

Rule: Divide the decimal number by 16 and develop the remainders in low-order to high-order sequence for the hexadecimal number.

Example. Convert 839_{10} to its hexadecimal equivalent.

```
        0 r = 3
    16)3  r = 4
   16)52  r = 7
  16)839
```

$$839_{10} = 347_{16}$$

The original number to be converted is divided by 16. The remainder of this first division becomes the low-order digit of the conversion (7). The quotient (received from the first division) is then divided by 16. Again the remainder becomes a part of the answer (next higher order, 4). This method is continued until the quotient is smaller than the divisor. The final quotient is considered the high order of the conversion (3).

Hexadecimal to Decimal

Rule. Multiply high-order hexadecimal digit by 16 and add next low-order digit to result; multiply sum by 16 and add next low-order digit. Repeat process until low-order digit of number has been added.

Example. Convert 347_{16} to its decimal equivalent.

```
  347
  X16
   48
  + 4
   52
  x16
  832
  + 7
  839(10)
```

The high-order digit is multiplied by 16 and the next lower-order digit is added to the result. The resultant answer is then multiplied by 16 and the next lower-order digit is added to the result. When the low-order digit has been added to the answer, the process ends.

Alternate Method, Expansion.

$$\begin{aligned} 347_{16} &= 3\,(16^2) + 4\,(16^1) + 7\,(16^0) \\ &= 768 + 64 + 7 \\ &= 839_{10} \end{aligned}$$

NOTE: In the following examples, where multiplication or division is used, detailed explanations are not given because the operations are similar to those previously described.

Hexadecimal to Binary

Rule. Express each hexadecimal number as a binary group of four bits.

Example. Convert $7E9_{16}$ to its binary equivalent.

Hexadecimal =	7	E	9
Binary =	0111	1110	1001

Binary to Hexadecimal

Rule. Express each binary group of four bits as a hexadecimal digit.

Example. Convert 011111101001_2 to its hexadecimal equivalent.

Binary =	0111	1110	1001
Hexadecimal =	7	E	9

Decimal to Binary

Rule. Divide the decimal number by 2 and develop the remainders in low-order to high-order sequence for the equivalent hexadecimal number.

Example. Convert 149_{10} to its binary equivalent.

```
       0 r = 1
    2)1 r = 0
    2)2 r = 0
    2)4 r = 1
    2)9 r = 0
   2)18 r = 1
   2)37 r = 0
   2)74 r = 1
  2)149
```

$$149_{10} = 10010101_2$$

Binary to Decimal

Rule. Multiply high-order binary digit by 2 and add next low-order digit to result; multiply sum by 2 and add next low-order digit. Repeat process until low-order digit of binary number has been added.

Example. Convert 10010101_2 to its decimal equivalent.

```
  1   0   0   1   0   1   0   1
 x 2
  2
 + 0
  2
 x 2
  4
 + 0
  4
 x 2
  8
 + 1
  9
 x 2
 18
 + 0
 18
 x 2
 36
 + 1
 37
 x 2
 74
 + 0
 74
 x 2
148
 + 1
149₁₀
```

Alternate Method, Expansion.

$$10010101_2 = 1(2^7) + 0(2^6) + 0(2^5) + 1(2^4) + 0(2^3) + 1(2^2) + 0(2^1) + 1(2^0)$$

$$= 128 + 16 + 4 + 1$$

$$= 149_{10}$$

Fractions

In a preceding section, Number Conversions, we reviewed what a number represents, i.e., in the decimal system, a number is the sum of its radix to the power of each position, multiplied by the value of that position. Fractions follow the same rules. The decimal fraction .123 can also be expressed as 1 tenth plus 2 hundredths plus 3 thousandths. Since a negative coefficient is the fractional power of the base, a tenth can be written 10^{-1} and a hundredth as 10^{-2}. Thus, it follows that

$$.123_{10} = 1(10^{-1}) + 2(10^{-2}) + 3(10^{-3})$$

$$= 1(\frac{1}{10}) + 2(\frac{1}{100}) + 3(\frac{1}{1000})$$

$$= \frac{100 + 20 + 3}{1000} = .123_{10}$$

By the same reasoning, the binary number .00100101 can be represented in the decimal system as

$$0(2^{-1}) + 0(2^{-2}) + 1(2^{-3}) + 0(2^{-4}) + 0(2^{-5}) + 1(2^{-6}) + 0(2^{-7}) + 1(2^{-8}).$$

With the coefficients converted to fractions, the binary number becomes

$$\frac{1}{8} + \frac{1}{64} + \frac{1}{256} = \frac{32 + 4 + 1}{256} = \frac{37}{256} = .145_{10}$$

In the hexadecimal system, the base of 16 with negative coefficients represents the same number in decimal notation. Thus, $.581_{16}$ may be stated as

$$5(16^{-1}) + 8(16^{-2}) + 1(16^{-3}) = \frac{5}{16} + \frac{8}{256} + \frac{1}{4096} = \frac{1409}{4096} = .344_{10}$$

Note that the hexadecimal fraction is always larger than the decimal equivalent; the opposite is true with integers.

Decimal to Hexadecimal

Rule. Multiply the fractional part of the decimal number by 16 and develop intergers as successive terms of the hexadecimal fraction. The hexadecimal fraction should be carried out to the number of places contained in the decimal fraction.

Example. Convert $.344_{10}$ to its hexadecimal equivalent.

$$\begin{array}{r} .344 \\ \underline{\times\ 16} \\ 5.504 \\ \underline{\times\ 16} \\ 8.064 \\ \underline{\times\ 16} \\ 1.004 \end{array}$$

$$.344_{10} = .581_{16}$$

Hexadecimal to Decimal

Rule. Express the hexadecimal numbers as powers of 16 (expansion).

Sample. Convert $.581_{16}$ to its decimal equivalent.

$$\begin{aligned} .581_{16} &= 5\,(16^{-1}) + 8\,(16^{-2}) + 1\,(16^{-3}) \\ &= \frac{5}{16} + \frac{8}{256} + \frac{1}{4096} \\ &= \frac{1409}{4096} \\ &= .344_{10} \end{aligned}$$

Hexadecimal/Binary Conversions

Rules for conversions of fractions from hexadecimal to binary, and from binary to hexadecimal are the same as for integers.

Decimal to Binary

Rule. Multiply the fractional part of the decimal number by 2 and develop integers as successive terms of the binary fraction. The binary fraction should be carried out to three times the number of places in the decimal fraction.

Example. Convert $.145_{10}$ to its binary equivalent. (The example is carried to only eight places because the ninth term is obviously zero.)

$$\begin{array}{l|r} \text{Read} & .145 \\ \downarrow & \underline{\times\ 2} \\ & 0.290 \\ & \underline{\times\ 2} \\ & 0.580 \\ & \underline{\times\ 2} \\ & 1.160 \\ & \underline{\times\ 2} \\ & 0.320 \\ & \underline{\times\ 2} \\ & 0.640 \\ & \underline{\times\ 2} \\ & 1.280 \\ & \underline{\times\ 2} \\ & 0.560 \\ & \underline{\times\ 2} \\ \downarrow & 1.120 \end{array}$$

$$.145_{10} = .00100101_2$$

Binary to Decimal

Rule. Express the binary numbers as powers of 2 (expansion).

Example. Convert $.00100101_2$ to its decimal equivalent.

$$\begin{aligned} .00100101_2 &= 2^{-3} + 2^{-6} + 2^{-8} \\ &= \frac{1}{8} + \frac{1}{64} + \frac{1}{256} \\ &= \frac{37}{256} = .145_{10} \end{aligned}$$

Improper Fractions

Decimal/Binary and Decimal/Hexadecimal

Since improper fractions are a combination of integers and fractions, the elements of the terms are converted separately and then combined for conversion from decimal to binary, binary to decimal, decimal to hexadecimal, and hexadecimal to decimal.

The example that follows, concerns decimal to binary conversion to illustrate the principle involved.

Example. Convert 149.145_{10} to its binary equivalent. The integer 149_{10} is converted to 10010101_2, as described under "Decimal to Binary" in the Integers section; the fraction $.145_{10}$ is converted to $.00100101_2$, as described under "Decimal to Binary" in the Fractions section. The combined result of the two conversions is:

$$149.145_{10} = 10010101.00100101_2$$

Hexadecimal/Binary

The conversion of improper fractions from hexadecimal to binary and from binary to hexadecimal is the same as described for integers. The numbers to be converted are simply expressed in terms of the new base.

Example. Convert $7E9.4AC_{16}$ to its binary equivalent.

Hexadecimal =	7	E	9	.	4	A	C
Binary =	0111	1110	1001	.	0100	1010	1100

FIXED-POINT AND TWO's COMPLEMENT NOTATION

A fixed-point number is a signed value, recorded as a binary integer. It is called fixed point because the programmer determines the fixed positioning of the binary point.

Fixed-point operands may be recorded in single precision words (16 bits) or double precision words (32 bits). In both, the first bit position (0) holds the sign of the number, with the remaining bit positions used to designate the magnitude of the number.

Positive fixed-point numbers are represented in true binary form with a zero sign bit. Negative fixed-point numbers are represented in two's complement notation with a one bit in the sign position. In all cases, the bits between the sign bit and the left-most significant bit of the integer are the same as the sign bit (i. e. all zeros for positive numbers, all ones for negative numbers).

Negative fixed-point numbers are formed in two's complement notation by inverting each bit of the positive binary number and adding one. For example, the true binary form of the decimal value (plus 26) is made negative (minus 26) in the following manner:

```
                   S  INTEGER
+ 26               0 0000000 00011010
Invert             1 1111111 11100101
Add 1                                1
                   ------------------
-26                1 1111111 11100110  (Two's complement form)
```

This is equivalent to the subtraction,

```
   1 00000000 00000000
 -   00000000 00011010
   -------------------
     11111111 11100110
```

The following addition examples illustrate two's complement arithmetic and the operation of the carry and overflow indicators. To simplify the illustrations, only eight bit positions are used. All negative numbers are in two's complement form.

```
   (+) 41  =  00101001
+  (+) 74  =  01001010
-----------------------
   (+)115  =  01110011
```

No overflow and no carry out: sum is correct

```
   (+) 84  =  01010100
   (+) 74  =  01001010
-----------------------
   (+) 158 ≠  10011110
```

Overflow into sign position, with no carry out: sum is not correct.

```
   (-) 41  =  11010111
+  (-) 74  =  10110110
-----------------------
   (-)115  =  10001101
```

Overflow into sign position resulted in carry out: sum is correct.

```
   (-) 84  =  10101011
+  (-) 74  =  10111110
-----------------------
   (-) 158 ≠  01100001
```

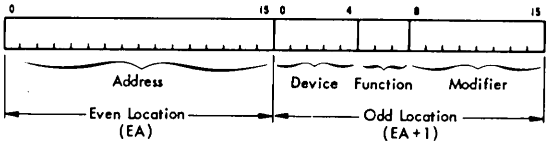
Carry out, with no overflow into sign position: sum is not correct.

Since subtraction is performed by changing the sign of the subtrahend and adding, the above examples also illustrate that function.

I/O Function Codes and Modifiers

I/O Device (Code) Instruction	Function Code	Modifier Bits		
		Bit No.	Bit	Function
Console Printer (00001)				
Write	001			
Sense Device	111	15	1	Reset Int. Level 4 ind.
Console Keyboard (00001)				
Read	010			
Control (Interrupt)	100			
Sense Device	111	15	1	Reset Int. Level 4 ind.
1442 Card Read-Punch (00010)				
Read	010			
Write	001			
Control	100	8	1	Stacker Select
		13	1	Initiate Read
		14	1	Feed Cycle
		15	1	Initiate Punch
Sense Device	111	14	1	Reset Int. Level 4 ind.
		15	1	Reset Int. Level 0 ind.
1134 Paper Tape Reader (00011) 1055 Paper Tape Punch (00011)				
Read	010			
Write	001			
Control	100			
Sense Device	111	15	1	Reset Int. Level 4 ind.
Disk Storage (00100)				
Initiate Write	101	13-15		Sector address
Initiate Read	110	13-15		Sector address
		8	0	Read operation
		8	1	Read-check operation
Control	100	13	0	Move access forward
		13	1	Move access backward
Sense Device	111	15	1	Reset Int. Level 2 ind.
1627 Plotter (00101)				
Write	001			
Sense Device	111	15	1	Reset Int. Level 3 ind.
1132 Printer (00110)				
Read Emitter	010			
Control	100	8	1	Start Printer
		9	1	Stop Printer
		13	1	Start Carriage
		14	1	Stop Carriage
		15	1	Space Carriage
Sense Device	111	15	1	Reset Int. Level 1 ind.
Console Entry Switches (00111)				
Read	010			

IOCC Format



Value Ranges - Single Precision Word

Positive Binary Values	Powers of 2	Absolute Values		Negative Binary Values
Bit Positions 11 1111 0123 4567 8901 2345		Decimal Notation Base-10	Hexa-decimal Notation Base-16	Bit Positions 11 1111 0123 4567 8901 2345
0000 0000 0000 0000	-	0	0	No negative zero
0000 0000 0000 0001	0	1	1	1111 1111 1111 1111
0000 0000 0000 0010	1	2	2	1111 1111 1111 1110
0000 0000 0000 0100	2	4	4	1111 1111 1111 1100
0000 0000 0000 1000	3	8	8	1111 1111 1111 1000
0000 0000 0001 0000	4	16	10	1111 1111 1111 0000
0000 0000 0010 0000	5	32	20	1111 1111 1110 0000
0000 0000 0100 0000	6	64	40	1111 1111 1100 0000
0000 0000 1000 0000	7	128	80	1111 1111 1000 0000
0000 0001 0000 0000	8	256	100	1111 1111 0000 0000
0000 0010 0000 0000	9	512	200	1111 1110 0000 0000
0000 0100 0000 0000	10	1,024	400	1111 1100 0000 0000
0000 1000 0000 0000	11	2,048	800	1111 1000 0000 0000
0001 0000 0000 0000	12	4,096	1,000	1111 0000 0000 0000
0010 0000 0000 0000	13	8,192	2,000	1110 0000 0000 0000
0100 0000 0000 0000	14	16,384	4,000	1100 0000 0000 0000
0111 1111 1111 1111	-	32,767	7,FFF	1000 0000 0000 0001
No positive equivalent	15	32,768	8,000	1000 0000 0000 0000

I/O Device Codes and Interrupt Levels

Device Code	I/O Device	Interrupt	
		Level	Core Storage Address
00001	Console Keyboard	4	0012
	Console Printer	4	0012
00010	1442 Card Read-Punch	0	0008
		4	0012
00011	1134 Paper Tape Rdr	4	0012
	1055 Paper Tape Punch	4	0012
00100	Disk Storage	2	0010
00101	1627 Plotter	3	0011
00110	1132 Printer	1	0009
00111	Console Entry Switches		
	Program Stop Switch	5	0013

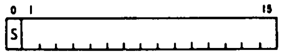
Reserved Core Storage Locations

Tag Bits	Core Storage Address	Description
00	--	Displacement
01	0001	Index Register 1
10	0002	Index Register 2
11	0003	Index Register 3
--	0008 - 0013	Interrupt Addresses
--	0032 - 0039	Printer Image Output

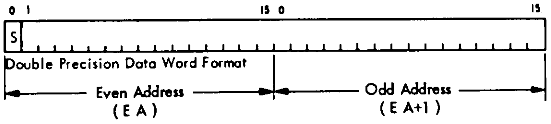
Value Ranges - Double Precision Word

Positive Binary Values	Powers of 2	Absolute Values		Negative Binary Values
Bit Positions 11 1111 1111 2222 2222 2233 0123 4567 8901 2345 6789 0123 4567 8901		Decimal Notation Base - 10	Hexidecimal Notation Base - 16	Bit Positions 11 1111 1111 2222 2222 2233 0123 4567 8901 2345 6789 0123 4567 8901
0000 0000 0000 0000 0000 0000 0000 0000	-	0	0	No negative zero
0000 0000 0000 0000 0000 0000 0000 0001	0	1	1	1111 1111 1111 1111 1111 1111 1111 1111
0000 0000 0000 0000 0000 0000 0000 0010	1	2	2	1111 1111 1111 1111 1111 1111 1111 1110
0000 0000 0000 0000 0000 0000 0000 0100	2	4	4	1111 1111 1111 1111 1111 1111 1111 1100
0000 0000 0000 0000 0000 0000 0000 1000	3	8	8	1111 1111 1111 1111 1111 1111 1111 1000
0000 0000 0000 0000 0000 0000 0001 0000	4	16	10	1111 1111 1111 1111 1111 1111 1111 0000
0000 0000 0000 0000 0000 0000 0010 0000	5	32	20	1111 1111 1111 1111 1111 1111 1110 0000
0000 0000 0000 0000 0000 0000 0100 0000	6	64	40	1111 1111 1111 1111 1111 1111 1100 0000
0000 0000 0000 0000 0000 0000 1000 0000	7	128	80	1111 1111 1111 1111 1111 1111 1000 0000
0000 0000 0000 0000 0000 0001 0000 0000	8	256	100	1111 1111 1111 1111 1111 1111 0000 0000
0000 0000 0000 0000 0000 0010 0000 0000	9	512	200	1111 1111 1111 1111 1111 1110 0000 0000
0000 0000 0000 0000 0000 0100 0000 0000	10	1,024	400	1111 1111 1111 1111 1111 1100 0000 0000
0000 0000 0000 0000 0000 1000 0000 0000	11	2,048	800	1111 1111 1111 1111 1111 1000 0000 0000
0000 0000 0000 0000 0001 0000 0000 0000	12	4,096	1,000	1111 1111 1111 1111 1111 0000 0000 0000
0000 0000 0000 0000 0010 0000 0000 0000	13	8,192	2,000	1111 1111 1111 1111 1110 0000 0000 0000
0000 0000 0000 0000 0100 0000 0000 0000	14	16,384	4,000	1111 1111 1111 1111 1100 0000 0000 0000
0000 0000 0000 0000 1000 0000 0000 0000	15	32,768	8,000	1111 1111 1111 1111 1000 0000 0000 0000
0000 0000 0000 0001 0000 0000 0000 0000	16	65,536	10,000	1111 1111 1111 1111 0000 0000 0000 0000
0000 0000 0000 0010 0000 0000 0000 0000	17	131,072	20,000	1111 1111 1111 1110 0000 0000 0000 0000
0000 0000 0000 0100 0000 0000 0000 0000	18	262,144	40,000	1111 1111 1111 1100 0000 0000 0000 0000
0000 0000 0000 1000 0000 0000 0000 0000	19	524,288	80,000	1111 1111 1111 1000 0000 0000 0000 0000
0000 0000 0001 0000 0000 0000 0000 0000	20	1,048,576	100,000	1111 1111 1111 0000 00000000 0000 0000
0000 0000 0010 0000 0000 0000 0000 0000	21	2,097,152	200,000	1111 1111 1110 0000 0000 0000 0000 0000
0000 0000 0100 0000 0000 0000 0000 0000	22	4,194,304	400,000	1111 1111 1100 0000 0000 0000 0000 0000
0000 0000 1000 0000 0000 0000 0000 0000	23	8,388,608	800,000	1111 1111 1000 0000 0000 0000 0000 0000
0000 0001 0000 0000 0000 0000 0000 0000	24	16,777,216	1,000,000	1111 1111 0000 0000 0000 0000 0000 0000
0000 0010 0000 0000 0000 0000 0000 0000	25	33,554,432	2,000,000	1111 1110 0000 0000 0000 0000 0000 0000
0000 0100 0000 0000 0000 0000 0000 0000	26	67,108,864	4,000,000	1111 1100 0000 0000 0000 0000 0000 0000
0000 1000 0000 0000 0000 0000 0000 0000	27	134,217,728	8,000,000	1111 1000 0000 0000 0000 0000 0000 0000
0001 0000 0000 0000 0000 0000 0000 0000	28	268,435,456	10,000,000	1111 0000 0000 0000 0000 0000 0000 0000
0010 0000 0000 0000 0000 0000 0000 0000	29	536,870,912	20,000,000	1110 0000 0000 0000 0000 0000 0000 0000
0100 0000 0000 0000 0000 0000 0000 0000	30	1,073,741,824	40,000,000	1100 0000 0000 0000 0000 0000 0000 0000
0111 1111 1111 1111 1111 1111 1111 1111	-	2,147,483,647	7F,FFF,FFF	1000 0000 0000 0000 0000 0000 0000 0001
No positive equivalent	31	2,147,483,648	80,000,000	1000 0000 0000 0000 0000 0000 0000 0000

Single Precision Data Word Format



Double Precision Data Word Format

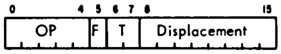


Effective Address Computation

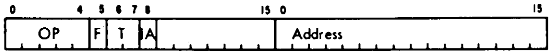
	F = 0 (Direct Addressing)	F = 1, IA = 0 (Direct Addressing)	F = 1, IA = 1 (Indirect Addressing)
T = 00	EA = Disp + IAR	EA = Add	EA = C/Add
T = 01	EA = Disp + XR1	EA = Add + XR1	EA = C/Add + XR1
T = 10	EA = Disp + XR2	EA = Add + XR2	EA = C/Add + XR2
T = 11	EA = Disp + XR3	EA = Add + XR3	EA = C/Add + XR3

Disp = Contents of Displacement field of instuction.
Add = Contents of Address field of instruction.
C = Contents of Location specified by Add or Add + XR.

Short Instruction Format



Long Instruction Format



Tag Bit Codes

Instruction	Tag Bits	Register/Operation
Load Index Store Index	00	IAR
	01	XR1
	10	XR2
	11	XR3
Shift Left Shift Right	00	Disp.
	01	XR1
	10	XR2
	11	XR3
Modify Index and Skip		
F = 0	00	Disp. added to IAR
	01	Disp. added to XR1
	10	Disp. added to XR2
	11	Disp. added to XR3
F = 1; IA = 0	00	Disp. added to C
	01	Add. added to XR1
	10	Add. added to XR2
	11	Add. added to XR3
F = 1; IA = 1	00	Disp. added to C
	01	C added to XR1
	10	C added to XR2
	11	C added to XR3

Disp. = Contents of Displacement field of instruction
Add. = Contents of Address field of instruction
C = Contents of location specified by Add.

Instruction Codes and Execution Times

Instruction	Mnemonic	Binary OP Code	Execution Times (in microseconds)							
			Single Word (F = 0)				Double Word (F = 1)			
			T = 00		T = 01,10,or 11		T = 00		T = 01,10, or 11	
			Avg.	Max.	Avg.	Max.	Avg.[1]	Max.[1]	Avg.[1]	Max.[1]
Load and Store										
Load ACC	LD	11000	7.6	-	11.2	-	10.8	-	14.8	-
Load Double	LDD	11001	11.2	-	14.9	-	14.4	-	18.0	-
Store ACC	STO	11010	7.6	-	11.2	-	10.8	-	14.8	-
Store Double	STD	11011	11.2	-	14.9	-	14.4	-	18.0	-
Load Index	LDX	01100	4.5	-	7.2	-	7.2	-	11.8	-
Store Index	STX	01101	7.6	-	11.2	-	11.8	-	15.4	-
Load Status*	LDS [7]	00100	3.6	-	3.6	-	-	-	-	-
Store Status	STS	00101	7.6	-	11.2	-	10.8	-	14.8	-
Arithmetic										
Add	A	10000	8.0	13.0	11.7	16.6	11.2	16.2	15.3	20.3
Add Double	AD	10001	12.2	22.0	15.8	25.6	15.3	25.2	19.3	29.5
Subtract	S	10010	8.0	13.0	11.7	16.6	11.2	16.2	15.3	20.3
Subtract Double	SD	10011	12.2	22.0	15.8	25.6	15.3	25.2	19.3	29.5
Multiply	M	10100	25.7	40.0	29.3	43.6	29.3	43.6	32.9	47.2
Divide	D	10101	76.0	150.8	79.6	154.4	79.6	154.4	83.2	150.0
And	AND	11100	7.6	-	11.2	-	10.8	-	14.8	-
Or	OR	11101	7.6	-	11.2	-	10.8	-	14.8	-
Exclusive Or	EOR	11110	7.6	-	11.2	-	10.8	-	14.8	-
Shift Left* Modifier Bits 8 & 9:										
Shift Left ACC 00	SLA [7]	00010								
Shift Left ACC and EXT 10	SLT [7]	00010								
Shift Left and Count ACC 01	[8] SLCA [7]	00010								
Shift Left and Count ACC and EXT 11	[8] SLC [7]	00010	[3]	-	[4]	-	-	-	-	-
Shift Right* Modifier Bits 8 & 9:										
Shift Right ACC 00 or 01	SRA [7]	00011								
Shift Right ACC and EXT 10	SRT [7]	00011								
Rotate Right 11	RTE [7]	00011	[5]		[6]					
Branch										
Branch and Store IAR	BSI	01000	7.6	-	11.2	-	10.8[2]	-	14.8	-
Branch or Skip on Condition	BSC	01001	3.6	-	3.6	-	7.2[2]	-	11.2	-
Modify Index and Skip	MDX	01110	4.5	9.9	11.2	16.2	18.5	23.4	18.5	23.4
Wait*	WAIT [7]	00110 [9]	3.6	-	3.6	-	-	-	-	-
Input/Output										
Execute I/O	XIO [10]	00001	11.2	-	14.8	-	14.8	-	18.4	-

* Valid in short format only

NOTES:

1 Indirect addressing, where applicable, adds 3.6 μsec to execution time
2 If branch is taken
3 3.6 + .45(N-4)
4 7.2 + .45(N-4)
5 N >16: 3.6 + .45(N-19)
N <16: 3.6 + .45(N-4)
6 N >16: 7.2 + .45(N-19)
N <16: 7.2 + .45(N-4)
where N=number of position shifted
7 Indirect addressing not allowed
8 If T=00, functions as SLA or SLT
9 All unassigned OP codes are defined as Wait operations
10 If XIO Read or Write, add 3.6 μsec

BSC Condition Codes

Bit Position	Condition
10	ACC zero
11	ACC negative
12	ACC positive, not zero
13	ACC even
14	Carry Indicator OFF
15	Overflow Indicator OFF

Short Instruction
Skip if any one condition is true
No-Op if all bits are zero

Long Instruction
Branch if none of the conditions is true
Unconditional branch if all bits are zero

AND, OR, EOR Operations

Memory → ACC →	Results		
	AND	OR	EOR
0 → 0 →	0	0	0
0 → 1 →	0	1	1
1 → 0 →	0	1	1
1 → 1 →	1	1	0

Device	Console Printer	Keyboard	1442	1134 1055	1627	Disk Storage	1132	Console
Device Code	1	1	2	3	5	4	6	7
Interrupt Level	4	4	0,4	4	3	2	1	5
Bit Position								
0	Service Response		Read Response		Response	Data Error	Read Emitter Response	Program Stop Key
1		Response	Punch Response	Reader Response		Operation Complete	Skip Response	Interrupt Run Mode
2		Request	Error Check			Busy, Not Ready	Space Response	
3		0 - Keyboard Entry 1 - Console Entry	Last Card	Punch Response		Busy	Carriage Busy	
4	Printer Busy		Operation Complete	Reader Busy		Carriage Home	Print Scan Check	
5	Printer Busy, Not Ready			Reader Busy, Not Ready			Forms Check	
6		Keyboard Busy		Punch Busy			Printer Busy	
7				Punch Busy, Not Ready				
8							Carriage Control Tape Channel No. 1	
9							2	
10							3	
11							4	
12							5	
13							6	
14			1442 Busy		1627 Busy	Sector Counts	9	
15			1442 Busy, Not Ready		1627 Not Ready		12	

20295A

APPENDIX D. POWERS OF TWO TABLE

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

11157

APPENDIX E. CHARACTER CODES

Ref No.	EBCDIC			IBM Card Code							Graphics and Control Names
	Binary		Hex	Rows						Hex	
	0123	4567		12	11	0	9	8	7-1		
0	0000	0000	00	12		0	9	8	1	B030	NUL
1		0001	01	12			9		1	9010	
2		0010	02	12			9		2	8810	
3		0011	03	12			9		3	8410	
4		0100	04	12			9		4	8210	PF Punch Off
5		0101	05	12			9		5	8110	HT Horiz.Tab
6		0110	06	12			9		6	8090	LC Lower Case
7		0111	07	12			9		7	8050	DEL Delete
8		1000	08	12			9	8		8030	
9		1001	09	12			9	8	1	9030	
10		1010	0A	12			9	8	2	8830	
11		1011	0B	12			9	8	3	8430	
12		1100	0C	12			9	8	4	8230	
13		1101	0D	12			9	8	5	8130	
14		1110	0E	12			9	8	6	80B0	
15		1111	0F	12			9	8	7	8070	
16	0001	0000	10	12	11		9	8	1	D030	
17		0001	11		11		9		1	5010	
18		0010	12		11		9		2	4810	
19		0011	13		11		9		3	4410	
20		0100	14		11		9		4	4210	RES Restore
21		0101	15		11		9		5	4110	NL New Line
22		0110	16		11		9		6	4090	BS Backspace
23		0111	17		11		9		7	4050	IDL Idle
24		1000	18		11		9	8		4030	
25		1001	19		11		9	8	1	5030	
26		1010	1A		11		9	8	2	4830	
27		1011	1B		11		9	8	3	4430	
28		1100	1C		11		9	8	4	4230	
29		1101	1D		11		9	8	5	4130	
30		1110	1E		11		9	8	6	40B0	
31		1111	1F		11		9	8	7	4070	
32	0010	0000	20		11	0	9	8	1	7030	
33		0001	21			0	9		1	3010	
34		0010	22			0	9		2	2810	
35		0011	23			0	9		3	2410	
36		0100	24			0	9		4	2210	BYP Bypass
37		0101	25			0	9		5	2110	LF Line Feed
38		0110	26			0	9		6	2090	EOB End of Block
39		0111	27			0	9		7	2050	PRE Prefix
40		1000	28			0	9	8		2030	
41		1001	29			0	9	8	1	3030	
42		1010	2A			0	9	8	2	2830	
43		1011	2B			0	9	8	3	2430	
44		1100	2C			0	9	8	4	2230	
45		1101	2D			0	9	8	5	2130	
46		1110	2E			0	9	8	6	20B0	
47		1111	2F			0	9	8	7	2070	
48	0011	0000	30	12	11	0	9	8	1	F030	
49		0001	31				9		1	1010	
50		0010	32				9		2	0810	
51		0011	33				9		3	0410	
52		0100	34				9		4	0210	PN Punch On
53		0101	35				9		5	0110	RS Reader Stop
54		0110	36				9		6	0090	UC Upper Case
55		0111	37				9		7	0050	EOT End of Trans.
56		1000	38				9	8		0030	
57		1001	39				9	8	1	1030	
58		1010	3A				9	8	2	0830	
59		1011	3B				9	8	3	0430	
60		1100	3C				9	8	4	0230	
61		1101	3D				9	8	5	0130	
62		1110	3E				9	8	6	00B0	
63		1111	3F				9	8	7	0070	
64	0100	0000	40	no punches						0000	(space)
65		0001	41	12		0	9		1	B010	
66		0010	42	12		0	9		2	A810	
67		0011	43	12		0	9		3	A410	
68		0100	44	12		0	9		4	A210	
69		0101	45	12		0	9		5	A110	
70		0110	46	12		0	9		6	A090	
71		0111	47	12		0	9		7	A050	
72		1000	48	12		0	9	8		A030	
73		1001	49	12				8	1	9020	
74		1010	4A	12				8	2	8820	¢
75		1011	4B	12				8	3	8420	. (period)
76		1100	4C	12				8	4	8220	<
77		1101	4D	12				8	5	8120	(
78		1110	4E	12				8	6	80A0	+
79		1111	4F	12				8	7	8060	| (logical OR)
80	0101	0000	50	12						8000	&
81		0001	51	12	11		9		1	D010	
82		0010	52	12	11		9		2	C810	
83		0011	53	12	11		9		3	C410	
84		0100	54	12	11		9		4	C210	
85		0101	55	12	11		9		5	C110	
86		0110	56	12	11		9		6	C090	
87		0111	57	12	11		9		7	C050	
88		1000	58	12	11		9	8		C030	
89		1001	59		11			8	1	5020	
90		1010	5A		11			8	2	4820	!
91		1011	5B		11			8	3	4420	$
92		1100	5C		11			8	4	4220	*
93		1101	5D		11			8	5	4120	)
94		1110	5E		11			8	6	40A0	;
95		1111	5F		11			8	7	4060	¬ (logical NOT)
96	0110	0000	60		11					4000	- (dash)
97		0001	61			0			1	3000	/
98		0010	62		11	0	9		2	6810	
99		0011	63		11	0	9		3	6410	
100		0100	64		11	0	9		4	6210	
101		0101	65		11	0	9		5	6110	
102		0110	66		11	0	9		6	6090	
103		0111	67		11	0	9		7	6050	
104		1000	68		11	0	9	8		6030	
105		1001	69			0		8	1	3020	
106		1010	6A	12	11					C000	
107		1011	6B			0		8	3	2420	, (comma)
108		1100	6C			0		8	4	2220	%
109		1101	6D			0		8	5	2120	_ (underscore)
110		1110	6E			0		8	6	20A0	>
111		1111	6F			0		8	7	2060	?
112	0111	0000	70	12	11	0				E000	
113		0001	71	12	11	0	9		1	F010	
114		0010	72	12	11	0	9		2	E810	
115		0011	73	12	11	0	9		3	E410	
116		0100	74	12	11	0	9		4	E210	
117		0101	75	12	11	0	9		5	E110	
118		0110	76	12	11	0	9		6	E090	
119		0111	77	12	11	0	9		7	E050	
120		1000	78	12	11	0	9	8		E030	
121		1001	79					8	1	1020	
122		1010	7A					8	2	0820	:
123		1011	7B					8	3	0420	#
124		1100	7C					8	4	0220	@
125		1101	7D					8	5	0120	' (apostrophe)
126		1110	7E					8	6	00A0	=
127		1111	7F					8	7	0060	"

20296

Ref No.	EBCDIC			IBM Card Code							Graphics and Control Names
	Binary		Hex	Rows						Hex	
	0123	4567		12	11	0	9	8	7-1		
128	1000	0000	80	12		0		8	1	B020	
129		0001	81	12		0			1	B000	a
130		0010	82	12		0			2	A800	b
131		0011	83	12		0			3	A400	c
132		0100	84	12		0			4	A200	d
133		0101	85	12		0			5	A100	e
134		0110	86	12		0			6	A080	f
135		0111	87	12		0			7	A040	g
136		1000	88	12		0		8		A020	h
137		1001	89	12		0	9			A010	i
138		1010	8A	12		0		8	2	A820	
139		1011	8B	12		0		8	3	A420	
140		1100	8C	12		0		8	4	A220	
141		1101	8D	12		0		8	5	A120	
142		1110	8E	12		0		8	6	A0A0	
143		1111	8F	12		0		8	7	A060	
144	1001	0000	90	12	11			8	1	D020	
145		0001	91	12	11				1	D000	j
146		0010	92	12	11				2	C800	k
147		0011	93	12	11				3	C400	l
148		0100	94	12	11				4	C200	m
149		0101	95	12	11				5	C100	n
150		0110	96	12	11				6	C080	o
151		0111	97	12	11				7	C040	p
152		1000	98	12	11			8		C020	q
153		1001	99	12	11		9			C010	r
154		1010	9A	12	11			8	2	C820	
155		1011	9B	12	11			8	3	C420	
156		1100	9C	12	11			8	4	C220	
157		1101	9D	12	11			8	5	C120	
158		1110	9E	12	11			8	6	C0A0	
159		1111	9F	12	11			8	7	C060	
160	1010	0000	A0		11	0		8	1	7020	
161		0001	A1		11	0			1	7000	
162		0010	A2		11	0			2	6800	s
163		0011	A3		11	0			3	6400	t
164		0100	A4		11	0			4	6200	u
165		0101	A5		11	0			5	6100	v
166		0110	A6		11	0			6	6080	w
167		0111	A7		11	0			7	6040	x
168		1000	A8		11	0		8		6020	y
169		1001	A9		11	0	9			6010	z
170		1010	AA		11	0		8	2	6820	
171		1011	AB		11	0		8	3	6420	
172		1100	AC		11	0		8	4	6220	
173		1101	AD		11	0		8	5	6120	
174		1110	AE		11	0		8	6	60A0	
175		1111	AF		11	0		8	7	6060	
176	1011	0000	B0	12	11	0		8	1	F020	
177		0001	B1	12	11	0			1	F000	
178		0010	B2	12	11	0			2	E800	
179		0011	B3	12	11	0			3	E400	
180		0100	B4	12	11	0			4	E200	
181		0101	B5	12	11	0			5	E100	
182		0110	B6	12	11	0			6	E080	
183		0111	B7	12	11	0			7	E040	
184		1000	B8	12	11	0		8		E020	
185		1001	B9	12	11	0	9			E010	
186		1010	BA	12	11	0		8	2	E820	
187		1011	BB	12	11	0		8	3	E420	
188		1100	BC	12	11	0		8	4	E220	
189		1101	BD	12	11	0		8	5	E120	
190		1110	BE	12	11	0		8	6	E0A0	
191		1111	BF	12	11	0		8	7	E060	
192	1100	0000	C0	12		0				A000	(+ zero)
193		0001	C1	12					1	9000	A
194		0010	C2	12					2	8800	B
195		0011	C3	12					3	8400	C
196		0100	C4	12					4	8200	D
197		0101	C5	12					5	8100	E
198		0110	C6	12					6	8080	F
199		0111	C7	12					7	8040	G
200		1000	C8	12				8		8020	H
201		1001	C9	12			9			8010	I
202		1010	CA	12		0	9	8	2	A830	
203		1011	CB	12		0	9	8	3	A430	
204		1100	CC	12		0	9	8	4	A230	
205		1101	CD	12		0	9	8	5	A130	
206		1110	CE	12		0	9	8	6	A0B0	
207		1111	CF	12		0	9	8	7	A070	
208	1101	0000	D0		11	0				6000	(- zero)
209		0001	D1		11				1	5000	J
210		0010	D2		11				2	4800	K
211		0011	D3		11				3	4400	L
212		0100	D4		11				4	4200	M
213		0101	D5		11				5	4100	N
214		0110	D6		11				6	4080	O
215		0111	D7		11				7	4040	P
216		1000	D8		11			8		4020	Q
217		1001	D9		11		9			4010	R
218		1010	DA	12	11		9	8	2	C830	
219		1011	DB	12	11		9	8	3	C430	
220		1100	DC	12	11		9	8	4	C230	
221		1101	DD	12	11		9	8	5	C130	
222		1110	DE	12	11		9	8	6	C0B0	
223		1111	DF	12	11		9	8	7	C070	
224	1110	0000	E0			0		8	2	2820	
225		0001	E1		11	0	9		1	7010	
226		0010	E2			0			2	2800	S
227		0011	E3			0			3	2400	T
228		0100	E4			0			4	2200	U
229		0101	E5			0			5	2100	V
230		0110	E6			0			6	2080	W
231		0111	E7			0			7	2040	X
232		1000	E8			0		8		2020	Y
233		1001	E9			0	9			2010	Z
234		1010	EA		11	0	9	8	2	6830	
235		1011	EB		11	0	9	8	3	6430	
236		1100	EC		11	0	9	8	4	6230	
237		1101	ED		11	0	9	8	5	6130	
238		1110	EE		11	0	9	8	6	60B0	
239		1111	EF		11	0	9	8	7	6070	
240	1111	0000	F0			0				2000	0
241		0001	F1						1	1000	1
242		0010	F2						2	0800	2
243		0011	F3						3	0400	3
244		0100	F4						4	0200	4
245		0101	F5						5	0100	5
246		0110	F6						6	0080	6
247		0111	F7						7	0040	7
248		1000	F8					8		0020	8
249		1001	F9				9			0010	9
250		1010	FA	12	11	0	9	8	2	E830	
251		1011	FB	12	11	0	9	8	3	E430	
252		1100	FC	12	11	0	9	8	4	E230	
253		1101	FD	12	11	0	9	8	5	E130	
254		1110	FE	12	11	0	9	8	6	E0B0	
255		1111	FF	12	11	0	9	8	7	E070	

20297

Decimal/ Hexadecimal Conversion Chart

The tables printed below are used to convert decimal numbers to hexadecimal and hexadecimal numbers to decimal. In the descriptions that follow, the explanation of each step is followed by an example in parentheses.

Decimal to Hexadecimal Conversion. Locate the decimal number (0489) in the body of the table. The two high-order digits (1E) of the hexadecimal number are in the left column on the same line, and the low-order digit (9) is at the top of the column. Thus, the hexadecimal number 1E9 is equal to the decimal number 0489.

Hexadecimal to Decimal Conversion. Locate the first two digits (1E) of the hexadecimal number (1E9) in the left column. Follow the line of figures across the page to the column headed by the low-order digit (9). The decimal number (0489) located at the junction of the horizontal line and the vertical column is the equivalent of the hexadecimal number.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

20290

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80_	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81_	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82_	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83_	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84_	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85_	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86_	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87_	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88_	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89_	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A_	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B_	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C_	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D_	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E_	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F_	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90_	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91_	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92_	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93_	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94_	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95_	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96_	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97_	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98_	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99_	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A_	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B_	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C_	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D_	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E_	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F_	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559
A0_	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1_	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2_	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3_	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4_	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5_	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6_	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7_	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8_	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9_	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA_	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB_	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC_	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD_	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE_	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF_	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0_	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1_	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2_	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3_	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4_	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5_	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6_	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7_	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8_	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9_	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA_	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB_	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC_	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD_	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE_	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF_	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C0_	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1_	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2_	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3_	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4_	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5_	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6_	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7_	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8_	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9_	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA_	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB_	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC_	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD_	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE_	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF_	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0_	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1_	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2_	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3_	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4_	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5_	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6_	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7_	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8_	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9_	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA_	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB_	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC_	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD_	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE_	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF_	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E0_	3584	3585	3586	3587	3583	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1_	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2_	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3_	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4_	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5_	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6_	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7_	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8_	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9_	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA_	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB_	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC_	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED_	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE_	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF_	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0_	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1_	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2_	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3_	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4_	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5_	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6_	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7_	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8_	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9_	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA_	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB_	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC_	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD_	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE_	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF_	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

Dec	Bin	Hex	Dec	Bin	Hex
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

The table to the left gives the decimal, binary, and hexadecimal coding for the full range of fourbinary bits, from zero through F_{16} and 15_{10}.

To convert a four-digit hexadecimal number to decimal, determine the decimal value of the three low-order hexadecimal digits in the main table, and add the value for the high-order digit, as shown in the extended chart to the right.

For conversion of decimal values beyond the main table, deduct the largest number in the table at the right that will yield a positive result. The related digit is the high-order hexadecimal digit. Determine the three remaining hexadecimal digits by converting the product of the above subtraction in the main table.

Hex	Dec	Hex	Dec
1000	4096	9000	36864
2000	8192	A000	40960
3000	12288	B000	45056
4000	16384	C000	49152
5000	20480	D000	53248
6000	24576	E000	57344
7000	28672	F000	61440
8000	32768		

APPENDIX G. DECIMAL/HEXADECIMAL CONVERSION CHART – FRACTIONS

Decimal to Hexadecimal Conversion. Locate the decimal fraction (. 1973) in the table. If the exact figure is not shown, locate the next higher and lower fractions (. 19726563, . 19750977). The first two digits of the hexadecimal fraction are at the top of the column (. 32). To locate the third digit, determine by observation or subtraction the smaller difference between the known fraction and each of the found fractions. The smaller difference identifies the correct line (. 008). The hexadecimal equivalent is . 328.
If the hexadecimal fraction is required to more places, multiply the decimal fraction by 16 and develop integers as successive terms of the hexadecimal fraction. Using the previous sample decimal fraction:

$$\begin{array}{r} .1973 \\ \underline{x16} \\ 3.1568 \\ \underline{x16} \\ 2.5085 \\ \underline{x16} \\ 8.1360 \\ \underline{x16} \\ 2.1760 \end{array} \qquad .1973_{10} = .3282_{16}$$

Hexadecimal to Decimal Conversion. Locate the first two digits (. 1E) of the hexadecimal fraction (. 1E9) in the horizontal row of column headings. Locate the third digit (. 009) in the leftmost column of the table. Follow the . 009 line horizontally to the right to the . lE column. The decimal equivalent is .11938477.
The decimal fractions in the table were carried to eight places and rounded. If 12 places are required, or if the hexadecimal fraction exceeds the capacity of the table, express the hexadecimal fraction as powers of 16 (expansion). For example:

$$.1E94_{16} = 1(16^{-1}) + 14(16^{-2}) + 9(16^{-3}) + 4(16^{-4})$$
$$= 1(.0625) + 14(.00390625) + 9(.000244140625) + 4(.0000152587890625)$$
$$= .11944580078125000_{10}$$

	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09	.0A	.0B	.0C	.0D	.0E	.0F
.000	.00000000	.00390625	.00781250	.01171875	.01562500	.01953125	.02343750	.02734375	.03125000	.03515625	.03906250	.04296875	.04687500	.05078125	.05468750	.05859375
.001	.00024414	.00415039	.00805664	.01196289	.01586914	.01977539	.02368164	.02758789	.03149414	.03540039	.03930664	.04321289	.04711914	.05102539	.05493164	.05883789
.002	.00048828	.00439453	.00830078	.01220703	.01611328	.02001953	.02392578	.02783203	.03173828	.03564453	.03955078	.04345703	.04736328	.05126953	.05517578	.05908203
.003	.00073242	.00463867	.00854492	.01245117	.01635742	.02026367	.02416992	.02807617	.03198242	.03588867	.03979492	.04370117	.04760742	.05151367	.05541992	.05932617
.004	.00097656	.00488281	.00878906	.01269531	.01660156	.02050781	.02441406	.02832031	.03222656	.03613281	.04003906	.04394531	.04785156	.05175781	.05566406	.05957031
.005	.00122070	.00512695	.00903320	.01293945	.01684570	.02075195	.02465820	.02856445	.03247070	.03637695	.04028320	.04418945	.04809570	.05200195	.05590820	.05981445
.006	.00146484	.00537109	.00927734	.01318359	.01708984	.02099609	.02490234	.02880859	.03271484	.03662109	.04052734	.04443359	.04833984	.05224609	.05615234	.06005859
.007	.00170898	.00561523	.00952148	.01342773	.01733398	.02124023	.02514648	.02905273	.03295898	.03686523	.04077148	.04467773	.04858398	.05249023	.05639648	.06030273
.008	.00195313	.00585938	.00976563	.01367188	.01757813	.02148438	.02539063	.02929688	.03320313	.03710938	.04101563	.04492188	.04882813	.05273438	.05664063	.06054688
.009	.00219727	.00610352	.01000977	.01391602	.01782227	.02172852	.02563477	.02954102	.03344727	.03735352	.04125977	.04516602	.04907227	.05297852	.05688477	.06079102
.00A	.00244141	.00634766	.01025391	.01416016	.01806641	.02197266	.02587891	.02978516	.03369141	.03759766	.04150391	.04541016	.04931641	.05322266	.05712891	.06103516
.00B	.00268555	.00659180	.01049805	.01440430	.01831055	.02221680	.02612305	.03002930	.03393555	.03784180	.04174805	.04565430	.04956055	.05346680	.05737305	.06127930
.00C	.00292969	.00683594	.01074219	.01464844	.01855469	.02246094	.02636719	.03027344	.03417969	.03808594	.04199219	.04589844	.04980469	.05371094	.05761719	.06152344
.00D	.00317383	.00708008	.01098633	.01489258	.01879883	.02270508	.02661133	.03051758	.03442383	.03833008	.04223633	.04614258	.05004883	.05395508	.05786133	.06176758
.00E	.00341797	.00732422	.01123047	.01513672	.01904297	.02294922	.02685547	.03076172	.03466797	.03857422	.04248047	.04638672	.05029297	.05419922	.05810547	.06201172
.00F	.00366211	.00756836	.01147461	.01538086	.01928711	.02319336	.02709961	.03100586	.03491211	.03881836	.04272461	.04663086	.05053711	.05444336	.05834961	.06225586

	.10	.11	.12	.13	.14	.15	.16	.17	.18	.19	.1A	.1B	.1C	.1D	.1E	.1F
.000	.06250000	.06640625	.07031250	.07421875	.07812500	.08203125	.08593750	.08984375	.09375000	.09765625	.10156250	.10546875	.10937500	.11328125	.11718750	.12109375
.001	.06274414	.06665039	.07055664	.07446289	.07836914	.08227539	.08618164	.09008789	.09399414	.09790039	.10180664	.10571289	.10961914	.11352539	.11743164	.12133789
.002	.06298828	.06689453	.07080078	.07470703	.07861328	.08251953	.08642578	.09033203	.09423828	.09814453	.10205078	.10595703	.10986328	.11376953	.11767578	.12158203
.003	.06323242	.06713867	.07104492	.07495117	.07885742	.08276367	.08666992	.09057617	.09448242	.09838867	.10229492	.10620117	.11010742	.11401367	.11791992	.12182617
.004	.06347656	.06738281	.07128906	.07519531	.07910156	.08300781	.08691406	.09082031	.09472656	.09863281	.10253906	.10644531	.11035156	.11425781	.11816406	.12207031
.005	.06372070	.06762695	.07153320	.07543945	.07934570	.08325195	.08715820	.09106445	.09497070	.09887695	.10278320	.10668945	.11059570	.11450195	.11840820	.12231445
.006	.06396484	.06787109	.07177734	.07568359	.07958984	.08349609	.08740234	.09130859	.09521484	.09912109	.10302734	.10693359	.11083984	.11474609	.11865234	.12255859
.007	.06420898	.06811523	.07202148	.07592773	.07983398	.08374023	.08764648	.09155273	.09545898	.09936523	.10327148	.10717773	.11108398	.11499023	.11889648	.12280273
.008	.06445313	.06835938	.07226563	.07617188	.08007813	.08398438	.08789063	.09179688	.09570313	.09960938	.10351563	.10742188	.11132813	.11523438	.11914063	.12304688
.009	.06469727	.06860352	.07250977	.07641602	.08032227	.08422852	.08813477	.09204102	.09594727	.09985352	.10375977	.10766602	.11157227	.11547852	.11938477	.12329102
.00A	.06494141	.06884766	.07275391	.07666016	.08056641	.08447266	.08837891	.09228516	.09619141	.10009766	.10400391	.10791016	.11181641	.11572266	.11962891	.12353516
.00B	.06518555	.06909180	.07299805	.07690430	.08081055	.08471680	.08862305	.09252930	.09643555	.10034180	.10424805	.10815430	.11206055	.11596680	.11987305	.12377930
.00C	.06542969	.06933594	.07324219	.07714844	.08105469	.08496094	.08886719	.09277344	.09667969	.10058594	.10449219	.10839844	.11230469	.11621094	.12011719	.12402344
.00D	.06567383	.06958008	.07348633	.07739258	.08129883	.08520508	.08911133	.09301758	.09692383	.10083008	.10473633	.10864258	.11254883	.11645508	.12036133	.12426758
.00E	.06591797	.06982422	.07373047	.07763672	.08154297	.08544922	.08935547	.09326172	.09716797	.10107422	.10498047	.10888672	.11279297	.11669922	.12060547	.12451172
.00F	.06616211	.07006836	.07397461	.07788086	.08178711	.08569336	.08959961	.09350586	.09741211	.10131836	.10522461	.10913086	.11303711	.11694336	.12084961	.12475586

	.20	.21	.22	.23	.24	.25	.26	.27	.28	.29	.2A	.2B	.2C	.2D	.2E	.2F
.000	.12500000	.12890625	.13281250	.13671875	.14062500	.14453125	.14843750	.15234375	.15625000	.16015625	.16406250	.16796875	.17187500	.17578125	.17968750	.18359375
.001	.12524414	.12915039	.13305664	.13696289	.14086914	.14477539	.14868164	.15258789	.15649414	.16040039	.16430664	.16821289	.17211914	.17602539	.17993164	.18383789
.002	.12548828	.12939453	.13330078	.13720703	.14111328	.14501953	.14892578	.15283203	.15673828	.16064453	.16455078	.16845703	.17236328	.17626953	.18017578	.18408203
.003	.12573242	.12963867	.13354492	.13745117	.14135742	.14526367	.14916992	.15307617	.15698242	.16088867	.16479492	.16870117	.17260742	.17651367	.18041992	.18432617
.004	.12597656	.12988281	.13378906	.13769531	.14160156	.14550781	.14941406	.15332031	.15722656	.16113281	.16503906	.16894531	.17285156	.17675781	.18066406	.18457031
.005	.12622070	.13012695	.13403320	.13793945	.14184570	.14575195	.14965820	.15356445	.15747070	.16137695	.16528320	.16918945	.17309570	.17700195	.18090820	.18481445
.006	.12646484	.13037109	.13427734	.13818359	.14208984	.14599609	.14990234	.15380859	.15771484	.16162109	.16552734	.16943359	.17333984	.17724609	.18115234	.18505859
.007	.12670898	.13061523	.13452148	.13842773	.14233398	.14624023	.15014648	.15405273	.15795898	.16186523	.16577148	.16967773	.17358398	.17749023	.18139648	.18530273
.008	.12695313	.13085938	.13476563	.13867188	.14257813	.14648438	.15039063	.15429688	.15820313	.16210938	.16601563	.16992188	.17382813	.17773438	.18164063	.18554688
.009	.12719727	.13110352	.13500977	.13891602	.14282227	.14672852	.15063477	.15454102	.15844727	.16235352	.16625977	.17016602	.17407227	.17797852	.18188477	.18579102
.00A	.12744141	.13134766	.13525391	.13916016	.14306641	.14697266	.15087891	.15478516	.15869141	.16259766	.16650391	.17041016	.17431641	.17822266	.18212891	.18603516
.00B	.12768555	.13159180	.13549805	.13940430	.14331055	.14721680	.15112305	.15502930	.15893555	.16284180	.16674805	.17065430	.17456055	.17846680	.18237305	.18627930
.00C	.12792969	.13183594	.13574219	.13964844	.14355469	.14746094	.15136719	.15527344	.15917969	.16308594	.16699219	.17089844	.17480469	.17871094	.18261719	.18652344
.00D	.12817383	.13208008	.13598633	.13989258	.14379883	.14770508	.15161133	.15551758	.15942383	.16333008	.16723633	.17114258	.17504883	.17895508	.18286133	.18676758
.00E	.12841797	.13232422	.13623047	.14013672	.14404297	.14794922	.15185547	.15576172	.15966797	.16357422	.16748047	.17138672	.17529297	.17919922	.18310547	.18701172
.00F	.12866211	.13256836	.13647461	.14038086	.14428711	.14819336	.15209961	.15600586	.15991211	.16381836	.16772461	.17163086	.17553711	.17944336	.18334961	.18725586

	.30	.31	.32	.33	.34	.35	.36	.37	.38	.39	.3A	.3B	.3C	.3D	.3E	.3F
.000	.18750000	.19140625	.19531250	.19921875	.20312500	.20703125	.21093750	.21484375	.21875000	.22265625	.22656250	.23046875	.23437500	.23828125	.24218750	.24609375
.001	.18774414	.19165039	.19555664	.19946289	.20336914	.20727539	.21118164	.21508789	.21899414	.22290039	.22680664	.23071289	.23461914	.23852539	.24243164	.24633789
.002	.18798828	.19189453	.19580078	.19970703	.20361328	.20751953	.21142578	.21533203	.21923828	.22314453	.22705078	.23095703	.23486328	.23876953	.24267578	.24658203
.003	.18823242	.19213867	.19604492	.19995117	.20385742	.20776367	.21166992	.21557617	.21948242	.22338867	.22729492	.23120117	.23510742	.23901367	.24291992	.24682617
.004	.18847656	.19238281	.19628906	.20019531	.20410156	.20800781	.21191406	.21582031	.21972656	.22363281	.22753906	.23144531	.23535156	.23925781	.24316406	.24707031
.005	.18872070	.19262695	.19653320	.20043945	.20434570	.20825195	.21215820	.21606445	.21997070	.22387695	.22778320	.23168945	.23559570	.23950195	.24340820	.24731445
.006	.18896484	.19287109	.19677734	.20068359	.20458984	.20849609	.21240234	.21630859	.22021484	.22412109	.22802734	.23193359	.23583984	.23974609	.24365234	.24755859
.007	.18920898	.19311523	.19702148	.20092773	.20483398	.20874023	.21264648	.21655273	.22045898	.22436523	.22827148	.23217773	.23608398	.23999023	.24389648	.24780273
.008	.18945313	.19335938	.19726563	.20117188	.20507813	.20898438	.21289063	.21679688	.22070313	.22460938	.22851563	.23242188	.23632813	.24023438	.24414063	.24804688
.009	.18969727	.19360352	.19750977	.20141602	.20532227	.20922852	.21313477	.21704102	.22094727	.22485352	.22875977	.23266602	.23657227	.24047852	.24438477	.24829102
.00A	.18994141	.19384766	.19775391	.20166016	.20556641	.20947266	.21337891	.21728516	.22119141	.22509766	.22900391	.23291016	.23681641	.24072266	.24462891	.24853516
.00B	.19018555	.19409180	.19799805	.20190430	.20581055	.20971680	.21362305	.21752930	.22143555	.22534180	.22924805	.23315430	.23706055	.24096680	.24487305	.24877930
.00C	.19042969	.19433594	.19824219	.20214844	.20605469	.20996094	.21386719	.21777344	.22167969	.22558594	.22949219	.23339844	.23730469	.24121094	.24511719	.24902344
.00D	.19067383	.19458008	.19848633	.20239258	.20629883	.21020508	.21411133	.21801758	.22192383	.22583008	.22973633	.23364258	.23754883	.24145508	.24536133	.24926758
.00E	.19091797	.19482422	.19873047	.20263672	.20654297	.21044922	.21435547	.21826172	.22216797	.22607422	.22998047	.23388672	.23779297	.24169922	.24560547	.24951172
.00F	.19116211	.19506836	.19897461	.20288086	.20678711	.21069336	.21459961	.21850586	.22241211	.22631836	.23022461	.23413086	.23803711	.24194336	.24584961	.24975586

	.40	.41	.42	.43	.44	.45	.46	.47	.48	.49	.4A	.4B	.4C	.4D	.4E	.4F
.000	.25000000	.25390625	.25781250	.26171875	.26562500	.26953125	.27343750	.27734375	.28125000	.28515625	.28906250	.29296875	.29687500	.30078125	.30468750	.30859375
.001	.25024414	.25415039	.25805664	.26196289	.26586914	.26977539	.27368164	.27758789	.28149414	.28540039	.28930664	.29321289	.29711914	.30102539	.30493164	.30883789
.002	.25048828	.25439453	.25830078	.26220703	.26611328	.27001953	.27392578	.27783203	.28173828	.28564453	.28955078	.29345703	.29736328	.30126953	.30517578	.30908203
.003	.25073242	.25463867	.25854492	.26245117	.26635742	.27026367	.27416992	.27807617	.28198242	.28588867	.28979492	.29370117	.29760742	.30151367	.30541992	.30932617
.004	.25097656	.25488281	.25878906	.26269531	.26660156	.27050781	.27441406	.27832031	.28222656	.28613281	.29003906	.29394531	.29785156	.30175781	.30566406	.30957031
.005	.25122070	.25512695	.25903320	.26293945	.26684570	.27075195	.27465820	.27856445	.28247070	.28637695	.29028320	.29418945	.29809570	.30200195	.30590820	.30981445
.006	.25146484	.25537109	.25927734	.26318359	.26708984	.27099609	.27490234	.27880859	.28271484	.28662109	.29052734	.29443359	.29833984	.30224609	.30615234	.31005859
.007	.25170898	.25561523	.25952148	.26342773	.26733398	.27124023	.27514648	.27905273	.28295898	.28686523	.29077148	.29467773	.29858398	.30249023	.30639648	.31030273
.008	.25195313	.25585938	.25976563	.26367188	.26757813	.27148438	.27539063	.27929688	.28320313	.28710938	.29101563	.29492188	.29882813	.30273438	.30664063	.31054688
.009	.25219727	.25610352	.26000977	.26391602	.26782227	.27172852	.27563477	.27954102	.28344727	.28735352	.29125977	.29516602	.29907227	.30297852	.30688477	.31079102
.00A	.25244141	.25634766	.26025391	.26416016	.26806641	.27197266	.27587891	.27978516	.28369141	.28759766	.29150390	.29541016	.29931641	.30322266	.30712891	.31103516
.00B	.25268555	.25659180	.26049805	.26440430	.26831055	.27221680	.27612305	.28002930	.28393555	.28784180	.29174805	.29565430	.29956055	.30346680	.30737305	.31127930
.00C	.25292969	.25683594	.26074219	.26464844	.26855469	.27246094	.27636719	.28027344	.28417969	.28808594	.29199219	.29589844	.29980469	.30371094	.30761719	.31152344
.00D	.25317383	.25708008	.26098633	.26489258	.26879883	.27270508	.27661133	.28051758	.28442383	.28833008	.29223633	.29614258	.30004883	.30395508	.30786133	.31176758
.00E	.25341797	.25732422	.26123047	.26513672	.26904297	.27294922	.27685547	.28076172	.28466797	.28857422	.29248047	.29638672	.30029297	.30419922	.30810547	.31201172
.00F	.25366211	.25756836	.26147461	.26538086	.26928711	.27319336	.27709961	.28100586	.28491211	.28881836	.29272461	.29663086	.30053711	.30444336	.30834961	.31225586

	.50	.51	.52	.53	.54	.55	.56	.57	.58	.59	.5A	.5B	.5C	.5D	.5E	.5F
.000	.31250000	.31640625	.32031250	.32421875	.32812500	.33203125	.33593750	.33984375	.34375000	.34765625	.35156250	.35546875	.35937500	.36328125	.36718750	.37109375
.001	.31274414	.31665039	.32055664	.32446289	.32836914	.33227539	.33618164	.34008789	.34399414	.34790039	.35180664	.35571289	.35961914	.36352539	.36743164	.37133789
.002	.31298828	.31689453	.32080078	.32470703	.32861328	.33251953	.33642578	.34033203	.34423828	.34814453	.35205078	.35595703	.35986328	.36376953	.36767578	.37158203
.003	.31323242	.31713867	.32104492	.32495117	.32885742	.33276367	.33666992	.34057617	.34448242	.34838867	.35229492	.35620117	.36010742	.36401367	.36791992	.37182617
.004	.31347656	.31738281	.32128906	.32519531	.32910156	.33300781	.33691406	.34082031	.34472656	.34863281	.35253906	.35644531	.36035156	.36425781	.36816406	.37207031
.005	.31372070	.31762695	.32153320	.32543945	.32934570	.33325195	.33715820	.34106445	.34497070	.34887695	.35278320	.35668945	.36059570	.36450195	.36840820	.37231445
.006	.31396484	.31787109	.32177734	.32568359	.32958984	.33349609	.33740234	.34130859	.34521484	.34912109	.35302734	.35693359	.36083984	.36474609	.36865234	.37255859
.007	.31420898	.31811523	.32202148	.32592773	.32983398	.33374023	.33764648	.34155273	.34545898	.34936523	.35327148	.35717773	.36108398	.36499023	.36889648	.37280273
.008	.31445313	.31835938	.32226563	.32617188	.33007813	.33398438	.33789063	.34179688	.34570313	.34960938	.35351563	.35742188	.36132813	.36523438	.36914063	.37304688
.009	.31469727	.31860352	.32250977	.32641602	.33032227	.33422852	.33813477	.34204102	.34594727	.34985352	.35375977	.35766602	.36157227	.36547852	.36938477	.37329102
.00A	.31494141	.31884766	.32275391	.32666016	.33056641	.33447266	.33837891	.34228516	.34619141	.35009766	.35400391	.35791016	.36181641	.36572266	.36962891	.37353516
.00B	.31518555	.31909180	.32299805	.32690430	.33081055	.33471680	.33862305	.34252930	.34643555	.35034180	.35424805	.35815430	.36206055	.36596680	.36987305	.37377930
.00C	.31542969	.31933594	.32324219	.32714844	.33105469	.33496094	.33886719	.34277344	.34667969	.35058594	.35449219	.35839844	.36230469	.36621094	.37011719	.37402344
.00D	.31567383	.31958008	.32348633	.32739258	.33129883	.33520508	.33911133	.34301758	.34692383	.35083008	.35473633	.35864258	.36254883	.36645508	.37036133	.37426758
.00E	.31591797	.31982422	.32373047	.32763672	.33154297	.33544922	.33935547	.34326172	.34716797	.35107422	.35498047	.35888672	.36279297	.36669922	.37060547	.37451172
.00F	.31616211	.32006836	.32397461	.32788086	.33178711	.33569336	.33959961	.34350586	.34741211	.35131836	.35522461	.35913086	.36303711	.36694336	.37084961	.37475586

	.60	.61	.62	.63	.64	.65	.66	.67	.68	.69	.6A	.6B	.6C	.6D	.6E	.6F
.000	.37500000	.37890625	.38281250	.38671875	.39062500	.39453125	.39843750	.40234375	.40625000	.41015625	.41406250	.41796875	.42187500	.42578125	.42968750	.43359375
.001	.37524414	.37915039	.38305664	.38696289	.39086914	.39477539	.39868164	.40258789	.40649414	.41040039	.41430664	.41821289	.42211914	.42602539	.42993164	.43383789
.002	.37548828	.37939453	.38330078	.38720703	.39111328	.39501953	.39892578	.40283203	.40673828	.41064453	.41455078	.41845703	.42236328	.42626953	.43017578	.43408203
.003	.37573242	.37963867	.38354492	.38745117	.39135742	.39526367	.39916992	.40307617	.40698242	.41088867	.41479492	.41870117	.42260742	.42651367	.43041992	.43432617
.004	.37597656	.37988281	.38378906	.38769531	.39160156	.39550781	.39941406	.40332031	.40722656	.41113281	.41503906	.41894531	.42285156	.42675781	.43066406	.43457031
.005	.37622070	.38012695	.38403320	.38793945	.39184570	.39575195	.39965820	.40356445	.40747070	.41137695	.41528320	.41918945	.42309570	.42700195	.43090820	.43481445
.006	.37646484	.38037109	.38427734	.38818359	.39208984	.39599609	.39990234	.40380859	.40771484	.41162109	.41552734	.41943359	.42333984	.42724609	.43115234	.43505859
.007	.37670898	.38061523	.38452148	.38842773	.39233398	.39624023	.40014648	.40405273	.40795898	.41186523	.41577148	.41967773	.42358398	.42749023	.43139648	.43530273
.008	.37695313	.38085938	.38476563	.38867188	.39257813	.39648438	.40039063	.40429688	.40820313	.41210938	.41601563	.41992188	.42382813	.42773438	.43164063	.43554688
.009	.37719727	.38110352	.38500977	.38891602	.39282227	.39672852	.40063477	.40454102	.40844727	.41235352	.41625977	.42016602	.42407227	.42797852	.43188477	.43579102
.00A	.37744141	.38134766	.38525391	.38916016	.39306641	.39697266	.40087891	.40478516	.40869141	.41259766	.41650391	.42041016	.42431641	.42822266	.43212891	.43603516
.00B	.37768555	.38159180	.38549805	.38940430	.39331055	.39721680	.40112305	.40502930	.40893555	.41284180	.41674805	.42065430	.42456055	.42846680	.43237305	.43627930
.00C	.37792969	.38183594	.38574219	.38964844	.39355469	.39746094	.40136719	.40527344	.40917969	.41308594	.41699219	.42089844	.42480469	.42871094	.43261719	.43652344
.00D	.37817383	.38208008	.38598633	.38989258	.39379883	.39770508	.40161133	.40551758	.40942383	.41333008	.41723633	.42114258	.42504883	.42895508	.43286133	.43676758
.00E	.37841797	.38232422	.38623047	.39013672	.39404297	.39794922	.40185547	.40576172	.40966797	.41357422	.41748047	.42138672	.42529297	.42919922	.43310547	.43701172
.00F	.37866211	.38256836	.38647461	.39038086	.39428711	.39819336	.40209961	.40600586	.40991211	.41381836	.41772461	.42163086	.42553711	.42944336	.43334961	.43725586

	.70	.71	.72	.73	.74	.75	.76	.77	.78	.79	.7A	.7B	.7C	.7D	.7E	.7F
.000	.43750000	.44140625	.44531250	.44921875	.45312500	.45703125	.46093750	.46484375	.46875000	.47265625	.47656250	.48046875	.48437500	.48828125	.49218750	.49609375
.001	.43774414	.44165039	.44555664	.44946289	.45336914	.45727539	.46118164	.46508789	.46899414	.47290039	.47680664	.48071289	.48461914	.48852539	.49243164	.49633789
.002	.43798828	.44189453	.44580078	.44970703	.45361328	.45751953	.46142578	.46533203	.46923828	.47314453	.47705078	.48095703	.48486328	.48876953	.49267578	.49658203
.003	.43823242	.44213867	.44604492	.44995117	.45385742	.45776367	.46166992	.46557617	.46948242	.47338867	.47729492	.48120117	.48510742	.48901367	.49291992	.49682617
.004	.43847656	.44238281	.44628906	.45019531	.45410156	.45800781	.46191406	.46582031	.46972656	.47363281	.47753906	.48144531	.48535156	.48925781	.49316406	.49707031
.005	.43872070	.44262695	.44653320	.45043945	.45434570	.45825195	.46215820	.46606445	.46997070	.47387695	.47778320	.48168945	.48559570	.48950195	.49340820	.49731445
.006	.43896484	.44287109	.44677734	.45068359	.45458984	.45849609	.46240234	.46630859	.47021484	.47412109	.47802734	.48193359	.48583984	.48974609	.49365234	.49755859
.007	.43920898	.44311523	.44702148	.45092773	.45483398	.45874023	.46264648	.46655273	.47045898	.47436523	.47827148	.48217773	.48608398	.48999023	.49389648	.49780273
.008	.43945313	.44335938	.44726563	.45117188	.45507813	.45898438	.46289063	.46679688	.47070313	.47460938	.47851563	.48242188	.48632813	.49023438	.49414063	.49804688
.009	.43969727	.44360352	.44750977	.45141602	.45532227	.45922852	.46313477	.46704102	.47094727	.47485352	.47875977	.48266602	.48657227	.49047852	.49438477	.49829102
.00A	.43994141	.44384766	.44775391	.45166016	.45556641	.45947266	.46337891	.46728516	.47119141	.47509766	.47900391	.48291016	.48681641	.49072266	.49462891	.49853516
.00B	.44018555	.44409180	.44799805	.45190430	.45581055	.45971680	.46362305	.46752930	.47143555	.47534180	.47924805	.48315430	.48706055	.49096680	.49487305	.49877930
.00C	.44042969	.44433594	.44824219	.45214844	.45605469	.45996094	.46386719	.46777344	.47167969	.47558594	.47949219	.48339844	.48730469	.49121094	.49511719	.49902344
.00D	.44067383	.44458008	.44848633	.45239258	.45629883	.46020508	.46411133	.46801758	.47192383	.47583008	.47973633	.48364258	.48754883	.49145508	.49536133	.49926758
.00E	.44091797	.44482422	.44873047	.45263672	.45654297	.46044922	.46435547	.46826172	.47216797	.47607422	.47998047	.48388672	.48779297	.49169922	.49560547	.49951172
.00F	.44116211	.44506836	.44897461	.45288086	.45678711	.46069336	.46459961	.46850586	.47241211	.47631836	.48022461	.48413086	.48803711	.49194336	.49584961	.49975586

	.80	.81	.82	.83	.84	.85	.86	.87	.88	.89	.8A	.8B	.8C	.8D	.8E	.8F
.000	.50000000	.50390625	.50781250	.51171875	.51562500	.51953125	.52343750	.52734375	.53125000	.53515625	.53906250	.54296875	.54687500	.55078125	.55468750	.55859375
.001	.50024414	.50415039	.50805664	.51196289	.51586914	.51977539	.52368164	.52758789	.53149414	.53540039	.53930664	.54321289	.54711914	.55102539	.55493164	.55883789
.002	.50048828	.50439453	.50830078	.51220703	.51611328	.52001953	.52392578	.52783203	.53173828	.53564453	.53955078	.54345703	.54736328	.55126953	.55517578	.55908203
.003	.50073242	.50463867	.50854492	.51245117	.51635742	.52026367	.52416992	.52807617	.53198242	.53588867	.53979492	.54370117	.54760742	.55151367	.55541992	.55932617
.004	.50097656	.50488281	.50878906	.51269531	.51660156	.52050781	.52441406	.52832031	.53222656	.53613281	.54003906	.54394531	.54785156	.55175781	.55566406	.55957031
.005	.50122070	.50512695	.50903320	.51293945	.51684570	.52075195	.52465820	.52856445	.53247070	.53637695	.54028320	.54418945	.54809570	.55200195	.55590820	.55981445
.006	.50146484	.50537109	.50927734	.51318359	.51708984	.52099609	.52490234	.52880859	.53271484	.53662109	.54052734	.54443359	.54833984	.55224609	.55615234	.56005859
.007	.50170898	.50561523	.50952148	.51342773	.51733398	.52124023	.52514648	.52905273	.53295898	.53686523	.54077148	.54467773	.54858398	.55249023	.55639648	.56030273
.008	.50195313	.50585938	.50976563	.51367188	.51757813	.52148438	.52539063	.52929688	.53320313	.53710938	.54101563	.54492188	.54882813	.55273438	.55664063	.56054688
.009	.50219727	.50610352	.51000977	.51391602	.51782227	.52172852	.52563477	.52954102	.53344727	.53735352	.54125977	.54516602	.54907227	.55297852	.55688477	.56079102
.00A	.50244141	.50634766	.51025391	.51416016	.51806641	.52197266	.52587891	.52978516	.53369141	.53759766	.54150391	.54541016	.54931641	.55322266	.55712891	.56103516
.00B	.50268555	.50659180	.51049805	.51440430	.51831055	.52221680	.52612305	.53002930	.53393555	.53784180	.54174805	.54565430	.54956055	.55346680	.55737305	.56127930
.00C	.50292969	.50683594	.51074219	.51464844	.51855469	.52246094	.52636719	.53027344	.53417969	.53808594	.54199219	.54589844	.54980469	.55371094	.55761719	.56152344
.00D	.50317383	.50708008	.51098633	.51489258	.51879883	.52270508	.52661133	.53051758	.53442383	.53833008	.54223633	.54614258	.55004883	.55395508	.55786133	.56176758
.00E	.50341797	.50732422	.51123047	.51513672	.51904297	.52294922	.52685547	.53076172	.53466797	.53857422	.54248047	.54638672	.55029297	.55419922	.55810547	.56201172
.00F	.50366211	.50756836	.51147461	.51538086	.51928711	.52319336	.52709961	.53100586	.53491211	.53881836	.54272461	.54663086	.55053711	.55444336	.55834961	.56225586

	.90	.91	.92	.93	.94	.95	.96	.97	.98	.99	.9A	.9B	.9C	.9D	.9E	.9F
.000	.56250000	.56640625	.57031250	.57421875	.57812500	.58203125	.58593750	.58984375	.59375000	.59765625	.60156250	.60546875	.60937500	.61328125	.61718750	.62109375
.001	.56274414	.56665039	.57055664	.57446289	.57836914	.58227539	.58618164	.59008789	.59399414	.59790039	.60180664	.60571289	.60961914	.61352539	.61743164	.62133789
.002	.56298828	.56689453	.57080078	.57470703	.57861328	.58251953	.58642578	.59033203	.59423828	.59814453	.60205078	.60595703	.60986328	.61376953	.61767578	.62158203
.003	.56323242	.56713867	.57104492	.57495117	.57885742	.58276367	.58666992	.59057617	.59448242	.59838867	.60229492	.60620117	.61010742	.61401367	.61791992	.62182617
.004	.56347656	.56738281	.57128906	.57519531	.57910156	.58300781	.58691406	.59082031	.59472656	.59863281	.60253906	.60644531	.61035156	.61425781	.61816406	.62207031
.005	.56372070	.56762695	.57153320	.57543945	.57934570	.58325195	.58715820	.59106445	.59497070	.59887695	.60278320	.60668945	.61059570	.61450195	.61840820	.62231445
.006	.56396484	.56787109	.57177734	.57568359	.57958984	.58349609	.58740234	.59130859	.59521484	.59912109	.60302734	.60693359	.61083984	.61474609	.61865234	.62255859
.007	.56420898	.56811523	.57202148	.57592773	.57983398	.58374023	.58764648	.59155273	.59545898	.59936523	.60327148	.60717773	.61108398	.61499023	.61889648	.62280273
.008	.56445313	.56835938	.57226563	.57617188	.58007813	.58398438	.58789063	.59179688	.59570313	.59960938	.60351563	.60742188	.61132813	.61523438	.61914063	.62304688
.009	.56469727	.56860352	.57250977	.57641602	.58032227	.58422852	.58813477	.59204102	.59594727	.59985352	.60375977	.60766602	.61157227	.61547852	.61938477	.62329102
.00A	.56494141	.56884766	.57275391	.57666016	.58056641	.58447266	.58837891	.59228516	.59619141	.60009766	.60400391	.60791016	.61181641	.61572266	.61962891	.62353516
.00B	.56518555	.56909180	.57299805	.57690430	.58081055	.58471680	.58862305	.59252930	.59643555	.60034180	.60424805	.60815430	.61206055	.61596680	.61987305	.62377930
.00C	.56542969	.56933594	.57324219	.57714844	.58105469	.58496094	.58886719	.59277344	.59667969	.60058594	.60449219	.60839844	.61230469	.61621094	.62011719	.62402344
.00D	.56567383	.56958008	.57348633	.57739258	.58129883	.58520508	.58911133	.59301758	.59692383	.60083008	.60473633	.60864258	.61254883	.61645508	.62036133	.62426758
.00E	.56591797	.56982422	.57373047	.57763672	.58154297	.58544922	.58935547	.59326172	.59716797	.60107422	.60498047	.60888672	.61279297	.61669922	.62060547	.62451172
.00F	.56616211	.57006836	.57397461	.57788086	.58178711	.58569336	.58959961	.59350586	.59741211	.60131836	.60522461	.60913086	.61303711	.61694336	.62084961	.62475586

	.A0	.A1	.A2	.A3	.A4	.A5	.A6	.A7	.A8	.A9	.AA	.AB	.AC	.AD	.AE	.AF
.000	.62500000	.62890625	.63281250	.63671875	.64062500	.64453125	.64843750	.65234375	.65625000	.66015625	.66406250	.66796875	.67187500	.67578125	.67968750	.68359375
.001	.62524414	.62915039	.63305664	.63696289	.64086914	.64477539	.64868164	.65258789	.65649414	.66040039	.66430664	.66821289	.67211914	.67602539	.67993164	.68383789
.002	.62548828	.62939453	.63330078	.63720703	.64111328	.64501953	.64892578	.65283203	.65673828	.66064453	.66455078	.66845703	.67236328	.67626953	.68017578	.68408203
.003	.62573242	.62963867	.63354492	.63745117	.64135742	.64526367	.64916992	.65307617	.65698242	.66088867	.66479492	.66870117	.67260742	.67651367	.68041992	.68432617
.004	.62597656	.62988281	.63378906	.63769531	.64160156	.64550781	.64941406	.65332031	.65722656	.66113281	.66503906	.66894531	.67285156	.67675781	.68066406	.68457031
.005	.62622070	.63012695	.63403320	.63793945	.64184570	.64575195	.64965820	.65356445	.65747070	.66137695	.66528320	.66918945	.67309570	.67700195	.68090820	.68481445
.006	.62646484	.63037109	.63427734	.63818359	.64208984	.64599609	.64990234	.65380859	.65771484	.66162109	.66552734	.66943359	.67333984	.67724609	.68115234	.68505859
.007	.62670898	.63061523	.63452148	.63842773	.64233398	.64624023	.65014648	.65405273	.65795898	.66186523	.66577148	.66967773	.67358398	.67749023	.68139648	.68530273
.008	.62695313	.63085938	.63476563	.63867188	.64257813	.64648438	.65039063	.65429688	.65820313	.66210938	.66601563	.66992188	.67382813	.67773438	.68164063	.68554688
.009	.62719727	.63110352	.63500977	.63891602	.64282227	.64672852	.65063477	.65454102	.65844727	.66235352	.66625977	.67016602	.67407227	.67797852	.68188477	.68579102
.00A	.62744141	.63134766	.63525391	.63916016	.64306641	.64697266	.65087891	.65478516	.65869141	.66259766	.66650391	.67041016	.67431641	.67822266	.68212891	.68603516
.00B	.62768555	.63159180	.63549805	.63940430	.64331055	.64721680	.65112305	.65502930	.65893555	.66284180	.66674805	.67065430	.67456055	.67846680	.68237305	.68627930
.00C	.62792969	.63183594	.63574219	.63964844	.64355469	.64746094	.65136719	.65527344	.65917969	.66308594	.66699219	.67089844	.67480469	.67871094	.68261719	.68652344
.00D	.62817383	.63208008	.63598633	.63989258	.64379883	.64770508	.65161133	.65551758	.65942383	.66333008	.66723633	.67114258	.67504883	.67895508	.68286133	.68676758
.C0E	.62841797	.63232422	.63623047	.64013672	.64404297	.64794922	.65185547	.65576172	.65966797	.66357422	.66748047	.67138672	.67529297	.67919922	.68310547	.68701172
.00F	.62866211	.63256836	.63647461	.64038086	.64428711	.64819336	.65209961	.65600586	.65991211	.66381836	.66772461	.67163086	.67553711	.67944336	.68334961	.68725586

	.B0	.B1	.B2	.B3	.B4	.B5	.B6	.B7	.B8	.B9	.BA	.BB	.BC	.BD	.BE	.BF
.000	.68750000	.69140625	.69531250	.69921875	.70312500	.70703125	.71093750	.71484375	.71875000	.72265625	.72656250	.73046875	.73437500	.73828125	.74218750	.74609375
.001	.68774414	.69165039	.69555664	.69946289	.70336914	.70727539	.71118164	.71508789	.71899414	.72290039	.72680664	.73071289	.73461914	.73852539	.74243164	.74633789
.002	.68798828	.69189453	.69580078	.69970703	.70361328	.70751953	.71142578	.71533203	.71923828	.72314453	.72705078	.73095703	.73486328	.73876953	.74267578	.74658203
.003	.68823242	.69213867	.69604492	.69995117	.70385742	.70776367	.71166992	.71557617	.71948242	.72338867	.72729492	.73120117	.73510742	.73901367	.74291992	.74682617
.004	.68847656	.69238281	.69628906	.70019531	.70410156	.70800781	.71191406	.71582031	.71972656	.72363281	.72753906	.73144531	.73535156	.73925781	.74316406	.74707031
.005	.68872070	.69262695	.69653320	.70043945	.70434570	.70825195	.71215820	.71606445	.71997070	.72387695	.72778320	.73168945	.73559570	.73950195	.74340820	.74731445
.006	.68896484	.69287109	.69677734	.70068359	.70458984	.70849609	.71240234	.71630859	.72021484	.72412109	.72802734	.73193359	.73583984	.73974609	.74365234	.74755859
.007	.68920898	.69311523	.69702148	.70092773	.70483398	.70874023	.71264648	.71655273	.72045898	.72436523	.72827148	.73217773	.73608398	.73999023	.74389648	.74780273
.008	.68945313	.69335938	.69726563	.70117188	.70507813	.70898438	.71289063	.71679688	.72070313	.72460938	.72851563	.73242188	.73632813	.74023438	.74414063	.74804688
.009	.68969727	.69360352	.69750977	.70141602	.70532227	.70922852	.71313477	.71704102	.72094727	.72485352	.72875977	.73266602	.73657227	.74047852	.74438477	.74829102
.00A	.68994141	.69384766	.69775391	.70166016	.70556641	.70947266	.71337891	.71728516	.72119141	.72509766	.72900391	.73291016	.73681641	.74072266	.74462891	.74853516
.00B	.69018555	.69409180	.69799805	.70190430	.70581055	.70971680	.71362305	.71752930	.72143555	.72534180	.72924805	.73315430	.73706055	.74096680	.74487305	.74877930
.00C	.69042969	.69433594	.69824219	.70214844	.70605469	.70996094	.71386719	.71777344	.72167969	.72558594	.72949219	.73339844	.73730469	.74121094	.74511719	.74902344
.00D	.69067383	.69458008	.69848633	.70239258	.70629883	.71020508	.71411133	.71801758	.72192383	.72583008	.72973633	.73364258	.73754883	.74145508	.74536133	.74926758
.00E	.69091797	.69482422	.69873047	.70263672	.70654297	.71044922	.71435547	.71826172	.72216797	.72607422	.72998047	.73388672	.73779297	.74169922	.74560547	.74951172
.00F	.69116211	.69506836	.69897461	.70288086	.70678711	.71069336	.71459961	.71850586	.72241211	.72631836	.73022461	.73413086	.73803711	.74194336	.74584961	.74975586

	.C0	.C1	.C2	.C3	.C4	.C5	.C6	.C7	.C8	.C9	.CA	.CB	.CC	.CD	.CE	.CF
.000	.75000000	.75390625	.75781250	.76171875	.76562500	.76953125	.77343750	.77734375	.78125000	.78515625	.78906250	.79296875	.79687500	.80078125	.80468750	.80859375
.001	.75024414	.75415039	.75805664	.76196289	.76586914	.76977539	.77368164	.77758789	.78149414	.78540039	.78930664	.79321289	.79711914	.80102539	.80493164	.80883789
.002	.75048828	.75439453	.75830078	.76220703	.76611328	.77001953	.77392578	.77783203	.78173828	.78564453	.78955078	.79345703	.79736328	.80126953	.80517578	.80908203
.003	.75073242	.75463867	.75854492	.76245117	.76635742	.77026367	.77416992	.77807617	.78198242	.78588867	.78979492	.79370117	.79760742	.80151367	.80541992	.80932617
.004	.75097656	.75488281	.75878906	.76269531	.76660156	.77050781	.77441406	.77832031	.78222656	.78613281	.79003906	.79394531	.79785156	.80175781	.80566406	.80957031
.005	.75122070	.75512695	.75903320	.76293945	.76684570	.77075195	.77465820	.77856445	.78247070	.78637695	.79028320	.79418945	.79809570	.80200195	.80590820	.80981445
.006	.75146484	.75537109	.75927734	.76318359	.76708984	.77099609	.77490234	.77880859	.78271484	.78662109	.79052734	.79443359	.79833984	.80224609	.80615234	.81005859
.007	.75170898	.75561523	.75952148	.76342773	.76733398	.77124023	.77514648	.77905273	.78295898	.78686523	.79077148	.79467773	.79858398	.80249023	.80639648	.81030273
.008	.75195313	.75585938	.75976563	.76367188	.76757813	.77148438	.77539063	.77929688	.78320313	.78710938	.79101563	.79492188	.79882813	.80273438	.80664063	.81054688
.009	.75219727	.75610352	.76000977	.76391602	.76782227	.77172852	.77563477	.77954102	.78344727	.78735352	.79125977	.79516602	.79907227	.80297852	.80688477	.81079102
.00A	.75244141	.75634766	.76025391	.76416016	.76806641	.77197266	.77587891	.77978516	.78369141	.78759766	.79150391	.79541016	.79931641	.80322266	.80712891	.81103516
.00B	.75268555	.75659180	.76049805	.76440430	.76831055	.77221680	.77612305	.78002930	.78393555	.78784180	.79174805	.79565430	.79956055	.80346680	.80737305	.81127930
.00C	.75292969	.75683594	.76074219	.76464844	.76855469	.77246094	.77636719	.78027344	.78417969	.78808594	.79199219	.79589844	.79980469	.80371094	.80761719	.81152344
.00D	.75317383	.75708008	.76098633	.76489258	.76879883	.77270508	.77661133	.78051758	.78442383	.78833008	.79223633	.79614258	.80004883	.80395508	.80786133	.81176758
.00E	.75341797	.75732422	.76123047	.76513672	.76904297	.77294922	.77685547	.78076172	.78466797	.78857422	.79248047	.79638672	.80029297	.80419922	.80810547	.81201172
.00F	.75366211	.75756836	.76147461	.76538086	.76928711	.77319336	.77709961	.78100586	.78491211	.78881836	.79272461	.79663086	.80053711	.80444336	.80834961	.81225586

	.D0	.D1	.D2	.D3	.D4	.D5	.D6	.D7	.D8	.D9	.DA	.DB	.DC	.DD	.DE	.DF
.000	.81250000	.81640625	.82031250	.82421875	.82812500	.83203125	.83593750	.83984375	.84375000	.84765625	.85156250	.85546875	.85937500	.86328125	.86718750	.87109375
.001	.81274414	.81665039	.82055664	.82446289	.82836914	.83227539	.83618164	.84008789	.84399414	.84790039	.85180664	.85571289	.85961914	.86352539	.86743164	.87133789
.002	.81298828	.81689453	.82080078	.82470703	.82861328	.83251953	.83642578	.84033203	.84423828	.84814453	.85205078	.85595703	.85986328	.86376953	.86767578	.87158203
.003	.81323242	.81713867	.82104492	.82495117	.82885742	.83276367	.83666992	.84057617	.84448242	.84838867	.85229492	.85620117	.86010742	.86401367	.86791992	.87182617
.004	.81347656	.81738281	.82128906	.82519531	.82910156	.83300781	.83691406	.84082031	.84472656	.84863281	.85253906	.85644531	.86035156	.86425781	.86816406	.87207031
.005	.81372070	.81762695	.82153320	.82543945	.82934570	.83325195	.83715820	.84106445	.84497070	.84887695	.85278320	.85668945	.86059570	.86450195	.86840820	.87231445
.006	.81396484	.81787109	.82177734	.82568359	.82958984	.83349609	.83740234	.84130859	.84521484	.84912109	.85302734	.85693359	.86083984	.86474609	.86865234	.87255859
.007	.81420898	.81811523	.82202148	.82592773	.82983398	.83374023	.83764648	.84155273	.84545898	.84936523	.85327148	.85717773	.86108398	.86499023	.86889648	.87280273
.008	.81445313	.81835938	.82226563	.82617188	.83007813	.83398438	.83789063	.84179688	.84570313	.84960938	.85351563	.85742188	.86132813	.86523438	.86914063	.87304688
.009	.81469727	.81860352	.82250977	.82641602	.83032227	.83422852	.83813477	.84204102	.84594727	.84985352	.85375977	.85766602	.86157227	.86547852	.86938477	.87329102
.00A	.81494141	.81884766	.82275391	.82666016	.83056641	.83447266	.83837891	.84228516	.84619141	.85009766	.85400391	.85791016	.86181641	.86572266	.86962891	.87353516
.00B	.81518555	.81909180	.82299805	.82690430	.83081055	.83471680	.83862305	.84252930	.84643555	.85034180	.85424805	.85815430	.86206055	.86596680	.86987305	.87377930
.00C	.81542969	.81933594	.82324219	.82714844	.83105469	.83496094	.83886719	.84277344	.84667969	.85058594	.85449219	.85839844	.86230469	.86621094	.87011719	.87402344
.00D	.81567383	.81958008	.82348633	.82739258	.83129883	.83520508	.83911133	.84301758	.84692383	.85083008	.85473633	.85864258	.86254883	.86645508	.87036133	.87426758
.00E	.81591797	.81982422	.82373047	.82763672	.83154297	.83544922	.83935547	.84326172	.84716797	.85107422	.85498047	.85888672	.86279297	.86669922	.87060547	.87451172
.00F	.81616211	.82006836	.82397461	.82788086	.83178711	.83569336	.83959961	.84350586	.84741211	.85131836	.85522461	.85913086	.86303711	.86694336	.87084961	.87475586

	.E0	.E1	.E2	.E3	.E4	.E5	.E6	.E7	.E8	.E9	.EA	.EB	.EC	.ED	.EE	.EF
.000	.87500000	.87890625	.88281250	.88671875	.89062500	.89453125	.89843750	.90234375	.90625000	.91015625	.91406250	.91796875	.92187500	.92578125	.92968750	.93359375
.001	.87524414	.87915039	.88305664	.88696289	.89086914	.89477539	.89868164	.90258789	.90649414	.91040039	.91430664	.91821289	.92211914	.92602539	.92993164	.93383789
.002	.87548828	.87939453	.88330078	.88720703	.89111328	.89501953	.89892578	.90283203	.90673828	.91064453	.91455078	.91845703	.92236328	.92626953	.93017578	.93408203
.003	.87573242	.87963867	.88354492	.88745117	.89135742	.89526367	.89916992	.90307617	.90698242	.91088867	.91479492	.91870117	.92260742	.92651367	.93041992	.93432617
.004	.87597656	.87988281	.88378906	.88769531	.89160156	.89550781	.89941406	.90332031	.90722656	.91113281	.91503906	.91894531	.92285156	.92675781	.93066406	.93457031
.005	.87622070	.88012695	.88403320	.88793945	.89184570	.89575195	.89965820	.90356445	.90747070	.91137695	.91528320	.91918945	.92309570	.92700195	.93090820	.93481445
.006	.87646484	.88037109	.88427734	.88818359	.89208984	.89599609	.89990234	.90380859	.90771484	.91162109	.91552734	.91943359	.92333984	.92724609	.93115234	.93505859
.007	.87670898	.88061523	.88452148	.88842773	.89233398	.89624023	.90014648	.90405273	.90795898	.91186523	.91577148	.91967773	.92358398	.92749023	.93139648	.93530273
.008	.87695313	.88085938	.88476563	.88867188	.89257813	.89648438	.90039063	.90429688	.90820313	.91210938	.91601563	.91992188	.92382813	.92773438	.93164063	.93554688
.009	.87719727	.88110352	.88500977	.88891602	.89282227	.89672852	.90063477	.90454102	.90844727	.91235352	.91625977	.92016602	.92407227	.92797852	.93188477	.93579102
.00A	.87744141	.88134766	.88525391	.88916016	.89306641	.89697266	.90087891	.90478516	.90869141	.91259766	.91650391	.92041016	.92431641	.92822266	.93212891	.93603516
.00B	.87768555	.88159180	.88549805	.88940430	.89331055	.89721680	.90112305	.90502930	.90893555	.91284180	.91674805	.92065430	.92456055	.92846680	.93237305	.93627930
.00C	.87792969	.88183594	.88574219	.88964844	.89355469	.89746094	.90136719	.90527344	.90917969	.91308594	.91699219	.92089844	.92480469	.92871094	.93261719	.93652344
.00D	.87817383	.88208008	.88598633	.88989258	.89379883	.89770508	.90161133	.90551758	.90942383	.91333008	.91723633	.92114258	.92504883	.92895508	.93286133	.93676758
.00E	.87841797	.88232422	.88623047	.89013672	.89404297	.89794922	.90185547	.90576172	.90966797	.91357422	.91748047	.92138672	.92529297	.92919922	.93310547	.93701172
.00F	.87866211	.88256836	.88647461	.89038086	.89428711	.89819336	.90209961	.90600586	.90991211	.91381836	.91772461	.92163086	.92553711	.92944336	.93334961	.93725586

	.F0	.F1	.F2	.F3	.F4	.F5	.F6	.F7	.F8	.F9	.FA	.FB	.FC	.FD	.FE	.FF
.000	.93750000	.94140625	.94531250	.94921875	.95312500	.95703125	.96093750	.96484375	.96875000	.97265625	.97656250	.98046875	.98437500	.98828125	.99218750	.99609375
.001	.93774414	.94165039	.94555664	.94946289	.95336914	.95727539	.96118164	.96508789	.96899414	.97290039	.97680664	.98071289	.98461914	.98852539	.99243164	.99633789
.002	.93798828	.94189453	.94580078	.94970703	.95361328	.95751953	.96142578	.96533203	.96923828	.97314453	.97705078	.98095703	.98486328	.98876953	.99267578	.99658203
.003	.93823242	.94213867	.94604492	.94995117	.95385742	.95776367	.96166992	.96557617	.96948242	.97338867	.97729492	.98120117	.98510742	.98901367	.99291992	.99682617
.004	.93847656	.94238281	.94628906	.95019531	.95410156	.95800781	.96191406	.96582031	.96972656	.97363281	.97753906	.98144531	.98535156	.98925781	.99316406	.99707031
.005	.93872070	.94262695	.94653320	.95043945	.95434570	.95825195	.96215820	.96606445	.96997070	.97387695	.97778320	.98168945	.98559570	.98950195	.99340820	.99731445
.006	.93896484	.94287109	.94677734	.95068359	.95458984	.95849609	.96240234	.96630859	.97021484	.97412109	.97802734	.98193359	.98583984	.98974609	.99365234	.99755859
.007	.93920898	.94311523	.94702148	.95092773	.95483398	.95874023	.96264648	.96655273	.97045898	.97436523	.97827148	.98217773	.98608398	.98999023	.99389648	.99780273
.008	.93945313	.94335938	.94726563	.95117188	.95507813	.95898438	.96289063	.96679688	.97070313	.97460938	.97851563	.98242188	.98632813	.99023438	.99414063	.99804688
.009	.93969727	.94360352	.94750977	.95141602	.95532227	.95922852	.96313477	.96704102	.97094727	.97485352	.97875977	.98266602	.98657227	.99047852	.99438477	.99829102
.00A	.93994141	.94384766	.94775391	.95166016	.95556641	.95947266	.96337891	.96728516	.97119141	.97509766	.97900391	.98291016	.98681641	.99072266	.99462891	.99853516
.00B	.94018555	.94409180	.94799805	.95190430	.95581055	.95971680	.96362305	.96752930	.97143555	.97534180	.97924805	.98315430	.98706055	.99096680	.99487305	.99877930
.00C	.94042969	.94433594	.94824219	.95214844	.95605469	.95996094	.96386719	.96777344	.97167969	.97558594	.97949219	.98339844	.98730469	.99121094	.99511719	.99902344
.00D	.94067383	.94458008	.94848633	.95239258	.95629883	.96020508	.96411133	.96801758	.97192383	.97583008	.97973633	.98364258	.98754883	.99145508	.99536133	.99926758
.00E	.94091797	.94482422	.94873047	.95263672	.95654297	.96044922	.96435547	.96826172	.97216797	.97607422	.97998047	.98388672	.98779297	.99169922	.99560547	.99951172
.00F	.94116211	.94506836	.94897461	.95288086	.95678711	.96069336	.96459961	.96850586	.97241211	.97631836	.98022461	.98413086	.98803711	.99194336	.99584961	.99975586

INDEX

A26-5881-1

Printed in U.S.A. A26-5881-1

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York

READER'S COMMENT FORM

IBM 1130 Functional Characteristics A26-5881-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis.

	Yes	No
• Does this publication meet your needs?	□	□
• Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

- What is your occupation? ______________________________
- How do you use this publication?

As an introduction to the subject?	□	As an instructor in a class?	□
For advanced knowledge of the subject?	□	As a student in a class?	□
For information about operating procedures?	□	As a reference manual?	□

Other ______________________________

- Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

FOLD

FOLD

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY
IBM CORPORATION
MONTEREY & COTTLE RDS.
SAN JOSE, CALIFORNIA
95114

ATTN: PRODUCT PUBLICATIONS DEPT. 455

FOLD

FOLD

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601

IBM 1130 Printed in U.S.A. A26-5881-1