File No. 1130-01
Order No. GA26-5881-6

IBM Systems Reference Library

IBM 1130 Functional Characteristics

Seventh Edition (April 1972)

This is a major revision of, and obsoletes GA26-5881-5 and Technical Newsletters GN34-0033 and GN34-0009. Material related to the following subjects has been added: a description of the IBM 1131 Central Processing Unit Models 1C, 1D, 5B, 5C, and 5D; a hardware description and programming information for the IBM 2311 Disk Storage Drive Models 11 and 12; and programming information pertaining to the IBM System/7 1130 Host Attachment. Other technical changes to the text and illustrations are indicated by a vertical line to the left of the changes.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the latest 1130 System SRL Newsletter, GN20-1130, for the editions that are applicable and current.

Some 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.

Requests for copies of IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, send your comments to IBM Corporation, Systems Publications, Department 27T, P. O. Box 1328, Boca Raton, Florida 33432. Comments become the property of IBM.

Contents

This reference manual contains CPU and I/O programming information, as well as related capacity and speed data for the IBM 1130 Computing System. Specific subjects described are:

- CPU functional characteristics
- CPU instructions
- Input/output interrupts
- Console functions
- Input/output devices
- Storage access channel
- Synchronous communications adapter
- Overlapped input/output and throughput considerations
- Character codes

This publication is intended primarily for programmers who need to know the functions of the machine instructions, (related to the assembler language mnemonics), the functions of attachable I/O devices, and the capacity and timing of the IBM 1130 Computing System. Because programming examples herein use 1130 assembler mnemonics, this manual should be used in conjunction with the *IBM 1130 Assembler Language* manual, Order No. GC26-5927, which describes the syntax of the assembler language as used in the 1130 system.

The reader of this publication should have a working knowledge of basic terminology and concepts of data processing systems. He should also have some experience in a programming language, preferably the assembler language. The reader will probably find that a basic background in communications terminology and concepts is necessary before he can effectively use the section of this manual dealing with the synchronous communications adapter. Certain sections of this manual assume that the reader be able to convert numbers that are represented in binary symbols to equivalent numbers represented in hexadecimal or decimal symbols. Such conversions are explained in *Number Systems,* Order No. GC20-1618.

Operator procedures for the CPU console and the I/O devices attachable to the 1130 system are described in *IBM 1130 Operating Procedures,* Order No. GA26-5717.

All 1130 system publications are listed and abstracted in the *IBM 1130 Bibliography,* Order No. GA26-5916.

Basic IBM 1130 Systems Reference Library (SRL) publications are:

- General Information
  IBM 1130 Configurator, GA26-5915
  IBM 1130 System Summary, GA26-5917
- Machine System
  IBM 1130 Operating Procedures, GA26-5717
- Input/Output
  IBM 1231, 1232 Optical Mark Page Readers, GA21-9012
  IBM 2250 Display Unit Model 4, GA27-2723
  IBM 2285 Display Copier, GA27-2730
  IBM 2501 Card Reader, Models A1 and A2, GA26-5892
- Physical Planning Specifications
  IBM 1130 Installation Manual–Physical Planning, GA26-5914
  IBM 1130 Physical Planning Template, GX26-5997
- Programming Systems–General
  IBM 1130 Card/Paper Tape Programming System Operators Guide, GC26-3629
- Symbolic Assembly Systems
  IBM 1130 Assembler Language, GC26-5927
- FORTRAN
  IBM 1130/1800 Basic FORTRAN IV Language, GC26-3715
- Report Program Generator
  IBM 1130 RPG Language, GC21-5002
- Input/Output Control System
  IBM 1130 Synchronous Communications Adapter Subroutines, GC26-3706
  IBM 1130 Subroutine Library, GC26-5929
- Monitors
  IBM 1130 Disk Monitor System, Version 2–System Introduction, GC26-3709
  IBM 1130 Disk Monitor System, Version 2–Programming and Operator's Guide, GC26-3717
- Systems Techniques
  IBM 1130 FORTRAN Programming Techniques, GC20-1642
- Reference Handbook
  IBM 1130 Reference Summary, GX26-3566

Note. The IBM 1131 Models 4A and 4B, and the IBM 1132 Model 2, are available only in the United States and Canada.

This manual describes the IBM 1130 Computing System. A brief summary of 1130 system facilities is presented first, with more detailed information in subsequent sections.

APPLICATIONS AND PROGRAMMING

The IBM 1130 Computing System is designed for general-purpose computing, which encompasses engineering, commercial, and scientific data-processing applications. This system is especially suited to individual operation by the person who requires a solution to a data-processing problem

On the other hand, because of the availability both of a wide variety of input/output (I/O) devices and of extensive programming support, the 1130 system is capable of meeting the processing requirements demanded by a broad range of applications that do not require such direct interaction between operator and computer.

IBM makes available control program and programming language facilities for the 1130 system. These facilities are basically the following:

- The FORTRAN programming language, applicable to engineering and scientific applications
- The RPG programming language, applicable to commercial applications
- The assembler programming language, mainly for those who desire more specific, direct control of machine functions
- A disk-monitor programming system (version 2) that lessens the application programmer's task of specifying necessary system functions
- A card/paper-tape programming system that lessens the application programmer's task of specifying system functions in a card/paper-tape I/O environment where the full capabilities of the disk-monitor system are not required

IBM also makes available a variety of specific application programs. In the aerospace, construction, engineering, fabrication, and assembly industries, the 1130 system can be be applied to the solution of such data-processing applications as:

- Complex mathematical problems
- Operations analysis and scheduling
- Estimating
- Design of machines and other equipment
- Simulation
- Job cost analysis

In processing industries, some of the applications suitable for the 1130 are:

- Formula blending
- Material balance
- Material evaluation
- Forecasting
- Unit operations

The preceding application lists are, of course, not exhaustive; they merely point to some areas in which the 1130 system can be used successfully. Many other data processing jobs in such business activities as transportation, marketing, finance, insurance, utility, and distribution are applicable to solution by the 1130 system.

For the availability, method of ordering, and cost information of IBM products, whatever the product – machine, education course, or program – consult with your IBM representative.

For more specific details concerning the disk-monitor or card/paper-tape programming systems, the FORTRAN, RPG, or assembler languages, or other program facilities that IBM makes available for use with the 1130 Computing System, refer to other System Reference Library publications. These publications are listed and abstracted in the *IBM 1130 Bibliography,* Order No. GA26-5916.

CENTRAL PROCESSING UNIT AND CORE STORAGE

The principal unit in the 1130 Computing System is the IBM 1131 Central Processing Unit (CPU). Very compact and desk-like in appearance, the CPU (Figures 1 and 2) houses the system console. The CPU contains core storage and electronic circuits—circuits that implement such functions as machine-instruction execution and interruption actions, and circuits necessary for the attaching of input/output devices to the system. The console printer, which operates at printing speeds up to 15.5 characters per second, is also located on the CPU frame.

Core-storage capacities of the models of the 1131 CPU are shown in Figure 3. The capacities are in words; a word is made up of 16 binary-digit positions in the 1130 system. (Other characteristics of data lengths, formats, and so forth are in subsequent sections of this book.) Also, in Figure 3, are listed the core-storage cycle times for the various models. A core-storage cycle is the time required to read a word from or store a word into core storage.

In addition to the core-storage capacities listed in Figure 3, the 1131 Models 2, 3, and 4 (*not* Models 1A, 1B, 1C, 1D, 5B, 5C, and 5D) contain a disk-storage drive that uses an IBM 2315 Disk Cartridge. This disk arrangement provides on-line storage for up to 512,000 words (1,024,000 bytes) of information on a single 2315 cartridge. Cartridges, however, can be manually changed, thus allowing for unlimited off-line storage of data.

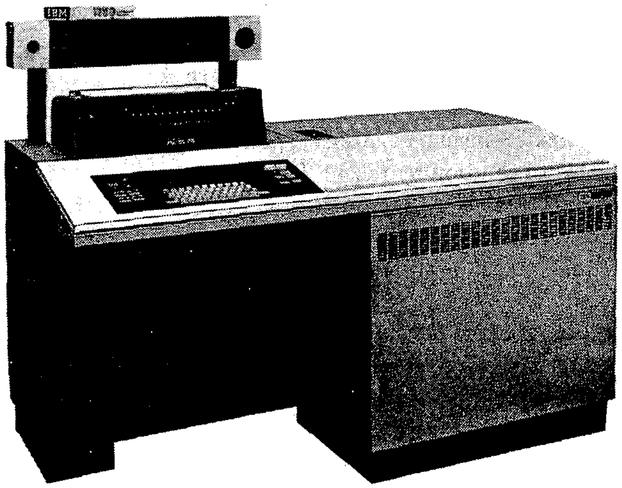


Figure 1. IBM 1131 Central Processing Unit (Model 1A, 1B, 2A, 2B, 4A, or 4B)

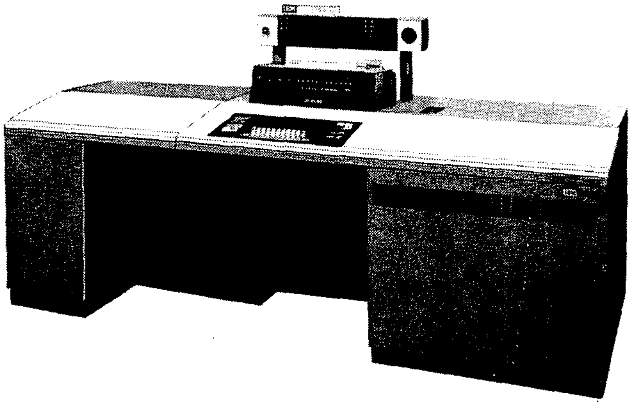


Figure 2. IBM 1131 Central Processing Unit (Model 1C, 1D, 2C, 2D, 3B, 3C, 3D, 5B, 5C, and 5D)

1131 CPU Model	Core-Storage Capacity (in 16-bit words)*	Core-Storage Cycle Time (in micro-seconds)**	Disk Storage
1A	4,096	3.6	No
2A, 4A	4,096	3.6	Yes
1B	8,192	3.6	No
2B, 4B	8,192	3.6	Yes
3B	8,192	2.2	Yes
5B	8,192	2.2	No
1C	16,384	3.6	No
2C	16,384	3.6	Yes
3C	16,384	2.2	Yes
5C	16,384	2.2	No
1D	32,768	3.6	No
2D	32,768	3.6	Yes
3D	32,768	2.2	Yes
5D	32,768	2.2	No

*The 16-bit word increment of information is accessed during a read or write operation. This 16-bit word is equivalent to two 8-bit bytes, a term used to describe an increment of information in other systems. Consequently, in terms of bytes, the available storage sizes in the 1130 system are 8,192, 16,384, 32,768, and 65,536 bytes.

**For Models 1, 2, 3, and 5, machine cycle time is the same as storage cycle time. For Model 4, machine cycle time is 5.85 microseconds. Machine cycle time is the time required for the CPU to perform one step in the execution of an instruction.

Figure 3. Storage Capacities and Cycle Times

I/O DEVICES

IBM input and output devices that can be used in the 1130 Computing System are listed in this section. Consult with your IBM representative for detailed information concerning the installing of any 1130 system configuration.

Except for the console printer that comes as a part of the CPU, an attachment feature or feature combination is required before any I/O device can be connected to and used with the system. (Refer to the *IBM 1130 Configurator*, Order No. GA26-5915.)

Card Readers and Punches

IBM 1442 Card Read Punch Model 6 or 7

IBM 1442 Card Punch Model 5

IBM 2501 Card Reader Model A1 or A2

Synchronous Communications Adapter (SCA)

This adapter permits the 1130 system to function as a remote processor terminal communicating with the following terminals or systems:

- IBM System/360 Models 25, 30, 40, 50, 65, 67 (in Model 65 mode), 75, or 85 by means of an IBM 2701 Data Adapter Unit or an IBM 2703 Transmission Control Unit in binary-synchronous-communications (BSC) mode
- IBM System/360 Models 25, 30, 40, 50, 65, 67, 75, or 85 by means of an IBM 2701 Data Adapter Unit in synchronous-transmit-receive (STR) mode
- IBM System/360 Model 25 by means of its integrated communications attachment in BSC mode
- IBM System/360 Model 20 by means of its communications adapter in STR mode
- IBM System/360 Model 20 by means of its communications adapter in BSC mode
- Another IBM 1130 Computing System by means of that system's synchronous communications adapter in BSC or STR mode
- IBM 1009 Data Transmission Unit in STR mode
- IBM 1013 Card Transmission Terminal in STR mode
- IBM 7702 Magnetic Tape Transmission Terminal in STR mode
- IBM 7711 Data Communication Unit in STR mode
- IBM 2770 Data Communications System
- IBM 2780 Data Transmission Terminal

Disk Storage

Single Disk Storage (in 1131 CPU, Models 2, 3, and 4 only)

IBM 2310 Disk Storage Model B1 or B2 (up to four drives on a system)

IBM 2311 Disk Storage Drive Models 11 and 12 (up to two drives on a system)

Graphic I/O

IBM 2250 Display Unit Model 4

IBM 2285 Display Copier (The 2285 does not require program control. It provides paper copy of images displayed on the 2250.)

Optical Mark Reader

IBM 1231 Optical Mark Page Reader Model 1

Paper Tape

IBM 1055 Paper Tape Punch

IBM 1134 Paper Tape Reader

Plotter

IBM 1627 Plotter Model 1 or 2

Printers

Console Printer (comes as a part of the 1131 CPU)

IBM 1132 Printer Model 1 or 2

IBM 1403 Printer Model 6 or 7

Sensor Based Systems

IBM System/7 (5010 Processor Module Models B2 through B16)

DATA FORMATS

In order to be accessible to the program during processing, data must be stored in core storage. Thus, input job data is read by an input device, stored in core storage, and then processed. Results (output data) are sent to an output device.

Input data can be represented in a variety of ways depending upon the input medium used. A medium is the material on which data is recorded. For example, a card that is punched with holes (which represent data) is a medium; the card can be read by a card reader. (Refer to subsequent sections for descriptions of the various input devices and the media they use.)

In the 1130 system, data is read from or stored in core storage on a *word* basis. A word is made up of sixteen positions, numbered 0 to 15.

Position → 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

The left-most position (0) is called the high-order position; the right-most position (15) is called the low-order position.

Each position can be at a value of 0 (also called off) or 1 (also called on). For example, all 16 positions of a word can be:

Position →	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value →	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Or:

Position →	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value →	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Or, any combination is possible:

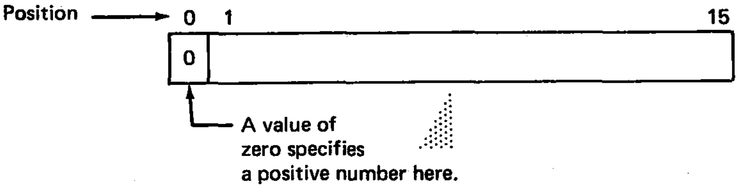
Position →	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value →	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	1
Or →	0	1	1	1	0	0	1	1	1	1	1	0	0	0	0	0
Or →	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1

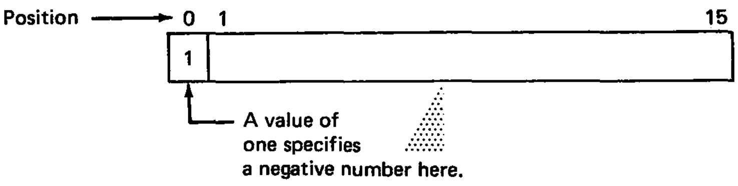
Each position within a word in core storage is called a *bit.* Numeric, alphabetic, special-character, or logical information can be represented by the bit values in a word. Both the combination of the bits in a word and the intention of the programmer who organizes the program and data in core storage determine whether the data is numeric, alphabetic, special-character, or logical.

Numeric Data Formats for Arithmetic Operations

Numeric computations in the 1130 system are performed in binary arithmetic by the arithmetic instructions. In other words, the 1130 operates on binary data to produce binary results.

Position 0 (the high-order position) of a word specifies whether the rest of the word contains a positive or negative number. If position 0 is at a value of 0, then the number in the rest of the word is positive; if position 0 is at a value of 1, then the number in the rest of the word is negative.



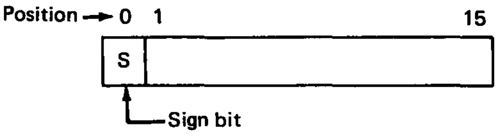


The assumption has been made that the reader can when given a numeric value represented by the symbols in the binary, hexadecimal, or decimal numbering system convert those symbols to an equivalent numeric value in any of the same systems. For example, the reader should be able to convert the binary symbols 11100010 to the hexadecimal symbols E2 or to the decimal symbols 226. The reader should also be able to do simple addition or subtraction with numeric values represented by binary or hexadecimal symbols. Refer to *Number Systems,* Order No. SC20-1618, for basic information on binary and hexadecimal numbering systems.

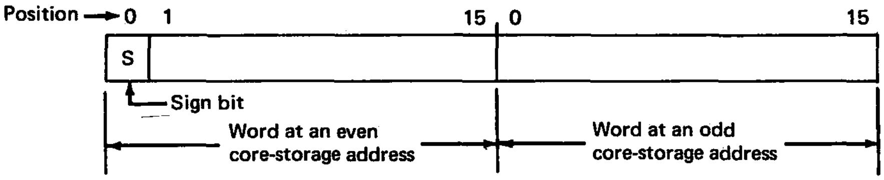
Numeric data can be handled in either single or double precision. Single precision is the single word binary format just described, where bit 0 is the sign bit and bits 1 through 15 contain a numeric value.

When a binary number is contained in two successive words in core storage, with the sign in the leftmost position of the leftmost word, then that number is in double precision format. The leftmost word of the pair of words must be at a core-storage location that has an even address.

Single Precision Format



Double Precision Format



Programming Note: A double-precision number may be stored in core storage starting at an odd address. But instructions that process double-precision operands use two successive words only if an even address is specified. If an odd address is specified, the operation is performed with the odd word twice. Anyone programming the machine to operate in this unusual manner should have an intimate understanding of the internal workings of the 1131 CPU.

Notice that only one sign position is used out of the 32 available positions in the double-precision format. In single precision, one position is used for the sign and 15 positions for the number. The ranges of numeric values that can be represented in single and double precision formats are shown in binary and decimal system symbology in Figure 4.

In Figure 4, negative numbers are in two's-complement binary form; the sign position, at a value of one, signifies the negative number. For example, the single precision number 1111 1111 1111 1111 is shown in Figure 4 as having an equivalent decimal value of -00001 (the left zeros are shown merely to maintain consistency between the binary and decimal notation shown). But this binary value is really the two's complement of negative 1. The true form can be obtained as follows:

Invert (change each position from 1 to 0):

1111 1111 1111 1111
invert
0000 0000 0000 0000

Add 1 to the inverted number:

0000 0000 0000 0000
+1
0000 0000 0000 0001

Negative results are always produced in two's-complement form.

Single Precision		
Binary		Decimal
Position 0 (Sign)	Numeric Value	
0	000 0000 0000 0000	00000
+	to	to
0	111 1111 1111 1111	32,767
1	000 0000 0000 0000	-32,768
-	to	to
1	111 1111 1111 1111	-00001

Double Precision		
Binary		Decimal
Position 0 (Sign)	Numeric Value	
0	000 0000 0000 0000 0000 0000 0000 0000	0000000000
+	to	to
0	111 1111 1111 1111 1111 1111 1111 1111	2,147,483,647
1	000 0000 0000 0000 0000 0000 0000 0000	-2,147,483,648
	to	to
1	111 1111 1111 1111 1111 1111 1111 1111	-0000000001

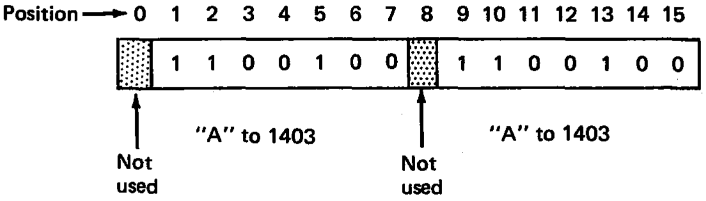
Figure 4. Numeric Value Ranges for Single and Double Precision Operands

Even though the arithmetic instructions handle data only in the binary formats just listed, other types of arithmetic operations can effectively be performed by use of arithmetic subroutines. These subroutines are part of IBM programming systems and are described in *IBM 1130 Subroutine Library,* Order No. GC26-5929.

Character Codes

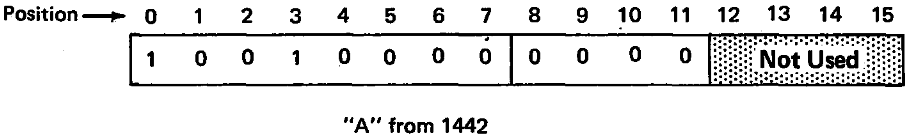
There is no fixed internal character code in the 1131 CPU. Conversions from one bit pattern to another must be performed, however, because many input/output devices are code sensitive. Such conversions must be done by program routines.

In order to print the letter A, for example, a 1403 printer must receive a bit pattern of 1100100 in the output print record. Other bit patterns cause other characters to be printed. The bit pattern to the printer comes from positions 1 through 7 or positions 9 through 15 of a word in core storage (part of an output print record):



If the preceding word is part of a print record to the 1403, then AA is printed at two adjacent print positions.

On the other hand, consider that the bit pattern for the letter A is received in an input record from the 1442 card reader. One core-storage word is required for each card column read by the 1442. The bit pattern for the A from the 1442 is stored into core storage as follows:



Clearly, the bit pattern for the A from the 1442 must undergo some conversion before it can be sent to the 1403 in an output print record.

But even if input data is from the 1442 card reader and output data goes to the 1442 card punch, conversion of bit patterns may be necessary. Assume, as an example, that two fields in an input card record are to be added together, and the result is to be punched into a card as output data. The input data read from the card is in card code, not in binary. This input data must be converted to binary format before the arithmetic instructions can handle it. (Arithmetic instructions operate only on binary data.) Then, after the computation of adding the two numbers, the result must be converted back to card code and sent in an output record to the 1442 card punch.

Note: Any peculiarities or restrictions related to the handling of bit patterns for the 1442, 1403, or other I/O devices are in sections in this book that are devoted to those devices. All conversions of data from one format to another must be program-controlled. Data conversion subroutines are in IBM programming systems for the 1130 system. For descriptions of the data conversion subroutines, refer to *IBM 1130 Subroutine Library,* Order No. GC26-5929. Programming methods of converting from one data code to another are not described in this functional-characteristics manual.

The various codes used in the 1130 system, via program conversion, are listed in chart form in Appendix A. The Extended Binary Coded Decimal Interchange Code (EBCDIC) shown in Appendix A is an eight-bit code that is given for reference because it is used in other IBM systems (for example, the IBM System/360).

CORE-STORAGE ADDRESSES

Core-storage capacity is dependent upon the 1131 CPU model but, at a maximum, is 32,768 words (see Figure 3). The address for any core-storage location in an 1131 CPU can be contained in a 16-bit word:

ADDRESS		
In Binary (16 Bits)	In Hexadecimal	In Decimal
0000 0000 0000 0000	0000	00000
0000 1111 1111 1111	0FFF	4, 095
0001 1111 1111 1111	1FFF	8, 191
0011 1111 1111 1111	3FFF	16, 383
0111 1111 1111 1111	7FFF	32, 767

The core-storage location specified by address 0000 (hexadecimal) is sequentially next to the highest location *in all 1131 models.* This means that if a word address that is one greater than the maximum for a particular system is specified, the location accessed is the location specified by address 0000 (hexadecimal). This addressing peculiarity is called *wraparound.*

In fact, any 16-bit address accesses a core-storage location regardless of the core-storage capacity of the system. The reason for this is that high-order (leftmost) address bits that specify addresses above the capacity of the system are not used and are ignored. For example, in a system that has 4,096 word locations in core storage, the maximum address is:

0000 1111 1111 1111 (for location 4,095 in decimal notation)

The four leftmost bits (for a 4,096 word capacity system) are ignored during actual addressing. Therefore, in this same capacity system, location 4,095 is addressed by all of the following addresses:

0000 1111 1111 1111
0001 1111 1111 1111
0100 1111 1111 1111
and so on.

If you use absolute addresses in conjunction with wraparound, you should be aware of the fact that the core-storage capacity of the system determines the actual location accessed. For example, assume that wraparound from location 4,095 to location 0000 is desired on a minimum capacity system. Wraparound does not occur if the same addresses are used on any other system. On the 4,096-word capacity system, wraparound occurs as follows:

Binary Address	Equivalent Decimal Address	Location Accessed
0000 1111 1111 1111	4,095	4,095
add 1 for wraparound		
0001 0000 0000 0000	4,096	0000

For an 8,192-word or any larger capacity system, the same addresses result in accessing storage as follows:

Binary Address	Equivalent Decimal Address	Location Accessed
0000 1111 1111 1111	4,095	4,095
+1		
0001 0000 0000 0000	4,096	4,096

Reserved Core-Storage Locations

All 1131 CPU's have certain reserved locations in core storage. The reserved locations have the same addresses regardless of the CPU model:

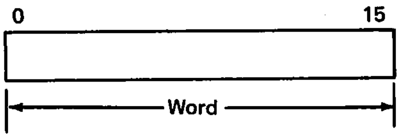
Core Storage Address		
Hexadecimal	Decimal	Name of Reserved Locations
0001	00001	Index-Register 1 (XR1)
0002	00002	Index-Register 2 (XR2)
0003	00003	Index-Register 3 (XR3)
0008-000D	00008-00013	Interrupt Vectors
0020-0027	00032-00039	1132 Printer Scan Field

You should not use these locations for purposes other than the ones for which they are intended; if you do, you may lose or destroy data. (Subsequent sections of this manual describe the items named in the preceding list.)

INSTRUCTION FORMATS

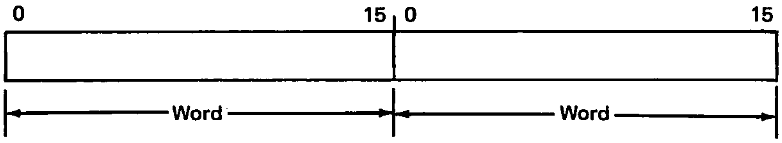
There are only two instruction formats in the 1130. The short-instruction format requires 16 bit positions; an instruction in the short-instruction format, therefore, requires one core-storage word location:

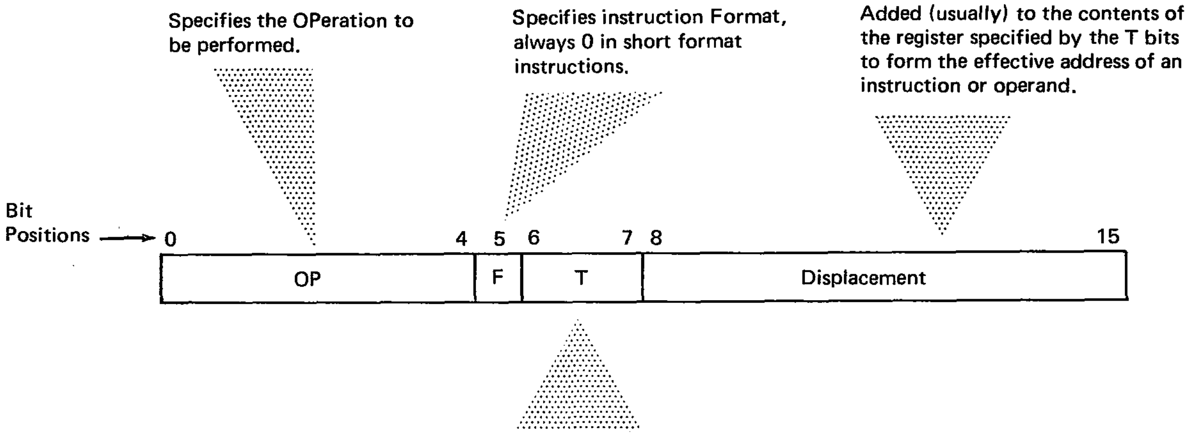
Short Instruction



The long-instruction format requires 32 bit positions; an instruction in the long-instruction format, therefore, requires two core-storage word locations:

Long Instruction





Specifies the OPeration to be performed.

Specifies instruction Format, always 0 in short format instructions.

Added (usually) to the contents of the register specified by the T bits to form the effective address of an instruction or operand.

Tag bits specify a register, the contents of which are (usually) added to bits 8-15 (the displacement) to form an operand or instruction address. The registers that can be specified are:

Bits 6 7	Specify
0 0	Instruction Address Register (IAR)
0 1	Index Register 1 (XR1)
1 0	Index Register 2 (XR2)
1 1	Index Register 3 (XR3)

Figure 5. Short-Instruction Format

A brief summary of what each group of bits represents in the short-instruction format is shown in Figure 5. Similarly, the significance of bit positions in the long-instruction format is shown in Figure 6. However, you should refer to the description of a specific instruction in order to find out how the bits are used in that instruction. Bits 8 and 9 in the short format, for example, specify the type of shift and not a displacement in certain instructions.

EFFECTIVE-ADDRESS GENERATION

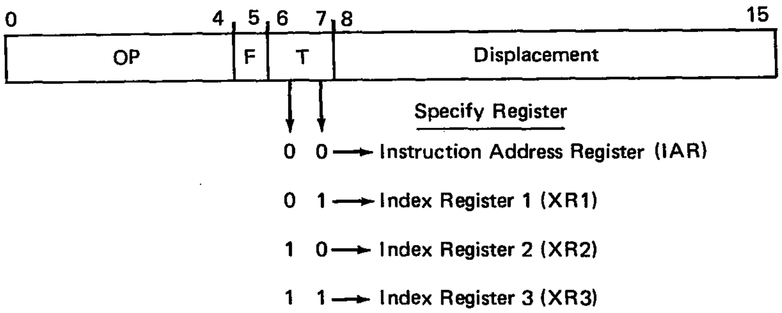
The actual core-storage location at which specific data (or an instruction) is located is identified by an *effective address.* In other words, the address of each core-storage location in the 1130 system is called the effective address of that location. For example, the effective address of the data 0404 in the following list is 0001.

Data (Hexadecimal)	Core-Storage Word Address (Hexadecimal)
FE01	0000
0404	0001
8424	0002
FFFF	0003

The term effective address is used because an address may be directly available or it may be computed, depending upon the operation in progress. The methods of generating effective addresses are different for short and for long format instructions.

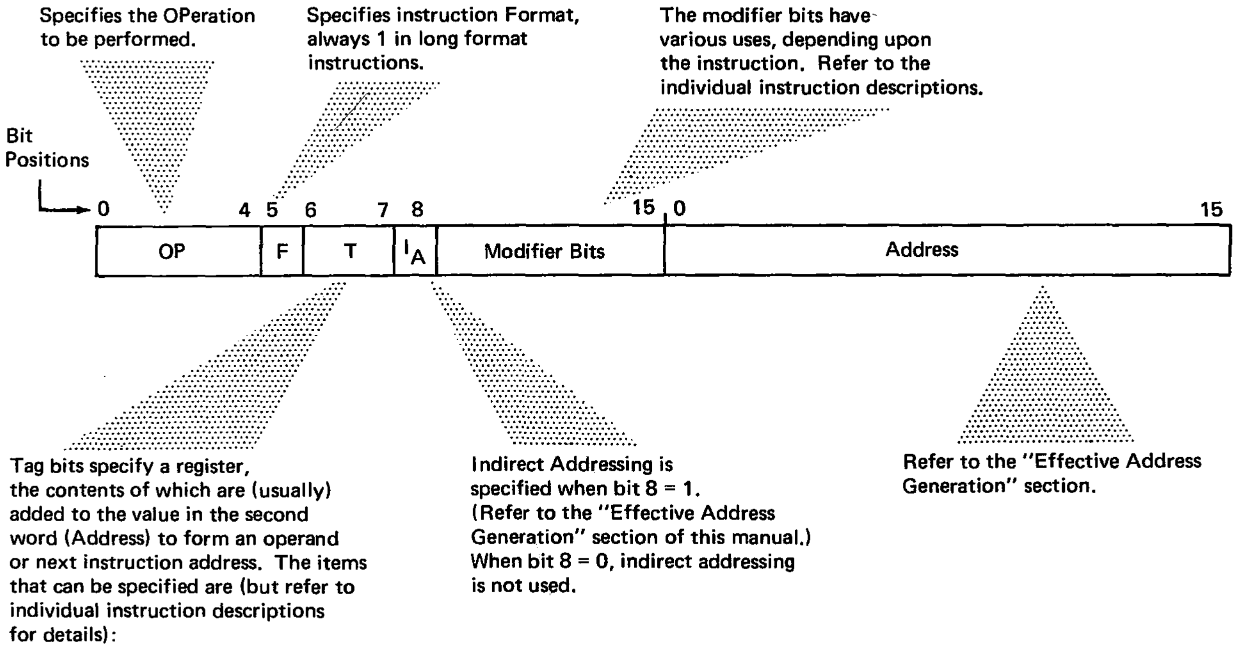
Short-Instruction Address Generation

In the short-instruction format, the tag (T) bits specify a register as follows:



The value of the displacement (bits 8 through 15 of the instruction) are added to the contents of the specified register. The result of this addition is the effective address:

Specified Register Contents + Displacement = Effective Address



Specifies the OPeration to be performed.

Specifies instruction Format, always 1 in long format instructions.

The modifier bits have various uses, depending upon the instruction. Refer to the individual instruction descriptions.

Tag bits specify a register, the contents of which are (usually) added to the value in the second word (Address) to form an operand or next instruction address. The items that can be specified are (but refer to individual instruction descriptions for details):

Indirect Addressing is specified when bit 8 = 1. (Refer to the "Effective Address Generation" section of this manual.) When bit 8 = 0, indirect addressing is not used.

Refer to the "Effective Address Generation" section.

Bits 6 7	Specify
0 0	No Index Register
0 1	Index Register 1 (XR1)
1 0	Index Register 2 (XR2)
1 1	Index Register 3 (XR3)

Figure 6. Long-Instruction Format

Tag Bits = 00 (Instruction Address Register)

The basic purpose of the instruction-address register is to point to each instruction that is to be executed next. Effective size of the instruction-address register, which is dependent upon the core-storage capacity of the system, is 12, 13, 14, or 15 bits long.

The maximum addresses in the instruction-address register for the various core-storage capacities are:

Instruction Address Register (Effective Size and Maximum Value)	Equivalent Core-Storage Address in Decimal
12 bits 1111 1111 1111	4,095
13 bits 1 1111 1111 1111	8,191
14 bits 11 1111 1111 1111	16,383
15 bits 111 1111 1111 1111	32,767

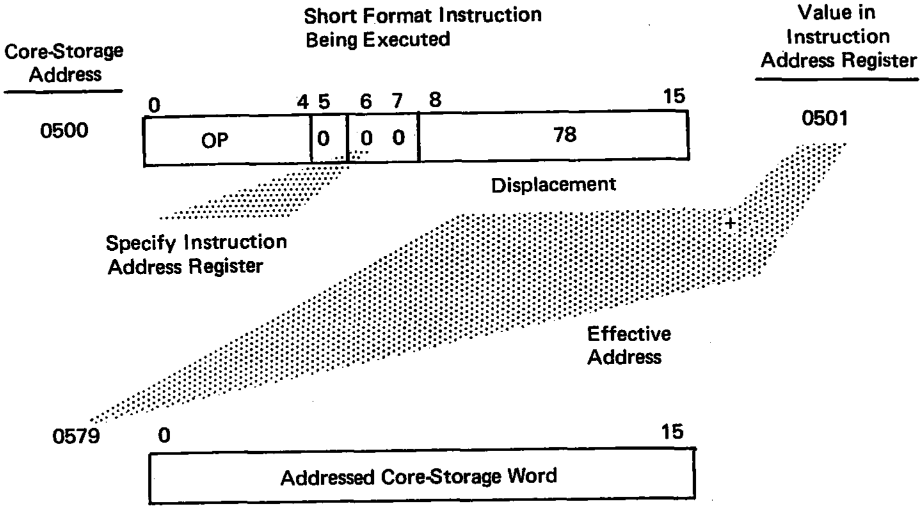
When the tag bits specify the instruction-address register, the displacement (from the instruction) is added to the current value in the instruction-address register in order to generate the effective address. The instruction-address register at this time contains the address of the core-storage location that immediately follows the instruction being executed.

For example, a short instruction from the core-storage location specified by address 0500 is being executed (Figure 7). During execution of this instruction, the instruction-address register is updated automatically to 0501, the address of the core-storage word immediately following the short instruction. The value in the instruction-address register is then 0501 when the effective address is generated.

Further assume that the displacement in the short instruction at location 0500 is:

Displacement

Binary									Hexadecimal Equivalent
Bit Positions →	8	9	10	11	12	13	14	15	
	0	1	1	1	1	0	0	0	78



Contents of Instruction Address Register	+	Displacement	=	Effective Address
0501	+	0078	=	0579

Figure 7. Positive Displacement (Short-Instruction Address Generation)

Also assume that the core-storage capacity is 4,096 words, and, therefore, the instruction-address register contains 12 bit positions. To form the effective address, the displacement is added to the value in the instruction-address register:

	Binary
0501 (Hex) from instruction-address register	0101 0000 0001
78 (Hex) from displacement in instruction +	0000 0000 0111 1000
Effective address 0579 (Hex)	0000 0101 0111 1001

The effective address is displaced by a value of 78 from the current value (0501) of the instruction-address register. This is why bits 8 through 15 of the instruction are called the *displacement.*

In the preceding example, the value of the high-order (leftmost) bit of the displacement is propagated to the left to form a 16-bit operand (0000 0000 0111 1000) and then added to the value from the instruction-address register. This is a normal operation that determines whether the displacement is positive or negative with respect to the value in the instruction-address register.

The eight available bits in the short instruction, which collectively form the displacement, provide a range from -128 words (decimal) to +127 words (decimal) of addressing capability. Whether the displacement is positive or negative is determined by the value of bit eight – the leftmost bit of the displacement in the instruction. When bit 8 = 0, the displacement is positive:

	Positive Displacement Value	
	Binary	Decimal Equivalent
Current value in instruction-address register +	0000 0000	000
	to	to
	0111 1111	127

When bit 8 = 1, the displacement is negative:

	Positive Displacement Value	
	Binary	Decimal Equivalent
Current value in instruction-address register +	1111 1111	-1
	to	to
	1000 0000	-128

The value in the instruction-address register is always considered positive, even if its leftmost bit is at a value of 1.

Tag Bits = 01, 10, or 11 (Index Register 1, 2, or 3)

When an index register is specified, address generation is similar to that used when the instruction-address register is specified. Two basic differences, however, are:

1. Each index register occupies a 16-bit word in core storage. (Index-register 1 is in core-storage word location 0001. Index-register 2 is in core-storage word location 0002. Index-register 3 is in core-storage word location 0003.)
2. An index register can be loaded under program control with any desired value.

Any index register can be loaded by the program with either a negative (bit 0 = 1) or a positive (bit 0 = 0) value. For the addressing scheme described here, however, the value in the index register is always considered positive; high-order, unused bits beyond the capacity of the system are ignored (refer to "Core Storage Addresses").

Figure 8 is an example of a negative displacement used with the contents of an index register to form an effective address.

Long-Instruction Address Generation

Two types of addressing are defined for long-format instructions: *direct* and *indirect* addressing. An effective address generated in the manner already described for short-format instructions is called a *direct address* – short-format instructions, in fact, can perform only direct addressing.

The indirect address (IA) bit (bit 8) in long-format instructions specifies whether addressing is direct or indirect:

Indirect Address (IA bit 8)	Type of Addressing
0	direct
1	indirect

Direct Addressing (IA Bit = 0)

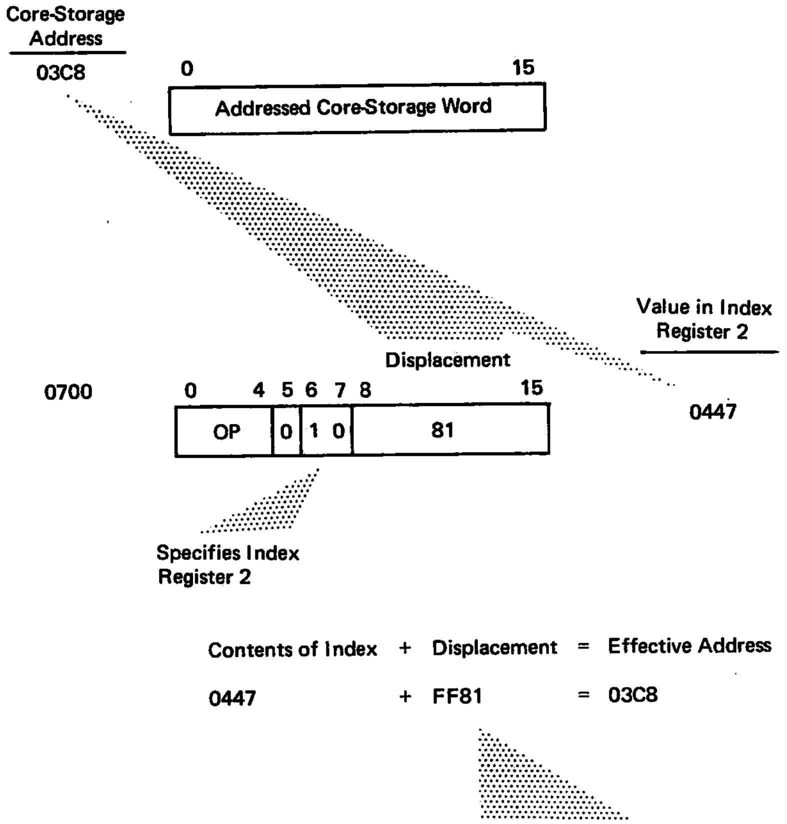
Here the tag bits can specify an index register (1, 2, or 3) as in short-format addressing, or no index register (0). The instruction register is not used to compute the effective-address as it is in the short-format instructions.

When an index register is specified, the second word (address field) of the long-format instruction is added (Figure 9) to the value in the index register to form the effective address.

Indirect Addressing (IA Bit = 1)

In indirect addressing, some word location in core storage contains a value that is the effective address. The desired effective address must be program-loaded into the word. This location, which can be any word in core storage, is itself first addressed by direct addressing. Then the contents of that word become the effective address (Figure 10).

When the tag bits equal 00, as in direct addressing in long-format instructions, the core-storage word that contains the effective address is located by the value in the address field in the instruction.



The leftmost bit of displacement 81 (equals 1000 0001 in binary) is propagated to the left to form FF81. Addition then occurs in the CPU to effectively subtract to produce the results shown. (A complement number has been added to the contents of index register 2.)

Figure 8. Negative Displacement

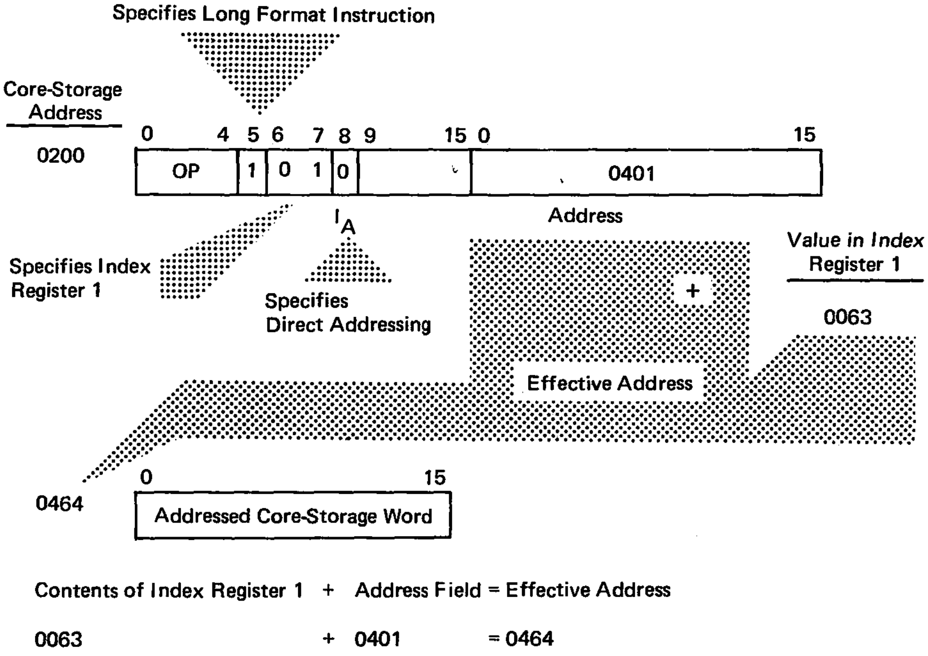


Figure 9. Direct Addressing (Long-Format Instruction)

Summary of Addressing Concepts

Exceptions to the general schemes of addressing are shown in Figure 11.

Short-Format Instructions (F=0)

Tag Bits	Specified Register	Effective Address (EA)
T=00	Instruction-Address Register (IAR)	EA = IAR + Displacement
T=01	Index-Register 1 (XR1)	EA = XR1 + Displacement
T=10	Index-Register 2 (XR2)	EA = XR2 + Displacement
T=11	Index-Register 3 (XR3)	EA = XR3 + Displacement

The high-order (leftmost) bit of the displacement determines whether the displacement is effectively added to or subtracted from the contents of the specified register:

Short Format Bit 8 Value (Leftmost Bit of Displacement)	Effective Operation
=0	Add displacement
=1	Subtract displacement (complement add)

The displacement range is -128 to +127 (decimal) word locations from the value in the specified register.

Long-Format Instructions (F=1)

Direct Addressing (IA=0):

Tag Bits	Specified Register	Effective Address (EA)
T=00	None	EA = Address field
T=01	Index-Register 1 (XR1)	EA = XR1 + Address field
T=10	Index-Register 2 (XR2)	EA = XR2 + Address field
T=11	Index-Register 3 (XR3)	EA = XR3 + Address field

High-order (leftmost) bits in the generated effective address that are beyond the storage capacity of the system are ignored.

Indirect Addressing (IA=1): A core-storage word location is specified. The contents of that word are the effective address. The core-storage word is specified in the same manner as that just described in direct addressing.

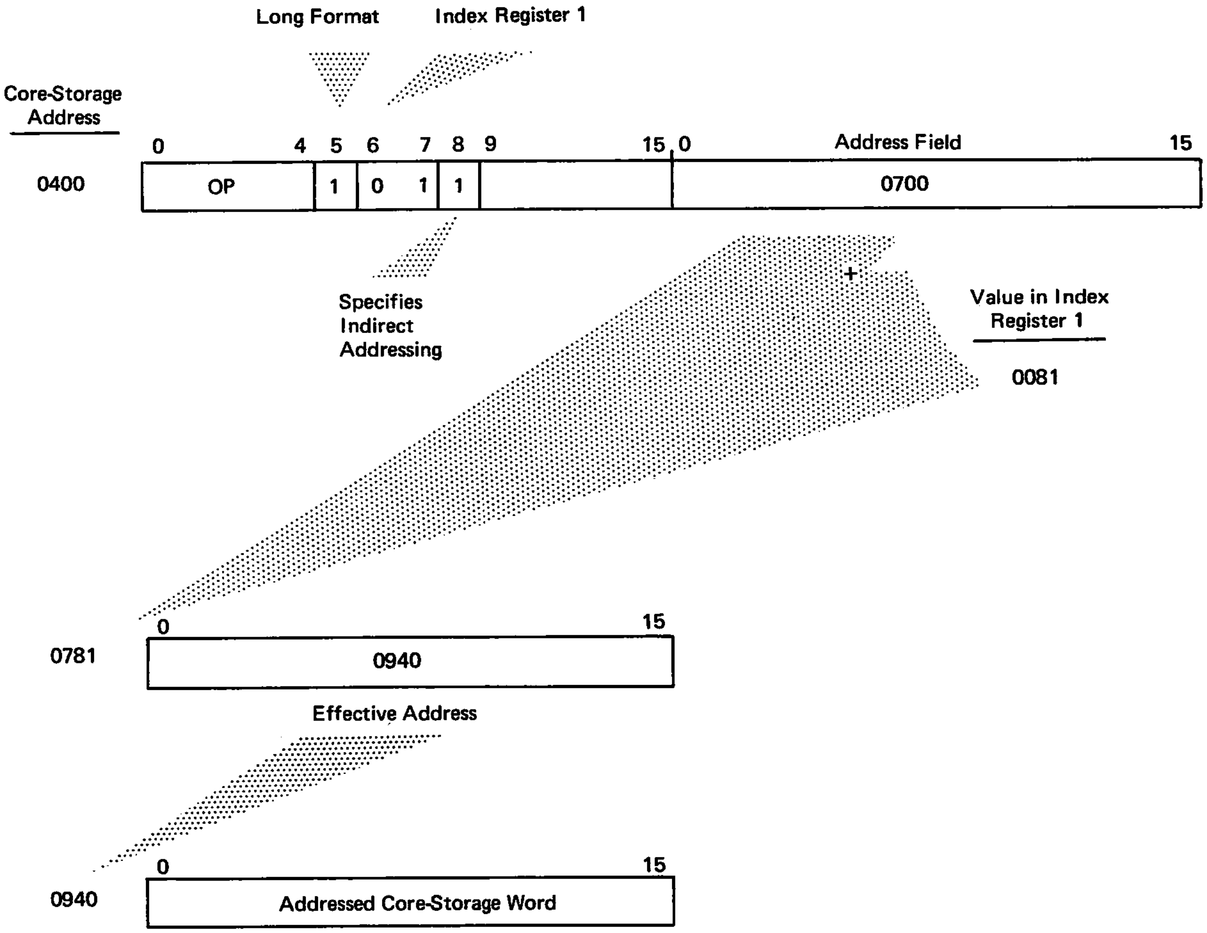


Figure 10. Indirect Addressing (Long-Format Instructions Only)

Instruction	Short Format	Long Format	
		IA=0	IA=1
LDS	The values of bits 14 and 15 of the instruction are used to set/reset the carry and overflow indicators.	-----------	-------------
LDX	The expanded displacement (16 bits) is loaded into IAR, XR1, XR2, or XR3.	The address field is loaded into IAR, XR1, XR2, or XR3.	The contents of the storage location specified by the address field are loaded into IAR, XR1, XR2, or XR3.
MDX	The expanded displacement (16 bits) is added to IAR, XR1, XR2, or XR3.	Tag=00: The expanded displacement (16 bits) is added to the contents of the storage location specified by the address field. Tag≠00: The contents of the address field are added to the index register specified.	Tag=00: The expanded displacement (a 16-bit negative number) is added to the contents of the storage location specified by the address field. Tag≠00: The contents of the storage location specified by the address field are added to the index register specified.
Shift Instructions	The number of shifted positions is controlled by the contents of the six low-order bits of the displacement (or register, if specified).	-----------	-------------
STX	The effective address is obtained by adding the displacement to the IAR.	The address field is the effective address.	The contents of the storage location specified by the address field is the effective address.
WAIT	The displacement is not used.	--------------	--------------

IAR = Instruction Address Register

XR1 = Index Register 1

XR2 = Index Register 2

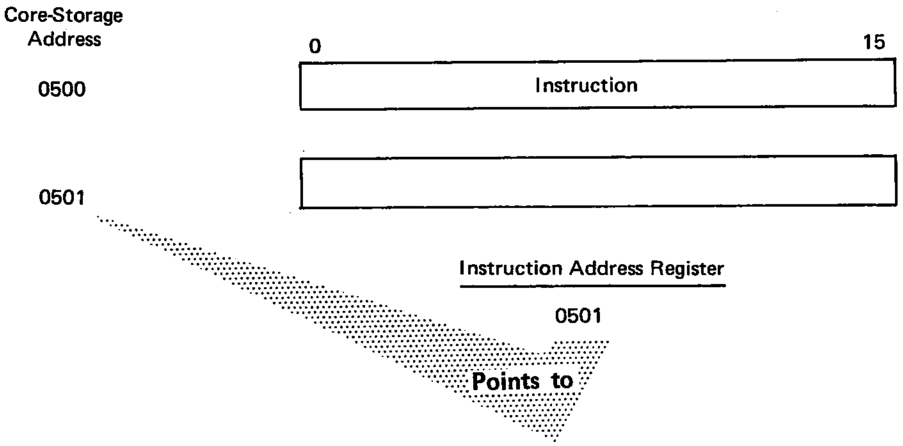
XR3 = Index Register 3

Figure 11. Exceptions to Effective Address Generation

PROGRAM REGISTERS AND PROGRAM INDICATORS

Instruction-Address Register

The instruction-address register holds the address of the next instruction to be executed. For example, assume that the following instruction is being executed:



The instruction-address register contains the address of the core-storage word immediately following the instruction being executed. In most cases, this next word is the next instruction to be executed. Sometimes, however, the contents of the instruction-address register are changed as a result of the instruction being executed. Execution of a branch instruction, for example, can cause accessing of the next instruction from a core-storage location other than the one immediately following the current instruction.

The effective size of the instruction-address register depends upon the core-storage capacity of the system:

Effective Size of Instruction-Address Register (Bits)	Core-Storage Capacity (Words)
12	4,096 (decimal)
13	8,192
14	16,384
15	32,768

In short-format instructions when the tag bits equal 00, the contents of the instruction-address register are used in effective address generation.

Index-Registers 1, 2, and 3

Index-registers 1, 2, and 3, each of which is a 16-bit word location in main storage, are used in address generation during instruction execution. Refer to "Effective Address Generation" for a description of addressing concepts.

Each index register, just as any other core-storage word, can contain a positive or negative number. When bit 0 = 0, the number is positive; when bit 0 = 1, the number is negative.

Accumulator (ACC)

This 16-bit register is used in arithmetic, shift, logical, and I/O operations.

In arithmetic operations, the accumulator is program-loaded with one of the two operands. Next, the operation is performed using one operand from core storage and the other from the value loaded into the accumulator. The result is in the accumulator at the end of the operation. Loading an operand into the accumulator and then specifying, in an arithmetic instruction, the operation and the location of the other operand in core storage are program-controlled operations. Development of the result in the accumulator is a machine function that is dependent upon the arithmetic instruction that is executed. The process is summarized in Figure 12. For arithmetic operations with 32-bit operands or 32-bit results, the accumulator and an accumulator extension are used (see "Accumulator Extension").

The accumulator can be loaded from any core-storage word by a load-accumulator instruction. Contents of the accumulator can be stored into any core-storage word by a store-accumulator instruction.

Refer to the specific instruction descriptions for discussions of the ways in which the contents of the accumulator can be manipulated in shift and logical operations.

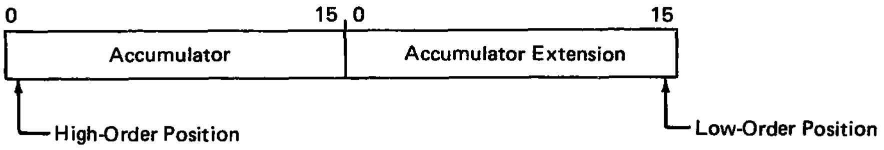
The accumulator is also used during I/O operations to hold status information. The status information indicates such conditions as:

- An interruption has occurred for a specific I/O device, or
- An I/O device is not ready

Status words are of two types: the interruption level status word (ILSW) and the device status word (DSW). A status word, ILSW or DSW, is loaded into the accumulator under program control. (The ILSW's and DSW's are described elsewhere in this manual.)

Accumulator Extension (Q)

This 16-bit register is used in arithmetic operations when 32-bit operands are used or 32-bit results are produced. The extension is to the right of the accumulator:



For descriptions of how the accumulator and accumulator extension function together, refer to the sections of this manual devoted to the following instructions:

Instruction	*Mnemonic*
Add double	AD
Divide	D
Load double	LDD
Multiply	M
Rotate right ACC and EXT	RTE
Shift left ACC and EXT	SLT
Shift left and count ACC and EXT	SLC
Shift right ACC and EXT	SRT
Store double	STD
Subtract double	SD

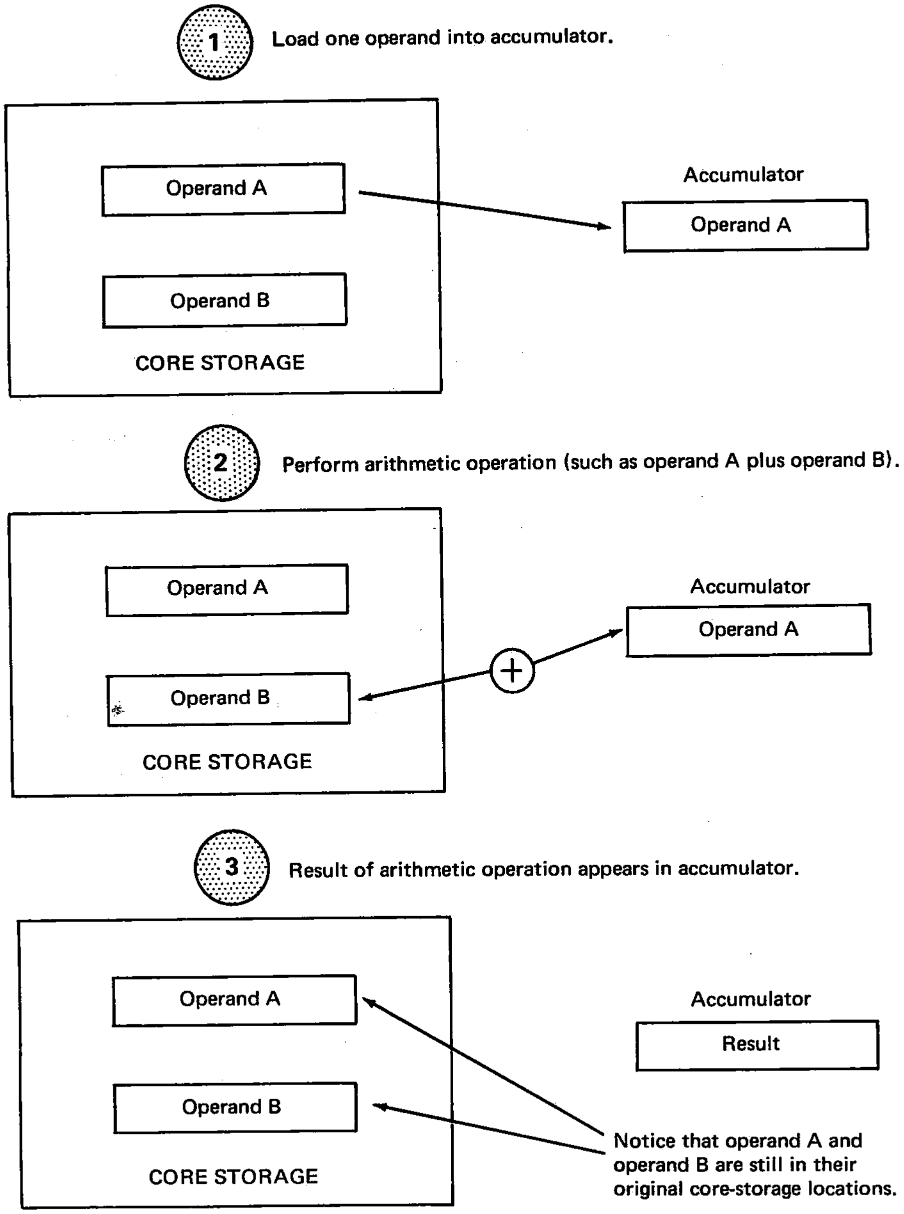


Figure 12. Use of the Accumulator in Arithmetic Operations

Carry and Overflow Indicators

Operations that affect the carry and overflow indicators are summarized in Figure 13.

The overflow indicator specifies when results of arithmetic operations exceed the capacity of the accumulator or accumulator plus accumulator extension.

The carry indicator is most useful in shift and logical operations. It can also be useful in program routines where arithmetic results may exceed the capacity of the accumulator and its extension (in double-precision arithmetic).

Instruction	Carry Indicator	Overflow Indicator
A	Set to zero prior to execution; set to one by a carry out of the high-order bit position of the accumulator.	Set to one if sum is greater than the capacity of the accumulator. If the overflow indicator is on before execution, its condition is not changed regardless of the result.
AD	Same as A.	Same as A.
BSC or BOSC	Not affected when tested.	Reset when tested.
BSI	Not affected in short format; not reset if tested in long format.	Not affected in short format; reset when tested in long format.
D	Not set, not reset.	Set to one when an attempt is made to divide by zero, or set to one when a quotient overflow occurs.
LDS	Set to the value of bit 14 of the instruction.	Set to the value of bit 15 of the instruction.
S	Set to one when a borrow occurs beyond the high-order position of the accumulator. Set to zero prior to execution of the instruction.	Same as A.
SD	Same as S.	Same as A.
SLA	Set to one if the last bit shifted out of the high-order position of the accumulator is a 1; set to zero if the same bit is a 0.	Not set, not reset.
SLC	Same as SLCA.	Not set, not reset.
SLCA	Set to one if the shift is terminated by a 1-bit in the high-order position of the accumulator; set to zero if the shift is terminated by the count (in CCC) going to zero (even if a 1-bit is in the high-order position of the accumulator).	Not set, not reset.
SLT	Same as SLA.	Not set, not reset.
STS	Set to zero.	Set to zero.

Note: Instructions not listed in this figure do not affect the carry and overflow indicators.

Figure 13. Effect of Instruction Execution on Carry and Overflow Indicators

The actual value of an arithmetic result that exceeds the capacity of the accumulator and its extension can be determined through combined testing of the carry and overflow indicators. In the following illustrative examples, only four bit positions are used to demonstrate the principle (the accumulator and its extension in reality hold 32 bits):

Example 1

Operation	Operand	Operand
Add	7 (hex)	7 (hex)
In binary notation:	In hexadecimal notation:	
0111	7	
+0111	+7	
1110	E	

In example 1, no carry occurs out of the high-order position. An overflow occurs because the result is negative (indicated by a value of 1 in the sign bit – the leftmost bit). The desired result of the addition, however, is a positive number. This can be achieved by appending (via programming) a high-order zero to the answer: 01110. In this example, the overflow indicator is on as a result of the operation, but the carry indicator is off.

Example 2

Operation	Operand	Operand
Add	-8	-8
In binary notation:	In hexadecimal notation:	
1000	-8	
+1000	+-8	
C 0000	-10	

In example 2, both a carry-out (C) and an overflow occur. Therefore, both the overflow and carry indicators are on as a result of the operation. Because the carry indicator is on, a high-order 1 should be appended to get the desired result (in two's-complement form): 10000.

Example 3

Operation	Operand	Operand
Subtract	-8	+7
In binary notation:	In hexadecimal notation:	
1000	-8	
-0111	-+7	
B 0001	-F	

In example 3, a borrow (B) beyond the high-order position and an overflow occur. Therefore, a high-order 1 should be appended to the result. (The result is in two's-complement form.)

MISCELLANEOUS MACHINE REGISTERS

The registers described here are not under direct program control but function automatically as required by the operation in progress. Contents of these registers can be displayed in indicators on the system console. Figure 14 shows the position of the registers in the CPU data flow.

Arithmetic-Factor Register (AFR)

This 16-bit register holds one operand during arithmetic and logical operations. (The other operand is in the accumulator.) The arithmetic-factor register is also referred to as the D register.

Cycle-Control Counter (CCC)

The cycle-control counter is a 6-bit register that is used primarily to count CPU cycles and control shift operations.

Operation Register (OP)

This 5-bit register holds the operation code of the instruction being executed.

Storage-Address Register (SAR)

The storage-address register contains the address of each location that is accessed in main storage, except for data transfers for cycle-steal I/O devices. Such devices provide main-storage addresses in circuitry separate from the storage-address register. The storage-address register is 12, 13, 14, or 15 bit positions in size, depending upon the core-storage capacity of the system. The SAR is also referred to as the M register.

Storage-Buffer Register (SBR)

This 16-bit register is the buffer between the CPU and core storage. Every word of data transferred to or from core storage passes through the storage-buffer register. The storage-buffer register is also called the B register.

Operation-Tag Register (TAG)

With a 3-bit capacity, the TAG register contains the F (format) and T (tag) bits of the instruction being executed. The format bit determines the instruction length (short or long) and the tag bits select the index register.

Temporary Accumulator (TAR)

This 16-bit register, which is the image of the accumulator, is used to store the contents of the accumulator during effective-address computation. The temporary accumulator is also referred to as the U register.

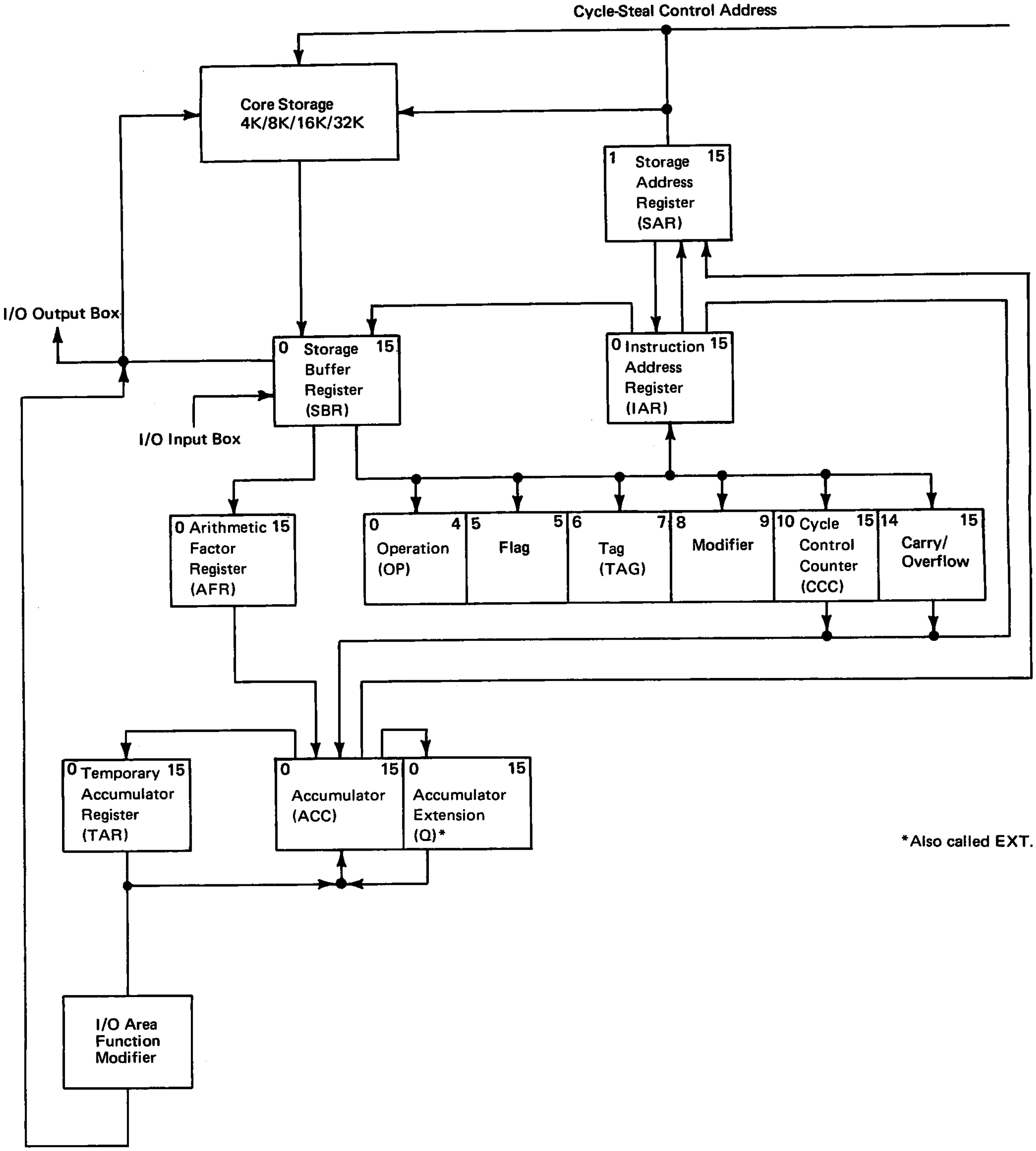


Figure 14. CPU Controls and Data Flow

CPU Instructions

The 1130 CPU instruction set is made up of five general classes of instructions:

1. Load and store
2. Arithmetic
3. Shift
4. Branch
5. Input/output

Names, assembler mnemonics, and execution times of the instructions are listed in Figure 15.

The execution times can be used by the programmer to calculate whether time-dependent functions can be performed -- for example, whether there is sufficient time to execute a specific series of instructions between the time that an input record is stored in core storage until the next record from the same device is stored in core storage. Similarly, the execution times are used to calculate which one of two or more equivalent series of instructions (for example, two or more equivalent loops) can be executed faster, thus contributing to decreased program execution time.

Instruction	Mnemonic	Binary OP Code	1131 Models 1 and 2 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										
No Operation	NOP	00010	3.6	—	—	—	—	—	—	—
Shift Left ACC,00	SLA[7]	00010								
Shift Left ACC and EXT,10	SLT[7]	00010								
Shift Left and Count ACC,01	SLCA[7] [8]	00010								
Shift Left and Count ACC and EXT,11	SLC[7] [8]	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	—	14.4	—	18.4	—

*Valid in short format only

Notes:

1. Indirect addressing, where applicable, adds one storage cycle (3.6 μsec) to execution time.
2. If branch is taken.
3. One storage cycle + 0.45(N-4). When N ≤4, only one storage cycle is used.
4. Two storage cycles + 0.45(N-4). When N≤4, only two storage cycles are used.
5. N > 16: one storage cycle + 0.45(N-19).
   N < 16: one storage cycle + 0.45(N-4).
   When N = 16, only one storage cycle is used.
6. N > 16: two storage cycles + 0.45(N-19).
   N < 16: two storage cycles + 0.45(N-4).
   where N = number of positions shifted,
   When N = 16, only two storage cycles are used.
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 one storage cycle.

Figure 15 (Part 1 of 3). 1130 Instruction Set and Execution Times

Instruction	Mnemonic	Binary OP Code	1131 Models 3 and 5 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	4.6	–	6.8	–	6.6	–	9.0	–
Load Double	LDD	11001	6.8	–	9.1	–	8.8	–	11.0	–
Store ACC	STO	11010	4.6	–	6.8	–	6.6	–	9.0	–
Store Double	STD	11011	6.8	–	9.1	–	8.8	–	11.0	–
Load Index	LDX	01100	2.7	–	4.4	–	4.4	–	7.2	–
Store Index	STX	01101	4.6	–	6.8	–	7.2	–	9.4	–
Load Status*	LDS[7]	00100	2.2	–	2.2	–	–	–	–	–
Store Status	STS	00101	4.6	–	6.8	–	6.6	–	9.0	–
Arithmetic										
Add	A	10000	4.9	7.9	7.1	10.1	6.8	9.9	9.4	12.4
Add Double	AD	10001	7.5	13.4	9.6	15.6	9.4	15.4	11.8	18.0
Subtract	S	10010	4.9	7.9	7.1	10.1	6.8	9.9	9.4	12.4
Subtract Double	SD	10011	7.5	13.4	9.6	15.6	9.4	15.4	20.1	26.1
Multiply	M	10100	15.7	24.4	17.9	26.6	17.9	26.6	18.8	28.8
Divide	D	10101	46.4	92.1	48.6	94.4	48.6	94.4	50.8	91.6
AND	AND	11100	4.6	–	6.8	–	6.6	–	9.0	–
OR	OR	11101	4.6	–	6.8	–	6.6	–	9.0	–
Exclusive OR	EOR	11110	4.6	–	6.8	–	6.6	–	9.0	–
Shift Left*/Modifier Bits 8 & 9										
No Operation	NOP	00010	2.2	–	–	–	–	–	–	–
Shift Left ACC,00	SLA[7]	00010								
Shift Left ACC and EXT,10	SLT[7]	00010								
Shift Left and Count ACC,01	SLCA[7] [8]	00010								
Shift Left and Count ACC and EXT,11	SLC[7] [8]	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	4.6	–	6.8	–	6.6[2]	–	9.0	–
Branch or Skip on Condition	BSC	01001	2.2	–	2.2	–	4.4[2]	–	6.8	–
Modify Index and Skip	MDX	01110	2.7	6.0	6.8	9.9	11.3	14.3	11.3	14.3
Wait*	WAIT[7]	00110[9]	2.2	–	2.2	–	–	–	–	–
Input/Output										
Execute I/O	XIO[10]	00001	6.8	–	9.0	–	8.8	–	11.2	–

*Valid in short format only

Notes:

1. Indirect addressing, where applicable, adds one storage cycle (2.2 μsec) to execution time.
2. If branch is taken.
3. One storage cycle + 0.275(N-4). When N ≤4, only one storage cycle is used.
4. Two storage cycles + 0.275(N-4). When N ≤4, only two storage cycles are used.
5. N > 16: one storage cycle + 0.275(N-19).
   N < 16: one storage cycle + 0.275(N-4).
   When N = 16, only one storage cycle is used.
6. N > 16: two storage cycles + 0.275(N-19).
   N < 16: two storage cycles + 0.275(N-4),
   where N = number of positions shifted.
   When N = 16, only two storage cycles are used.
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 one storage cycle.

Figure 15 (Part 2 of 3). 1130 Instruction Set and Execution Times

Instruction	Mnemonic	Binary Op Code	1131 Model 4 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	12.1	–	18.0	–	17.5	–	23.8	–
Load Double	LDD	11001	18.0	–	23.8	–	23.4	–	29.2	–
Store ACC	STO	11010	12.1	–	18.0	–	17.5	–	23.8	–
Store Double	STD	11011	18.0	–	23.8	–	23.4	–	29.2	–
Load Index	LDX	01100	6.8	–	11.7	–	11.7	–	18.5	–
Store Index	STX	01101	12.1	–	18.0	–	18.5	–	24.3	–
Load Status*	LDS[7]	00100	5.9	–	5.9	–	–	–	–	–
Store Status	STS	00101	12.1	–	18.0	–	17.5	–	23.8	–
Arithmetic										
Add	A	10000	12.6	19.8	18.4	25.6	18.0	25.2	24.3	31.5
Add Double	AD	10001	18.9	35.5	24.8	41.5	24.3	40.9	30.6	47.7
Subtract	S	10010	12.6	19.8	18.4	25.6	18.0	25.2	24.3	31.5
Subtract Double	SD	10011	18.9	35.5	24.8	41.5	24.3	40.9	30.6	47.7
Multiply	M	10100	41.4	64.8	47.5	70.6	47.5	70.6	53.1	76.5
Divide	D	10101	123.3	243.0	129.1	248.8	129.1	248.8	135.0	242.1
AND	AND	11100	12.1	–	18.0	–	17.5	–	23.8	–
OR	OR	11101	12.1	–	18.0	–	17.5	–	23.8	–
Exclusive OR	EOR	11110	12.1	–	18.0	–	17.5	–	23.8	–
Shift Left*, Modifier Bits 8 & 9										
No Operation	NOP	00010	5.9	–	–	–	–	–	–	–
Shift Left ACC,00	SLA[7]	00010								
Shift Left ACC and EXT,10	SLT[7]	00010								
Shift Left and Count ACC,01	SLCA[7] [8]	00010								
Shift Left and Count ACC and EXT,11	SLC[7] [8]	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	12.1	–	18.0	–	17.5[2]	–	23.8	–
Branch or Skip on Condition	BSC	01001	5.9	–	5.9	–	11.7[2]	–	18.0	–
Modify Index and Skip	MDX	01110	6.8	14.4	18.0	25.2	29.6	36.9	29.6	36.9
Wait*	WAIT[7]	00110[9]	5.9	–	5.9	–	–	–	–	–
Input/Output										
Execute I/O	XIO[10]	00001	18.0	–	23.8	–	23.4	–	29.5	–

*Valid in short format only

**Execution times are the same as for Models 1 and 2 when either interruption level 0 or 1 is active.

Notes:

1. Indirect addressing, where applicable, adds one storage cycle (5.9 or 3.6 μsec) to execution time.
2. If branch is taken.
3. One storage cycle + 0.45(N-4). When ≤4, only one storage cycle is used.
4. Two storage cycles + 0.45(N-4). When ≤4, only two storage cycles are used.
5. N > 16: one storage cycle + 0.45(N-19).
   N < 16: one storage cycle + 0.45(N-4).
   When N = 16, only one storage cycle is used.
6. N > 16: two storage cycles + 0.45(N-19).
   N < 16: two storage cycles + 0.45(N-4),
   where N = number of positions shifted.
   When N = 16, only two storage cycles are used.
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 one storage cycle.

Figure 15 (Part 3 of 3). 1130 Instruction Set and Execution Times

Symbols and Organization of Instruction Descriptions

The following instruction descriptions generally include:

1. The name of the instruction
2. Next, the assembler machine-language menomic
3. The bit structure for both the short and long formats, when applicable. Hexadecimal values appear under the bit structures. Frequently, a range of values is shown when the instruction can, in fact, have more than one bit structure for a particular field. (X represents any hexadecimal digit.)
4. A description of the function(s) and exceptional conditions of the instruction
5. A short paragraph, following the instruction description, that specifies how the carry and overflow indicators are affected by the instruction.
6. Examples at the end of each instruction. Shown in these examples are the assembler language coding and the hexadecimal value that is assembled (the X again represents any valid hexadecimal digit) with a brief description of the operation. For example, for the load-accumulator instruction:

Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		LD					DISP	C0XX	Contents of CSL at EA (I + DISP) are loaded into A

In this example, the contents of the core-storage location (CSL) specified by the effective address (EA), which is determined by the contents of the instruction address register (I) plus the displacement (DISP), are loaded into the accumulator (A). The description is condensed through use of the abbreviated notation shown to the right of the hexadecimal value.

The symbols used in the instruction examples and the meanings of such symbols are:

Symbol	*Meaning*
A	Accumulator
Q	Accumulator extension
ADDR or ADDRESS	Contents of the address field of a long-format instruction
CSL	Core-storage location
DISP	Contents of the displacement field in a short-format instruction
EA	Effective address (refer to "Effective-Address Generation")
EA + 1	The next word location after the one specified by the EA
I	Contents of the instruction-address register
V	Value
XR1	Contents of index-register 1
XR2	Contents of index-register 2
XR3	Contents of index-register 3
X	Any valid hexadecimal digit

Pictorial representations appear in the "Description" portion of text for most of the instructions. The purpose of these illustrations is merely to clarify the main points of the operations. They are not meant to present all the variations or exceptions that are discussed in the descriptive narrative. In the pictorial representations, the outlined numerals (●) simply point out the order of the steps within the illustration (not necessarily the order of steps within an instruction execution) so that you may follow the presentation more easily; these outlined numerals have no other purpose.

This manual does not present a detailed explanation of assembler-language coding. Rules of assembler-language coding are in *IBM 1130 Assembler Language,* Order No. GC26-5927, which should be used with this functional-characteristics manual whenever reference to assembler-language *rules is* required.

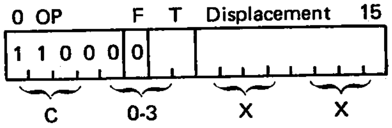
LOAD AND STORE INSTRUCTIONS

Load Accumulator

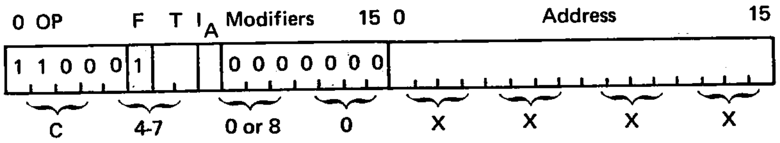
Mnemonic

LD

Short Format

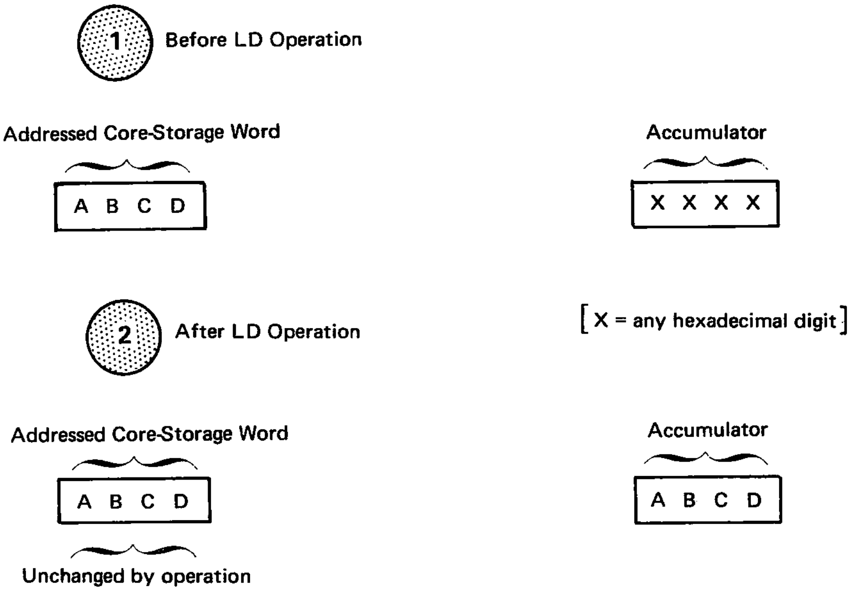


Long Format



Description

The contents of the addressed core-storage location replace the contents of the accumulator. Contents of the addressed core-storage location are unchanged by the operation.



There are no addressing exceptions for the load accumulator instruction; all forms of addressing that are described under "Effective Address Generation" apply to the LD instruction.

Indicators: The carry and the overflow indicators are not affected during an LD operation.

Examples

Load Accumulator

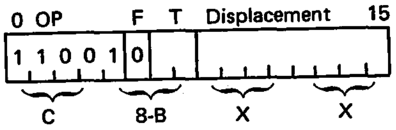
Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		LD					DISP	C0XX	Contents of CSL at EA (I + DISP) are loaded into A
		LD			1		DISP	C1XX	Contents of CSL at EA (XR1 + DISP) are loaded into A
		LD			2		DISP	C2XX	Contents of CSL at EA (XR2 + DISP) are loaded into A
		LD			3		DISP	C3XX	Contents of CSL at EA (XR3 + DISP) are loaded into A
		LD		L			ADDR	C400XXXX	Contents of CSL at EA (Addr) are loaded into A
		LD		L	1		ADDR	C500XXXX	Contents of CSL at EA (Addr +XR1) are loaded into A
		LD		L	2		ADDR	C600XXXX	Contents of CSL at EA (Addr + XR2) are loaded into A
		LD		L	3		ADDR	C700XXXX	Contents of CSL at EA (Addr + XR3) are loaded into A
		LD		I			ADDR	C480XXXX	Contents of CSL at EA (V in CSL at Addr) are loaded into A
		LD		I	1		ADDR	C580XXXX	Contents of CSL at EA (V in CSL at "Addr + XR1") are
									loaded into A
		LD		I	2		ADDR	C680XXXX	Contents of CSL at EA (V in CSL at "Addr + XR2") are
									loaded into A
		LD		I	3		ADDR	C780XXXX	Contents of CSL at EA (V in CSL at "Addr + XR3") are
									loaded into A

Load Double

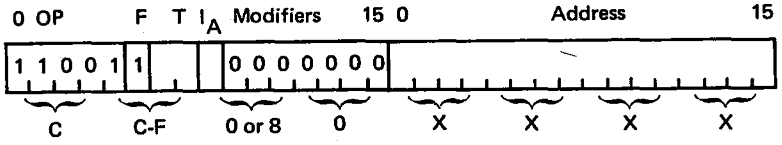
Mnemonic

LDD

Short Format



Long Format



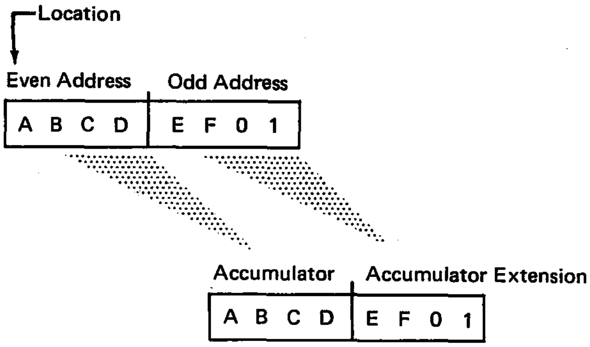
Description

The accumulator and accumulator extension are loaded with two consecutive words from core storage. The two consecutive words in core storage are located by the effective address as follows:

1. The first word is at the location specified by the effective address generated during instruction execution. The effective address should specify an even-word location.
2. The second word is at the word location immediately following the location specified by the effective address.

LDD Operation

Addressed Core-Storage



Contents of the two core storage words remain undisturbed as a result of the operation. The accumulator and extension can contain any value before the LDD instruction is executed; however, the value of the two core storage words appears in the accumulator and its extension at the end of the operation.

Note: If an odd-word location is addressed first, then the contents of that location are loaded into both the accumulator and its extension. For example, if an odd-word location contains DFEA (hexadecimal), and that word is addressed first during execution of an LDD instruction, the DFEA appears in both the accumulator and its extension. In normal operation, then, the effective address generated as a result of execution of the LDD instruction should point to an even word location in core storage.

There are no addressing exceptions for the LDD instruction; all forms of addressing that are described under "Effective Address Generation" apply to the LDD instruction.

Indicators: The carry and overflow indicators are not affected during an LDD operation.

Examples

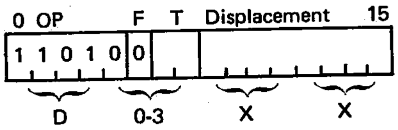
Load Double

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	LDD			DISP	C8XX	Contents of CSL at EA (I + DISP) and EA + 1 are loaded into A and Q
	LDD		1	DISP	C9XX	Contents of CSL at EA (XR1 + DISP) and EA + 1 are loaded into A and Q
	LDD		2	DISP	CAXX	Contents of CSL at EA (XR2 + DISP) and EA + 1 are loaded into A and Q
	LDD		3	DISP	CBXX	Contents of CSL at EA (XR3 + DISP) and EA+1 are loaded into A and Q
	LDD	L		ADDR	CC00XXXX	Contents of CSL at EA (Addr) and EA+1 are loaded into A and Q
	LDD	L	1	ADDR	CD00XXXX	Contents of CSL at EA (Addr +XR1) and EA+1 are loaded into A and Q
	LDD	L	2	ADDR	CE00XXXX	Contents of CSL at EA (Addr +XR2) and EA+1 are loaded into A and Q
	LDD	L	3	ADDR	CF00XXXX	Contents of CSL at EA (Addr +XR3) and EA+1 are loaded into A and Q
	LDD	I		ADDR	CC80XXXX	Contents of CSL at EA (V in CSL at Addr) and EA+1 are loaded into A and Q
	LDD	I	1	ADDR	CD80XXXX	Contents of CSL at EA (V in CSL at "Addr +XR1") and EA+1 are loaded into A and Q
	LDD	I	2	ADDR	CE80XXXX	Contents of CSL at EA (V in CSL at " Addr +XR2") and EA+1 are loaded into A and Q
	LDD	I	3	ADDR	CF80XXXX	Contents of CSL at EA (V in CSL at "Addr +XR3") and EA+1 are loaded into A and Q

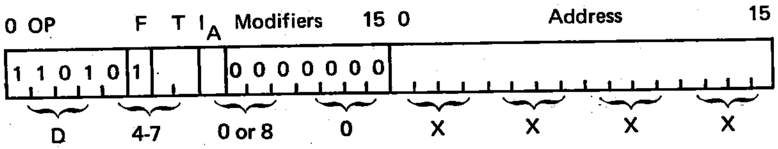
Store Accumulator

Mnemonic
STO

Short Format

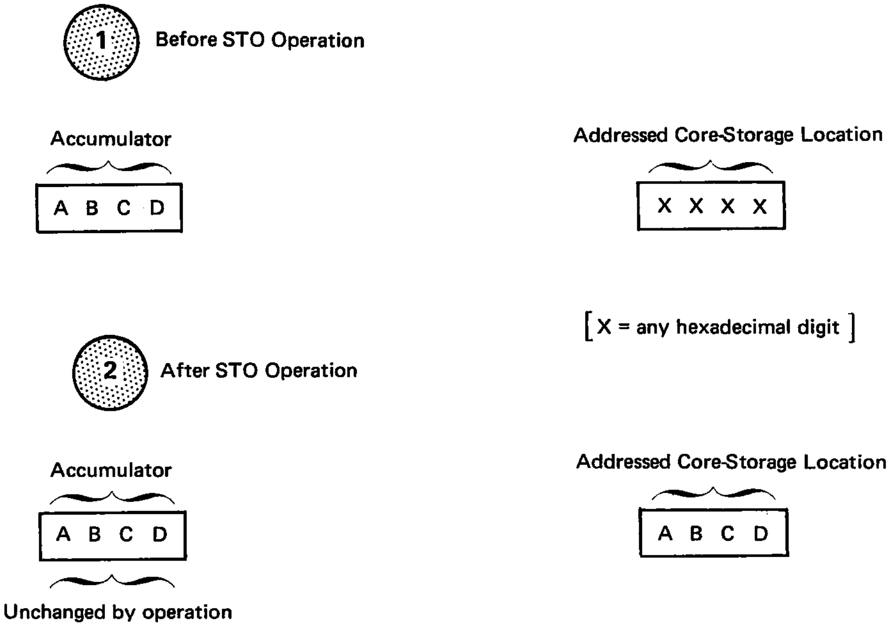


Long Format



Description

The contents of the accumulator replace the contents of the addressed core-storage location. Contents of the accumulator are unchanged by the operation.



There are no addressing exceptions for the load accumulator instruction; all forms of addressing that are described under "Effective Address Generation" apply to the STO instruction.

Indicators: The carry and the overflow indicators are not affected during the STO operation.

Examples

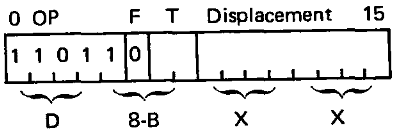
Store Accumulator

Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		STO					DISP	D0XX	Contents of A are stored in CSL at EA (I + DISP)
		STO			1		DISP	D1XX	Contents of A are stored in CSL at EA (XR1+DISP)
		STO			2		DISP	D2XX	Contents of A are stored in CSL at EA (XR2+DISP)
		STO			3		DISP	D3XX	Contents of A are stored in CSL at EA (XR3+DISP)
		STO		L			ADDR	D400XXXX	Contents of A are stored in CSL at EA (Addr)
		STO		L	1		ADDR	D500XXXX	Contents of A are stored in CSL at EA (Addr +XR1)
		STO		L	2		ADDR	D600XXXX	Contents of A are stored in CSL at EA (Addr +XR2)
		STO		L	3		ADDR	D700XXXX	Contents of A are stored in CSL at EA (Addr +XR3)
		STO		I			ADDR	D480XXXX	Contents of A are stored in CSL at EA (V in CSL at Addr)
		STO		I	1		ADDR	D580XXXX	Contents of A are stored in CSL at EA (V in CSL at
									"Addr +XR1")
		STO		I	2		ADDR	D680XXXX	Contents of A are stored in CSL at EA (V in CSL at "Addr
									+XR2")
		STO		I	3		ADDR	D780XXXX	Contents of A are stored in CSL at EA (V in CSL at "Addr
									+XR3")

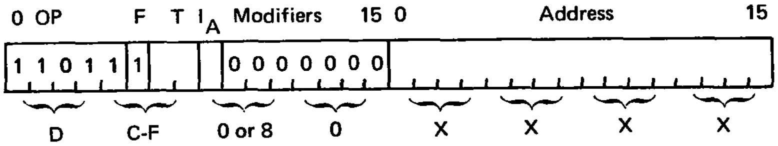
Store Double

Mnemonic
STD

Short Format



Long Format

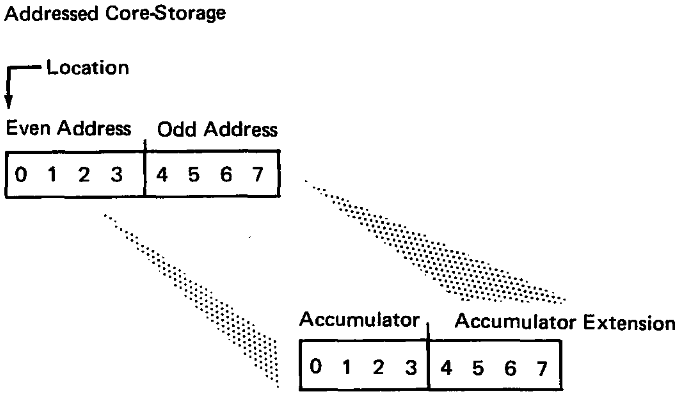


Description

The contents of the accumulator and its extension are loaded into two consecutive words in core storage. These two consecutive words in core storage are located by the effective address as follows:

1. The first word is at the location specified by the effective address generated during instruction execution. The effective address should specify an even-word location.
2. The second word is at the word location immediately following the location specified by the effective address.

STD Operation



Contents of the accumulator and its extension remain undisturbed as a result of the operation.

Note: If an odd-word location is addressed first, the contents of the accumulator are stored into that location, and the contents of the accumulator extension are not stored. In normal operation, then, the effective address generated as a result of execution of the STD instruction should point to an even-word location in core storage. If only the accumulator contents are to be stored into a core-storage word, then the STO instruction should be used.

There are no addressing exceptions for the STD instruction; all forms of addressing described under "Effective Address Generation" apply to the STD instruction.

Indicators: The carry and overflow indicators are not affected during execution of an STD instruction.

Examples

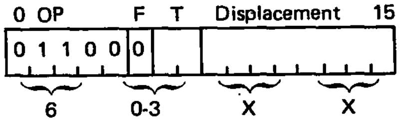
Store Double

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	STD			DISP	D8XX	Contents of A and Q are stored in CSL at EA (I+DISP) and EA+1
	STD		1	DISP	D9XX	Contents of A and Q are stored in CSL at EA (XR1 +DISP) and EA+1
	STD		2	DISP	DAXX	Contents of A and Q are stored in CSL at EA (XR2 +DISP) and EA+1
	STD		3	DISP	DBXX	Contents of A and Q are stored in CSL at EA (XR3 +DISP) and EA+1
	STD	L		ADDR	DC00XXXX	Contents of A and Q are stored in CSL at EA (Addr) and EA+1
	STD	L	1	ADDR	DD00XXXX	Contents of A and Q are stored in CSL at EA (Addr +XR1) and EA+1
	STD	L	2	ADDR	DE00XXXX	Contents of A and Q are stored in CSL at EA (Addr +XR2) and EA+1
	STD	L	3	ADDR	DF00XXXX	Contents of A and Q are stored in CSL at EA (Addr +XR3) and EA+1
	STD	I		ADDR	DC80XXXX	Contents of A and Q are stored in CSL at EA (V in CSL at Addr) and EA+1
	STD	I	1	ADDR	DD80XXXX	Contents of A and Q are stored in CSL at EA (V in CSL at "Addr +XR1") and EA+1
	STD	I	2	ADDR	DE80XXXX	Contents of A and Q are stored in CSL at EA (V in CSL at "Addr +XR2") and EA+1
	STD	I	3	ADDR	DF80XXXX	Contents of A and Q are stored in CSL at EA (V in CSL at "Addr +XR3") and EA+1

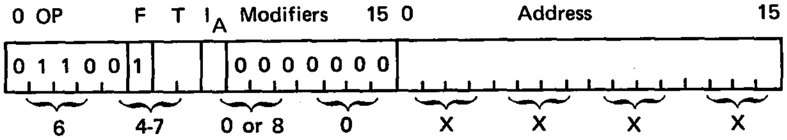
Load Index

Mnemonic
LDX

Short Format



Long Format



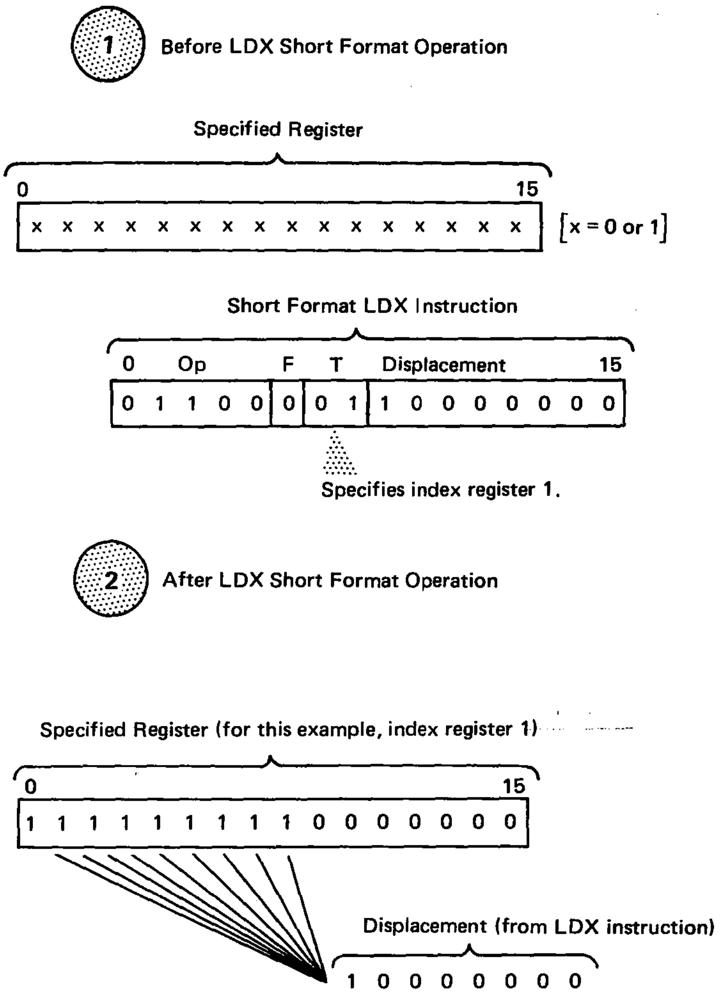
Description

The purpose of this instruction is to load the instruction-address register or an index register with a value. How this is done is dependent upon the format of the instruction. Whether the instruction-address register or an index register is specified is dependent upon the T bits (in either the short or the long format):

T Bits (bits 6 and 7 of the instruction)	Register
00	Instruction address
01	Index-register 1
10	Index-register 2
11	Index-register 3

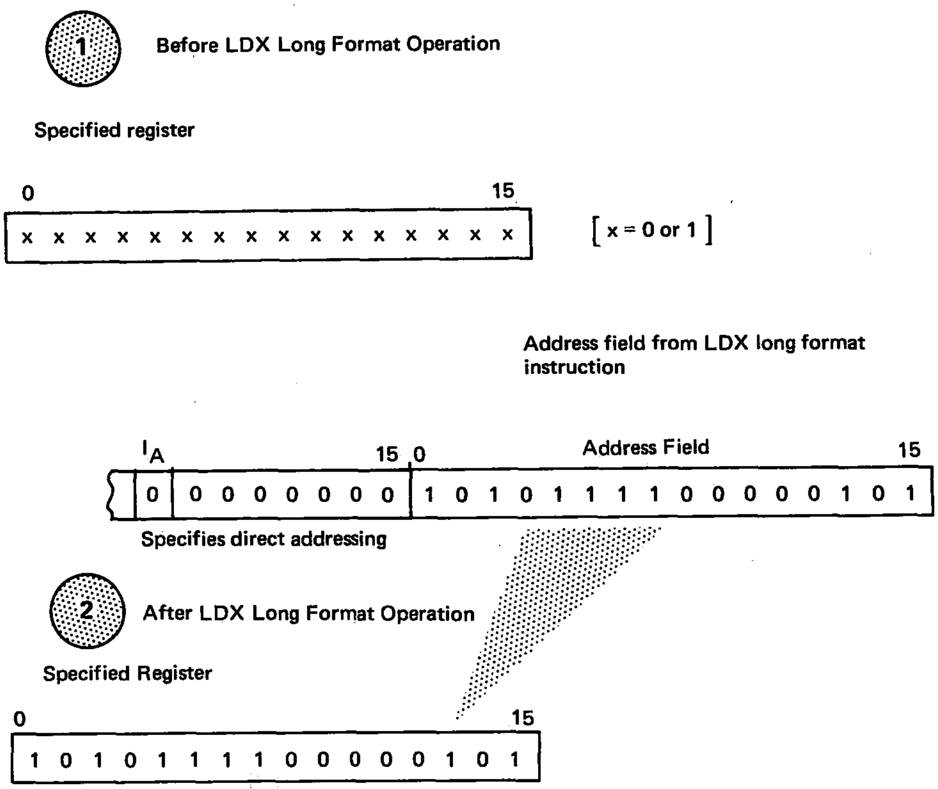
When the value is loaded into the instruction-address register, an unconditional branch occurs to the address loaded.

For the short-instruction format, the specified register is loaded with an *expanded* displacement. The displacement is from the displacement field in the instruction; the value of bit 8 in the instruction is propagated to the left to form the leftmost 8 bits of the word.

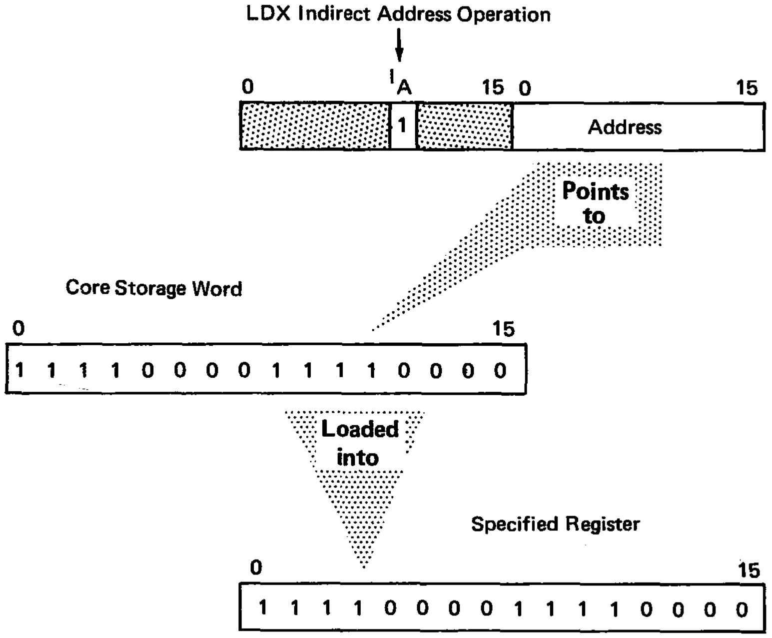


Leftmost bit is propagated to the left.

For the long-instruction format, when indirect addressing is not specified, the specified register is loaded with the value in the address field in the instruction.



When indirect addressing is specified, the contents of the word addressed by the address field are loaded into the specified register.



Indicators: The carry and overflow indicators are not affected during execution of an LDX instruction.

Examples

Load Index

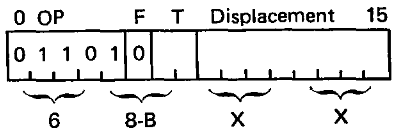
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	LDX			DISP	60XX	Load expanded DISP into the Instruction Register
	LDX		1	DISP	61XX	Load expanded DISP into Index Register 1
	LDX		2	DISP	62XX	Load expanded DISP into Index Register 2
	LDX		3	DISP	63XX	Load expanded DISP into Index Register 3
	LDX	L		ADDR	6400XXXX	Load Addr into the Instruction Register
	LDX	L	1	ADDR	6500XXXX	Load Addr into Index Register 1
	LDX	L	2	ADDR	6600XXXX	Load Addr into Index Register 2
	LDX	L	3	ADDR	6700XXXX	Load Addr into Index Register 3
	LDX	I		ADDR	6480XXXX	Load contents of CSL at Addr into the Instruction Register
	LDX	I	1	ADDR	6580XXXX	Load contents of CSL at Addr into Index Register 1
	LDX	I	2	ADDR	6680XXXX	Load contents of CSL at Addr into Index Register 2
	LDX	I	3	ADDR	6780XXXX	Load contents of CSL at Addr into Index Register 3

Store Index

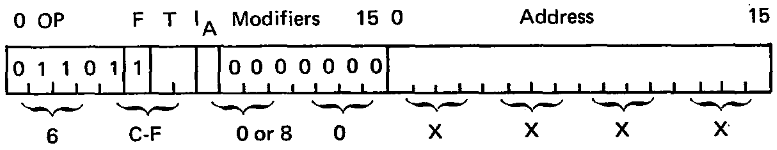
Mnemonic

STX

Short Format



Long Format



Description

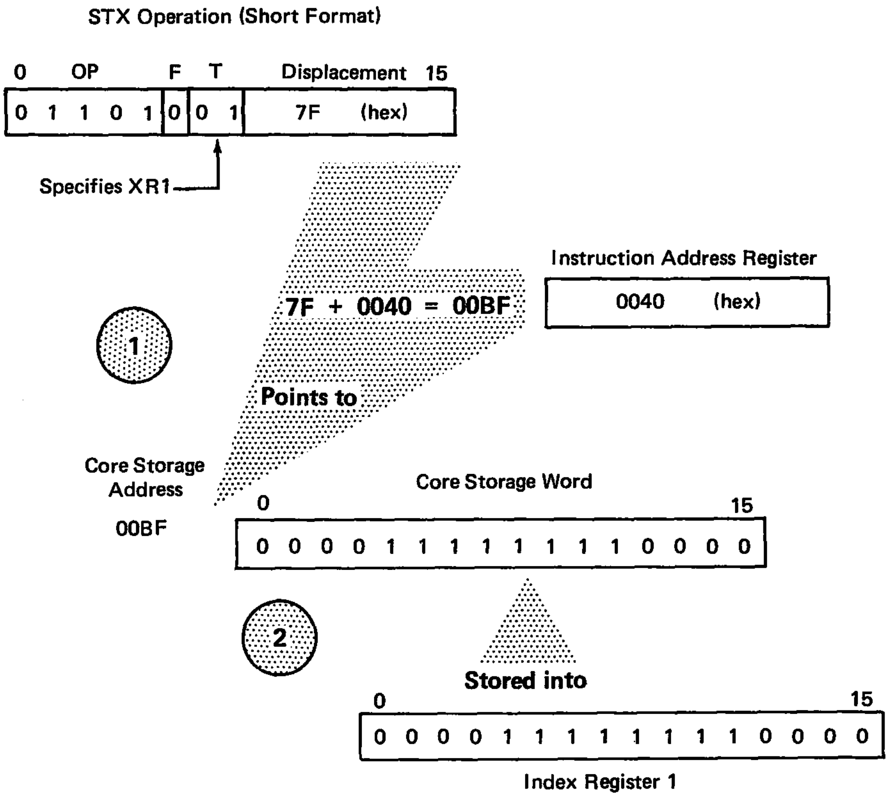
Contents of the specified register are stored in the addressed core-storage location. The T bits, in the short or the long format, specify the register.

T bits	Register
00	Instruction address
01	Index-register 1
10	Index-register 2
11	Index-register 3

The contents of the register remain unchanged as a result of the operation.

For the short-instruction format, the addressed core-storage location is always specified by adding the displacement to the contents of the instruction-address register. The contents of XR1, XR2, and XR3 are never used to form the effective address in an STX operation.

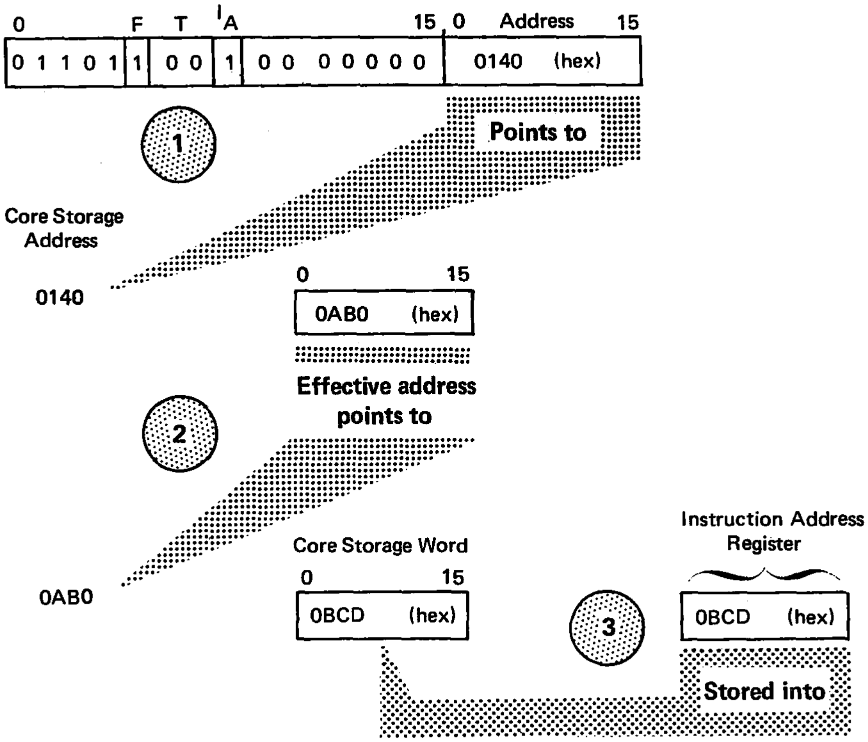
Pictorially, a short-format STX operation can be shown as follows:



For the long-instruction format, addressing is either direct or indirect in the normal manner. However, as in the short format of this same instruction, the index registers are not used to form the effective address. Also, as is the case with other long-format instructions, the instruction-address register is not used in the long format to form the effective address. The value in the instruction-address register can, nevertheless, be stored in the specified storage location.

A long-format STX operation with indirect addressing can be represented pictorially as follows:

STX Operation (Long Format, Indirect Addressing)



Indicators: The carry and overflow indicators are not affected during execution of an STX instruction.

Examples

Store Index

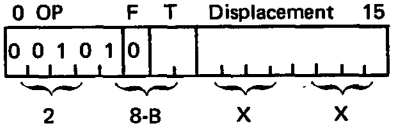
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	STX			DISP	68XX	Store I in CSL at EA (I+DISP)
	STX		1	DISP	69XX	Store XR1 in CSL at EA (I+DISP)
	STX		2	DISP	6AXX	Store XR2 in CSL at EA (I+DISP)
	STX		3	DISP	6BXX	Store XR3 in CSL at EA (I+DISP)
	STX	L		ADDR	6C00XXXX	Store I in CSL at EA (Addr)
	STX	L	1	ADDR	6D00XXXX	Store XR1 in CSL at EA (Addr)
	STX	L	2	ADDR	6E00XXXX	Store XR2 in CSL at EA (Addr)
	STX	L	3	ADDR	6F00XXXX	Store XR3 in CSL at EA (Addr)
	STX	I		ADDR	6C80XXXX	Store I in CSL at EA (V in CSL at Addr)
	STX	I	1	ADDR	6D80XXXX	Store XR1 in CSL at EA (V in CSL at Addr)
	STX	I	2	ADDR	6E80XXXX	Store XR2 in CSL at EA (V in CSL at Addr)
	STX	I	3	ADDR	6F80XXXX	Store XR3 in CSL at EA (V in CSL at Addr)

Store Status

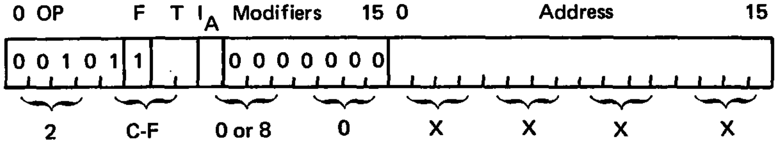
Mnemonic

STS

Short Format



Long Format



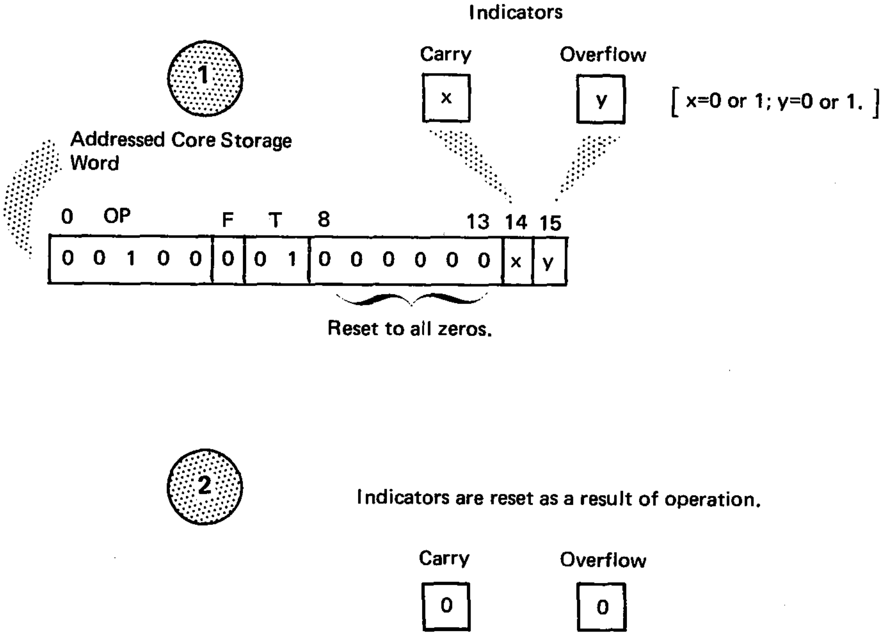
Description

The values of the carry and overflow indicators are stored in bits 14 and 15, respectively, of the addressed core-storage word. Remaining bits of the addressed core-storage word are affected as follows:

Bits	Condition
0 through 7	Unchanged
8 through 13	Reset to zeros

The addressed core-storage word is usually a load-status instruction. Storing the values of the carry and overflow indicators in this word provides for setting these indicators to the stored values when the load-status instruction is subsequently executed. This procedure is used when, for example, a program routine is temporarily interrupted so that some other routine can be executed. Then, before a return to the interrupted routine, the load-status instruction can be executed to restore the carry and overflow indicators to their previous values.

The STS operation can be shown pictorially as follows:



There are no addressing exceptions for the store-status instruction; all forms of addressing that are described under "Effective Address Generation" apply to the STS instruction.

Indicators: Both the carry and the overflow indicators are reset to zero as a result of execution of the STS instruction.

Examples

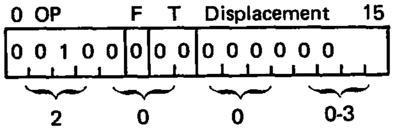
Store Status

Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		STS					DISP	28XX	Store status of indicators in CSL at EA (I+DISP)
		STS			1		DISP	29XX	Store status of indicators in CSL at EA (XR1+DISP)
		STS			2		DISP	2AXX	Store status of indicators in CSL at EA (XR2+DISP)
		STS			3		DISP	2BXX	Store status of indicators in CSL at EA (XR3+DISP)
		STS		L			ADDR	2C00XXXX	Store status of indicators in CSL at EA (Addr)
		STS		L	1		ADDR	2D00XXXX	Store status of indicators in CSL at EA (Addr+XR1)
		STS		L	2		ADDR	2E00XXXX	Store status of indicators in CSL at EA (Addr+XR2)
		STS		L	3		ADDR	2F00XXXX	Store status of indicators in CSL at EA (Addr+XR3)
		STS		I			ADDR	2C80XXXX	Store status of indicators in CSL at EA (V in CSL at Addr)
		STS		I	1		ADDR	2D80XXXX	Store status of indicators in CSL at EA (V in CSL at "Addr +XR1")
		STS		I	2		ADDR	2E80XXXX	Store status of indicators in CSL at EA (V in CSL at "Addr +XR2")
		STS		I	3		ADDR	2F80XXXX	Store status of indicators in CSL at EA (V in CSL at "Addr +XR3")

Load Status

Mnemonic
LDS

Short Format



Description

This instruction sets or resets the carry and overflow indicators. Bits 14 and 15 of the instruction set or reset the indicators as follows:

Bit 14	Bit 15	Set or Reset Carry Indicator	Set or Reset Overflow Indicator
0	0	reset (=0)	reset (=0)
0	1	reset (=0)	set (=1)
1	0	set (=1)	reset (=0)
1	1	set (=1)	set (=1)

This instruction is not valid in the long format. If an attempt is made to execute a load-status instruction in which the F bit = 1, the instruction is still treated as a short-format instruction by the system.

A load-status instruction is usually the core-storage word addressed by a store-status instruction. The load-status instruction is subsequently executed before a return to the routine originally interrupted – the routine whose status was stored by the store-status instruction.

Core storage is not addressed as a result of execution of a load-status instruction.

Indicators: The carry and overflow indicators are affected as described under the description of this instruction.

Examples

Load Status

Assembler Language Coding Label 21–25	Operation 27–30	F 32	T 33	35–40	Hexadecimal Value	Description of Instruction
	LDS			0	2000	Set CARRY and OVERFLOW indicators OFF
	LDS			1	2001	Set OVERFLOW ON and CARRY OFF
	LDS			2	2002	Set OVERFLOW OFF and CARRY ON
	LDS			3	2003	Set CARRY and OVERFLOW indicator ON

Intentionally Blank

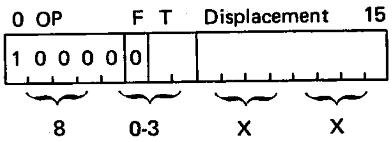
ARITHMETIC INSTRUCTIONS

Add

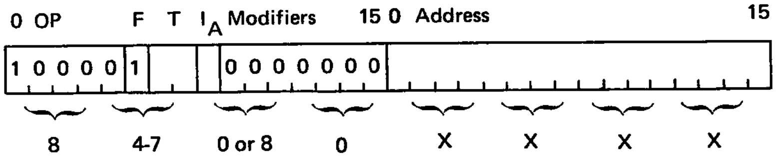
Mnemonic

A

Short Format



Long Format



Description (Add)

The basic purpose of this instruction is to add two 16-bit operands. One of the operands must first be loaded into the accumulator, such as by means of execution of a load-accumulator instruction. The add instruction then provides the address of the other operand, which must be in main storage. Addition takes place, and the result is placed in the accumulator:

(Sign bit 0 = 0 specifies + number.)

0 000 0000 1001 1101 = Contents of accumulator
+0 000 0010 0011 0101 = Contents of storage location addressed by add instruction
0 000 0010 1101 0010 = Result loaded into accumulator

Although the result replaces the contents of the accumulator, the contents of the addressed storage location remain unchanged.

The result of the addition is either positive or negative, depending upon the magnitude of the values used and whether the signs of the two operands are the same:

+ plus a + = +
- plus a - = -
+ plus a - = sign of the larger operand
- plus a + = sign of the larger operand

The value in the accumulator is positive if the leftmost bit is at a value of 0; the value in the accumulator is negative if the leftmost bit is at a value of 1. Negative numbers are in two's-complement form.

There are no addressing exceptions for the add instruction; all forms of addressing that are described under "Effective Address Generation" apply to the A instruction.

Indicators: The carry indicator is automatically reset to 0 at the beginning of an add-instruction execution. If, during the add-instruction execution, a carry-out of the high-order (leftmost) position of the accumulator occurs, then the carry indicator is set to 1; if no such carry-out of the high-order position occurs, the carry indicator remains at its reset condition of 0. It can subsequently be set or reset by the various actions listed under "Carry and Overflow Indicators" (see Figure 13).

The overflow indicator must be reset to 0 if it is to be used during execution of an add instruction. If the overflow indicator is at a value of 1 at the start of an add operation, it is not changed regardless of the result of the add operation. If the overflow indicator is at a value of zero at the start of an add operation, it is set to a value of 1 if the addition produces a result that exceeds the capacity of the accumulator. For example, when the following two 16-bit operands are added together,

S

0 100 0000 0000 0000		Operand in accumulator — a positive number
+0 100 0000 0000 0000		Operand in main storage — a positive number
1 000 0000 0000 0000	=	Result in accumulator — a negative number

(S = Sign bit)

the result is greater than the capacity of the accumulator because the accumulator specifies a negative result (the leftmost bit is at a value of 1). In this case, the overflow indicator is set to 1. The carry indicator, however, is not set to one because a carry-out of the high-order position of the accumulator does not occur. Refer to "Carry and Overflow Indicators" for a discussion of how these two indicators can be used together in certain arithmetic operations.

The maximum capacity of the accumulator is:

Power-of-2 Notation	Decimal Notation	Hexadecimal Notation
$+2^{15}-1$	+ 32,767	+ 7FFF
-2^{15}	- 32,768	- 8000

Examples

Add

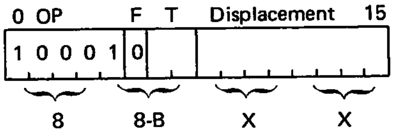
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	A			DISP	80XX	Add contents of CSL at EA (I+DISP) to A
	A		1	DISP	81XX	Add contents of CSL at EA (XR1+DISP) to A
	A		2	DISP	82XX	Add contents of CSL at EA (XR2+DISP) to A
	A		3	DISP	83XX	Add contents of CSL at EA (XR3+DISP) to A
	A	L		ADDR	8400XXXX	Add contents of CSL at EA (Addr) to A
	A	L	1	ADDR	8500XXXX	Add contents of CSL at EA (Addr+XR1) to A
	A	L	2	ADDR	8600XXXX	Add contents of CSL at EA (Addr+XR2) to A
	A	L	3	ADDR	8700XXXX	Add contents of CSL at EA (Addr +XR3) to A
	A	I		ADDR	8480XXXX	Add contents of CSL at EA (V in CSL at Addr) to A
	A	I	1	ADDR	8580XXXX	Add contents of CSL at EA (V in CSL at "Addr+XR1") to A
	A	I	2	ADDR	8680XXXX	Add contents of CSL at EA (V in CSL at "Addr+XR2") to A
	A	I	3	ADDR	8780XXXX	Add contents of CSL at EA (V in CSL at "Addr+XR3") to A

Add Double

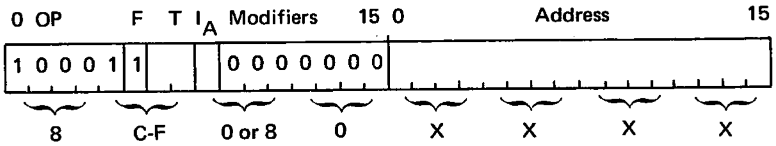
Mnemonic

AD

Short Format

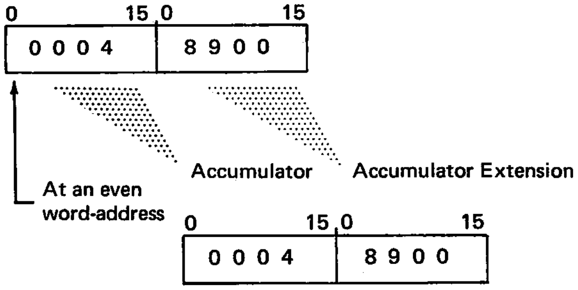


Long Format



Description

The purpose of this instruction is to add two 32-bit operands. One of the operands must be loaded into the accumulator and accumulator extension before the add-double operation is performed. A load-double instruction can be used to do this.



The add-double instruction addresses another operand in core storage; this operand must also be at an even-word address, just as the leftmost word addressed by the load-double instruction.

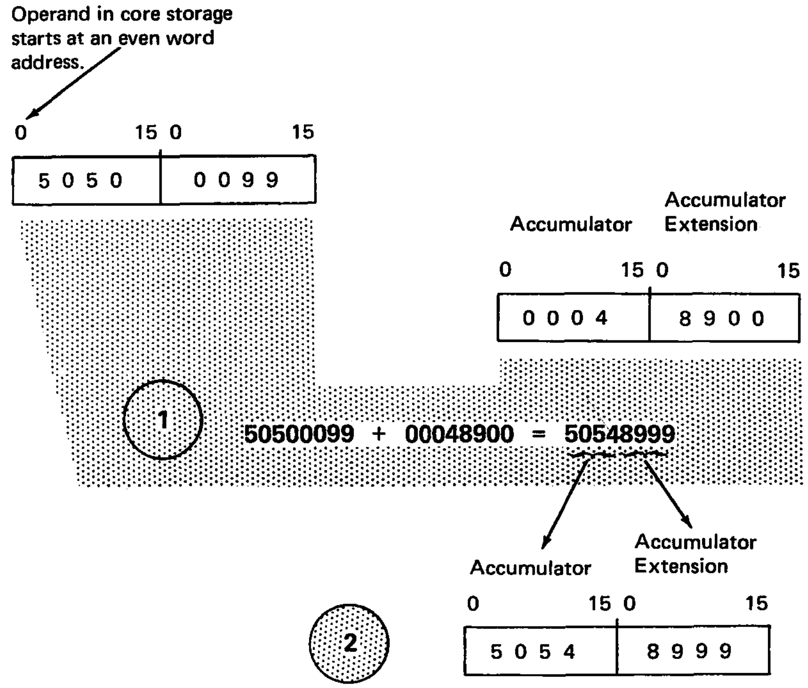
Except for the size of the operands, the add-double operation proceeds in much the same manner as the add operation:

1. The result of the add-double operation is placed in the accumulator and accumulator extension.
2. Result of the addition is either positive or negative:

+ plus a + = +
- plus a - = -
+ plus a - = sign of the larger operand
- plus a + = sign of the larger operand

3. The value in the accumulator and accumulator extension has a sign as signified by bit 0 in the accumulator. The leftmost bit of each operand determines that operand's sign.
4. The operand in core storage is unchanged by the operation. Pictorially, the operation proceeds as follows:

Add Double Operation



If the add-double instruction addresses an operand starting at an odd address, the contents of that single word are added to both the accumulator and to the accumulator extension.

There are no other addressing exceptions for the add-double instruction; all forms of addressing that are described under "Effective Address Generation" apply to the AD instruction.

Indicators: The carry indicator is automatically reset to 0 at the beginning of execution of an add-double instruction. If, during the add-double execution, a carry occurs out of the leftmost position of the accumulator, then the carry indicator is set to 1; if no such carry-out occurs, the carry indicator remains reset. It can subsequently be set or reset by the various actions listed under "Carry and Overflow Indicators" (see Figure 13).

The overflow indicator must be reset to 0 if it is to be used during execution of an add-double instruction. If the overflow indicator is at a value of 1 at the start of an add-double operation, it is not changed regardless of the result of the operation.

If the overflow indicator is at a value of zero at the start of an add-double operation, it is set to a value of 1 if the addition produces a result that exceeds the capacity of the accumulator plus the accumulator extension. For example, assume that the following two negative numbers are added together:

S

1 000 0000 0000 0000 0000 0000 0000 0000
+1 000 0000 0000 0000 0000 0000 0000 0000
C 0 000 0000 0000 0000 0000 0000 0000 0000

(S = sign bit; C = carry out of high-order position.)

This result is greater than the capacity of the accumulator plus accumulator extension because adding the two largest negative numbers should not yield a positive zero, which is the result in the accumulator plus its extension. In this case, the overflow indicator is set to 1. The carry indicator would also be set to 1 because of the carry-out of the leftmost position of the accumulator. Refer to "Carry and Overflow Indicators" for a discussion of how these two indicators can be used together in certain arithmetic operations.

Maximum capacity of the accumulator and accumulator extension is:

Power of 2 Notation	Decimal Notation	Hexadecimal Notation
$+2^{31}-1$	2,147,483,647	7FFFFFFF
-2^{31}	2,147,483,648	80000000

Examples

Add Double

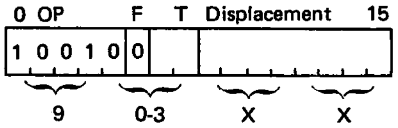
Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		AD					DISP	88XX	Add contents of CSL at EA (I+DISP) and EA+1 to A and Q
		AD			1		DISP	89XX	Add contents of CSL at EA (XR1+DISP) and EA+1 to A
									and Q
		AD			2		DISP	8AXX	Add contents of CSL at EA (XR2+DISP) and EA+1 to A
									and Q
		AD			3		DISP	8BXX	Add contents of CSL at EA (XR3+DISP) and EA+1 to A
									and Q
		AD		L			ADDR	8C00XXXX	Add contents of CSL at EA (Addr) and EA+1 to A and Q
		AD		L	1		ADDR	8D00XXXX	Add contents of CSL at EA (Addr+XR1) and EA+1 to A
									and Q
		AD		L	2		ADDR	8E00XXXX	Add contents of CSL at EA (Addr+XR2) and EA+1 to A
									and Q
		AD		L	3		ADDR	8F00XXXX	Add contents of CSL at EA (Addr+XR3) and EA+1 to A
									and Q
		AD		I			ADDR	8C80XXXX	Add contents of CSL at EA (V in CSL at Addr) and EA+1
									to A and Q
		AD		I	1		ADDR	8D80XXXX	Add contents of CSL at EA (V in CSL at "Addr+XR1") and
									EA+1 to A and Q
		AD		I	2		ADDR	8E80XXXX	Add contents of CSL at EA (V in CSL at "Addr+XR2")
									and EA+1 to A and Q
		AD		I	3		ADDR	8F80XXXX	Add contents of CSL at EA (V in CSL at "Addr+XR3")
									and EA+1 to A and Q

Subtract

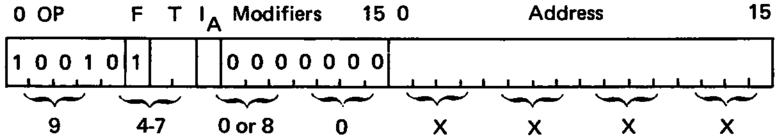
Mnemonic

S

Short Format



Long Format



Description

The subtract instruction is used to subtract one 16-bit operand from the 16-bit operand in the accumulator. The accumulator must first be loaded with the 16-bit value; then the subtract instruction addresses the other operand during execution of the subtract instruction. The operand in core storage and the operand in the accumulator can have the same or different signs; negative numbers are in two's complement form.

Result of the subtraction is in the accumulator at the end of the operation. The operand addressed in core storage by the subtract instruction is unchanged by the operation. Several examples are:

Example 1

0 000 0000 0000 0011 = Operand in accumulator
-0 000 0000 0000 0010 = Operand addressed by subtract instruction
0 000 0000 0000 0001 = Result placed in accumulator

(In decimal: 3 - 2 = 1)

Example 2

1 000 0000 0000 0011 = Operand in accumulator
-0 000 0000 0000 0010 = Operand addressed by subtract instruction
1 000 0000 0000 0001 = Result placed in accumulator

(In decimal: -32,765 - 2 = -32,767)

Example 3

1 000 0000 0000 0011 = Operand in accumulator
-1 000 0000 0000 0000 = Operand addressed by subtract instruction
0 000 0000 0000 0011 = Result placed in accumulator

(In decimal: -32,765 - (-)32,768 = -32,765 + 32,768 = +3)

The sign of the result is dependent upon the signs and magnitudes of both operands. Possible combinations (where operand B is always numerically greater than operand A, regardless of signs) are:

Operand in Accumulator		Operand in Core Storage		Sign of Result in Accumulator
+B	-	(+)A	=	+
+B	-	(-)A	=	+
-B	-	(+)A	=	-
-B	-	(-)A	=	-
+A	-	(+)B	=	-
+A	-	(-)B	=	+
-A	-	(+)B	=	-
-A	-	(-)B	=	+

There are no addressing exceptions for the subtract instruction; all forms of addressing that are described under "Effective Address Generation" apply to the S instruction.

Indicators: The carry indicator is automatically reset to 0 at the beginning of execution of a subtract instruction. If, during execution of the subtract instruction, a borrow occurs beyond the leftmost position of the accumulator, the carry indicator is set to 1 (on). It can subsequently be set or reset by the various actions listed under "Carry and Overflow Indicators" (see Figure 13).

The overflow indicator must be reset to 0 if it is to be used during execution of a subtract instruction. If the overflow indicator is at a value of 1 at the start of a subtract operation, it is not changed regardless of the result of the subtract operation.

If the overflow indicator is at a value of 0 at the start of an add operation, it is set to a value of 1 if the subtraction produces a result that exceeds the capacity of the accumulator. For example, assume that the following subtraction operation is performed:

S

1 000 0000 0000 0000	Operand in accumulator — a negative number
-0 000 0000 0000 0001	Operand in main storage — a positive number
0 111 1111 1111 1111	Result in accumulator — a positive number

(S = sign bit)

In the first place, subtracting any positive number from any negative number should produce a negative result. But the sign bit is at a value of 0 in the result, and an overflow has occurred. The correct answer is obtained by appending (via programming) a 1 to the left of the sign bit. The overflow indicator is turned on (set to 1) for this operation. Refer to "Carry and Overflow Indicators" for a discussion of how these two indicators can be used together in certain arithmetic operations.

Maximum capacity of the accumulator is:

Power of 2 Notation	Decimal Notation	Hexadecimal Notation
$+2^{15}-1$	+32,767	+7FFF
-2^{15}	-32,768	-8000

Examples

Subtract

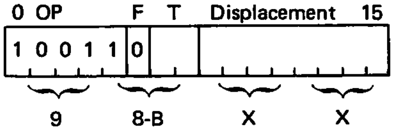
Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		S					DISP	90XX	Subtract contents of CSL at EA (I+DISP) from A
		S			1		DISP	91XX	Subtract contents of CSL at EA (XR1+DISP) from A
		S			2		DISP	92XX	Subtract contents of CSL at EA (XR2+DISP) from A
		S			3		DISP	93XX	Subtract contents of CSL at EA (XR3+DISP) from A
		S		L			ADDR	9400XXXX	Subtract contents of CSL at EA (Addr) from A
		S		L	1		ADDR	9500XXXX	Subtract contents of CSL at EA (Addr+XR1) from A
		S		L	2		ADDR	9600XXXX	Subtract contents of CSL at EA (Addr+XR2) from A
		S		L	3		ADDR	9700XXXX	Subtract contents of CSL at EA (Addr+XR3) from A
		S		I			ADDR	9480XXXX	Subtract contents of CSL at EA (V in CSL at Addr) from A
		S		I	1		ADDR	9580XXXX	Subtract contents of CSL at EA (V in CSL at “Addr+XR1”)
									from A
		S		I	2		ADDR	9680XXXX	Subtract contents of CSL at EA (V in CSL at “Addr+XR2”)
									from A
		S		I	3		ADDR	9780XXXX	Subtract contents of CSL at EA (V in CSL at “Addr+XR3”)
									from A

Subtract Double

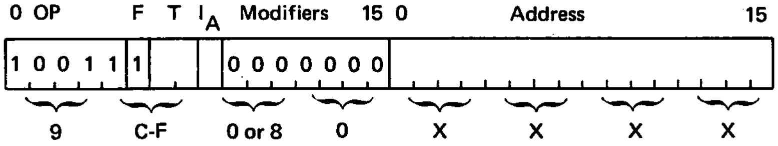
Mnemonic

SD

Short Format



Long Format



Description

Purpose of this instruction is to subtract a 32-bit operand that is in core storage from the contents of the accumulator and accumulator extension. The accumulator and accumulator extension must be loaded with the desired operand before the subtract-double operation is performed. A load-double instruction can be used to load the accumulator and accumulator extension.

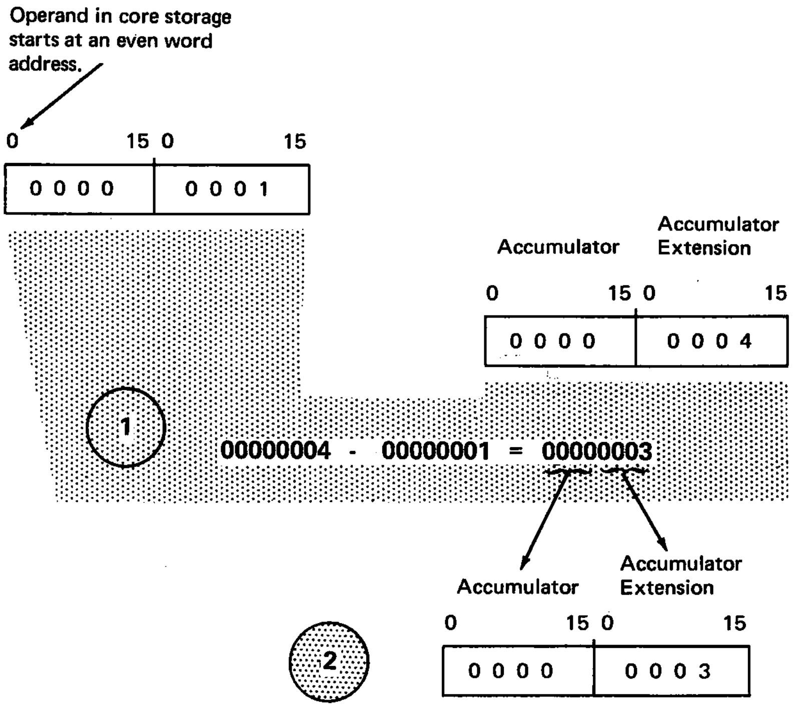
The operand addressed by the subtract-double instruction must start at an even-word address in core storage. If an odd-word address is specified by the subtract-double instruction, the single word at that address is subtracted from the accumulator and from the accumulator extension.

Except for the size of the operands, the subtract-double operation proceeds in much the same manner as the subtract operation:

1. The result of the subtract-double operation is placed in the accumulator and accumulator extension.
2. Result of the operation is either positive or negative, depending upon the magnitude and signs of the participating operands. (For a list of the magnitude and size of operands that give the sign of the result, see "Subtract Instruction".)
3. The value in the accumulator and accumulator extension has a sign as signified by bit 0 in the accumulator. The leftmost bit of each operand determines that operand's sign.
4. The operand in core storage is unchanged by the operation.

Pictorially, the operation proceeds as follows:

Subtract Double Operation



Except for the fact that the operand addressed should start at an even word address, there are no addressing exceptions for the subtract-double instruction; all forms of addressing that are described under "Effective Address Generation" apply to the SD instruction.

Indicators: The carry indicator is automatically reset to 0 at the beginning of execution of a subtract-double instruction. If, during execution of the subtract-double instruction, a borrow occurs to the left of the high-order (leftmost) position of the accumulator, then the carry indicator is set to 1 (on); if no such borrow occurs, the carry indicator remains reset. It can subsequently be set or reset by the various actions listed under "Carry and Overflow Indicators" (see Figure 13).

The overflow indicator must be reset to 0 if it is to be used during execution of a subtract-double instruction. If the overflow indicator is at a value of 1 at the start of a subtract-double operation, it is not changed regardless of the result of the operation.

If the overflow indicator is at a value of zero at the start of a subtract-double operation, it is set to a value of 1 if the subtraction produces a result that exceeds the capacity of the accumulator plus the accumulator extension. For example, assume that the number 1 is subtracted from the largest negative number that can be held in the accumulator and accumulator extension:

S	
1 000 0000 0000 0000 0000 0000 0000 0000	In accumulator and extension
-0 000 0000 0000 0000 0000 0000 0000 0001	From core storage
0 111 1111 1111 1111 1111 1111 1111 1111	

(S = sign bit)

The capacity of the accumulator and accumulator extension is exceeded because subtracting 1 from a negative number should produce a negative number result that is 1 greater than the original number. But the result is clearly a positive number (bit 0 in the accumulator equals 0). Therefore, for this operation, the overflow indicator is set on.

Maximum capacity of the accumulator and accumulator extension is:

Power of 2 Notation	Decimal Notation	Hexadecimal Notation
$+2^{31}-1$	2,147,483,647	7FFFFFFF
-2^{31}	- 2,147,483,648	- 80000000

Refer to "Carry and Overflow Indicators" for a discussion of how these two indicators can be used together in certain arithmetic operations.

Examples

Subtract Double

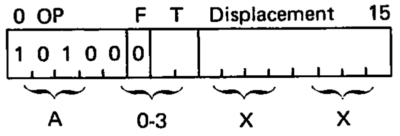
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	SD			DISP	98XX	Subtract contents of CSL at EA (I+DISP) and EA+1 from A and Q
	SD		1	DISP	99XX	Subtract contents of CSL at EA (XR1+DISP) and EA+1 from A and Q
	SD		2	DISP	9AXX	Subtract contents of CSL at EA (XR2+DISP) and EA+1 from A and Q
	SD		3	DISP	9BXX	Subtract contents of CSL at EA (XR3+DISP) and EA+1 from A and Q
	SD	L		ADDR	9C00XXXX	Subtract contents of CSL at EA (Addr) and EA+1 from A and Q
	SD	L	1	ADDR	9D00XXXX	Subtract contents of CSL at EA (Addr+XR1) and EA+1 from A and Q
	SD	L	2	ADDR	9E00XXXX	Subtract contents of CSL at EA (Addr+XR2) and EA+1 from A and Q
	SD	L	3	ADDR	9F00XXXX	Subtract contents of CSL at EA (Addr+XR3) and EA+1 from A and Q
	SD	I		ADDR	9C80XXXX	Subtract contents of CSL at EA (V in CSL at Addr) and EA+1 from A and Q
	SD	I	1	ADDR	9D80XXXX	Subtract contents of CSL at EA (V in CSL at "Addr+XR1") and EA+1 from A and Q
	SD	I	2	ADDR	9E80XXXX	Subtract contents of CSL at EA (V in CSL at "Addr+XR2") and EA+1 from A and Q
	SD	I	3	ADDR	9F80XXXX	Subtract contents of CSL at EA (V in CSL at "Addr+XR3") and EA+1 from A and Q

Multiply

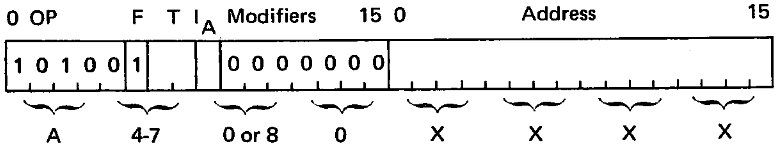
Mnemonic

M

Short Format



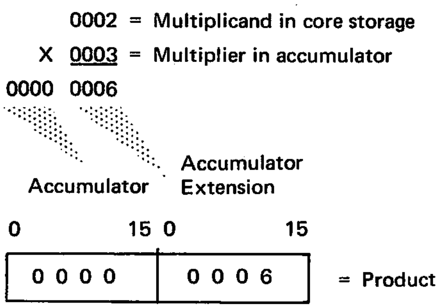
Long Format



Description

Execution of this instruction results in multiplication of a 16-bit multiplicand (in core storage) by a 16-bit multiplier (in the accumulator). The 32-bit product that is developed replaces the contents of the accumulator and accumulator extension. The product is developed so that the more significant bits are in the accumulator. (The sign of the product is indicated by bit 0 in the accumulator.) The operation can be portrayed pictorially as follows:

Multiplication (in Hexadecimal Notation)



The multiplier must be loaded into the accumulator before the multiply operation is performed. The multiplicand is addressed in the normal manner by the multiply instruction. The multiplicand is unchanged in core storage as a result of the operation.

There are no addressing exceptions for the multiply instruction; all forms of addressing that are described under "Effective Address Generation" apply to the M instruction.

The largest product that can be developed is 2^{30}. This product results from multiplying the largest 16-bit negative number (-2^{15}) by itself.

Indicators: The carry and overflow indicators are not affected during the M operation.

Examples

Note: There is only one multiply instruction. No separate multiply instruction exists for double-precision operands.

Multiply

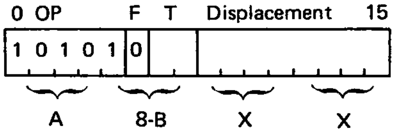
Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		M					DISP	A0XX	Multiply contents of CSL at EA (I+DISP) by A
		M			1		DISP	A1XX	Multiply contents of CSL at EA (XR1+DISP) by A
		M			2		DISP	A2XX	Multiply contents of CSL at EA (XR2+DISP) by A
		M			3		DISP	A3XX	Multiply contents of CSL at EA (XR3+DISP) by A
		M		L			ADDR	A400XXXX	Multiply contents of CSL at EA (Addr) by A
		M		L	1		ADDR	A500XXXX	Multiply contents of CSL at EA (Addr+XR1) by A
		M		L	2		ADDR	A600XXXX	Multiply contents of CSL at EA (Addr+XR2) by A
		M		L	3		ADDR	A700XXXX	Multiply contents of CSL at EA (Addr+XR3) by A
		M		I			ADDR	A480XXXX	Multiply contents of CSL at EA (V in CSL at Addr) by A
		M		I	1		ADDR	A580XXXX	Multiply contents of CSL at EA (V in CSL at "Addr+XR1")
									by A
		M		I	2		ADDR	A680XXXX	Multiply contents of CSL at EA (V in CSL at "Addr+XR2")
									by A
		M		I	3		ADDR	A780XXXX	Multiply contents of CSL at EA (V in CSL at "Addr+XR3")
									by A

Divide

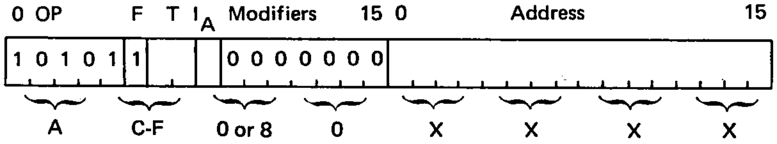
Mnemonic

D

Short Format



Long Format



Description

The divide instruction causes a 32-bit value to be divided by a 16-bit word from core storage. Result of the operation is placed in the accumulator; the remainder is placed in the accumulator extension.

Before the divide operation is started, the 32-bit value must be loaded into the accumulator and accumulator extension. (The units position is in bit 15 of the accumulator extension.) The divide instruction then addresses a 16-bit word (in core storage) that is used as the divisor. Examples of the arithmetic are:

Example 1

$$\overbrace{0000}^{\text{In Accumulator}}\quad \underbrace{0008}_{\text{In Accumulator Extension}} \div \overbrace{0004}^{\text{From Core Storage}} \quad = \quad \underbrace{0002}_{\text{In Accumulator}}\quad \overbrace{0000}^{\text{Remainder in Accumulator Extension}}$$

Example 2

$$\overbrace{0000}^{\text{In Accumulator}}\quad \underbrace{0008}_{\text{In Accumulator Extension}} \div \overbrace{0003}^{\text{From Core Storage}} \quad = \quad \underbrace{0002}_{\text{In Accumulator}}\quad \overbrace{0002}^{\text{Remainder in Accumulator Extension}}$$

The names of the values in the operation are:

$$\overbrace{00000008}^{\text{Dividend}} \div \overbrace{0004}^{\text{Divisor}} \quad = \quad \overbrace{0002}^{\text{Quotient}}\quad \overbrace{0000}^{\text{Remainder}}$$

The sign of the remainder is always the same as the sign of the original dividend. Before the operation, bit 0 of the accumulator specifies the sign of the dividend: bit 0 = 0 specifies a positive dividend; bit 0 = 1 specifies a negative dividend. Therefore, the value of bit 0 of the accumulator extension (after the operation when the extension contains the remainder) is the same as the original value of bit 0 of the accumulator (when it contained the original dividend).

The sign of the quotient is determined as follows:

Dividend	÷	Divisor	=	Quotient
+	÷	+	=	+
–	÷	+	=	–
+	÷	–	=	–
–	÷	–	=	+

If a 16-bit dividend (in the accumulator) is the result of some prior operation, it must be shifted to the right 16 places into the accumulator extension before the divide operation is performed. (A shift-right-accumulator-and-extension instruction can be used for this purpose.)

There are no addressing exceptions for the divide instruction; all forms of addressing that are described under "Effective Address Generation" apply to the D instruction.

The largest dividend that can be correctly operated on is $2^{30} + 2^{15} - 1$ (1,073,774,591 decimal) if divided by the largest negative divisor, -2^{15} (– 32,768 decimal).

Indicators: The carry indicator is not affected during a divide operation. The overflow indicator must be reset to 0 before the divide operation if it is to be used. When the overflow indicator is initially at a value of 0, it is set to 1 during a divide operation for either of two conditions:

1. An attempt is made to divide by zero. (The divisor, in core storage, has a value of 0000 0000 0000 0000.)
2. A quotient overflow occurs. A quotient overflow occurs when the quotient exceeds the range -2^{15} to $+2^{15} - 1$. A quotient overflow causes the accumulator and accumulator extension to be left in an undefined state.

Examples

Divide

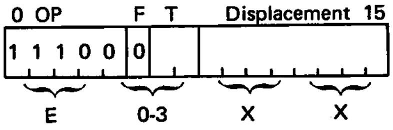
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	D			DISP	A8XX	Divide A and Q by contents of CSL at EA (I+DISP)
	D		1	DISP	A9XX	Divide A and Q by contents of CSL at EA (XR1+DISP)
	D		2	DISP	AAXX	Divide A and Q by contents of CSL at EA (XR2+DISP)
	D		3	DISP	ABXX	Divide A and Q by contents of CSL at EA (XR3+DISP)
	D	L		ADDR	AC00XXXX	Divide A and Q by contents of CSL at EA (Addr)
	D	L	1	ADDR	AD00XXXX	Divide A and Q by contents of CSL at EA (Addr+XR1)
	D	L	2	ADDR	AE00XXXX	Divide A and Q by contents of CSL at EA (Addr+XR2)
	D	L	3	ADDR	AF00XXXX	Divide A and Q by contents of CSL at EA (Addr+XR3)
	D	I		ADDR	AC80XXXX	Divide A and Q by contents of CSL at EA (V in CSL at Addr)
	D	I	1	ADDR	AD80XXXX	Divide A and Q by contents of CSL at EA (V in CSL at "Addr+XR1")
	D	I	2	ADDR	AE80XXXX	Divide A and Q by contents of CSL at EA (V in CSL at "Addr+XR2")
	D	I	3	ADDR	AF80XXXX	Divide A and Q by contents of CSL at EA (V in CSL at "Addr+XR3")

Logical AND

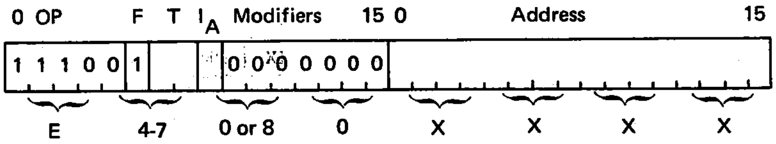
Mnemonic

AND

Short Format



Long Format



Description

The contents of the accumulator are ANDed, bit by bit, with the contents of the addressed core-storage location. The result replaces the contents of the accumulator. ANDing occurs only between corresponding bit positions in the accumulator and the core-storage word: bit 0 is ANDed only with bit 0, bit 1 only with bit 1, and so on. The four possible ANDing results are:

Bit Values		
From Core Storage Word	From Accumulator	Result in Accumulator
0	0	0
0	1	0
1	0	0
1	1	1

Contents of the addressed core-storage word are not changed as a result of the operation.

An example of ANDing is:

	0101 0000 1111 1010	Word in accumulator
AND	1010 1111 1010 1111	Word from core storage
	0000 0000 1010 1010	Result in accumulator

There are no addressing exceptions for the logical AND instruction; all forms of addressing that are described under "Effective Address Generation" apply to the AND instruction.

Indicators: The carry and overflow indicators are not affected during the AND operation.

Programming Note: The AND instruction is particularly useful in two applications.

1. It can be used to set a specific bit off as:

	XXXX XXXX XXXX XXXX	Word in accumulator
AND	1111 1101 1111 1111	Mask in storage
	XXXX XX0X XXXX XXXX	Result in accumulator

2. It can be used to isolate a bit for testing as:

	XXXX XXXX XXXX XXXX	Word in accumulator
AND	0001 0000 0000 0000	Mask in storage
	000X 0000 0000 0000	Result in accumulator

Examples

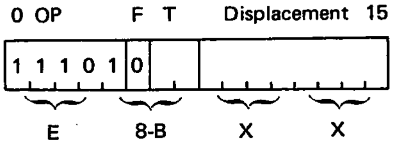
Logical AND

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	AND			DISP	E0XX	AND contents of CSL at EA (I+DISP) with A
	AND		1	DISP	E1XX	AND contents of CSL at EA (XR1+DISP) with A
	AND		2	DISP	E2XX	AND contents of CSL at EA (XR2+DISP) with A
	AND		3	DISP	E3XX	AND contents of CSL at EA (XR3+DISP) with A
	AND	L		ADDR	E400XXXX	AND contents of CSL at EA (Addr) with A
	AND	L	1	ADDR	E500XXXX	AND contents of CSL at EA (Addr+XR1) with A
	AND	L	2	ADDR	E600XXXX	AND contents of CSL at EA (Addr+XR2) with A
	AND	L	3	ADDR	E700XXXX	AND contents of CSL at EA (Addr+XR3) with A
	AND	I		ADDR	E480XXXX	AND contents of CSL at EA (V in CSL at Addr) with A
	AND	I	1	ADDR	E580XXXX	AND contents of CSL at EA (V in CSL at "Addr+XR1")
						with A
	AND	I	2	ADDR	E680XXXX	AND contents of CSL at EA (V in CSL at "Addr+XR2")
						with A
	AND	I	3	ADDR	E780XXXX	AND contents of CSL at EA (V in CSL at "Addr+XR3")
						with A

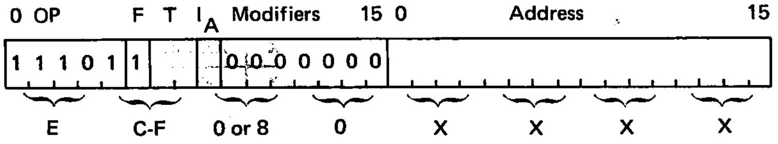
Logical OR

Mnemonic
OR

Short Format



Long Format



Description

The contents of the accumulator are ORed, bit by bit, with the contents of the addressed core-storage location. The result replaces the contents of the accumulator. ORing occurs only between corresponding bit positions in the accumulator and the core-storage word: bit 0 is ORed only with bit 0, bit 1 only with bit 1, and so on. The four possible ORing results are:

	Bit Values	
From Core Storage Word	From Accumulator	Result in Accumulator
0	0	0
0	1	1
1	0	1
1	1	1

Contents of the addressed core-storage word are not changed as a result of the operation.

An example of ORing is:

	0011 0101 1111 1010	Word in accumulator
OR	0101 0001 1010 0000	Word from core storage
	0111 0101 1111 1010	Result in accumulator

There are no addressing exceptions for the logical OR instruction; all forms of addressing that are described under "Effective Address Generation" apply to the OR instruction.

Indicators: The carry and overflow indicators are not affected during the OR operation.

Programming Note: A common use of the OR instruction is in setting a particular bit to the on (1) condition.

For example:

	XXXX XXXX XXXX XXXX	Word in accumulator
OR	0000 0010 0000 0000	Mask in storage
	XXXX XX1X XXXX XXXX	Result in accumulator

Examples

Logical OR

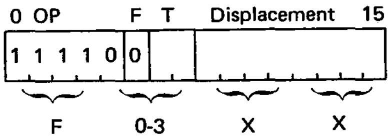
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	OR			DISP	E8XX	OR contents of CSL at EA (I+DISP) with A
	OR		1	DISP	E9XX	OR contents of CSL at EA (XR1+DISP) with A
	OR		2	DISP	EAXX	OR contents of CSL at EA (XR2+DISP) with A
	OR		3	DISP	EBXX	OR contents of CSL at EA (XR3+DISP) with A
	OR	L		ADDR	EC00XXXX	OR contents of CSL at EA (Addr) with A
	OR	L	1	ADDR	ED00XXXX	OR contents of CSL at EA (Addr+XR1) with A
	OR	L	2	ADDR	EE00XXXX	OR contents of CSL at EA (Addr+XR2) with A
	OR	L	3	ADDR	EF00XXXX	OR contents of CSL at EA (Addr+XR3) with A
	OR	I		ADDR	EC80XXXX	OR contents of CSL at EA (V in CSL at Addr) with A
	OR	I	1	ADDR	ED80XXXX	OR contents of CSL at EA (V in CSL at "Addr+XR1")
						with A
	OR	I	2	ADDR	EE80XXXX	OR contents of CSL at EA (V in CSL at "Addr+XR2")
						with A
	OR	I	3	ADDR	EF80XXXX	OR contents of CSL at EA (V in CSL at "Addr+XR3")
						with A

Logical Exclusive OR

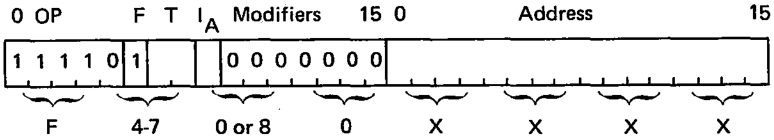
Mnemonic

EOR

Short Format



Long Format



Description

The contents of the accumulator are exclusive-ORed, bit by bit, with the contents of the addressed core-storage location. The result replaces the contents of the accumulator. Exclusive-ORing occurs only between corresponding bit positions in the accumulator and the core-storage word: bit 0 is exclusive-ORed only with bit 0, bit 1 only with bit 1, and so on. The four possible exclusive-ORing results are:

Bit Values		
From Core Storage Word	From Accumulator	Result in Accumulator
0	0	0
0	1	1
1	0	1
1	1	0

Contents of the addressed core-storage word are not changed as a result of the operation.

An example of exclusive-ORing is:

	0000 1111 0000 1111	Word in accumulator
EOR	1111 0000 0000 1111	Word from core storage
	1111 1111 0000 0000	Result in accumulator

There are no addressing exceptions for the logical exclusive-OR instruction; all forms of addressing that are described under "Effective Address Generation" apply to the EOR instruction.

Indicators: The carry and overflow indicators are not affected during the EOR operation.

Programming Note: Typical uses for the exclusive OR instruction are shown below.

1. Obtaining the complement of a number such as:

0000 0000 0000 1111	Accumulator = 15_{10}
1111 1111 1111 1111	Storage mask = -1_{10}
1111 1111 1111 0000	EOR result = -16_{10}
-1111 1111 1111 1111	Subtract storage mask = $-(-1_{10})$
1111 1111 1111 0001	Accumulator value = -15_{10}

2. Flipping and testing a switch such as:

LD SWITCH	Accumulator = XXXX XXXX XXXX XXXX
EOR MASK	Accumulator = XXXX XXZX XXXX XXXX
STO SWITCH	Location switch = XXXX XXZX XXXX XXXX
AND MASK	Accumulator = 0000 00Z0 0000 0000
BSC L A, Z	Branch to a specified location (A) if bit 6 was 0
0000 0010 0000 0000	Mask in storage
XXXX XXXX XXXX XXXX	Switch value in storage

Examples

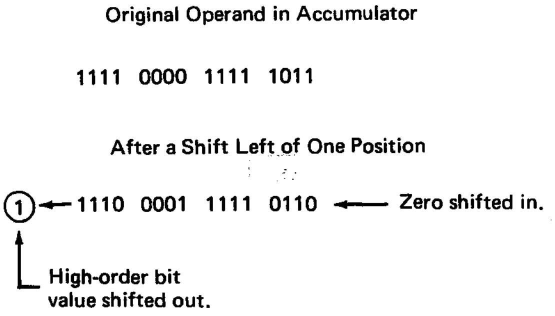
Logical Exclusive OR

Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		EOR					DISP	F0XX	EOR contents of CSL at EA (I+DISP) with A
		EOR			1		DISP	F1XX	EOR contents of CSL at EA (XR1+DISP) with A
		EOR			2		DISP	F2XX	EOR contents of CSL at EA (XR2+DISP) with A
		EOR			3		DISP	F3XX	EOR contents of CSL at EA (XR3+DISP) with A
		EOR		L			ADDR	F400XXXX	EOR contents of CSL at EA (Addr) with A
		EOR		L	1		ADDR	F500XXXX	EOR contents of CSL at EA (Addr+XR1) with A
		EOR		L	2		ADDR	F600XXXX	EOR contents of CSL at EA (Addr+XR2) with A
		EOR		L	3		ADDR	F700XXXX	EOR contents of CSL at EA (Addr+XR3) with A
		EOR		I			ADDR	F480XXXX	EOR contents of CSL at EA (V in CSL at Addr) with A
		EOR		I	1		ADDR	F580XXXX	EOR contents of CSL at EA (V in CSL at "Addr+XR1")
									with A
		EOR		I	2		ADDR	F680XXXX	EOR contents of CSL at EA (V in CSL at "Addr+XR2")
									with A
		EOR		I	3		ADDR	F780XXXX	EOR contents of CSL at EA (V in CSL at "Addr+XR3")
									with A

SHIFT INSTRUCTIONS

The purpose of each shift instruction is to shift an operand, bit by bit, to the right or to the left. Direction of shift (left or right) is dependent upon the specific instruction. The operand to be shifted must first be loaded into the accumulator (or accumulator and accumulator extension, depending upon which shift instruction is to be executed). All shift instructions are in the short format only; there are no long-format shift instructions.

The manner of shifting is dependent upon the specific shift instruction. In shift-left operations, zeros are shifted into low-order vacated positions. For example:



In shift right operations, bits that are shifted in can be:

1. Zeros – a logical shift right (SRA instruction)
2. The original value of the sign bit (always from bit 0 of the accumulator) – an arithmetic right shift (SRT instruction)
3. The bits that are shifted out of the low-order position (bit 15) of the accumulator extension (RTE instruction)

Depending upon the instruction being executed, shift operations can be ended in one or both of two ways:

1. By a specified count (the shift count) decrementing to zero. (Refer to the individual descriptions for information about where a count is set up before a shift instruction is executed.)
2. By a bit value of 1 shifting into the high-order position (bit 0) of the accumulator.

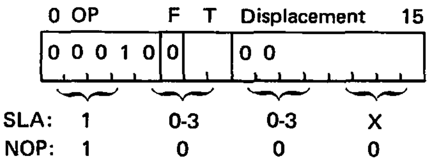
Refer to the individual descriptions for information about how a particular shift operation is ended.

Shift Left Accumulator (Or No-Operation)

Mnemonic

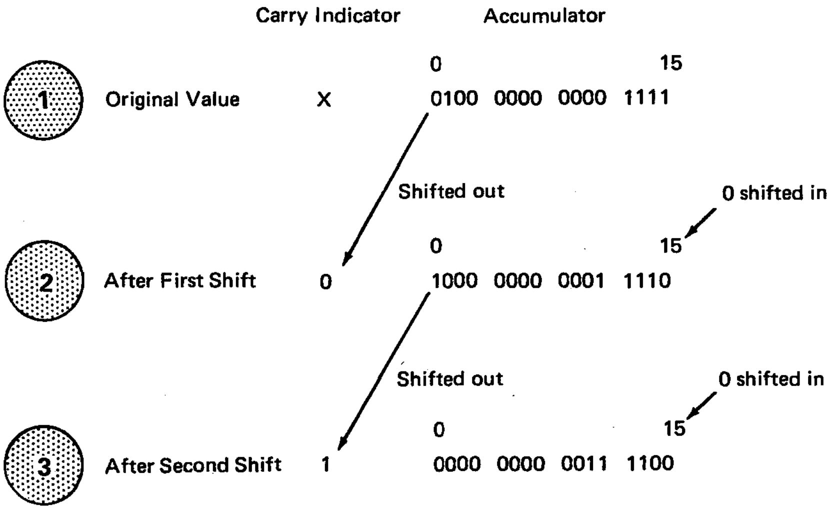
SLA
NOP

Short Format



Description

Contents of the accumulator are shifted left, a bit at a time. Each bit value shifted out (from accumulator position 0) is shifted into the carry indicator. Low-order bits shifted into accumulator-position 15 (and then to the left) are always zeros. An example of a left shift of two positions is:



The number of positions shifted is specified by a shift count. Location of the shift count is determined by the T bits in the instruction:

T Bits (6 and 7)	Shift Count Location
00	Low-order six bits of displacement (in the shift instruction)
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

A value must be put into the shift-count location before the operation is started. The operation is ended when the specified number of shifts have been performed. The shift count, however, need be set up only once. After the operation is ended, the original shift count (for example, a count in index-register 1) is the same as its initial value. (Decrementing of the count is performed in separate circuits.)

The *maximum* shift count that can be specified is 63 (111111 in binary). A left-shift of sixteen, however, puts zeros in all positions of the accumulator. In this case, the carry indicator, at the end of the operation, is at the value that accumulator-bit 15 had at the start of the operation.

A shift count of zero (000000 in binary) causes this instruction to perform as a no-operation (NOP): contents of the accumulator remain unchanged.

Core storage is not addressed during execution of the shift-left-accumulator instruction.

Indicators: Each bit value that is shifted out of accumulator-bit-position 0 is placed in the carry indicator. The carry indicator holds only one bit. Consequently, the carry indicator, at the end of the operation, is at the value of the last bit shifted out. If any shifting is to be performed, the carry indicator need not be program-reset before the shift operation because the first left-shift sets or resets the carry indicator to the first bit value shifted out. The carry indicator is not changed if the initial shift count is zero.

The overflow indicator is not affected by a shift-left-accumulator operation.

Examples

Shift Left Accumulator

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	SLA			DISP	10*X	Contents of A shift left the number of shift counts in DISP
	SLA		1		1100	Contents of A shift left the number of shift counts in XR1
	SLA		2		1200	Contents of A shift left the number of shift counts in XR2
	SLA		3		1300	Contents of A shift left the number of shift counts in XR3
	NOP			00	1000	Perform no operation

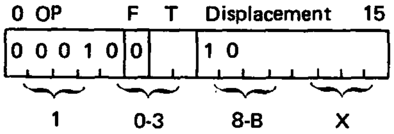
*This hexadecimal digit can be 0, 1, 2, or 3 depending upon the desired shift count.

Shift Left Accumulator and Extension

Mnemonic

SLT

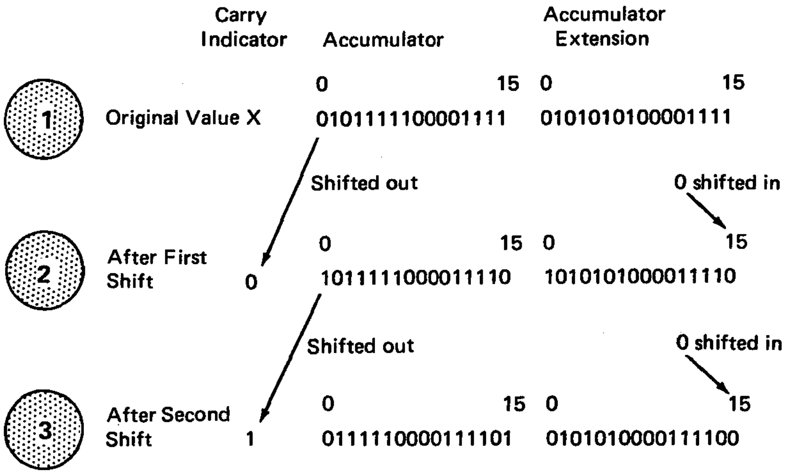
Short Format



Description

Contents of the accumulator and accumulator extension are shifted left, a bit at a time, as if they were a single 32-bit register. (The accumulator contains the leftmost 16 bits of the 32, the accumulator extension contains the rightmost 16 bits.) Each bit value shifted out from accumulator-position 0 is shifted into the carry indicator. Low-order bits shifted into accumulator-extension position 15 (and then to the left) are always zeros.

An example of a left shift of two positions is:



The number of positions shifted is specified by a shift count. Location of the shift count is determined by the T bits in the instruction:

T Bits (6 and 7)	Shift Count Location
00	Low-order six bits of displacement (in the shift instruction)
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

A value must be put into the shift-count location before the operation is started. The operation is ended when the specified number of shifts have been performed. The shift count, however, need be set up only once. After the operation is ended, the original shift count (for example, a count in index-register 1) is the same as its initial value. (Decrementing of the count is performed in separate circuits.)

Maximum shift count that can be specified is 63 (111111 in binary). A left shift of 32, however, puts zeros in all positions of the accumulator and extension. In this case, the carry indicator, at the end of the operation, is at the value that accumulator-extension bit 15 had at the start of the operation.

A shift count of zero (000000 in binary) causes this instruction to perform as a no-operation: contents of the accumulator and accumulator extension remain unchanged.

Core storage is not addressed during execution of the shift-left-accumulator-and-extension instruction.

Indicators: Each bit value that is shifted out of the accumulator bit-position 0 is placed in the carry indicator. The carry indicator holds only one bit. Consequently, the carry indicator, at the end of the operation, is at the value of the last bit shifted out. If any shifting is to be performed, the carry indicator need not be program-reset before the shift operation because the first left-shift sets or resets the carry indicator to the first bit value shifted out. The carry indicator is not changed if the initial shift count is zero.

The overflow indicator is not affected by a shift-left-accumulator-and-extension operation.

Examples

Shift Left Accumulator and Extension

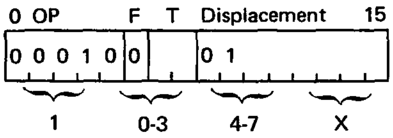
Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		SLT					DISP	10*X	Contents of A and Q shift left the number of shift counts in DISP
		SLT			1			1180	Contents of A and Q shift left the number of shift counts in XR1
		SLT			2			1280	Contents of A and Q shift left the number of shift counts in XR2
		SLT			3			1380	Contents of A and Q shift left the number of shift counts in XR3

*This hexadecimal digit can be 8, 9, A, or B depending upon the desired shift count.

Shift Left and Count Accumulator

Mnemonic
SLCA

Short Format



Description

The purpose of this instruction is to shift the contents of the accumulator left, a bit at a time, as in the shift-left-accumulator instruction. Zeros are shifted into position 15 of the accumulator and then to the left in order to fill vacated positions. Execution of the shift-left-and-count-accumulator instruction stops when:

1. A 1-bit value is in bit-position 0 of the accumulator, or
2. The shift count is decremented to zero.

The location of the shift count is specified by the T bits of the instruction as follows:

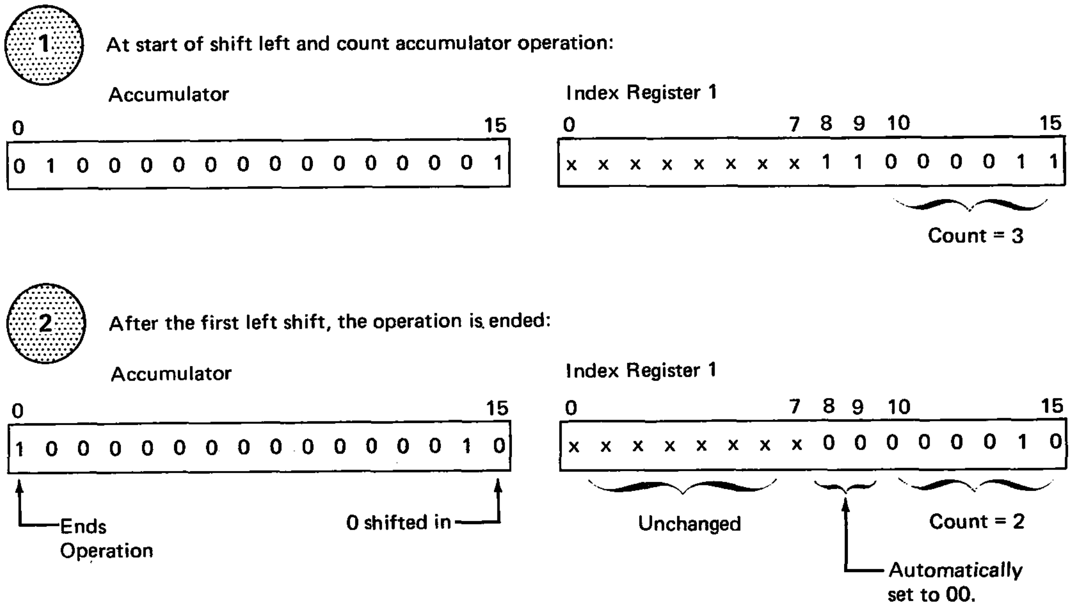
T Bits (6 and 7)	Shift Count Location
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

Note: When the T bits are 00, the shift-left-and-count-accumulator instruction is executed in exactly the same manner as a shift-left-accumulator instruction (SLA) with its T bits set to 00.

When execution of a shift-left-and-count-accumulator instruction starts, the shift count is automatically moved from the specified index register to CPU circuits that do the counting. When the operation is ended, the specified index register is updated as follows:

Index-Register Bits	Condition at End of Operation
0-7	Unchanged
8	Reset to 0
9	Reset to 0
10-15	Contain residual count

If the count is decremented to zero before a 1 is shifted into accumulator-bit 0, the residual count is zero. If a 1 is shifted into accumulator-bit 0 before the count is decremented to 0, the operation is ended. In this latter case, the remaining count value is loaded back into the index register. For example:



In this example, the initial count is 3, but after one shift has occurred, a 1 is in accumulator bit-position 0. This 1 ends the operation. One shift has occurred, so the residual count stored into the index register is 2 (3 - 1 = 2).

If the shift count is initially zero, the instruction performs as a no-operation, regardless of the value of accumulator-bit 0. In this case, no shifting occurs, the carry indicator is not changed, and none of the bits in the specified index register are altered.

If accumulator-bit 0 is initially at a value of 1 and the initial count does not equal zero, the carry indicator is turned on. Also, bits 8 and 9 of the specified index register are reset to 00, but the count is not changed. Again, shifting does not occur.

Core storage is not addressed during execution of a shift-left-and-count-accumulator instruction.

Programming Note: This instruction is particularly useful in determining specific device status and interrupt conditions. After a device status word or interrupt-level status word has been loaded into the accumulator, a shift-left-and-count-accumulator instruction can be executed so that the first 1 into accumulator position 0 stops the operation. The residual count, stored back into the specified index register, can then be used to index to the desired sub-routine. A unique subroutine is then specified for each bit in the accumulator and is indexed by the residual count. Refer to "I/O Interrupts" for a general description of the device status words and the interrupt-level status words. Specific device-status-word bit definitions are in each I/O device description.

Indicators: The carry indicator is set or reset differently for the shift-left-and-count-accumulator instruction than it is for the shift-left-accumulator instruction. The carry indicator can have the following values after execution of a shift-left-and-count-accumulator instruction:

Count	Accumulator Bit 0 Equals	Carry Indicator Equals
≠ 0	1	1
= 0	1	0
= 0	0	0

The carry indicator is set or reset by bits shifted out of accumulator-bit 0 in the shift-left-accumulator operation but not in the shift-left-and-count-accumulator operation. In the shift-left-and-count-accumulator operation, the carry indicator is affected as shown in the immediately preceding table.

If, however, the T bits (in a shift-left-and-count instruction) are 00, the carry indicator is affected exactly as described in the shift-left-accumulator instruction description.

The overflow indicator is not affected by the shift-left-and-count-accumulator operation.

Examples

Shift Left and Count Accumulator

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	SLCA			DISP	10*X	Contents of A shift left the number of shift counts in DISP
	SLCA		1		1140	Contents of A shift left the number of shift counts in XR1**
	SLCA		2		1240	Contents of A shift left the number of shift counts in XR2**
	SLCA		3		1340	Contents of A shift left the number of shift counts in XR3**

*This hexadecimal digit can be 4, 5, 6, or 7 depending upon the desired count.

**If a 1 bit shifts into accumulator position 0, the operation ends regardless of shift counts specified.

For the four examples below, assume that the index register was previously loaded by an LDX instruction. Only the low-order bit positions (10–15) of the index register (XR) are shown, and only the high-order bit positions (0–5) of the accumulator (A) are shown. Those bit positions containing an X can be 0 or 1.

Example Number	1	2	3	4
XR before SLCA	000011	000100	000101	000110
XR after SLCA	000000	000000	000001	000010
A before SLCA	00001X	00001X	00001X	00001X
A after SLCA	01XXXX	1XXXXX	1XXXXX	1XXXXX
Carry indicator after SLCA	OFF*	OFF*	ON**	ON**

*If no 1 bits were contained in the field defined by the index register (examples 1 and 2), the program can determine the value of accumulator bit 0 only by testing the accumulator sign. (Carry indicator is OFF and the index register is 0.)

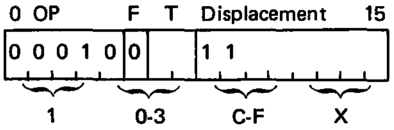
**If a 1 bit was contained in the field defined by the index register (examples 3 and 4), the SLCA instruction was terminated when an attempt was made to shift the 1 out of the high-order position, leaving the carry indicator ON and the index register at a nonzero condition. (The 1 bit remains in the high-order position.)

Shift Left and Count Accumulator and Extension

Mnemonic

SLC

Short Format



Description

The purpose of this instruction is to shift the contents of the accumulator and accumulator extension left, a bit at a time, as if these two registers made up a single 32-bit register. Zeros are shifted into position 15 of the accumulator extension and then to the left in order to fill vacated positions. Bits shifted out of accumulator-extension position 0 are shifted to the left into accumulator-position 15. Execution of the shift-left-and-count-accumulator-and-extension instruction stops when:

1. A 1-bit is in bit position 0 of the accumulator, or
2. The shift count is decremented to zero.

Location of the shift count is specified by the T bits of the instruction as follows:

T Bits (6 and 7)	Shift Count Location
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

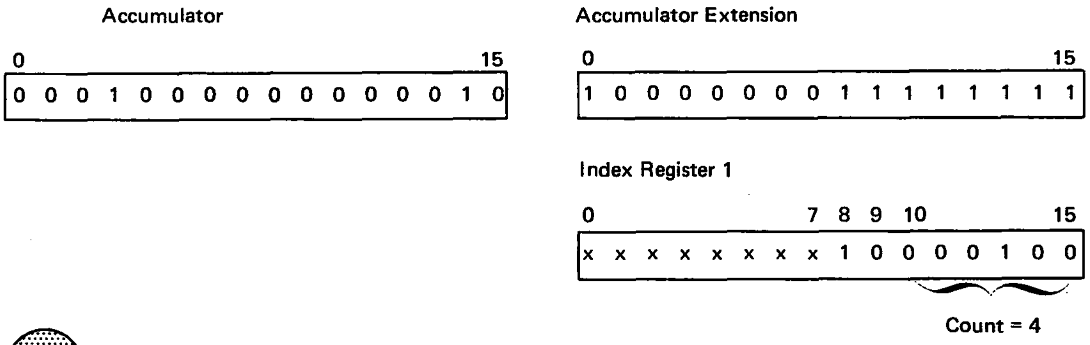
Note: When the T bits are 00, the shift-left-and-count-accumulator-and-extension instruction is executed in exactly the same manner as a shift-left-accumulator-and-extension instruction (SLT) with its T bits set to 00.

When execution of a shift-left-and-count-accumulator-and-extension instruction starts, the shift count is automatically moved from the specified index register to CPU circuits that do the counting. When the operation is ended, the specified index register is updated as follows:

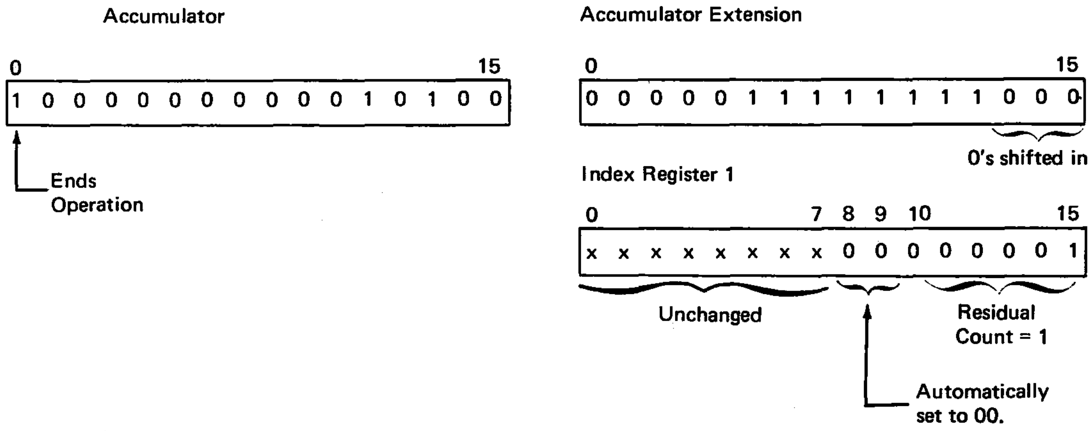
Index-Register Bits	Condition at End of Operation
0-7	Unchanged
8	Reset to 0
9	Reset to 0
10-15	Contain residual count

If the count is decremented to zero before a 1 is shifted into accumulator-bit 0, the residual count is 0. If a 1 is shifted into accumulator-bit 0 before the count is decremented to 0, the operation is ended. The residual count value is loaded back into the index register. For example:

① At start of shift left and count accumulator and extension operation:



② After the third left shift, the operation is ended:



In this example, the initial count is 4, but after three shifts have occurred, a 1 is in accumulator-position 0. This condition ends the operation. Three shifts have occurred, so the residual count stored into the index register is 1 (4 - 3 = 1).

If the shift count is initially zero, the instruction performs as a no-operation, regardless of the value of accumulator-bit 0. In this case, no shifting occurs, the carry indicator is not changed, and none of the bits in the specified index register are altered.

If accumulator-bit 0 is initially at a value of 1 and the initial count does not equal zero, the carry indicator is turned on. Also, bits 8 and 9 of the specified index register are reset to 00, but the count is not changed. Again, shifting does not occur.

Core storage is not addressed during execution of a shift-left-and-count-accumulator-and-extension instruction.

Indicators: The carry indicator is set or reset differently for the shift-left-and-count-accumulator-and-extension instruction than it is for the shift-left-accumulator-and-extension instruction. The carry indicator can have the following values after execution of a shift-left-and-count-accumulator-and-extension instruction:

Count	Accumulator Bit 0 Equals	Carry Indicator Equals
≠ 0	1	1
= 0	1	0
= 0	0	0

The carry indicator is set or reset by bits shifted out of accumulator-bit 0 in the shift-left instruction but not in the shift-left-and-count instruction. In the shift-left-and-count operation, the carry indicator is affected as shown in the immediately preceding table.

If, however, the T bits in a shift-left-and-count-accumulator-and-extension instruction are 00, the carry indicator is affected exactly as described in the shift-left-accumulator-and-extension instruction description.

The overflow indicator is not affected by the shift left and count accumulator and extension operation.

Examples

Shift Left and Count Accumulator and Extension

Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		SLC					DISP	10*X	Contents of A and Q shift left the number of shift counts in
									DISP
		SLC			1			11C0	Contents of A and Q shift left the number of shift counts in
									XR1 unless 1 bit shifts into accumulator position 0
		SLC			2			12C0	Contents of A and Q shift left the number of shift counts in
									XR2 unless 1 bit shifts into accumulator position 0
		SLC			3			13C0	Contents of A and Q shift left the number of shift counts in
									XR3 unless 1 bit shifts into accumulator position 0

***This hexadecimal digit can be C, D, E, or F depending upon the desired count.**

For the four examples below, assume that the index register was previously loaded by an LDX instruction. Only the low-order bit positions (10–15) of the index register (XR) are shown and only the high-order bit positions (0–5) of the accumulator (A) are shown. Those bit positions containing an X can be 0 or 1.

Example Number	1	2	3	4
XR before SLC	000011	000100	000101	000110
XR after SLC	000000	000000	000001	000010
A before SLC	00001X	00001X	00001X	00001X
A after SLC	01XXXX	1XXXXX	1XXXXX	1XXXXX
Carry indicator after SLC	OFF*	OFF*	ON**	ON**

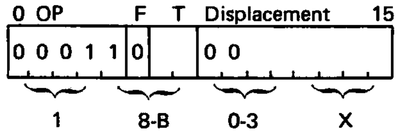
*If no 1 bits were contained in the field defined by the index register (examples 1 and 2), the program can determine the value of accumulator bit 0 only by testing the accumulator sign. (Carry indicator is OFF and the index register is 0.)

**If a 1 bit was contained in the field defined by the index register (examples 3 and 4), the SLCA instruction was terminated when an attempt was made to shift 1 out of the high-order position, leaving the carry indicator ON and the index register at a nonzero condition. (The 1 bit remains in the high-order position.)

Shift Right Logical Accumulator

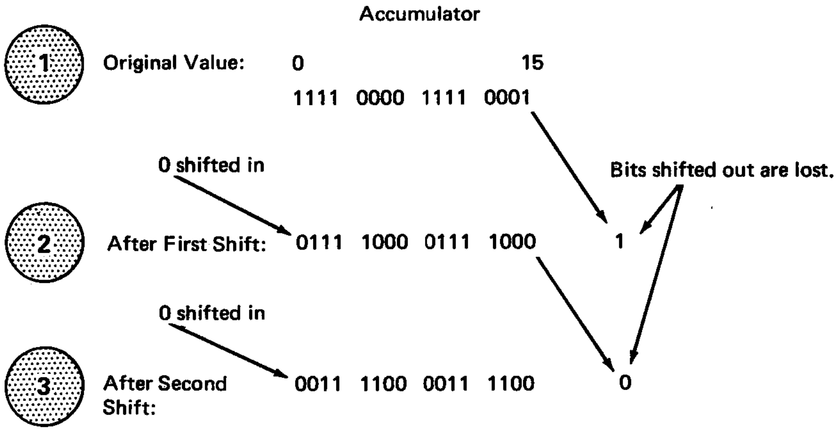
Mnemonic
SRA

Short Format



Description

The contents of the accumulator are shifted right, a bit at a time. Zeros (one for each right-shifted bit) are shifted into accumulator bit-position 0, regardless of the initial value of bit-position 0. Bits are shifted out of accumulator-position 15 and lost. An example of a right shift of two positions is:



The number of positions shifted is specified by a shift count. Location of the shift count is determined by the T bits in the instruction:

T Bits (6 and 7)	Shift Count Location
00	Low-order six bits of displacement (in the shift instruction)
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

A value must be put into the shift-count location before the operation is started. The operation is ended when the specified number of shifts have been performed. The shift count, however, need be set up only once. After the operation is ended, the original shift count (for example, a count in index-register 1) is the same as its initial value. (Decrementing of the count is performed in separate circuits.)

The maximum shift count that can be specified is 63 (111111 in binary). A right-shift of sixteen, however, puts zeros in all positions of the accumulator.

A shift count of zero (000000 in binary) causes this instruction to perform as a no-operation: contents of the accumulator remain unchanged.

Core storage is not addressed during execution of the shift-right-accumulator instruction.

Indicators: The carry and overflow indicators are not affected during execution of the SRA operation.

Examples

Shift Right Logical Accumulator

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	SRA			DISP	18*X	Contents of A shift right the number of shift counts in DISP
	SRA		1		1900	Contents of A shift right the number of shift counts in XR1
	SRA		2		1A00	Contents of A shift right the number of shift counts in XR2
	SRA		3		1B00	Contents of A shift right the number of shift counts in XR3

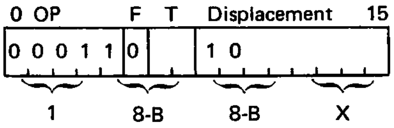
*This hexadecimal digit can be 0, 1, 2, or 3 depending upon the desired count.

Shift Right Accumulator and Extension

Mnemonic

SRT

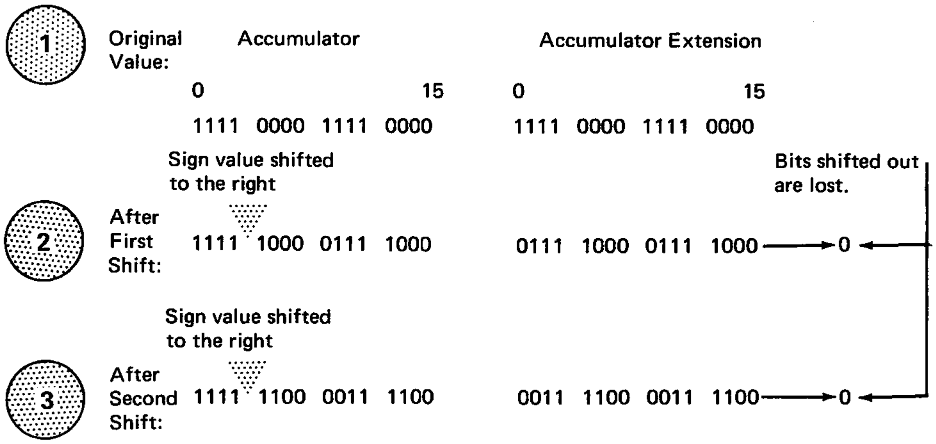
Short Format



Description

The contents of the accumulator and accumulator extension are shifted right, a bit at a time, as if these two registers made up a single 32-bit register. Bits shifted out of extension-position 15 are lost.

Bits shifted in the high-order end (bit 0) of the accumulator are the same value as the original sign bit (that is, the value of the bit in position 0 at the start of the operation). This type of shifting is called arithmetic shifting. An example of a right shift of two positions of a negative number is:



The number of positions shifted is specified by a shift count. Location of the shift count is determined by the T bits in the instruction:

T Bits (6 and 7)	Shift Count Location
00	Low-order six bits of displacement (in shift instruction)
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

A value must be put into the shift-count location before the operation is started. The operation is ended when the specified number of shifts have been performed. The shift count, however, need be set up only once. After the operation is ended, the original shift count is the same as its initial value. (Decrementing of the count is performed in separate circuits.)

The maximum shift count that can be specified is 63 (111111 in binary). A right-shift of 32 propagates the sign bit from accumulator-position 0 into all positions of the accumulator and the accumulator extension.

A shift count of zero (000000 in binary) causes this instruction to perform as a no-operation: contents of the accumulator and extension remain unchanged.

Core storage is not addressed during execution of the shift-right-accumulator-and-extension instruction.

Indicators: The carry and overflow indicators are not affected during execution of the SRT instruction.

Examples

Shift Right Accumulator and Extension

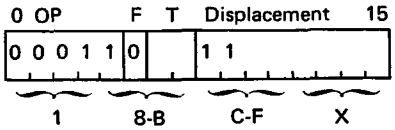
Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	SRT			DISP	18*X	Contents of A and Q shift right the number of shift counts in DISP
	SRT		1		1980	Contents of A and Q shift right the number of shift counts in XR1
	SRT		2		1A80	Contents of A and Q shift right the number of shift counts in XR2
	SRT		3		1B80	Contents of A and Q shift right the number of shift counts in XR3

*This hexadecimal digit can be 8, 9, A, or B depending upon the desired count.

Rotate Right Accumulator and Extension

Mnemonic
RTE

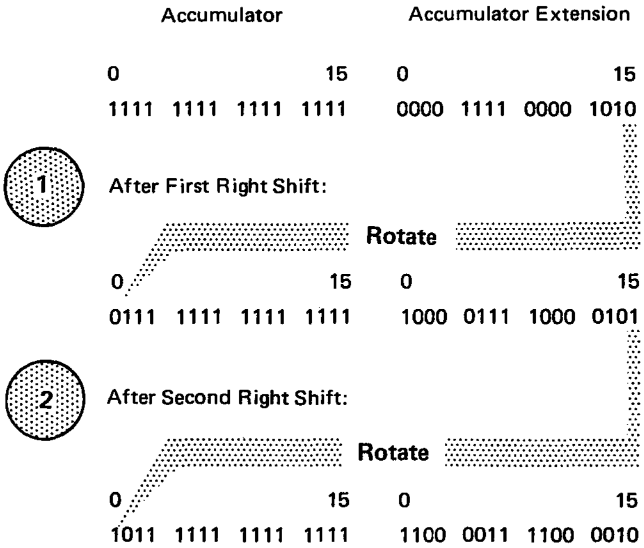
Short Format



Description

The basic purpose of this instruction – indicated by the term rotate – is to take the bits shifted out of accumulator-extension position 15 and shift them back into accumulator-position 0 (no bits are lost).

Original Value:



The number of positions shifted is specified by a shift count. Location of the shift count is determined by the T bits in the instruction:

T Bits (6 and 7)	Shift Count Location
00	Low-order six bits of displacement (in shift instruction)
01	Low-order six bits in index-register 1
10	Low-order six bits in index-register 2
11	Low-order six bits in index-register 3

A value must be put into the shift-count location before the operation is started. The operation is ended when the specified number of shifts have been performed. The shift count, however, need be set up only once. After the operation is ended, the original shift count is the same as its initial value. (Decrementing of the count is performed in separate circuits.)

Maximum shift count that can be specified is 63 (111111 in binary). A count of 31 shifts the value of accumulator-position 0 to extension-position 15; a count of 16 (or 48) reverses the positions of the two words – the one in the accumulator and the one in the accumulator extension:

① Original Value:

Accumulator	Accumulator Extension
0 15	0 15
1111 1111 1111 1111	1000 0000 1000 0000

② After Shift and Rotate 16:

Accumulator	Accumulator Extension
0 15	0 15
1000 0000 1000 0000	1111 1111 1111 1111

A shift count of zero (000000 in binary) causes this instruction to perform as a no-operation: contents of the accumulator and extension remain unchanged.

Core storage is not addressed during execution of the rotate-right-accumulator-and-extension instruction.

Indicators: The carry and overflow indicators are not affected during execution of the RTE instruction.

Examples

Rotate Right Accumulator and Extension

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	RTE			DISP	18*X	Contents of A and Q rotate right the number of counts
						in DISP
	RTE		1		19C0	Contents of A and Q rotate right the number of counts in XR1
	RTE		2		1AC0	Contents of A and Q rotate right the number of counts in XR2
	RTE		3		1BC0	Contents of A and Q rotate right the number of counts in XR3

*This hexadecimal digit can be C, D, E, or F depending upon the desired count.

BRANCH INSTRUCTIONS

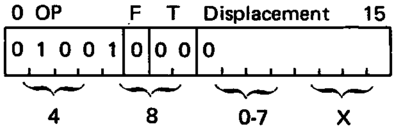
Branch instructions provide the means for departing from a sequential series of instructions, by testing to determine if a stated condition or combination of conditions exists, and returning to the point from which the departure was made. Note the unique differences between the short- and long-format instructions in the following descriptions.

Branch or Skip on Condition

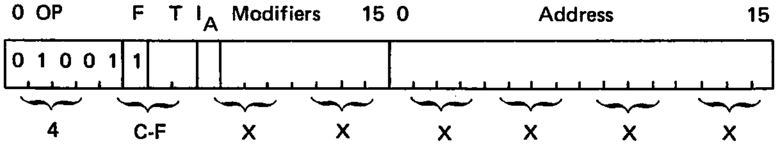
Mnemonic

BSC or BOSC

Short Format



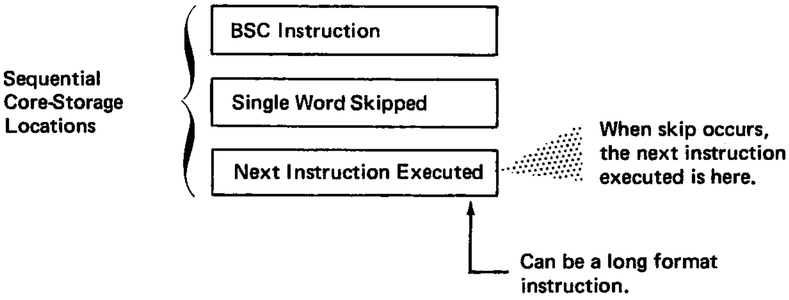
Long Format



Short-Format Description

For the short format of the instruction, the action taken as a result of execution of the instruction is one of the following:

1. Execute the word immediately following the branch-or-skip-on-condition (BSC) instruction as the next instruction, or
2. Skip the word immediately following the BSC instruction. In this case, the next instruction is at the location that starts two words after the BSC instruction.



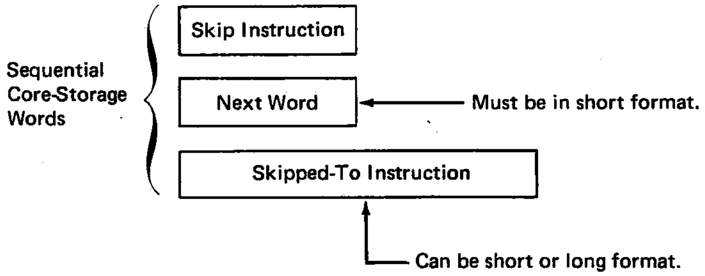
The next word must be a short format instruction. Otherwise, when the skip occurs, the second word of a long-format instruction would be accessed, resulting in a programming error.

Whether the skip is taken is dependent upon what testable items are specified by the modifier bits in the instruction and the conditions of these testable items:

Modifier Bit (When = 1)	Specifies Testing of
15	Overflow indicator off
14	Carry indicator off
13	Accumulator even
12	Accumulator plus (greater than zero)
11	Accumulator negative
10	Accumulator zero

If any specified testable item is at the stated condition, the skip is taken; if no specified testable item is at the stated condition, the skip is not taken.

For example, the carry-indicator-off and overflow-indicator-off conditions are to be tested (modifier bits 14 and 15 in the instruction are on). If either or both indicators (carry or overflow) are off, the skipped-to instruction is executed after the branch-or-skip-on-condition instruction. If both of the indicators are on, however, the next word is the instruction executed after the BSC instruction:



Examples of skipping for various combinations of specified testable items are shown in Figure 16. A skip occurs only if a specified testable item is at the stated condition – for example, the carry indicator off. (The long format of this instruction, on the other hand, operates in exactly the opposite manner.)

Core storage is not explicitly addressed during execution of the branch-or-skip-on-condition instruction. Either the very next word becomes the next instruction, or the skip is taken to the second word.

Long-Format Description

The same testable items can be specified by the same modifier bits in the long-format of the branch-or-skip-on-condition instruction as in the short format of the instruction. With the long format, however, a branch occurs rather than a skip. The core storage location branched-to contains the next instruction to be executed. Any core-storage location can be addressed for branching. The branch, however, occurs only when none of the specified testable items are at the stated conditions. The branch is to the location specified by the effective address, which is formed in the usual manner (refer to "Long Instruction Address Generation"). If no condition is specified, the instruction performs an unconditional branch.

For example, the accumulator-zero and carry-indicator-off testable items are specified (modifier bits 10 and 14 are on). If the accumulator is not zero and if the carry indicator is on, a branch occurs. Otherwise, the next sequential instruction (located immediately after the branch-or-skip-on-condition instruction) is executed.

Examples of branching for various combinations of specified testable items are shown in Figure 16.

When the indirect-address bit (bit 8) is equal to 1, indirect addressing is specified. In this case, the branch-or-skip-on-condition instruction enables the program to return to a mainline program from a subroutine or interrupt routine. This is accomplished by making the effective address of the instruction identical to the effective address of a previously executed branch-and-store-instruction-address-register instruction.

Programming Note: For an overall description of interrupts refer to "I/O Interrupts." When an interrupt request has been detected by a priority level, an interrupt occurs. The interrupt action causes a forced CPU branch (and store instruction register) to an interrupt handling subroutine. During execution of the subroutine, all interrupt requests of equal or lower priority status are prevented from interrupting the subroutine. If a request for a higher priority interrupt is detected, however, the program (subroutine) is immediately interrupted again.

At the completion of servicing any level of interrupt, the priority-status of the highest level is reset. This reset permits lower priority requests that may have been temporarily prevented to be accepted again. (Requests that have been made are recorded and are initiated again.) The reset to the current priority level (made at the end of the subroutine) is effected through execution of a branch-or-skip-on-condition instruction in which bit 9 = 1. A branch-or-skip-on-condition instruction with bit 9 = 1 is called a branch out of interrupts. (The mnemonic is BOSC.)

Bit Positions:	10	11	12	13	Skip	Branch
ACC Conditions:	Zero	Minus	Plus	Even	(F = 0)	(F=1)
Test Conditions	1	1	1	0	Always	Never
	0	0	0	0	Never	Always
	0	0	1	0	Plus	Not Plus
	1	1	0	0	Not Plus	Plus
	0	1	0	0	Minus	Not Minus
	1	0	1	0	Not Minus	Minus
	1	0	0	0	Zero	Not Zero
	0	1	1	0	Not Zero	Zero
	0	0	0	1	Even	Odd
	0	0	1	1	Even or Plus	Odd and Minus
	0	1	0	1	Even or Minus	Odd and Plus

Notes: 1. ACC Zero is not a plus condition.

2. Skip and Branch columns specify action or ACC condition required for Skip or Branch.

3. Skip on Odd condition, Carry ON, or Overflow ON are not possible.

Figure 16. Branch Examples for Branch or Skip on Condition Instruction

But the branch on condition is still conditional (if conditions are specified when bit 9 = 1). Reset of the interrupt level occurs only if the branch or skip occurs. If the branch or skip does not occur, the interrupt level is not reset. Branching or skipping is dependent upon the format, the conditions tested, and the state of the conditions tested.

This programmed branch back from an interrupt subroutine should not be confused with a normal linkage from one routine to another in which bit 9 equals 0.

Indicators: The overflow indicator is reset when tested. The carry indicator is not reset when tested. Contents of the accumulator are not changed by testing.

Examples

Branch or Skip On Condition

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	BSC			COND	48*X	Skip the next one-word instruction if ANY condition is
						sensed
	BSC	L		ADDR,COND	4C*XXXXX	Branch to CSL at EA (Addr) on NO condition
	BSC	L	1	ADDR,COND	4D*XXXXX	Branch to CSL at EA (Addr + XR1) on NO condition
	BSC	L	2	ADDR,COND	4E*XXXXX	Branch to CSL at EA (Addr + XR2) on NO condition
	BSC	L	3	ADDR,COND	4F*XXXXX	Branch to CSL at EA (Addr + XR3) on NO condition
	BSC	I		ADDR,COND	4C*XXXXX	Branch to CSL at EA (V in CSL at Addr) on NO condition
	BSC	I	1	ADDR,COND	4D*XXXXX	Branch to CSL at EA (V in CSL at "Addr + XR1") on NO
						condition
	BSC	I	2	ADDR,COND	4E*XXXXX	Branch to CSL at EA (V in CSL at "Addr + XR2") on NO
						condition
	BSC	I	3	ADDR,COND	4F*XXXXX	Branch to CSL at EA (V in CSL at "Addr + XR3") on NO
						condition

A BOSC can be used in any of the above examples instead of a BSC.

If COND is not specified:

1. In the short format, a skip is not taken.
2. In the long format, a branch is always taken.

COND represents a specified condition. The assembler codes are specified in the 1130 Assembler Language Manual.

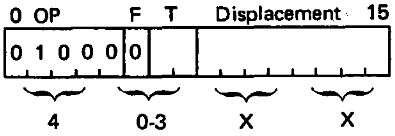
* This digit is determined by the desired condition(s) specified.

Branch and Store Instruction Address Register

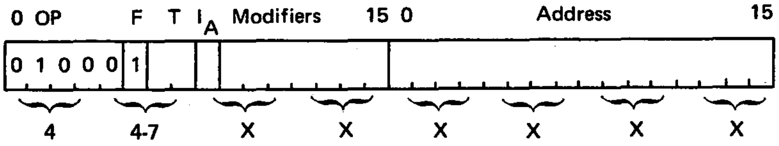
Mnemonic

BSI

Short Format



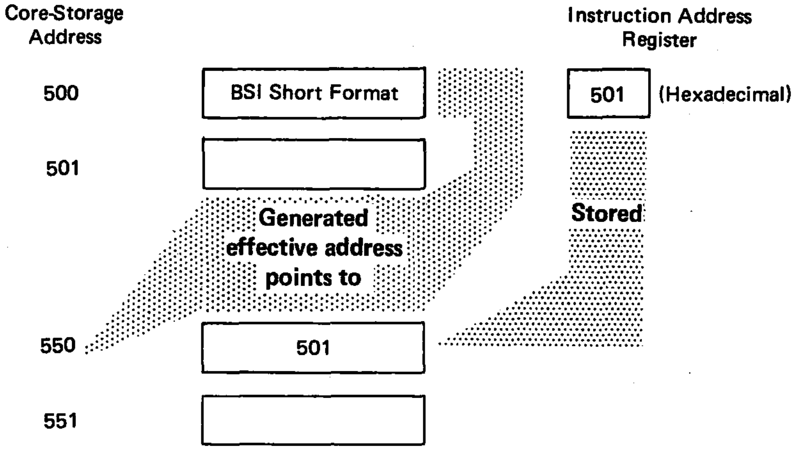
Long Format



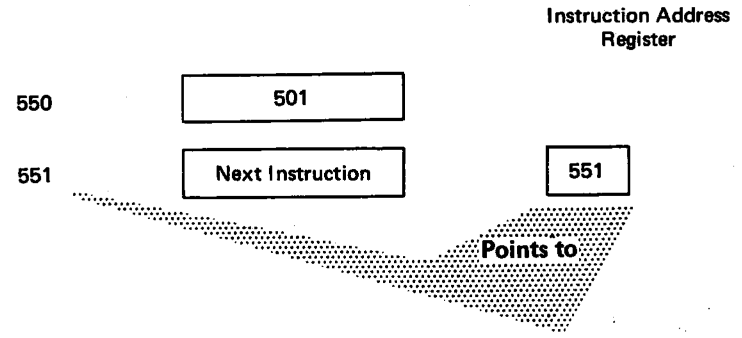
Short-Format Description

For the short format of this instruction, an unconditional branch always occurs, and the address in the instruction-address register is stored at a specified core-storage location. The address stored is that of the instruction immediately following the location of the branch instruction (BSI). This address is stored at the location specified by the effective address generated as a result of execution of the branch instruction. The instruction-address register is then loaded with the generated effective address plus one. Consequently, the location of the next instruction that is executed starts at the word immediately following the location specified by the generated effective address. Pictorially, the operation is as follows:

① The contents of the instruction address register are stored at the location specified by the effective address.



② The instruction address register is updated to the effective address plus one.



Normal short format addressing applies to this instruction (refer to "Short Instruction Address Generation"). Notice that the address stored (the address from the instruction-address register) is that of the instruction located immediately after (in core storage) the branch-and-store-instruction-address-register instruction.

Long-Format Description

In the long format, this instruction operates in a manner similar to the long-format branch-or-skip-on-condition instruction (BSC). Two actions are conditional:

1. Storing of the address contents of the instruction-address register
2. Branching to the instruction at the location specified by the effective address plus one

The testable items are the same as those in the branch-or-skip-on-condition instruction:

Modifier Bit (When = 1)	Specifies Testing of
15	Overflow indicator off
14	Carry indicator off
13	Accumulator even
12	Accumulator plus (greater than zero)
11	Accumulator negative
10	Accumulator zero

If any one or more of the specified items is at the stated condition, the branch is not performed, and the contents of the instruction-address register are not stored. For example, if overflow-off is to be tested and the overflow indicator is off, the branch is not performed and the address is not stored.

On the other hand, if none of the testable items are at the stated condition (or if no condition is specified):

1. The contents of the instruction-address register are stored at the location specified by the effective address.
2. The next instruction executed is at the effective address plus one.

These two actions are done in the same manner as described for the short format of this instruction. Addressing is performed in the normal manner (refer to "Long Instruction Address Generation").

A branch-and-store-instruction-address-register instruction that specifies indirect addressing is forced by the CPU during initiation of interrupt action. In this case, the branch instruction is not in core storage but is forced into circuitry by the CPU. At the end of the interrupt subroutine, a branch-or-skip-on-condition instruction (BOSC) should be executed to return to the stored address. Instruction execution can then resume at the point at which the program was stopped when the interrupt originally occurred. (Refer to "I/O Interrupts" for further details.)

Indicators: For the short-instruction format, neither indicator (carry or overflow) is affected. In the long format, the overflow indicator is reset if tested. The carry indicator is not reset by testing. Contents of the accumulator are not changed by testing.

Examples

Branch and Store Instruction Address Register

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	BSI			DISP	40XX	Store next sequential address in CSL at EA (I + DISP)
						and Branch to EA + 1
	BSI		1	DISP	41XX	Store next sequential address in CSL at EA (XR1 + DISP)
						and Branch to EA + 1
	BSI		2	DISP	42XX	Store next sequential address in CSL at EA (XR2 + DISP)
						and Branch to EA + 1
	BSI		3	DISP	43XX	Store next sequential address in CSL at EA (XR3 + DISP)
						and Branch to EA + 1
	BSI	L		ADDR,COND	44*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (Addr) and Branch to EA + 1
	BSI	L	1	ADDR,COND	45*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (Addr + XR1) and Branch to EA + 1
	BSI	L	2	ADDR,COND	46*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (Addr + XR2) and Branch to EA + 1
	BSI	L	3	ADDR,COND	47*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (Addr + XR3) and Branch to EA + 1
	BSI	I		ADDR,COND	44*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (V in CSL at Addr) and Branch to EA + 1
	BSI	I	1	ADDR,COND	45*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (V in CSL at "Addr + XR1") and Branch to EA + 1
	BSI	I	2	ADDR,COND	46*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (V in CSL at "Addr + XR2") and Branch to EA + 1
	BSI	I	3	ADDR,COND	47*XXXXX	If NO condition is true, store next sequential address in CSL
						at EA (V in CSL at "Addr + XR3") and Branch to EA + 1

In the long format, if COND is not specified, the branch is always taken.

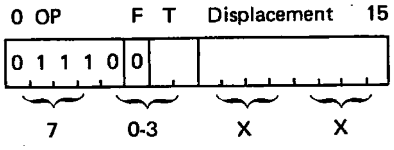
* This hexadecimal digit is determined by the testable items specified.

Modify Index and Skip

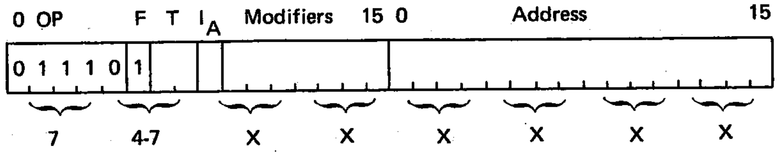
Mnemonic

MDX

Short Format



Long Format



The purpose of this instruction is:

1. To modify (increment or decrement) the contents of an index register (1, 2, or 3), the contents of the instruction-address register, or the contents of a word in core storage, and
2. To skip a word or to branch when the location being modified (except the instruction-address register contents) either changes sign or is zero after the modification.

Note: In no case are the contents of the accumulator, its extension, the overflow indicator, or the carry indicator modified or changed.

A skip causes the program to skip over the next word in storage and go to the second word in sequence. This means that when this instruction could cause a skip it should be followed by a short-format instruction. If a long-format instruction were to follow, a skip would send the program to the second word in the instruction and a programming error would result. A skip does not occur if the contents of the modified location do not change sign or do not go to zero as a result of the modification.

Short-Format Description

The displacement is expanded to 16 bits by duplication of the sign bit eight positions to the left of the high-order position. The expanded displacement is added to the register specified by the tag bits of the instruction as follows:

Tag Bits	Operation
00	Displacement added to instruction-address register
01	Displacement added to index-register 1
10	Displacement added to index-register 2
11	Displacement added to index-register 3

When the tag bits are 00, this instruction performs a no-operation or modifies the instruction-address register, depending on the value of the displacement. Because the instruction-address register contains the address of the next instruction, a displacement of zero accesses the next instruction. Any displacement results in a branch to the modified address. The displacement can be either positive or negative.

A typical operation is:

1. Assume:
   a. Tag bits = 01, specifying index-register 1.
   b. Index-register 1 contains FFFF (a minus one).
   c. The displacement is 04.
2. The expanded displacement is added to the contents of index-register 1:

   FFFF From XR1
   +0004 Expanded displacement
   0003 Result in XR1
3. A skip of one word occurs because the sign of the contents of XR1 changes from negative to positive.

Long-Format Description

Modification is accomplished according to the tag and modifier bits or indirect-address fields of the instruction. If the tag bits are 00, the expanded displacement (bits 8 through 15 of the first word of the instruction) is added to the contents of the storage location specified by the address field in the instruction. The displacement is expanded to 16 bits by duplicating the sign bit eight positions to the left of the high-order position. If the tag bits are not 00, the contents of the address field of the instruction are added to the contents of the index register that is specified by the tag bits.

Tag Bits	Index Register
01	1
10	2
11	3

Indicators: The carry and overflow indicators are not affected by execution of a modify-index-and-skip instruction.

Examples

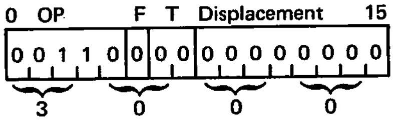
Modify Index and Skip

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	MDX			DISP	70XX	Add expanded DISP to I (no skip can occur)
	MDX		1	DISP	71XX	Add expanded DISP to XR1
	MDX		2	DISP	72XX	Add expanded DISP to XR2
	MDX		3	DISP	73XX	Add expanded DISP to XR3
	MDX	L		ADDR,DISP	74XXXXXX	Add expanded positive or negative DISP to CSL at Addr
						(Add to memory)
	MDX	L	1	ADDR	7500XXXX	Add Addr to XR1
	MDX	L	2	ADDR	7600XXXX	Add Addr to XR2
	MDX	L	3	ADDR	7700XXXX	Add Addr to XR3
	MDX	I	1	ADDR	7580XXXX	Add V in CSL at Addr to XR1
	MDX	I	2	ADDR	7680XXXX	Add V in CSL at Addr to XR2
	MDX	I	3	ADDR	7780XXXX	Add V in CSL at Addr to XR3

Wait

Mnemonic
WAIT

Short Format



Description

This instruction is valid in the short format only. When a wait instruction is executed, the CPU stops in a wait condition and can be restarted manually or by an interrupt. A manual restart (operation of the program-start key) causes resumption of the program with the next sequential instruction; an interrupt causes resumption at a point determined by the interrupt branch operation (executed at the end of the interrupt subroutine). Cycle stealing operations for transfer of data to or from main storage continue and are not affected by the wait condition.

Indicators: The carry and overflow indicators are not affected by execution of the wait instruction.

Example

Wait

Assembler Language Coding								Hexadecimal Value	Description of Instruction
Label 21 25		Operation 27 30		F 32	T 33		35 40		
		WAIT						3005	WAIT until manual start or an Interrupt occurs

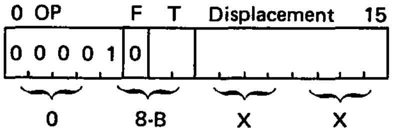
Immediate data
(determined by programmer)

EXECUTE I/O

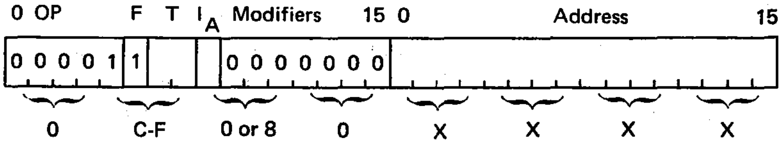
Mnemonic

XIO

Short Format



Long Format



Description

Execute I/O is the only CPU instruction that can be used to service I/O devices. Operation for the short format of the instruction is identical to operation for the long format, except for addressing, which is performed in the normal manner (refer to "Effective Address Generation" for details).

The four basic types of I/O operations, which are initiated by an execute I/O instruction by means of input/output control commands, are:

1. *Write.* Sending data from core storage to an output device.
2. *Read.* Receiving data into core storage from an input device.
3. *Control.* An operation (peculiar to the specific I/O device involved) that generally does not involve data transfer. For example, moving the access mechanism in a disk-storage drive from one cylinder to another cylinder is a control operation. In some cases, a control operation prepares an I/O device for subsequent data transfer to or from core storage.
4. *Sense Status.* Information that specifies the condition of an I/O device (such as the device is ready to be used or a printer is out of paper forms) must be obtained from the device and examined by the program so that subsequent operations can be carried out properly.

The execute I/O instruction does not in itself specify any of the preceding operations; they must be specified by input/output control commands (IOCC's).

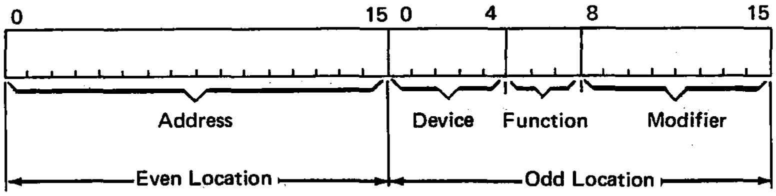
The function of execute I/O is to address the appropriate IOCC. Then the CPU uses the addressed IOCC to specify the operation to the desired device. The IOCC operation is automatic once an IOCC has been accessed by an execute I/O instruction.

The IOCC must start at an even word address in core storage. Also, the contents of the accumulator must be saved (via programming) before an execute I/O instruction is executed. The reason for this is that the accumulator is used in the analysis of the IOCC; any data in the accumulator at the start of this analysis is destroyed by the IOCC information.

Indicators: The carry and overflow indicators are not affected as a result of execution of an execute I/O instruction.

Input/Output Control Commands (IOCC's)

The format of the IOCC is:



The leftmost word of the IOCC is addressed during execution of the execute I/O instruction. The effective address generated by the execute I/O must be even. Consequently, the leftmost word of the IOCC must be located at an even-word location.

Specific fields in the IOCC are described in the following paragraphs. For descriptions of IOCC's for the various I/O devices, refer to the I/O device descriptions in this manual.

Address Field

The function of this 16-bit field depends on both the operation and the device specified. During data-transfer operations, the address field specifies the core-storage location of the word to be transferred to or from an I/O device.

With respect to the use of the address field, data transfers may be performed in either of two ways, depending upon the I/O device involved in the operation:

1. Cycle steal
2. Direct program control

In cycle-steal operations, initial information is sent to the I/O device. This initial information is sent as a result of an IOCC addressed by an execute I/O instruction. The cycle-steal I/O device then accesses core storage as required by any data transfers involved in the operation. This cycle-steal activity goes on independently of the program being executed in the CPU. Updating of the storage address that specifies each sequential core-storage location for each word of input or output data is handled independently of the CPU program. In the same manner, any data count that indicates the amount of data to be transferred is decremented automatically as the cycle-steal operations proceed.

On the other hand, in direct-program-control operations, updating of any data count or data-address information must be explicitly performed by the program. In other words, each time a data word is to be transferred, the address in the IOCC must first be updated to point to the word location in core storage that is to be used. Also, any data count must be program controlled so the operation is stopped when the required amount of data has been transferred. In direct program control, execution of an IOCC does not in itself cause automatic updating of either the data address or any associated data count. Such direct-program-control operations are applicable to certain devices that are not attached to the storage-access channel.

All devices attached to a storage-access channel (I or II) transfer data on a cycle-steal basis. (They do require direct program control, however, for control or sense operations.) Also, the single-disk drive (in the 1131 CPU Models 2, 3, and 4 only), the 1132 Printer Model 1 and 2, and the 2501 Card Reader Model A1 or A2 use a cycle-steal, data-transfer method of operation. Refer to descriptions of the individual I/O devices and the storage-access channel for further details.

Device Field

This five-bit field identifies the I/O device to which the IOCC is directed. This field is also called device address, area code, or device code. The device-code assignments are:

Device Code (Binary)	Device
00001	Console keyboard and console printer
00010	1442 Card Read Punch
00011	1134 Paper Tape Reader and Paper Tape Punch
00100	CPU single disk storage
00101	1627 Plotter
00110	1132 Printer
00111	Console entry switches, program stop key, and interrupt run mode
01000	1231 Optical Mark Page Reader
01001	2501 Card Reader
01010	Synchronous Communications Adapter
01100	IBM System/7
10001	2310 Disk Storage, Drive 1, or 2311 Disk Storage Drive, Drive 1, Disk 1
10010	2310 Disk Storage, Drive 2, or 2311 Disk Storage Drive, Drive 1, Disk 2
10011	2310 Disk Storage, Drive 3, or 2311 Disk Storage Drive, Drive 1, Disk 3
10100	2310 Disk Storage, Drive 4, or 2311 Disk Storage Drive, Drive 1, Disk 4
10101	1403 Printer
10110	2311 Disk Storage Drive, Drive 1, Disk 5
10111	2311 Disk Storage Drive, Drive 2, Disk 1 through 5
11001	2250 Display Unit

Function Field

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

Function Code	Function Specified
000	Not used
001	Write — transfers a single word from core storage to an I/O unit. The address of the storage location is provided by the address field of the IOCC.
010	Read — transfers a single word from an I/O unit to core storage. The address of the storage location is provided by the address field of the IOCC.
011	Sense Interrupt — loads the accumulator with the interrupt-level status word (ILSW) for the level being serviced at the time it is issued. This command is common to all I/O devices; therefore, no device code is needed. (Refer to "I/O Interrupts" for a description of the ILSW's.)
100	Control — causes the selected device to interpret the modifier and/or address fields as a specific control action.
101	Initiate Write — provides the ability to initiate a write operation on a device or unit that subsequently makes data transfers from core storage via a data channel.
110	Initiate Read — provides the ability to initiate a read operation on a device or unit that subsequently makes data transfers to core storage via a data channel.
111	Sense Device — loads the accumulator with the device-status word (DSW) from the device that is addressed by the IOCC. The status indicators (conditions) that can cause interrupts are reset by modifier bits in the IOCC as follows: bit 15 (= 1) for the highest level interrupt for the device; bit 14 (= 1) for the next highest level, and so on.

Note. When, during execution of a sense device command, the device field addresses a device that is not in the system configuration, every position of the accumulator is set to zero (an all-zero device-status word is in the accumulator). Normally, when a device is in the system, this all-zero condition indicates that the device is ready and not busy. But, for a device that is not in the system, this indication is meaningless. No separate indication is given to indicate that a device is not in the system.

Modifier Field

This portion of the IOCC provides additional information for the device and function specified. Where modifier bits affect a command, they are defined in the command-description section in subsequent portions of this manual.

Examples

Execute I/O

Assembler Language Coding					Hexadecimal Value	Description of Instruction
Label 21 25	Operation 27 30	F 32	T 33	35 40		
	XIO			DISP	08XX	Execute IOCC in CSL at EA (I + DISP) and EA + 1
	XIO		1	DISP	09XX	Execute IOCC in CSL at EA (XR1 + DISP) and EA + 1
	XIO		2	DISP	0AXX	Execute IOCC in CSL at EA (XR2 + DISP) and EA + 1
	XIO		3	DISP	0BXX	Execute IOCC in CSL at EA (XR3 + DISP) and EA + 1
	XIO	L		ADDR	0C00XXXX	Execute IOCC in CSL at EA (Addr) and EA + 1
	XIO	L	1	ADDR	0D00XXXX	Execute IOCC in CSL at EA (Addr + XR1) and EA + 1
	XIO	L	2	ADDR	0E00XXXX	Execute IOCC in CSL at EA (Addr + XR2) and EA + 1
	XIO	L	3	ADDR	0F00XXXX	Execute IOCC in CSL at EA (Addr + XR3) and EA + 1
	XIO	I		ADDR	0C80XXXX	Execute IOCC in CSL at EA (V in CSL at Addr) and EA + 1
	XIO	I	1	ADDR	0D80XXXX	Execute IOCC in CSL at EA (V in CSL at "Addr + XR1")
						and EA + 1
	XIO	I	2	ADDR	0E80XXXX	Execute IOCC in CSL at EA (V in CSL at "Addr + XR2")
						and EA + 1
	XIO	I	3	ADDR	0F80XXXX	Execute IOCC in CSL at EA (V in CSL at "Addr + XR3")
						and EA + 1

This section is provided for those who require an understanding of the 1130 CPU interrupt scheme. Normally, application programs (used in conjunction with the facilities of IBM-supplied programming systems for the 1130) interact with the 1130 CPU interrupt scheme by means of subroutines. Subroutines are supplied by IBM for the 1130, or they may be written by the application programmer. If IBM-supplied subroutines are used, it may not be necessary for the programmer to understand how the interrupt scheme is implemented in the 1130 CPU.

General Purpose of Interrupts

The temporary stopping of the current program routine, in order to execute some higher priority I/O subroutine, is called an interrupt. The interrupt mechanism in the CPU forces a branch out of the current program routine to one of several subroutines, depending upon which level of interrupt occurs.

I/O operations are started as a result of the execution of a program instruction. Once started, the I/O device continues its operation at the same time that the job program is being executed. Eventually the I/O operation reaches a point at which a program routine that is related to the I/O operation must be executed. At that point an interrupt is requested by the I/O device involved. The interrupt action results in a forced branch to the required subroutine.

In addition to the routine needed to start an I/O operation, subroutines are required to:

1. Transfer a data word between an I/O device and main storage (for write or read operations)
2. Handle unusual (or check) conditions related to the I/O device
3. Handle the ending of the I/O device operation

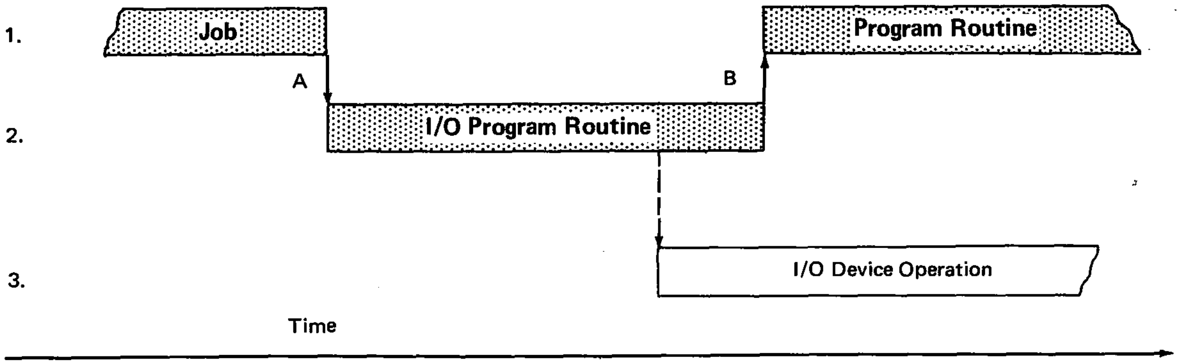
Note: Some I/O devices do not require program-instruction handling of data transfers to or from core storage. These devices are described in subsequent sections of this book. Their method of transferring data is called cycle steal and is not related to the interrupt program-subroutine method of handling data described in this section.

In order to understand the interrupt scheme, first examine the start of an I/O operation. Then contrast this operation with an interrupt occurrence, which in many respects is similar to the beginning of the I/O operation.

Starting an I/O Operation

Shown in the following diagram are:

1. The job program routine (which, for this discussion, includes those program steps not used for I/O operation handling)



2. The I/O program routine that starts an I/O device
3. The I/O device operation (such as moving a punched card through the read feed of a card reader)

Branching from the job routine to the I/O routine occurs at point A in the diagram. This branch is a program-controlled operation that is started because the job program is at a point at which the I/O operation is required (such as the reading of a card in the card reader). Similarly, when the end of the I/O routine is reached, a program-controlled branch is made back to the job routine (point B). (Program-controlled means that the logical point at which the branch occurs is determined by the program; no forcing is performed by the CPU.)

Because the CPU can execute only one instruction at a time, the I/O routine and the job routine are not overlapped.

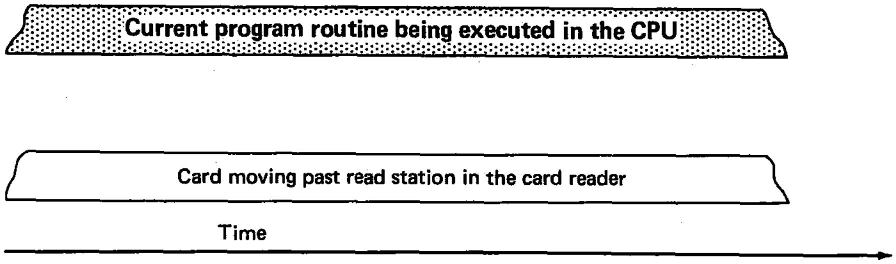
But the I/O device operation is overlapped with program-instruction execution because the I/O device can perform its functions without using the CPU mechanism required for execution of program instructions. In other words, the I/O device operation, once started, continues while program-instruction execution takes place in the CPU.

In summary, the important points to notice about starting the I/O operation are:

1. Branching operations to the I/O routine and then back to the job routine are under program control.
2. Once started, the I/O device operates at the same time as the current program routine (I/O or job) is being executed.

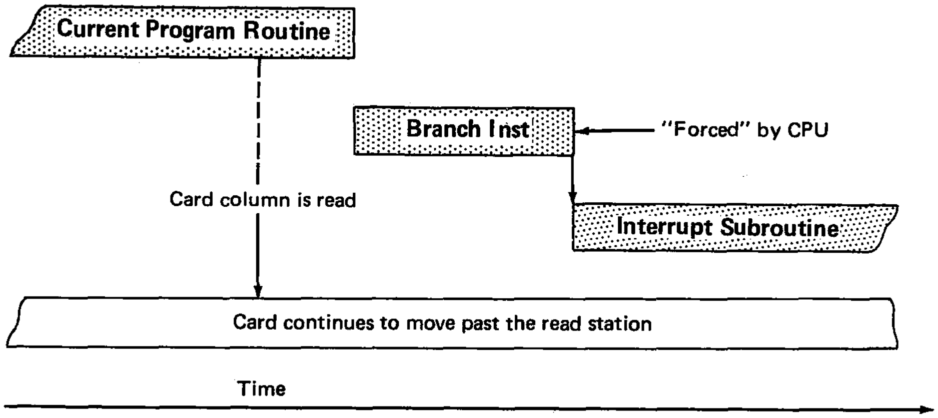
Interrupt Action

Assume that the I/O device in operation is the card reader (1442). The I/O device operation that has been started, then, is the moving of a card past the read station.



As soon as the card is moved far enough for one card column to be read, the card reader signals the CPU. This signal to the CPU is an interrupt request. The interrupt, however, does not occur until execution of the current instruction is completed. At that time, a forced branch occurs to an interrupt-handling subroutine.

The interrupt action can be shown pictorially in the following way:



Logical requirements of the interrupt subroutine are carried out before a return to the interrupted program.

To return to the program routine that was in progress before the interrupt occurred, another branch must be provided. This, however, is a program-controlled branch and, therefore, is not forced by the CPU.

In summary,

1. An interrupt request occurs at the time that an I/O operation, which otherwise proceeds independently of the program, requires program intervention.
2. The interrupt occurs after the current instruction has been completely executed.
3. The CPU then forces execution of a branch instruction, which causes a branch to a subroutine.
4. At the end of the subroutine the program branches back to the program routine that was in progress when the interrupt occurred.

As discussed, the two ways of branching into an I/O routine are:

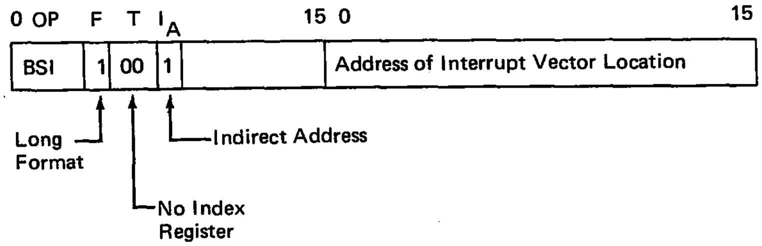
1. Program-controlled branching
2. CPU-forced branching

In a program-controlled branch to an I/O routine, a properly written program has address information for the branch-to location and the return address, if needed. Similarly, when a CPU-forced branch occurs, a way must be provided to branch to the proper subroutine and return (via another branch) to the interrupted program at the end of the subroutine. Therefore, the following information must be available.

1. The core-storage address of the beginning of the interrupt subroutine
2. The core-storage address of the next instruction to be executed in the current program (that is, the address of the place to return to after the interrupt routine is completed)

But at the time of the actual interrupt (after the current instruction is completely executed) the address of the next instruction is in the instruction-address register.(The next instruction is the one that would have been executed if the interrupt had not occurred.)

The next instruction address is stored into the first word of the interrupt-subroutine location. Storing is performed during execution of the CPU-forced branch at the time of the interrupt. The CPU-forced branch is a branch-and-store-IAR instruction (a BSI instruction). Format of this forced branch instruction is:

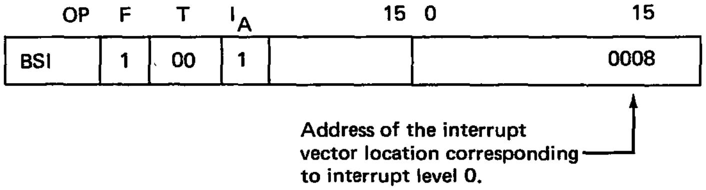


The beginning (first word) of the interrupt subroutine is specified by an address in the interrupt-vector location, which is the location that contains the effective address used during execution of the BSI instruction. The address field in the BSI instruction points to the interrupt vector. There are several interrupt-vector, fixed locations in core storage because there are several levels of interrupt. Interrupt levels and their associated vector locations are:

Interrupt Level	Interrupt Vector Location (in core storage)	Device
0	00008 (in decimal)	1442 Card Read Punch (column read, punch)
1	00009	1132 Printer, synchronous communications adapter
2	00010	Disk storage, storage access channel
3	00011	1627 Plotter, 2250 Display Unit, storage access channel, System/7
4	00012	1442 (operation complete), keyboard, console printer, 1134 Paper Tape Reader, 1055 Paper Tape Punch, 2501 Card Reader, 1403 Printer, 1231 Optical Mark Page Reader, storage access channel
5	00013	Console (program stop switch and interrupt run), storage access channel

Entering an Interrupt Subroutine

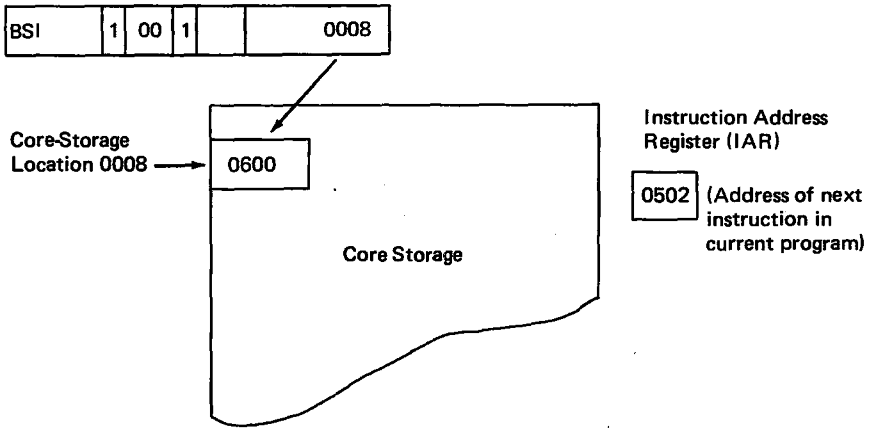
Suppose that an interrupt occurs at level 0. For this example, the address of the next instruction in the current program is 0502. This value (0502) is in the instruction-address register (IAR) when the CPU-forced branch occurs. Because the interrupt is at level 0, the CPU forces execution of the following branch instruction.



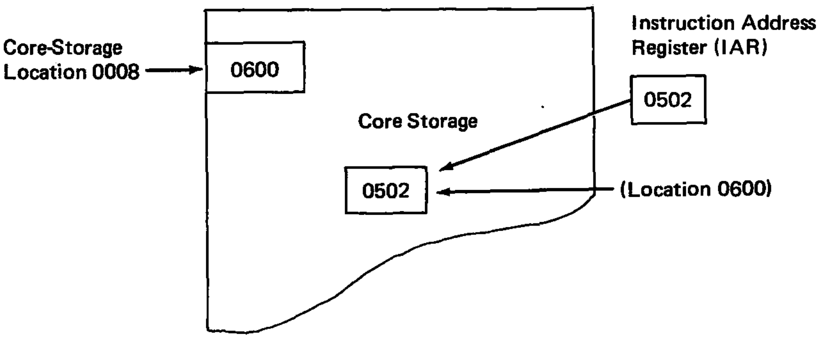
Execution of this instruction stores the contents (0502) of the instruction-address register into the first word location of the interrupt subroutine. The contents are kept there until needed for branching back to the interrupted program after the interrupt has been handled.

The first word of the interrupt subroutine is at the core-storage location specified by the contents of location 0008 (the interrupt-vector location for level 0). When the program is originally loaded into the system, the appropriate subroutine addresses must be stored (via program control) into the interrupt vectors. Therefore, for this example, the contents of core-storage location 0008 are 0600. Core-storage address 0600 is the address of the first word of the interrupt subroutine for interrupt-level 0. For this example, then, execution of the branch-and-store-IAR (the CPU-forced branch) results in:

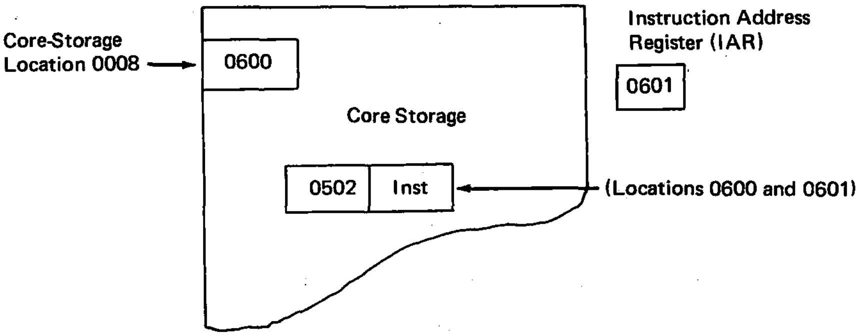
1. The contents of the second word of the BSI instruction point to the core-storage location (0008) that contains the effective address (0600) used during execution of the BSI instruction.



2. The contents (0600) of location 0008 are an address that points to the first word of the interrupt-level-0 subroutine. The value in the instruction-address register is stored at this location.



3. The instruction-address register is updated to the effective address plus 1 (the effective address used during execution of the forced BSI – for this example, 0600).



4. Because the instruction-address register now points to 0601, the first instruction of the subroutine is taken from that location. This is normal operation for the BSI instruction.

Saving Data Used by The Interrupted Program

The interrupt subroutine must save the contents of certain index and machine registers (by storing their contents into core storage) before such registers are used for data manipulation within the subroutine. Only the contents of those registers that are used by the subroutine need be saved. The data integrity of such registers must be maintained because the interrupted program may use the same registers. At the end of the interrupt subroutine, the contents of the affected registers are loaded back into the registers from core storage.

For example, the accumulator is used by all subroutines. Therefore, the contents of the accumulator must be stored into core storage before the accumulator is used by the subroutine. At the end of the subroutine, before a return to the interrupted program, the accumulator is loaded with the data that is saved.

Cause of Interrupt

In general, two methods are used to determine what condition is causing an interrupt:

1. Examination of the device status word
2. Examination of the interrupt-level status word

The I/O device causing an interrupt can be determined by examination (via programming) of the interrupt-level status word, when necessary. (The following discussion indicates when this is not necessary.) Then, after the device is known, the device-status word is examined to determine the condition in the device that caused the interrupt to occur.

First consider a level-0 interrupt. An interrupt-level status word is not used at level 0. Only the device status word need be examined at interrupt level 0.

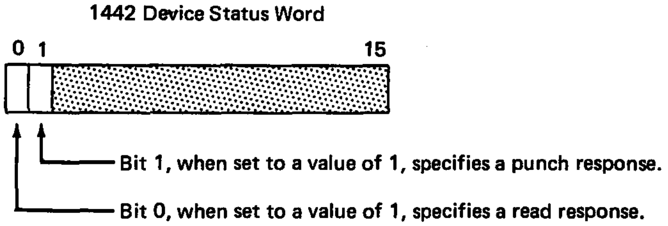
Either one of two conditions can cause level-0 interrupt to occur:

1. The card reader (1442) has a data word ready for storing into core storage during a read operation.
2. Or the card punch (1442) is ready to accept data for punching a column during a punch operation.

Because the 1442 can perform only a read or a punch operation at any one time, an interrupt at level 0 is for either a read or a punch operation. The program must determine which operation occurs. Clearly, sending a data word from core storage to the card punch during a read operation would be inappropriate.

Determination of whether the reader or the punch caused the level-0 interrupt is done in the following way: An execute I/O instruction addresses a sense-device I/O control command. Execution of this command, which specifically addresses the 1442, results in the loading of the 1442 device status word into the accumulator. Also, bit 15 in the second word of the command is on (that is, equals a value of 1). That bit, when on, causes a reset of the interrupt response. (The interrupt response is the signal, from the 1442, that originally requested the level-0 interrupt.)

Bits 0 and 1 of the 1442 device status word specify which operation (punch or read) causes the level-0 interrupt.



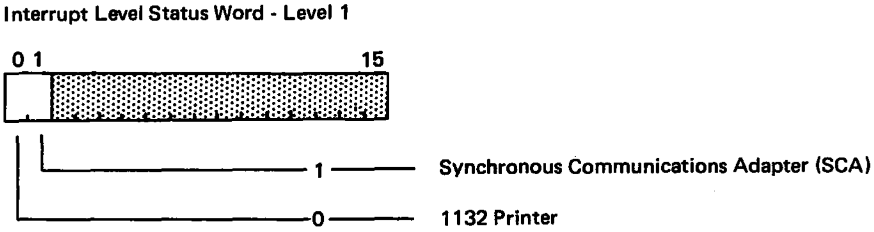
After the subroutine determines whether the interrupt is for a read or punch response, the appropriate action is taken:

1. For a read response, an execute I/O instruction addresses a read I/O control command that moves the data word from the 1442 to core storage.
2. For a punch response, an execute I/O instruction addresses a write I/O control command that moves the data word from core storage to the 1442 punch.

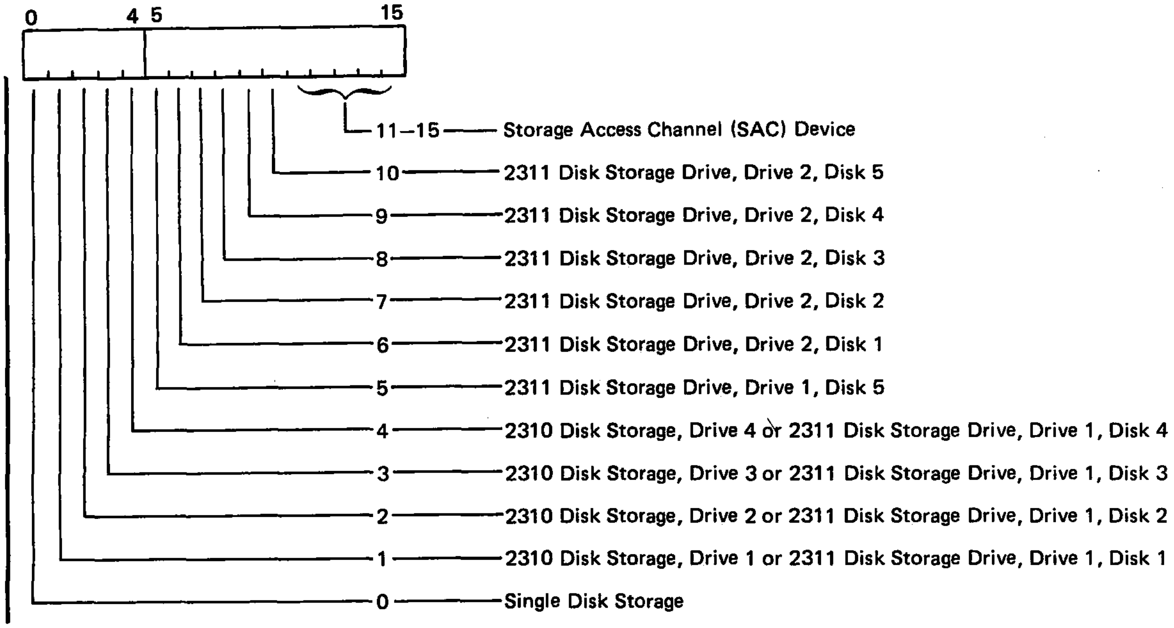
At the end of the level-0 interrupt subroutine:

1. The saved contents of registers are loaded from core storage back into the registers.
2. A branch (BOSC indirect) is made back to the interrupted program. (This resets the interrupt level.) The branch is carried out through use of the address stored in the first word of the subroutine. (Recall that the first word of the subroutine is loaded with the address of the instruction that would have been executed if the interrupt had not occurred.)

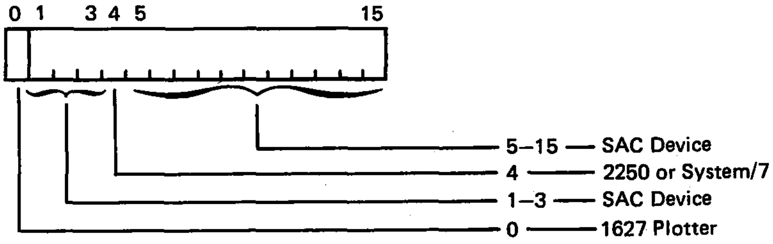
Interrupt-level status words are defined for interrupt levels 1, 2, 3, 4, and 5. During execution of the subroutine, the interrupt-level status word for the level of interrupt can be put into the accumulator by an execute I/O instruction and sense-interrupt command. The contents of the interrupt-level status words at the various interrupt levels are:



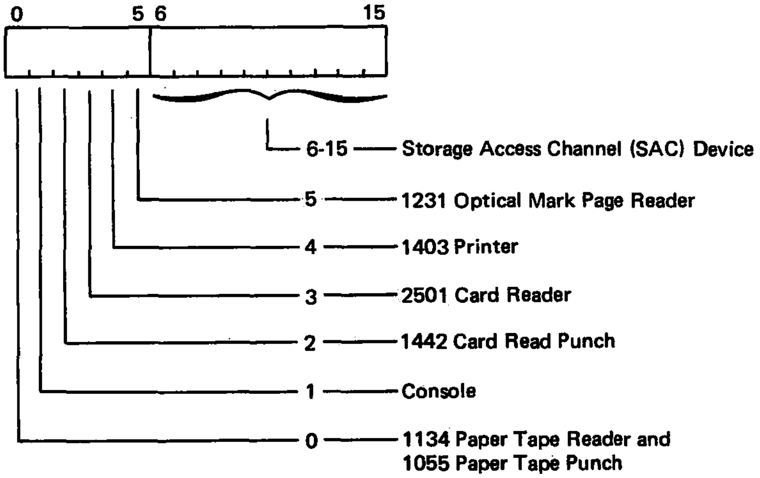
Interrupt Level Status Word - Level 2



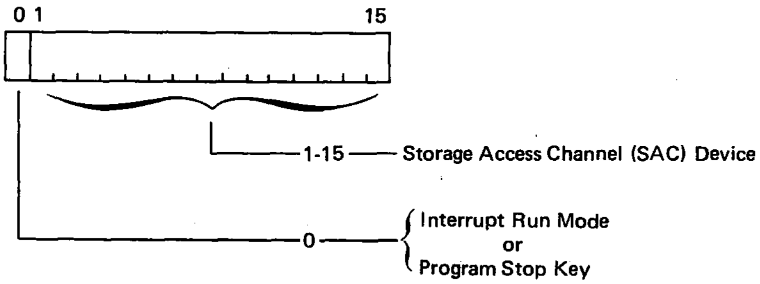
Interrupt Level Status Word - Level 3



Interrupt Level Status Word - Level 4

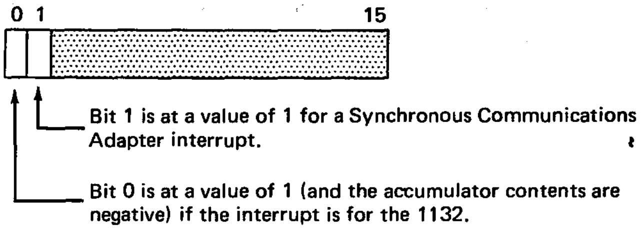


Interrupt Level Status Word - Level 5



After an interrupt-level status word is loaded into the accumulator, the contents of that word can be inspected to determine the device causing the interrupt. For example, the procedure for level 1, for either the 1132 printer or the synchronous communications adapter, involves branching on the contents of the accumulator. If the value in the accumulator is negative (bit 0 set to a value of 1), then the interrupt is for the 1132:

Interrupt Level Status Word (for Level 1) in the Accumulator



If bit 0 is not at a value of 1 during a level-1 interrupt, the interrupt must be for the synchronous communications adapter because that is the only other device that can cause an interrupt at level 1.

After the device causing the interrupt has been determined, a sense-device-status command can be executed (via an execute I/O instruction) to determine the condition within the device that caused the interrupt.

Figure 17 contains a sample interrupt-recognition routine.

Special Considerations for Level-5 Interrupt

Programming for level-5 interrupts is different from normal interrupt programming. The major difference for level 5 is the reset function in the sense-device-status-word command.

Program stop. The normally generated branch-and-store-instruction-address-register instruction (BSI) is performed with an indirect branch via storage-location 000D (0013 in decimal). The execute I/O and sense DSW sequence is then performed to determine what caused the interrupt.

At completion of the branch-out at the end of the routine (a BOSC) another level-5 interrupt is requested, as if the program stop key were operated again. The request remains active until the start key or the reset key is operated to turn off the program-stop interrupt request.

Interrupt Run Mode. The normally generated branch-and-store-instruction-address-register instruction (BSI) is performed with an indirect branch via storage location 000D (0013 in decimal). The execute I/O and sense DSW sequence is then performed to determine what caused the interrupt. The branch-out at the end of the routine (a BOSC) resets level-5 interrupt request and level-5 priority.

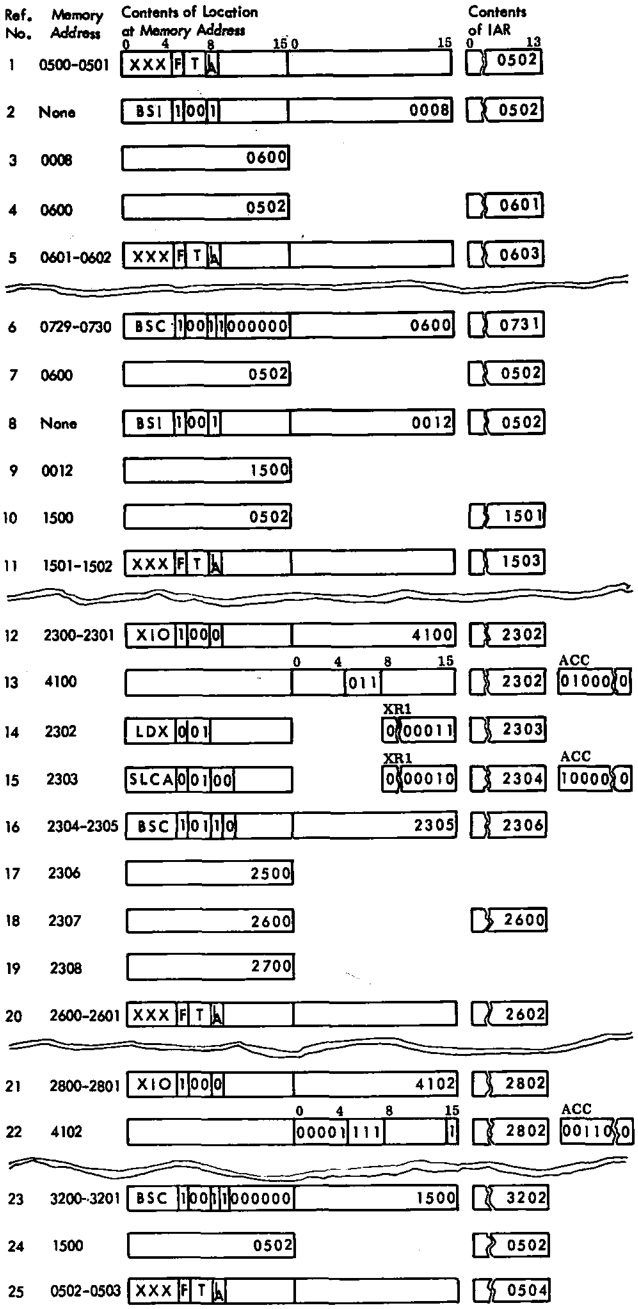
Interrupts at level 5 do not occur during other interrupt service routines. Therefore, interrupt routines cannot be traced.

Sample Interrupt Recognition Procedure

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 conclusion of 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 Level 0 BSI is 0008; EA at 0008 is 0600.
4. IAR (0502) is stored at EA (0600); IAR is then loaded with EA+1 (0601).
5. First instruction of Level 0 interrupt subroutine. Subroutine must store status of each register, all data, etc., that could be altered by execution of subroutine. Before leaving subroutine, program must restore all registers, data, etc., to condition that existed when interrupt occurred.
6. Last instruction of interrupt subroutine is a BSC instruction with 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 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 . BSC is shown as an unconditional branch (Bits 10-15 = 0); branch could have been conditional, i.e., branch executed only if no 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. EA (0502) is the location of the next mainline program instruction and is loaded into the IAR.
8. To illustrate an interrupt with low priority occurring while a higher priority interrupt is being serviced, we assume the console printer-keyboard initiated a Level 4 interrupt while the card read interrupt was being serviced. We assume no Level 0, 1, 2, or 3 interrupts are waiting and the Level 4, CPU-generated BSI can be interposed, as in ② for Level 0.
9. 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. XIO instruction EA (4100) is the memory location of the IOCC. IAR contents remain at 2302 because the IOCC controls an I/O device and is not a sequential program instruction.
13. 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 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 insturction 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. 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 as in ⑥ and ⑦ . IA (1500) of the BSC instruction is equal to EA of the CPU-generated BSI instruction that initiated the interrupt; see ⑧ , ⑨ , and ⑩ .
24. 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 17. Sample Interrupt Recognition Program

Console

INTRODUCTION

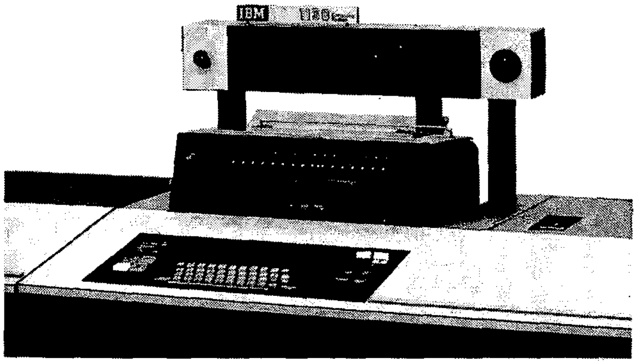
The console is made up of several functional units (Figure 18):

- Console printer
- Input keyboard
- Display panel
- Console entry switches
- Function switches and lights

Program control is required for the first two of these units, the printer and the keyboard. Program control is also required for certain console function switches, such as for the program-stop key or for one type of data-entry operation performed through the console entry switches. See the descriptions of the individual units for further details.

The *console printer* is a program-to-operator output printer. Program control is needed for the following two frequently performed operations:

1. Printing of output for program-to-operator communication
2. Printing of input typed on the input keyboard for verification of operator initiated input and for a hard-copy record of keyboard input



BR2682

Figure 18. 1131 Console

The primary function of the *input keyboard,* which looks like a typewriter keyboard (Figure 19), is to provide a means for operator-to-system input. Operation of a character key on the keyboard does not in itself cause printing at the console printer; the keyboard and console printer combination do not automatically function like a typewriter; each must be separately program-controlled.

The *display panel* is made up of visual indicators that can be examined to determine either the data contents of specific registers or various conditions in the system. For example, the wait (W) indicator is on when the CPU is in the wait-state.

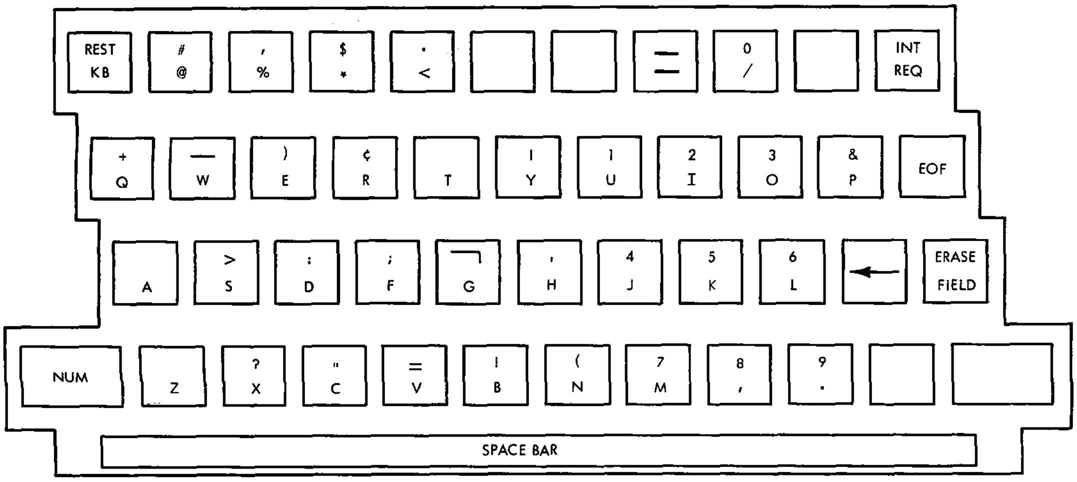


Figure 19. 1131 Console Keyboard

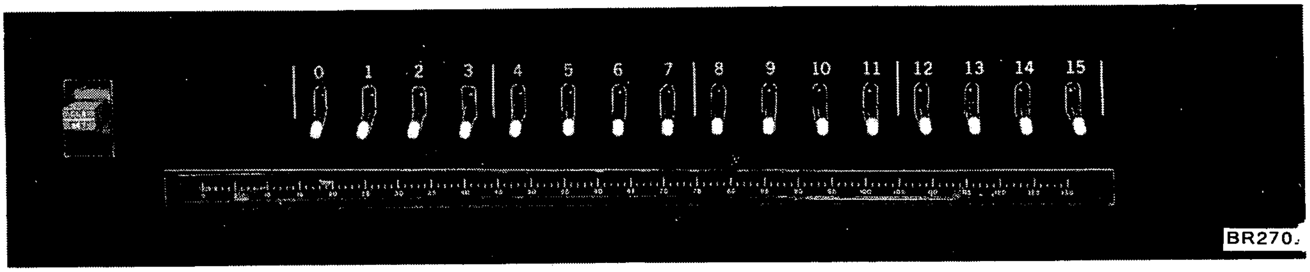


Figure 20. Console Entry Switches

The term wait-state means that the CPU, when in that state, is doing no instruction processing but is waiting for either a manual or program-controlled start. (Refer to "Wait Instruction" for further details.)

Console entry switches are used to enter data (which can be instructions) into core storage (see Figure 20). Each of the 16 switches corresponds to a specific bit (0 to 15) in the 16-bit word. Two methods for entering data from these switches are:

1. Manual control (no programming) from the console, which requires stopping the CPU
2. Program control, which can be done while application programming is being executed without stopping the CPU

Function switches and lights, which are located on both sides of the input keyboard (Figure 21), have special uses for various system operations. Refer to "Console Function Lights and Switches" for descriptions of each light and switch.

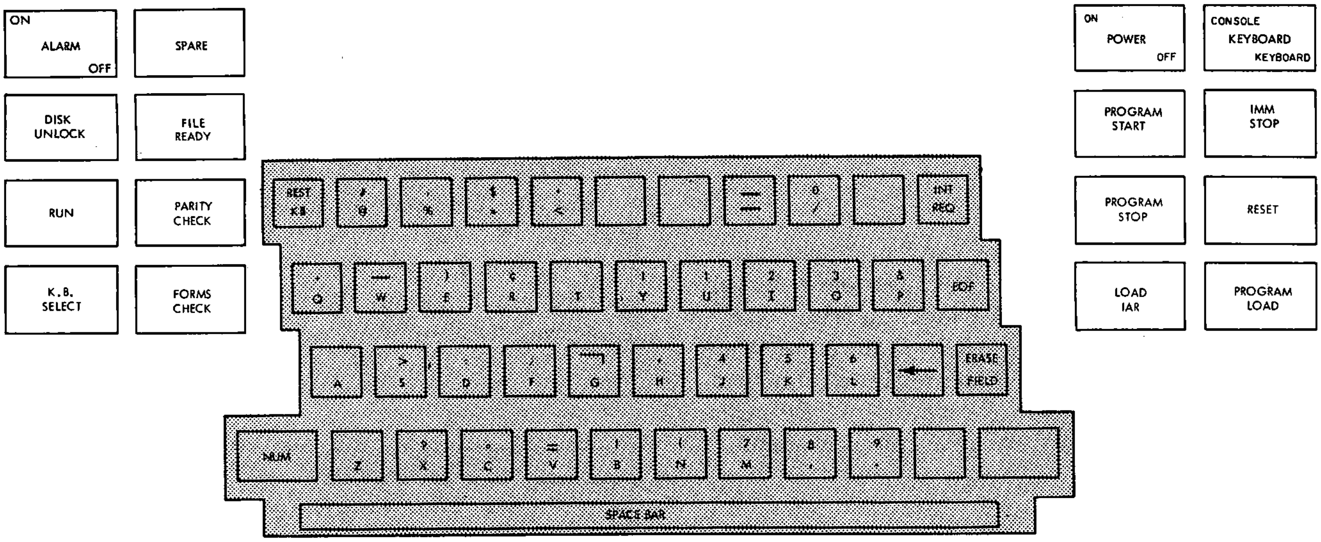


Figure 21. Console Function Lights and Switches

CONSOLE PRINTER

Printing Speed

Data to be printed must be transferred from core storage to the console printer under program control. Printing can be performed at various rates up to a maximum of 15.5 characters per second.

Data Coding

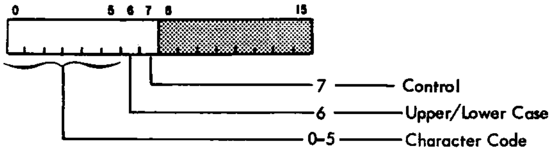
Bit patterns for both data and control characters (such as space and tabulate) are sent to the printer by means of a write IOCC. A separate write IOCC is required for each character bit pattern that is sent to the printer.

Control- and data-bit patterns are sent to the printer in the same manner. Therefore, control-character bit patterns must be in the output print record. For example, the ƀ's in the following example represent blank (or space) control characters:

AFTERƀEACHƀWORDƀAƀBLANK

If these blank control characters are not in the output record as shown, no spaces appear between the printed words.

The bit-pattern format that must be set up in each core-storage word to be sent to the printer is:

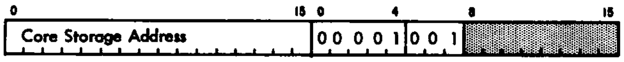


Each word sent to the printer contains the bit pattern for only one data or control character. Bit-positions 8 through 15 are not used. Bit 6 = 0 specifies lower case (LC); bit 6 = 1 specifies upper case (UC). When bit 7 = 0, the bit pattern represents a data character; when bit 7 = 1, the bit pattern represents a control character. The bit pattern for the data and control characters are shown in hexadecimal notation in Appendix A.

Commands

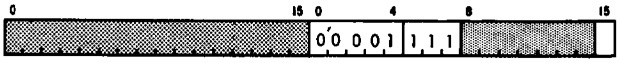
The console printer is programmed by means of two commands (IOCC's). The device code for the console printer in these IOCC's is 00001.

Write



The write command causes bits 0 through 7 of the addressed core-storage data word to be sent to the printer. If, in the addressed core-storage word, bit 7 = 0, the bit pattern represents a data character. If bit 7 = 1, the bit pattern represents a control character. A separate write command must be executed to send each data- or control-character bit pattern to the printer.

Sense Device



The sense-device IOCC allows the program to examine the device status word. Execution of the sense-device IOCC places the device status word in the accumulator.

Device Status Word Indicators

The device status word for the console keyboard and console printer is loaded into the accumulator as the result of execution of a sense-device IOCC with device-code 00001.

This device status word is for both the console keyboard and console printer (Figure 22). Bit definitions for the console printer are in this section. Refer to "Console Keyboard" for definitions of the other bits.

Note: There are two device status words for console operations. The one shown in Figure 22 is specified by device-code 00001. The other device status word, which is associated with interrupt-level 5, pertains to interrupt-run-mode and program-stop-key operations. (These operations are described under "Console Keyboard".) The latter device status word is loaded into the accumulator by means of execution of a sense-device IOCC with device-code 00111.

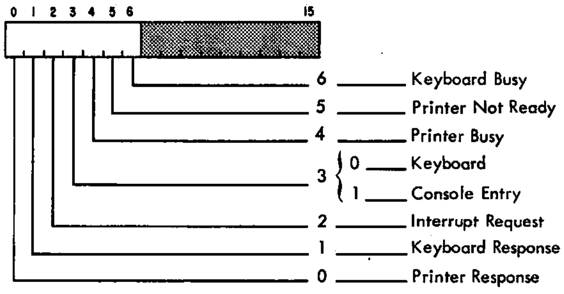


Figure 22. Keyboard/Console Printer Device Status Word

Printer-Response Interrupt (Bit 0 = 1): Each time a print or control function is completed, the console printer requests a level-4 interrupt. A sense-interrupt command is then needed to load the level-4 ILSW into the accumulator. Examination of this ILSW, for the interrupt under discussion, indicates that bit 1 = 1, which specifies a console interrupt. A sense device IOCC must then be executed for the console. This sense-device IOCC, addressing the console printer, loads the device status word into the accumulator. Bit 0, when equal to 1 in this device status word, indicates that the printer is available for the next print operation or control function.

Printer Busy (Bit 4 = 1): When bit 4 = 1, the console printer is printing a character or performing a control function. A write command should not be sent to the printer when the printer is busy. If a write command is sent to the printer when bit 4 (printer busy) is on, the data sent to the printer (either character or control data) is ignored, and no further indication is given to the program.

Bit 4 (printer busy) is on from the time data is sent to the printer until the printer has completed the required print or control action.

Printer Not Ready (Bit 5 = 1): The printer is not ready when bit 5 is on. The following two conditions must exist in order for the printer to be made ready:

1. The printer must be properly loaded with forms.
2. The printer must not be busy.

If the two preceding conditions are both met, bit 5 is at a value of 0 in the device status word.

Programming Considerations

Program routines for console-printer control are available in the 1130 Disk Monitor, Version 2, Programming System. Such programming for the console printer is not described here. Rather, this section provides a brief description of factors that should be considered in writing programs for the console printer.

Before any write operation is initiated for the console printer, its device status word should be examined. If bit 5 (printer not ready) is on, a write operation or control operation cannot be performed. If bit 5 is off, the printer is ready. If the printer is ready and an output print record is in core storage, the first write command can be issued. Both the address in the write IOCC and a data count (not in the IOCC, but kept at some desired location in core storage) must then be updated.

The data address in the IOCC must, at the beginning of the output operation, point to the first word of the output record. After the first word is sent, via a write command, the address must be incremented by one; the address then points to the next word in the output record. Incrementing must be done by separate program control; it is not automatic.

A desired data-count value can be used to end the operation (that is, branch out of the routine). The data count is decremented by one for each word sent to the printer by a write IOCC. This decrementing is not automatic but must be provided separately by the program. When the count equals 0, the last write IOCC has been executed and the operation can be ended. If the write operation is dependent upon operator input at the keyboard, the operation may also be ended by the operator pressing the EOF key. Programming must also be provided to handle this type of ending.

A possible general procedure for handling printing or control functions after the first print or control function is shown in Figure 23. The procedure shown is general in nature and is for informational purposes only; the procedure is not meant to represent any specific application for the printer.

After the first print or control function is performed, the printer interrupts the CPU (see Figure 23). For this interrupt, bit 1 = 1 in the level-4 interrupt-level status word. A sense-interrupt IOCC can be executed to load the interrupt level status word in the accumulator.

A sense-device IOCC can then be executed (it must address, with device-code 00001, the keyboard/console). As a result, the printer device status word is loaded into the accumulator. Because a printer-response interrupt has occurred, bit 0 = 1 in the device status word.

Finally, after determining that the console printer requires service, the program can issue a write IOCC to the console printer. The next bit pattern is then sent to the printer. The operation continues in a similar manner until the data count goes to zero (or the EOF key is pressed when printing is dependent upon a keyboard input operation.)

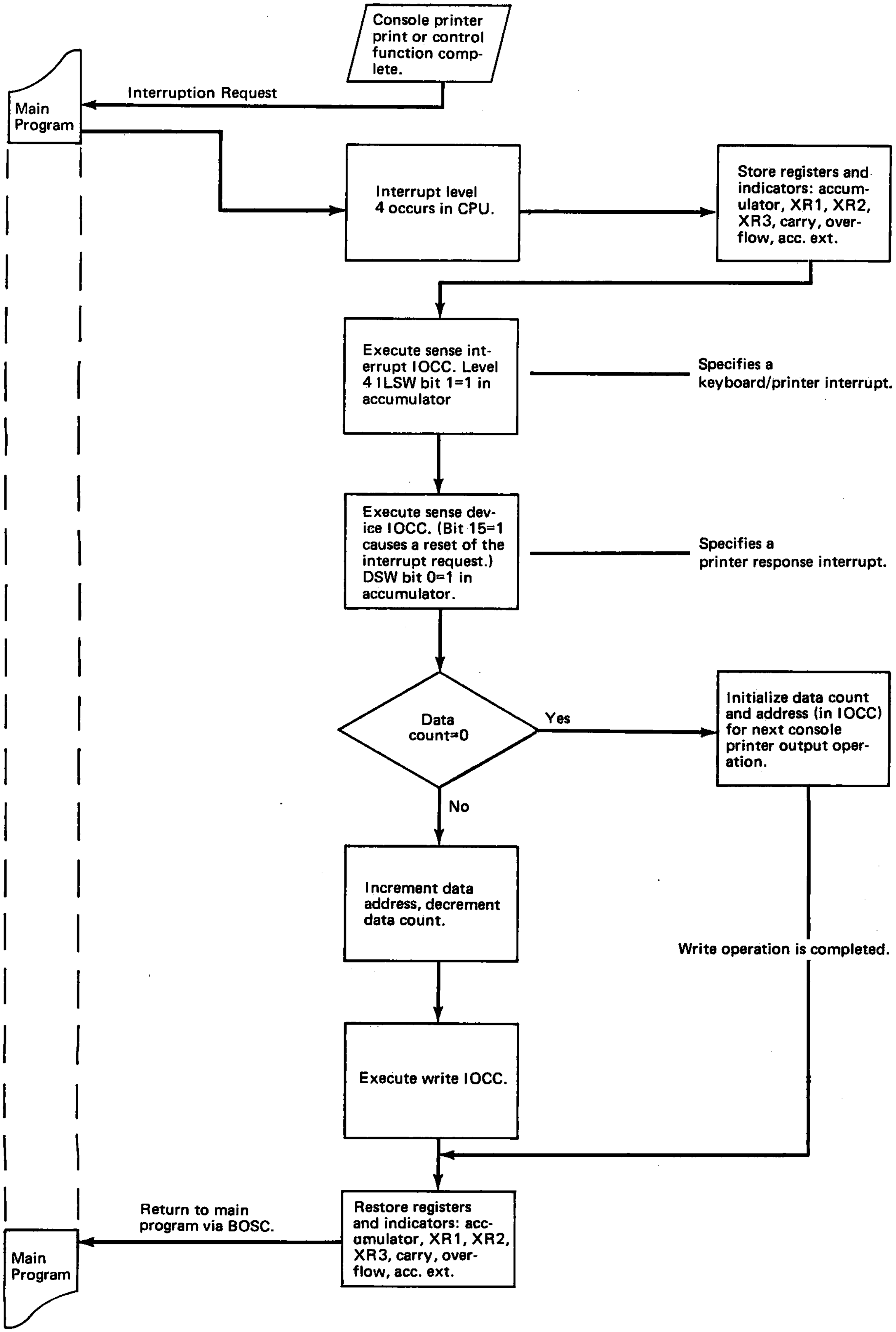


Figure 23. General Procedure for Console Print Operation (for a Printer Response Interrupt)

KEYBOARD FUNCTIONAL DESCRIPTION

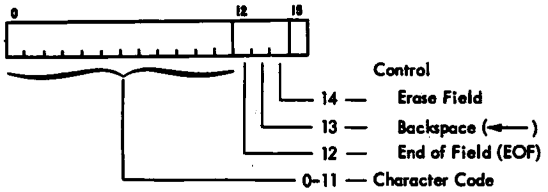
The input speed of the keyboard (Figure 24) is limited only by the speed of the operator.

Keyboard entries are not automatically printed unless the CPU is programmed to provide an output (of the entry) to the printer. The keyboard sends a bit pattern to the CPU for each key struck by the operator. The bit pattern is related to IBM card coding (see the character code chart in Appendix A). For example, operation of the A-character key results in the following bit pattern in the addressed core-storage word:

1001 0000 0000 0000

This bit pattern is identical to that received into core storage when the character A is read from a card column in card code by the 1442 Card Reader. Other bit patterns are listed in Appendix A. The bit definitions are:

Keyboard Data Format



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

The numbers next to the descriptions are the same as the numbers that point to the corresponding keys in Figure 24:

1. The REST KB (restore keyboard) key allows the operator to restore the keys if they should become locked.
2. The INT REQ (interrupt request) key initiates a keyboard restore and causes a level-4 interrupt in the CPU. Bit 1 in the ILSW for level 4 is on as a result of execution of a sense-interrupt IOCC. The DSW for the keyboard/console should then be examined by means of a sense-device IOCC. If the interrupt is due to operation of the interrupt request key, bit 2 is on in the DSW.
3. The EOF (end of field) key places a word containing only the 12-bit equal to 1 in a core-storage word. The program determines from the analysis of this word that no further characters are to be sent in the message.
4. The ←(backspace) key places a word containing only the 13-bit equal to 1 in a core-storage word. 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.

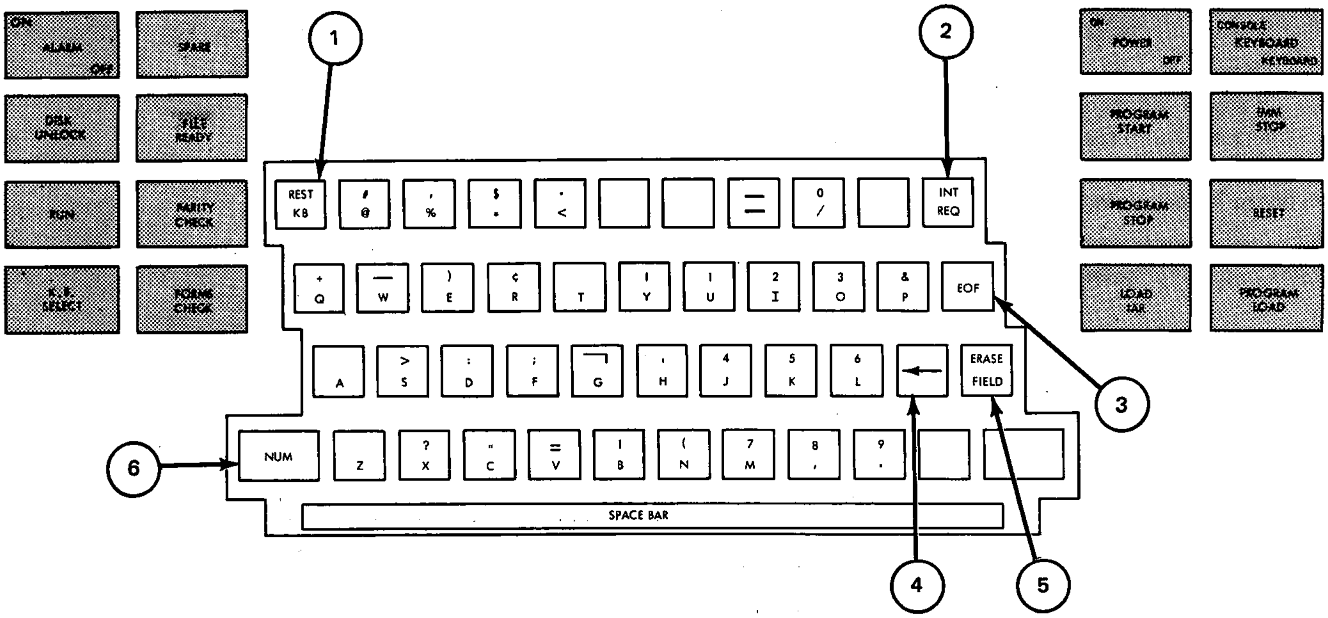


Figure 24. 1131 Console Keyboard

5. The ERASE FIELD key places a word containing only the 14-bit equal to 1 in a core-storage word. Analysis of this word allows the program to determine that the message being entered is to be deleted and replaced by a corrected message.
6. The NUM (numeric) key places the keyboard in the numeric mode. The numeric key must be held down continuously while numeric data is being entered.

Manual Start Operating Procedure

The following manual-start procedure is a typical use of the keyboard:

1. Press the interrupt-request key, which initiates a request interrupt and places the keyboard in restored status. When the CPU honors the request interrupt, the program must determine that the keyboard is the device that caused the interrupt. The program then must issue a control command to select the keyboard.
2. When the keyboard is selected, the select light is turned on to signal that a character can be entered.
3. If you then press a character key, the keyboard initiates a keyboard-response interrupt to the CPU. In response to the interrupt, the program should execute a read command. This command enters the character into core storage and removes the keyboard from the selected status. Before another character can be entered, the program must issue another control command to select the keyboard.

   Note: When the request is initiated by the program, the operation is basically the same but starts at the issuance of the control command to select the keyboard.

   If a read command is issued when the keyboard is not selected, no bits are entered into core storage.

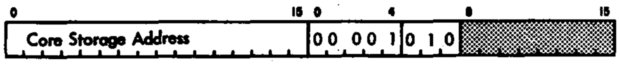
Keyboard Programming

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

I/O Control Commands (IOCC's)

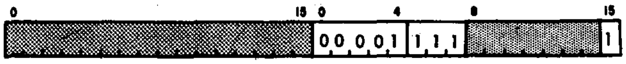
The keyboard is addressed by the same device code used by the console printer, 00001.

Read (010)



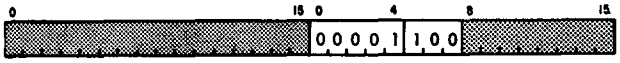
This command enters a single input character from the keyboard into the core-storage location specified by the address of the IOCC. A control command must have been previously executed to place the keyboard in a select status.

Sense Device (111)



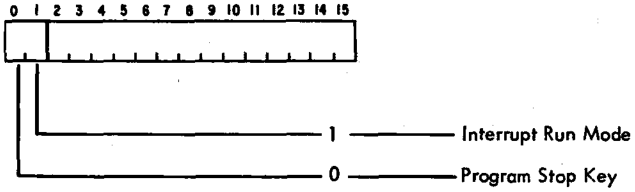
This command reads the keyboard/console printer device status word (Figure 22) into the ACC. Modifier bit 15 on specifies that all keyboard/console printer responses are to be reset.

Control (100)



This command places the keyboard in a select status so that a character can be entered. The read command resets the keyboard select status.

Special Keyboard Console Programming: Either the program stop key or the interrupt run mode causes an interrupt on level 5. The 0-bit in the ILSW is set on. After determining the interrupt is on level 5, the program must issue a sense device (with area code 7). The following DSW is presented to determine the cause of the interrupt.



DSW Indicators

Refer to Figure 22.

Keyboard Response (Interrupt): Bit 1 indicates an interrupt which signals that a character key has been pressed and that a character is ready to be entered into core storage.

Interrupt Request (Interrupt): Bit 2 indicates an interrupt which is initiated by the request key located on the keyboard.

Keyboard/Console Entry: Bit 3 indicates to the program the position of the keyboard/console switch. The two-position keyboard/console switch indicates the desired source of the console input data, either the keyboard or the console entry switches.

Keyboard Busy: Bit 6 indicates the keyboard is busy. This bit is on from the time an XIO Control is issued to the keyboard until the next XIO Read is issued to the keyboard. This bit is on any time the keyboard select light is on.

CONSOLE DISPLAY PANEL

The contents of the registers within the computer are displayed on the console panel (Figure 25) 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.

Indicator Displays

The Instruction Address Indicator represents the status of the 15 bits in the instruction address register. The instruction address register holds the address of the next sequential instruction.

The Storage Address Indicator represents the status of the 15 bits in the storage address register.

The storage address register contains the address of the last reference to a core storage word.

The Storage Buffer Indicator represents the status of the 16 bits in the storage buffer register.

The storage buffer register is the buffer between the central processing unit and core storage. Each word of data transferred into or transferred out of core storage passes through the storage buffer register.

The Arithmetic Factor Indicator represents the status of the 16 bits in the arithmetic factor register.

The arithmetic factor register holds one of the two operands during arithmetic and logical operations.

The Accumulator Indicator represents the status of the 16 bits in the accumulator register.

Data can be loaded into the accumulator register from core storage; conversely, data can be stored in core storage from the accumulator register. Data in the register can also be shifted to the right or to the left and can be manipulated by arithmetic and logical instructions. The accumulator register contains the binary number or expression resulting from an arithmetic or logical operation.

Error conditions, which are generally not-ready conditions or FORTRAN pause conditions, are indicated by the accumulator indicator.

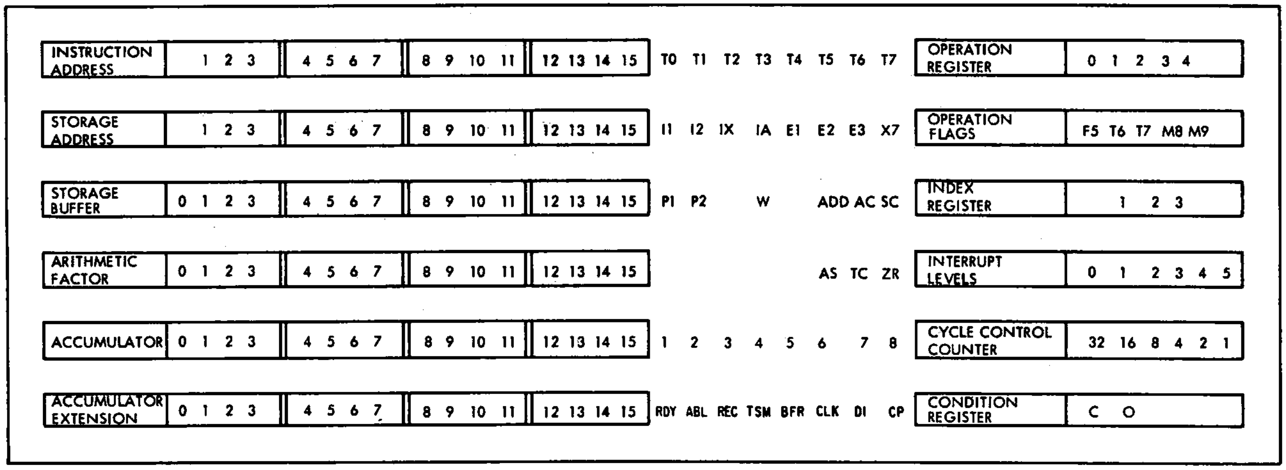


Figure 25. Console Panel

The Accumulator Extension Indicator represents the status of the 16 bits in the accumulator extension register. The accumulator extension register and the accumulator register are used as a 32-bit register. The 16-bit accumulator extension register is the low-order extension of the accumulator register. The accumulator extension register receives the data shifted to the right by the accumulator register or by a load double command code. The accumulator extension register is also used for multiplication and division operations and double-word arithmetic.

T0 through T7 Indicators represent the last clock step completed.

I1, I2, IX, IA, E1, E2, and E3 Cycle Indicators 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.

The X7 Indicator turns on when the cycle-steal clock is in X7; that is, stopped.

The P1 and P2 Indicators indicate the parity of the storage buffer register. P1 is on when bits 0-7 contain an even number of bits, and P2 is on when bits 8-15 contain an even number of bits.

The W (Wait) Indicator is on when the central processing unit is in a wait condition. The following conditions turn on the W indicator:

1. The machine executes a wait instruction or a FORTRAN pause.
2. The machine attempts to execute an operation code that is illegal, particularly a blank (b̸).

When a W condition is indicated, the central processing unit can be restarted at the next sequential instruction (after the wait instruction) by pressing the program start key.

The central processing unit is also restarted when an interrupt occurs. This restart is under control of the program and requires no operator intervention.

The requirements of the application being processed determine when the operator should press the program start key to resume program operation.

The ADD, AC, SC, AS, TC, and ZR Indicators indicate the status of the following functions: add, arithmetic control, shift control, accumulator sign, accumulator carry, and zero remainder.

1 through 8 Indicators are used by the customer engineer. Each lamp can be wired by a customer engineer to give a visual indication of any condition in the machine.

The next eight indicators are associated only with the synchronous communications adapter.

The RDY (Ready) Indicator turns on when the data set is ready.

The ABL (Enable) Indicator turns on when the program has enabled the adapter to respond to a ring indicator signal from the data set.

The REC (Receive) Indicator turns on when the receive trigger of the adapter is on.

The TSM (Transmit Mode) Indicator turns on when the adapter is in the transmit mode.

The BFR (Buffer Loaded) Indicator turns on when the buffer contains data.

The CLK (Clock) Indicator turns on when the receive clock is running.

The DI (Data In) Indicator turns on when the receive data line from the data set is at a zero or space level.

The CP (Character Phase) Indicator turns on when the adapter is operating in character phase.

The Operation Register Indicator indicates the operation in process when in single step (SS) mode or single machine cycle (SMC) mode. The indicator indicates the operation just completed when in any other mode.

Error conditions in the operation register generally occur when the machine has executed (1) a wait condition, (2) a FORTRAN pause, or (3) an invalid instruction such as all blanks.

Illegal instructions are as follows:

00000 10110 00111 01010
01111 01011 10111 11111

The Operation Flags Indicator indicates the status of the format, tag, and modifier bits of the instruction shown in the operation register.

The Index Register Indicator shows which one of the three index registers is being used.

The Interrupt Levels Indicator shows the interrupt level being serviced. The level indicator that is on aids in identifying the device that is being serviced by the interrupt subroutine.

The following indicators aid in identifying 1130 system devices:

1. 0 – 1442 Card Punch Model 5
   1442 Card Read Punch Models 6 and 7
2. 1 – 1132 Printer Models 1 and 2
   Synchronous communications adapter
3. 2*– Disk Drive
4. 3*– 1627 Plotter Models 1 and 2
   2250 Display Unit Model 4, or System/7
5. 4*– 1055 Paper Tape Punch Model 1
   1134 Paper Tape Reader
   1231 Optical Mark Page Reader Model 1
   1403 Printer Models 6 and 7
   1442 Card Punch Model 5
   1442 Card Read Punch Models 6 and 7
   2501 Card Reader
   Console interrupt request key
6. 5*– Program stop key
   Console mode switch

The Cycle Control Counter Indicator represents the binary value contained in the shift counter.

The Condition Register Indicator represents the status of the carry indicator (C) and the overflow indicator (O).

Mode Switch

The mode switch (Figure 26) selects one of seven operating modes:

The SS (Single Step) Setting with each depression and release of the start key, causes the 1131 clock to advance one step; for example, from T1 to T2.

The SMC (Single Memory Cycle) Setting, with each depression of the start key, causes the central processing unit to advance one machine cycle; for example, I1 to I2.

The INT RUN (Interrupt Run) Setting causes a level 5 interrupt after each mainline program instruction is completed. This setting is convenient for program trace routines.

The RUN (Program Run) Setting causes the 1131 to advance through its stored program when the start key is pressed.

The SI (Single Instruction) Setting causes the 1131 to interpret and execute a single instruction when the start key is pressed.

The DISP (Display Core Storage) Setting, after pressing the start key, displays (in the storage buffer register) the core storage word at the location specified by the address in the instruction address register and advances the instruction address register.

The LOAD (Load Core Storage) Setting, after pressing the start key, loads the data from the console entry switches into core storage at the location specified by the address in the instruction address register and advances the instruction address register. Pressing the LOAD IAR switch causes the data from the console entry switches to be loaded into the IAR.

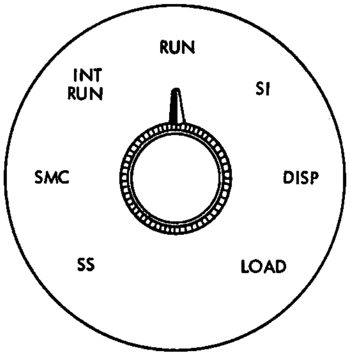


Figure 26. Console Mode Switch

---

* These interrupt levels may also have a Storage Access Channel (SAC) device.

CONSOLE ENTRY SWITCHES

These 16 toggle switches (Figure 27) 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 console entry switches (CES) by means of manual control, keyboard interrupt, or XIO instruction.

Manual Branching: This procedure allows the operator to begin processing from the instruction word located at any point in the program.

1. Set the mode switch to LOAD.
2. Set the console entry switches to the binary core storage address of the first instruction word to be executed.
3. Press the load IAR switch.
4. Set the mode switch to RUN.
5. Press the program start key.

Manual Entry: This procedure causes the bits set in the console entry switches to be loaded into the word at the core storage address in the instruction address register (IAR).

1. Set the mode switch to load
2. Set the CES to the binary core storage address where the data is to be stored.
3. Press the load IAR switch.

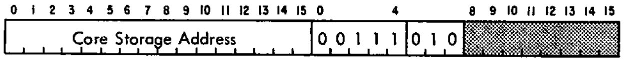
   Note: The parity indicators P1 and P2 may not show correct parity.

4. Set the data word in the CES.
5. Press the program 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. The subroutine must analyze the ILSW to determine the interrupting device (keyboard). The keyboard/console printer DSW is loaded into the ACC by a sense device command.
4. The DSW must be analyzed by the subroutine to determine the cause of the interrupt. DSW bit 2 = 1 indicates an operator interrupt request was initiated by the interrupt request key on the keyboard. DSW bit 3 = 1 indicates to the interrupt subroutine that the CES should be read by a read command.
5. Return to the mainline program is by the regular method of a BOSC instruction.

Read (010)



This command reads the settings of the CES. Each CES that is on causes a 1-bit to be placed in the corresponding location of the core storage word located at the address specified by the address field of the IOCC. The area code is 7 (00111).

CONSOLE FUNCTION LIGHTS AND SWITCHES

These lights and switches are located on both sides of the keyboard (Figure 28).

Function Lights

The Disk Unlock Light turns on when the disk cartridge can be removed from the disk drive.

The File Ready Light turns on when the disk storage is available for reading and writing.

The Run Light turns on when the central processing unit is operating and the meter is running.

The Parity Check Light turns on when a parity error is detected in either half of a word read out of core storage.

The KB Select Light turns on when an instruction requests input data from the keyboard.

The Forms Check Light turns on when the last form is detected by the console printer forms contact.

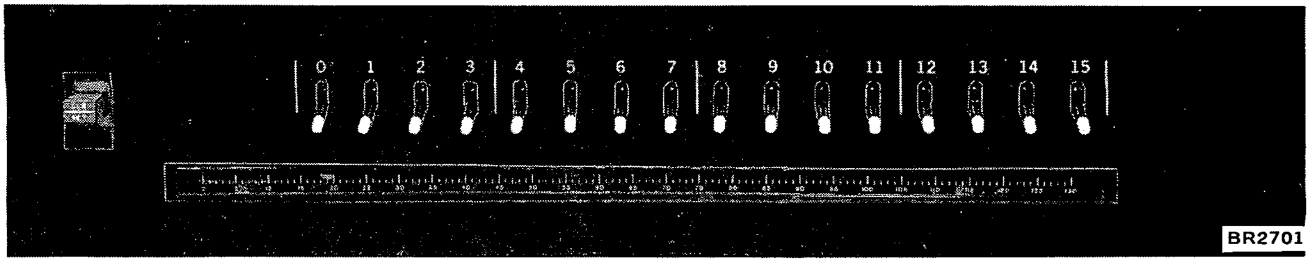


Figure 27. Console Entry Switches

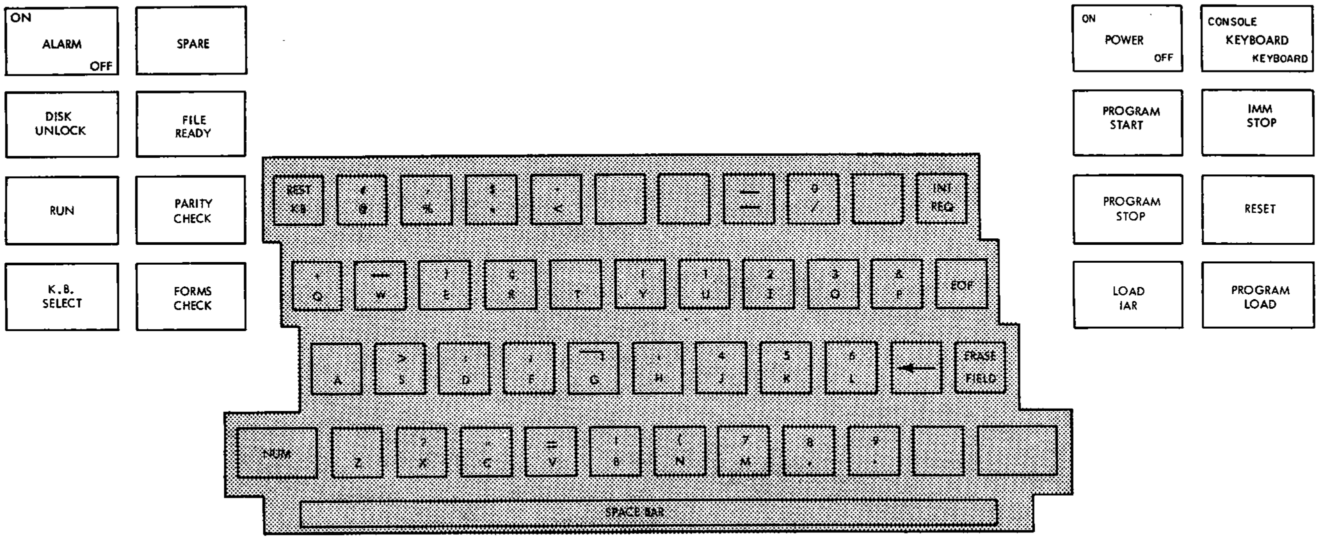


Figure 28. Console Function Lights and Switches

Function Switches

The Alarm On/Off Switch turns off the synchronous communications adapter (special feature) alarm in the 1131. The operator uses this switch to turn the alarm off only if the program should fail to turn it off.

The Emergency Pull Switch, as the name of the switch implies, should be initiated only under unusual circumstances. Once the emergency pull switch is pulled, it is mechanically locked so that system power cannot be brought up again until the customer engineer has reset this switch. All power – including that to all on-line input/output units – is dropped without regard to sequencing. Therefore, the contents of core storage may be partially destroyed during an emergency pull operation. The customer engineer should be contacted whenever system power cannot be maintained.

The Power On/Off Switch turns on the electrical power to the 1130 system.

The Keyboard Console/Keyboard Switch indicates (to the program) the desired source of the console input data – either the keyboard or the console entry switches.

The Program Start Switch causes the 1131 to take one clock step or one machine cycle (and to continue taking additional cycles), depending on the setting of the console mode switch. Eight clock steps complete one machine cycle, and one or more machine cycles complete an instruction.

The IMM (Immediate) Stop Switch causes an immediate stop of the processor interrupt, although the input/output devices will finish their present cycle. Data from the devices may be lost if they are operating when the IMM stop key is pressed. A complete program restart is normally required.

The Program Stop Switch causes a level-5 interrupt. The program must then provide the control to cycle down I/O devices and stop the CPU. If routines that service this interrupt are not in the program, loss of information may result if the program stop switch is operated. Operational and programming considerations are:

1. The program stop switch is pressed and a level-5 interrupt occurs.
2. Bit 0 = 1 in the console-keyboard device status word. This DSW is accessed by area code 00111.
3. A user-supplied wait loop is required to block mainline operations until the operator intervenes.
4. The interrupt program routine should allow the program to continue when the start switch is pressed.
5. If an overlapping (or malfunction) operation is in progress (such as printing or moving disk data), the program operation must be completed or data and operating system integrity can be destroyed. (The preceding factors must be considered in the program-stop interrupt routine.)

The Reset Switch resets all input/output and machine registers, cycle and control triggers, and status indicators.

The Load IAR (Instruction Address Register) Switch places the status of the 16 console entry switches in the instruction address register. The console mode switch must be set to LOAD.

The Program Load Switch loads the first card or paper tape record into core storage, beginning at 00000.

1131 CPU Usage Meter

This meter runs when either of the following conditions is present:

1. When the 1131 clock is running or
2. When any of the attached I/O equipment is operating to finish an instruction given by the 1131 program.

For the second condition the meter will run regardless of the state of the clock, and will stop when the I/O equipment has finished the particular operation the instruction specified.

A customer engineering meter and key lock switch are also provided in the 1131. When this key switch is turned on it activates the CE meter and deactivates the 1131 usage meter and all usage meters on the attached I/O equipment. This CE meter then records time in the same manner as specified for the usage meter. The purpose of the CE meter is to record system time during maintenance.

The 1131 usage meter provides the master time measurement of useful work done by the 1130 computing system. All I/O unit meters are interlocked by this meter running and cannot record time if it is stopped. No I/O meter can ever record more time than the 1131 usage meter.

Disk storage for the 1130 system offers random- or sequential-access data storage and consists of two units: a disk cartridge (or disk pack) and a disk storage drive. (Note that *disk cartridge* actually refers to the IBM 2315 Disk Cartridge and *disk pack* refers to the IBM 1316 Disk Pack. To eliminate dual references, *disk cartridge* will be used to refer to both the 2315 and 1316 units.)

The disk cartridge's storage medium is an oxide-coated disk (or disks) that can store data magnetically. The disk cartridge is mounted on a disk storage drive that rotates the disk(s) and contains the necessary electrical and mechanical components to record data on and read data from the disk cartridge. The disk cartridge can be removed from the disk drive and interchanged with another disk cartridge of the same type, allowing almost unlimited, off-line storage.

The 1130 system uses three types of disk storage drives: the single disk storage drive; the IBM 2310 Disk Storage; and the IBM 2311 Disk Storage Drive.

Two types of disk cartridges are used with the 1130 system: the IBM 2315 Disk Cartridge, and the IBM 1316 Disk Pack. The type of cartridge used depends on the type of disk storage drive installed on the system. The single disk storage drive and the 2310 drive use a 2315 Disk Cartridge; 2311 drives use a 1316 Disk Pack.

STORAGE CAPACITY

The amount of on-line disk storage capacity depends on the 1130 system configuration. Off-line storage is virtually unlimited because the disk cartridge is easily removed and interchanged with another cartridge.

The 1131 Models 2, 3, and 4, include a single disk storage drive capable of storing 512,000 sixteen-bit words.

Up to two 2310s (containing up to two disk storage drives each) or up to two 2311 Disk Storage Drives, Models 11 or 12, in any model combination, can be attached to the 1131 Models 1 (not 1A), 2, 3, and 5. (An 1130 system can have either 2310 drives or 2311 drives but not both.) Each drive within the 2310 is capable of storing 512,000 sixteen-bit words. Each 2311 Model 12 can store 1,536,000 sixteen-bit words; the model 11 can store 2,560,000 sixteen-bit words.

DISK CARTRIDGE

The IBM 2315 Disk Cartridge (Figure 29) is a single disk completely enclosed in a protective housing. The recording medium is an oxide-coated disk with two surfaces for the magnetic recording of data. When the cartridge is mounted on a storage drive, the disk rotates at 1,500 revolutions per minute.

The IBM 1316 Disk Pack (Figure 30) contains 6 disks mounted on a vertical shaft. The top surface of the top disk and the bottom surface of the bottom disk cannot be used for recording data, which leaves 10 recording surfaces.

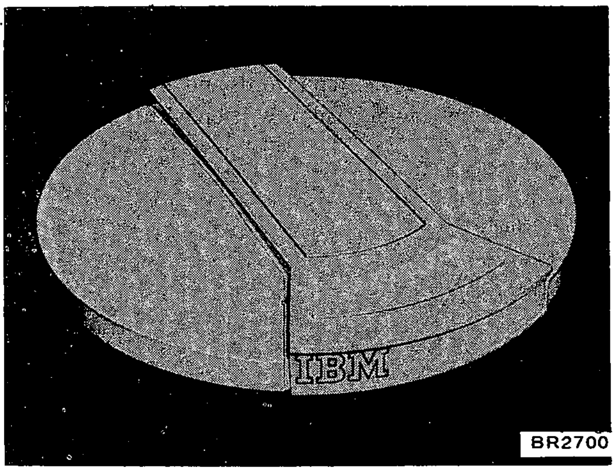


Figure 29. 2315 Disk Cartridge

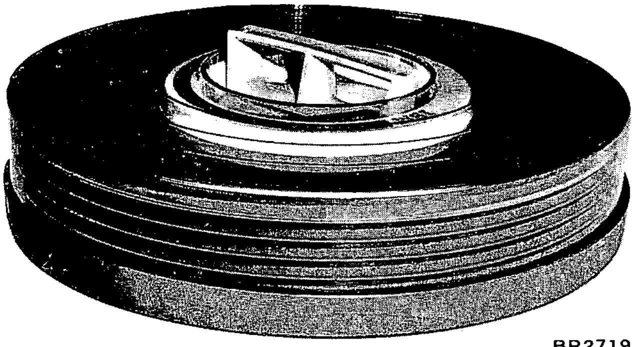
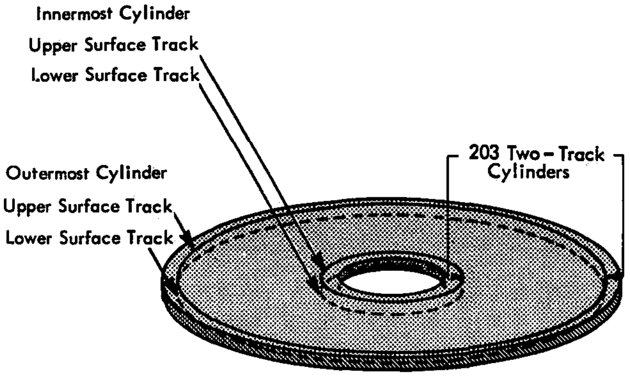


Figure 30. IBM 1316 Disk Pack

Data Organization

The disk access mechanism, located in the disk drive, is moved back and forth by programmed commands and can be placed in any one of 203 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. The 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 31 shows the innermost and outermost cylinders of two tracks each. To complete the picture, the 201 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 32. 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.



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 31. 2315 Disk Storage Cylinder Schematic

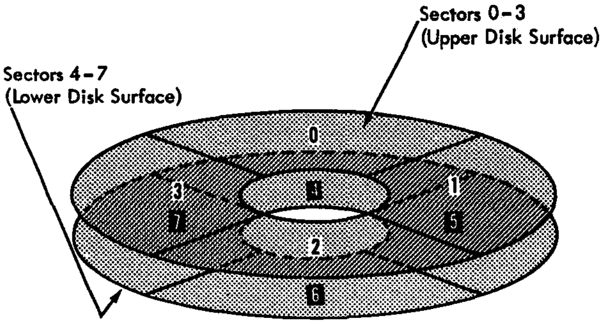


Figure 32. Disk Storage Sector Numbers

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 the cylinder sector number. Therefore, the first word of the sector cannot be used by the programmer if the assembler program or other components of the monitor system are to be used.

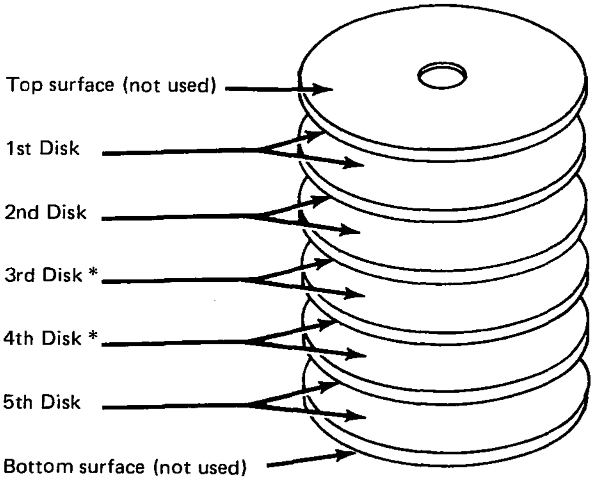
A disk storage word comprises 16 data bits and four check and space bits.

Figure 33 shows the organizational components of disk storage. Note that capacities are based on the 320-word sector; also, the number of cylinders is 200 rather than 203. Three cylinders (24 sectors) are provided as alternates to be used if a surface is defective.

The 1316 Disk Pack has 10 recording surfaces which the 1130 disk monitor system program considers to be either three or five disks, depending on the model of 2311 Disk Storage Drive the 1316 Disk Pack is used on. Model 11 drives are considered to be five disks; Model 12 drives are considered to be three disks. The manner in which the 1316 Disk Pack is arranged for both drives is shown in Figure 34. Note that disks 3 and 4 are not addressed when used on Model 12 drives. Other than having four additional disks, the 1316 Disk Pack has the same sector, track, and cylinder format as that of the 2315 Disk Cartridge.

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

Figure 33. Disk Storage Data Organization



*Not used on the 2311 Disk Storage Drive Model 12.

Figure 34. Logical Disk Format of the 1316 Disk Pack

Data Checking

Data is checked on each data transfer between core storage and disk storage. When writing on disk storage, the number of 1-bits in each word is effectively divided by four, as the word is shifted out of the file data register in the disk storage attachment, by incrementing a two-position counter. The number of bits necessary to make the division even (modulo 4) is added to the end of the word as shown in the following chart.

Number Of Data Bits Written 0-15	0	1	2	3
	4	5	6	7
	8	9	10	11
	12	13	14	15
Modulo 4 Counter	00	01	10	11
Check Bits				
16	0	1	1	1
17	0	1	1	0
18	0	1	0	0
19	0	0	0	0

The modulo 4 check is performed as each word is read from disk storage. A word that is not modulo 4 causes the data error bit to be set in the disk storage DSW.

The data checking provided in write operations only ensures that data was transferred correctly to the disk drive; however, several factors–chipped or dirty disk, etc.– could keep data from actually being recorded correctly on the disk surface. For this reason, the programmer is encouraged to always perform a read-check operation immediately after writing and while source data is still available.

SINGLE DISK STORAGE DRIVE

One single disk storage drive is contained in the CPU cabinet (1131 Model 2, Model 3, and Model 4) and is connected to the CPU by a high-speed data channel. It is composed of two components: the disk cartridge, and the drive assembly and access mechanism.

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.

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

Timing

Timing considerations of single disk storage operation involve three elements: access time, reading and writing data, and the time during which the CPU is tied up. (See the section entitled Overlapping Input/Output Operations and Throughput Considerations.)

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 stabilization period (22.5 ms) that follows the last incremental movement, a read or write command can be given and will be started at the end of the stabilization period.

Access time (ms) = 7.5N+T
Where T = 22.5 (±2.5)ms

Read/Write: Reading or writing of data in disk storage is at the rate of 27.8 us. per word. Average rotational delay is 20 ms, based on 1,500 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 235 us. between sectors; the interval is increased by 27.8 us. for each word less than 321 read or written.

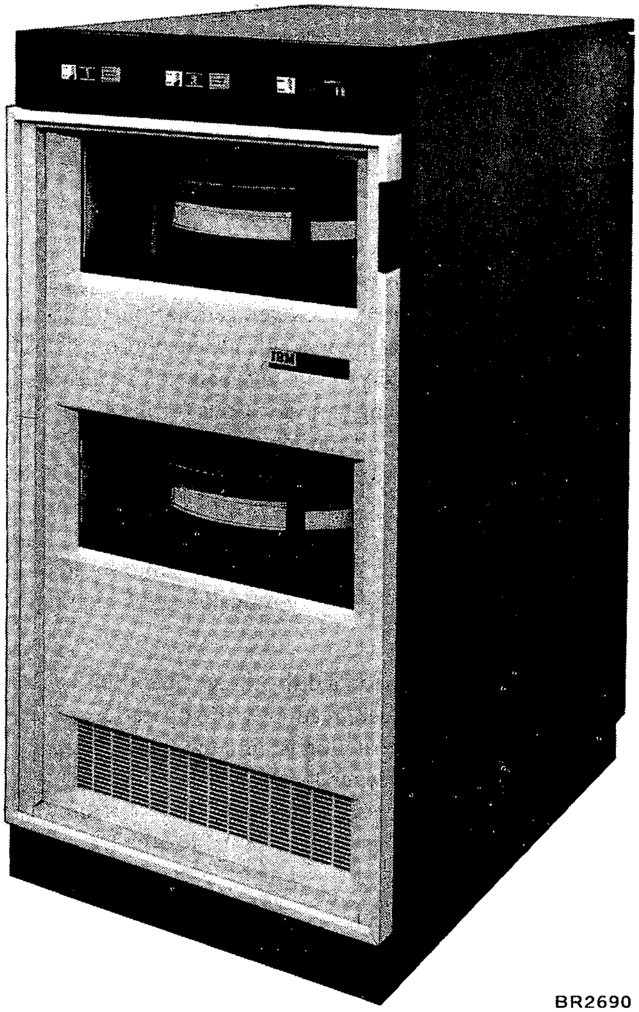
A full cylinder of eight 321-word sectors can be read or written in 100 ms because the rotational delay is required for only the first sector.

CPU Time: An interrupt in a disk storage operation occurs only at the end of the seek, read, or write operations. 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 us. for the transmission of the next word.

IBM 2310 DISK STORAGE

The IBM 2310 Disk Storage Model B, (Figure 35) provides additional random-access storage capabilities for the 1130. The 2310 Model B1 contains one single disk storage drive, whereas the 2310 Model B2 contains two single disk storage drives. A maximum of two 2310 Model B2s (containing 4 drives) may be attached to the 1130 via a channel multiplexer in the 1133. If two 2310s are attached, at least one must be a model B2.

The functional description – that is, capacity, data organization, data checking, access mechanism, timing, etc. – is the same as for the single disk storage drive just described.



BR2690

Figure 35. IBM 2310 Disk Storage Model B

IBM 2311 DISK STORAGE DRIVE

The IBM 2311 Disk Storage Drive (Figure 36), provides the 1130 system with higher speed and greater storage capacity than that of the 2310 Disk Storage. The 2311 Disk Storage Drive is available in two models: Model 11 and Model 12. The 2311 Model 11 can store 2,560,000 sixteen-bit words (the equivalent storage of five 2310 drives); the 2311 Model 12 can store 1,536,000 sixteen-bit words (the equivalent storage of three 2310 drives). Note that the 1130 system can have either 2310 drives or 2311 drives installed, but not both.

The 2311 Disk Storage Drives uses the 1316 Disk Pack for its storage device. The adapter that attaches the drives to the 1130 system is designed to make each 1316 Disk Pack simulate either 3 or 5 single disk drives, depending on which model of 2311 drive uses the disk pack. The 2311 Model 11 simulates 5 single disk drives and the model 12 simulates 3 single disk drives.

BR2679

Figure 36. IBM 2311 Disk Storage Drive

Access Mechanism

The access mechanism for the 2311 drives is a single, multiple-head unit. Therefore, when the access mechanism is at a specified cylinder for one disk, it is at the same cylinder for all of the other disks on that drive.

No access motion occurs at the time a control command is issued to a drive; instead, the motion occurs when a read or write command is issued to the drive and the access mechanism is not at the specified cylinder. Cylinders do not carry an identifying number. Therefore, the program must maintain the necessary information relative to the position of the access mechanism. Programs for the 1130 system regard each disk on a drive as independent of all other disks on the same drive. For this reason, it is necessary only to maintain relative position information for each disk without regard to the actual cylinder location.

A control command can be issued only when the 2311 is ready and not busy. The 2311 is busy to all commands when a read or write command is being executed, so access mechanism motion and data transfer cannot be overlapped on the same drive.

Timing

Reading or writing data on the disk is accomplished at a rate of 16.0 microseconds per word, with an average rotational delay of 12.5 milliseconds. Maximum rotational delay is 25 milliseconds.

Average access time is 75 milliseconds, minimum time is 25 milliseconds, and maximum time is 135 milliseconds.

PROGRAMMING DISK STORAGE

The disk storage drives attached to the 1130 system are controlled by I/O control commands (IOCC) provided by the program in CPU core storage. Each of the installed drives responds to its assigned device code. Each of the disks for the 2311 Disk Storage Drive is individually addressed. This requires the use of modifier bits to address all disks on the second drive of a two-drive system. The five-bit device codes and the modifier bits required for each drive and disk are as follows:

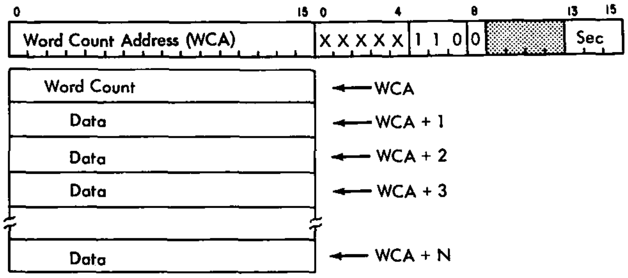
Device Code	*Modifier Bits*			*Device Location*
	9	*10*	*11*	
00100	0	0	0	CPU
10001	0	0	0	2310 drive 1 or 2311 drive 1, disk 1
10010	0	0	0	2310 drive 2 or 2311 drive 1, disk 2
10011	0	0	0	2310 drive 3 or 2310 drive 1, disk 3
10100	0	0	0	2310 drive 4 or 2311 drive 1, disk 4
10110	0	0	0	2311 drive 1, disk 5
10111	0	0	0	2311 drive 2, disk 1
10111	0	0	1	2311 drive 2, disk 2
10111	0	1	0	2311 drive 2, disk 3
10111	0	1	1	2311 drive 2, disk 4
10111	1	0	0	2311 drive 2, disk 5

In the illustrations of IOCC's that follow, bits 9, 10, and 11 are not shown.

I/O Control Commands

Initiate Read (110)

This command causes the number of words specified by the word count to be read from the disk storage drive identified by the device code. The sector to be read is identified by modifier bits 13-15.



The address word of the command contains the word count address (WCA), and modifier bit 8 determines whether the command is a read command (0) or a read-check command (1).

A full sector, 321 words, is the maximum transmission with one command. Succeeding sectors, or parts of sectors, require the initiate read command for each one.

An operation-complete interrupt occurs when the number of words in the word count has been transmitted.

Note for 2311 disk drives only: If the access mechanism in the drive is not located at the cylinder specified by the arm register, an access operation will occur before data transfer begins. The arm register is specified by the area code which is the same as for the disk addressed. The address in the arm register is updated by a control instruction.

Read (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, 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 (Bit 8 = 1): Data is read from disk storage, as in the read command, and the number of bits of each word is checked for modulo 4. Unlike the read command, data is not transferred into core storage. Therefore, a storage area does not need to be provided, and no time demand is placed on the CPU. Once the read-check command has been started, the modulo 4 checking is independent of the CPU. If the number of bits in a word, including check bits, is not even when divided by four, the data error indicator is set in the disk storage DSW. Neither disk storage nor core storage is affected by the read-check command.

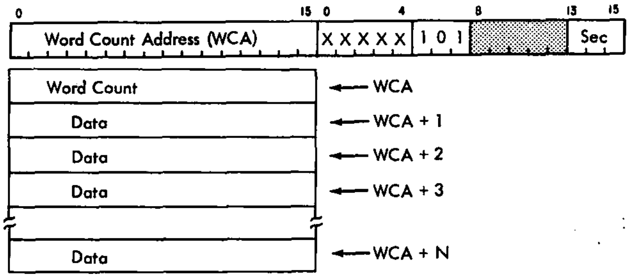
To achieve the maximum level of performance that the disk storage is capable of providing, the program should provide error recovery procedures. Errors are often due to temporary conditions which can be successfully recovered by re-executing the read or write command.

A write command which does not write correctly because of temporary or intermittent conditions can be detected by immediately verifying the data just written. In this way, any such "soft" write error can be corrected while the data is still available in core storage. If this write checking procedure is not followed, the "soft" write error becomes a "hard" error, which can be corrected only by reconstruction or adjustment. In almost all cases, permanent data files should be verified as soon as written, while for transient or work files, verification may not always be required. The programmer should weigh the possible reconstruction time versus the time consumed in write verification before deciding not to verify write data.

An initiate read with a word count of zero should not be used. Recovery requires a console DC reset.

Initiate Write (101)

This command 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 disk storage drive to be used is designated by the device code, bits 0-4.



The address word of the command contains the address of the word count. The data is transmitted from core storage location WCA + 1 and ascending addresses until the number of words specified by the word count has been written. If the word count is less than 321 words, the remainder is written with all 0's. Succeeding sectors, or parts of sectors, require an initiate write command for each one.

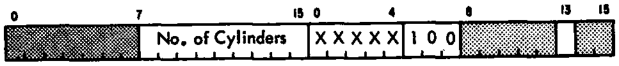
An operation-complete interrupt occurs when the number of words in the word count has been transferred.

An initiate write command should be followed immediately by a read-check command to verify that data can be read correctly.

An initiate write command with a word count of zero should not be used. Recovery requires a console DC reset.

Note for 2311 disk drives only: If the access mechanism in the drive is not located at the cylinder specified by the arm register, an access operation will occur before data transfer begins. The arm register is specified by the area code which is the same as for the disk addressed. The address in the arm register is updated by a control instruction.

Control (100)



For 2310 Disk Storage: This command causes the access mechanism of the drive designated by the device code to move in increments of two cylinders for the number of cylinders specified by the address word of the command. 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.

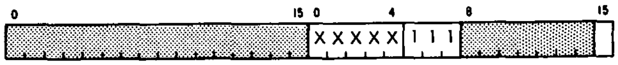
Note: Cylinders do not carry an identifying number. It is the responsibility of the program, therefore, to maintain the necessary information relative to the position of the access mechanism. A control command which specifies an access motion of zero cylinders is treated as a no-operation and does *not* result in an operation-complete interrupt.

For 2311 Disk Storage Drives: This command causes the hardware counter associated with each simulated disk (addressed by the specific area codes) to be updated by the number of cylinders specified by the address word of the instruction. Modifier bit 13 = 1 causes the number of cylinders specified to be subtracted from the address arm register and the result to be placed back into the same register. Modifier bit 13 = 0 causes a similar addition to occur.

No arm movement takes place in the drive during a control command. Accessing, if necessary, occurs when the initiate read or initiate write command is issued and before data transfer begins. An operation-complete interrupt is requested at the end of the control command.

Sense Device (111)

This command causes the device status word (Figure 37) of the disk storage identified by the device code to be read into the ACC.



Operation complete and data error (except select and unsafe) indicators are reset if modifier bit 15 is a 1.

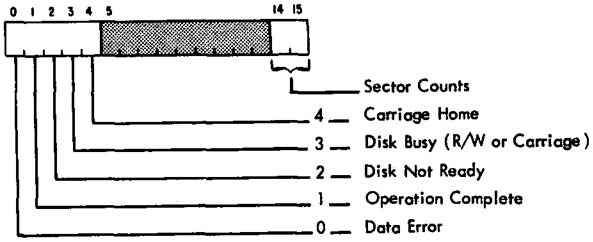


Figure 37. Disk Storage Device Status Word

DSW Indicators

Operation Complete (Interrupt): This is the only interrupt associated with disk storage, and is turned on at the end of a read, read-check, write, or control (access) operation (where there is access movement). It also occurs if the disk storage is in a read, read-check, or write operation at the leading edge of a sector pulse; this occurs if the word count specified is greater than 321.

Data Error: This indicator is turned on when:

1. A modulo 4 error is detected during a read, read-check, or write operation.
2. The disk storage is in a read or write mode at the leading edge of a sector pulse.
3. A seek-incomplete signal is received from the 2311.
4. A write select error has occurred in the disk storage drive.
5. The power unsafe latch is set in the attachment.

Conditions 1, 2, and 3 are turned off by a sense device command with modifier bit 15 set to 1. Conditions 4 and 5 are reset by turning off the disk storage drive, allowing for the cartridge unlock indicator to light, turning on the drive, and waiting until the disk ready indicator (heads loaded for the 2310 or 2311) to light. Disk operation can resume after this sequence of events.

Disk Not Ready: This indicator is turned on with disk not ready or busy or disabled or off-line or power unsafe latch set. Also included in the disk not ready is the write select error, which can be a result of power unsafe or write select. (Bit 0 and bit 2 will be turned on.)

Disk Busy (R/W or Carriage): This indicator is on during execution of a disk storage command. It turns off when the operation is completed.

Carriage Home: This indicator is on when the access mechanism is at the home position (cylinder 000).

Sector Count: These bits represent the sector number of the next available sector to be used for reading or writing.

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

If an error is detected by the CPU circuitry, the following procedure should 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.

Note: IBM-supplied disk subroutines perform standard error recovery procedures.

Usage Meter

The usage meter for the 2310 or 2311 disk storage drives will run when the following conditions are present:

1. The 1133 is in an enabled status.
2. The disk drive enable/disable switch is in the enable position. The CPU must be stopped when the switch is changed from one position to another to affect the enable/disable status.
3. The 1131 usage meter is running. The disk drive meter will run simultaneously with the 1131 meter until the status is disabled or the 1133 status is disabled.

Eighty-column punched card input and output is provided to the 1130 system by the IBM 1442 Card Read Punch, Model 5, 6, or 7, and/or the IBM 2501 Card Reader.

IBM 1442 Card Read Punch

The IBM 1442 Card Read Punch (Figure 38), Model 6 or Model 7, provides both card input and card output for the 1130. The 1442 Model 5 is a card punch only and is considered the companion unit to the 2501 to provide a separate card path for card input and output. However, a model 6 or 7 may be installed with the 2501 in place of a model 5; in this case the 2501 should be considered the primary input unit. Functionally, the 1442 model 5 has the same punching characteristics as the 1442 model 7.

The 1442 is a single unit that processes cards serially, column by column, from a single supply hopper. All cards first pass the read station (model 6 and 7), then the punch station. This permits each card to be read, punched, or read and punched. Reading and punching cannot occur simultaneously – that is, one card cannot be punched while the following card is being read – because the reading and punching rates are different.

Maximum machine speeds are:

Card Reading

Model	Cards per minute
6	300
7	400

Card Punching

Model	Columns per second
5	160
6	80
7	160

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, model 6 (or 25 milliseconds, model 7) after the operation-complete 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 for model 6 and 375 cards per minute for model 7.

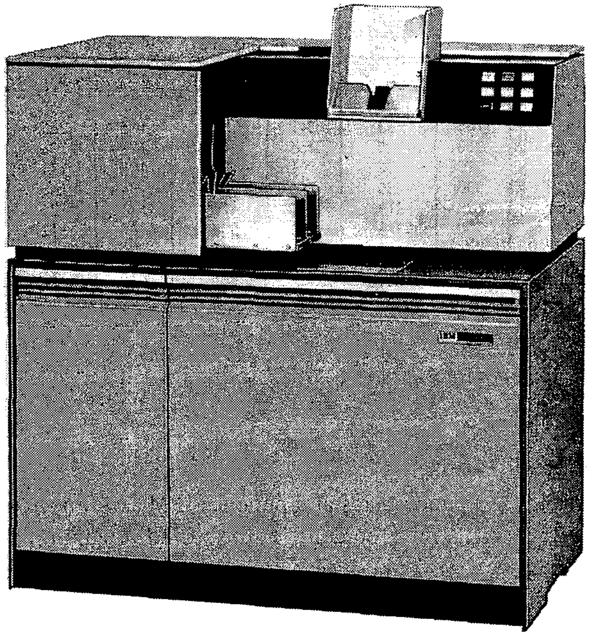

BR2684

Figure 38. IBM 1442 Card Read Punch

Punching rates depend on the position of the card when the last column has been spaced or punched. The punching speed ranges are:

Model 6 – 49 cpm to 262 cpm
Model 5 and 7 – 91 cpm to 355 cpm

The approximate time required to process a single card is:

Model 6 – 216 ms + 12.5 ms per card column spaced or punched
Model 5 and 7 – 163 ms + 6.25 ms per card column spaced or punched

The following table shows the approximate punch cycle times and cards-per-minute rates based upon the last column punched.

Last Column Punched	Punch Time (ms)		Total Punch Cycle Time (ms)		Cards per Minute	
	Model 6	Model 5&7	Model 6	Model 5&7	Model 6	Model 5&7
1	13	6	229	169	262	355
10	125	63	341	226	176	265
20	250	125	466	288	127	208
30	375	188	591	351	102	171
40	500	250	716	413	84	145
50	625	313	841	476	71	126
60	750	375	966	538	62	112
70	875	438	1091	601	55	100
80	1000	500	1216	663	49	91

Data Coding

The card read punch reads and punches IBM card image only. Code translation must be done by the stored program. As shown in Figure 39, the twelve rows (12–9) in a card column correspond to the 0–11 bits, respectively, of a core storage word. Bits 12 through 15 are reset to zero. A 1-bit represents a punched hole; a 0-bit represents a card position not punched. Thus, the word in Figure 39 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 initated by pressing the program load key on the 1130 console. In the load mode data is split (Figure 40) as it enters core storage to form the load program.

Card Feeding

An initial feed cycle results when the 1442 start key is pressed; this feeds the first card into position at the read station (sense station – 1442 Model 5). The initial feed cycle places the 1442 in a ready condition, which is necessary before reading or punching may begin.

A constant-speed drive moves the cards through the serial path during a feed cycle. A feed cycle is initiated by a control command with modifier bits designating feed cycle, start read, or start punch. The feed cycle does three things:

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

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. The last card indicator in the DSW is turned on if the card read is the last card in the deck.

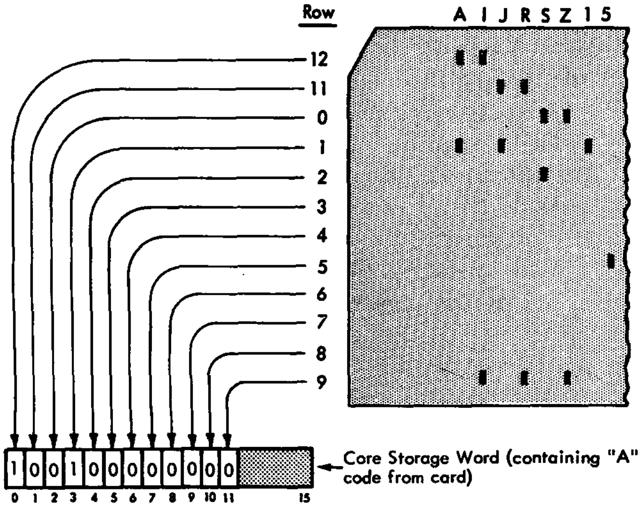


Figure 39. Normal Mode Read

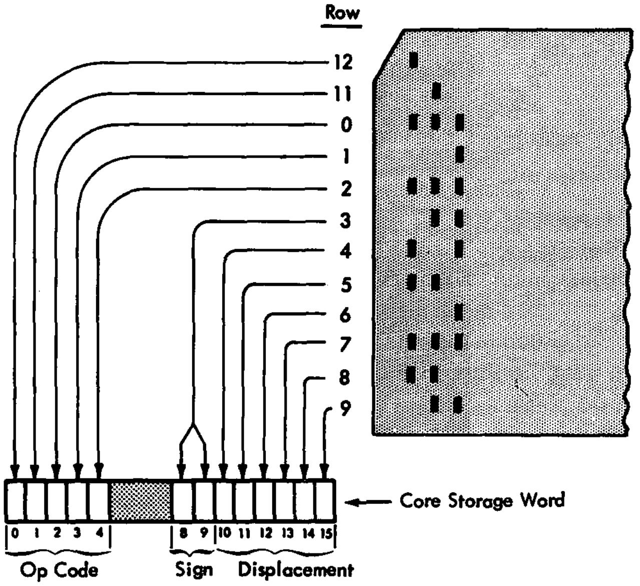


Figure 40. Load Mode Read

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, that column is punched and the card is moved to the next column. An operation complete interrupt is given. No more punch response interrupts are given. No further punching can take place on the card.

Failure to have an end-punch bit 12 results in more than 80 columns being punched. No indication of this programming error is presented in the DSW.

A feed cycle is necessary to eject a punched card to the stacker and can be initiated by either of the three control commands: feed cycle, start read, or start punch.

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 start read.

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 storage positions beginning at 00000, then causes the CPU to go to position 00000 for its next instruction.

Last Card Sequence

The last-card indicator is turned on in the 1442 DSW when the last card passes the read station. The program determines when to enter the last-card sequence by testing the indicator.

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 process the last card.

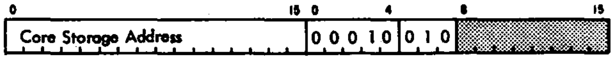
PROGRAMMING

The 1442 operates under direct program control of the CPU.

I/O Control Commands

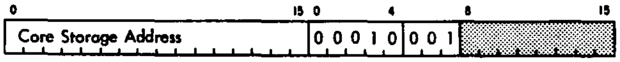
The card read punch is addressed by the 5-bit device code, 00010.

Read (010)



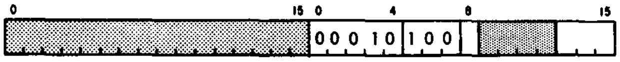
This command causes a card column image to be entered from the card reader into the core storage location specified by the address.

Write (001)



This command causes the data in the core storage location specified by the address of the IOCC to be punched as a column of the card.

Control (100)



This command causes the 1442 to perform the function specified by the modifier.

Modifier bits that have significance are:

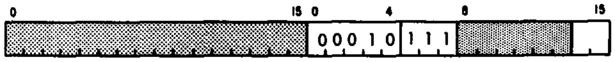
Bit 14	*Feed Cycle* – causes all cards in the feed path to advance one station. Cards pass through the read and punch stations without being processed.
Bit 13	*Start Read* – causes the card to move through the read station. As each column is read and checked, the card read punch initiates a read column 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 being read.

Bit 8 *Stacker Select* – (model 6 or 7 only) causes the card leaving the punch area to enter the alternate stacker. This control applies only to the next card leaving the punch station after this command has been issued. Selection of the desired card will not occur if the stacker select command is given while the card is still being processed at the read station.

Modifier bits 13, 14, and 15 of this control command should not be used in combination with each other.

Sense Device (111)



This command directs the 1442 to place its device status word (Figure 41) into the ACC. Modifier bit 15 on resets responses for level 0; modifier bit 14 on resets responses for level 4.

DSW Indicators

The three interrupts associated with the 1442 are divided into two groups. The sense interrupt (011) command causes the active ILSW to be loaded into the ACC.

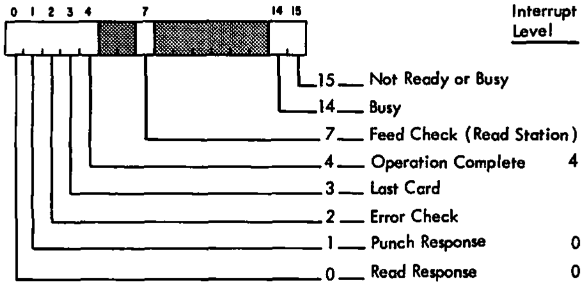


Figure 41. 1442 Device Status Word

Level 0 Interrupt Indicator

Read Response (Interrupt): This indicates an interrupt which signals that a column of data is ready to be entered into core storage. This interrupt request must be serviced within 800 us. for the 1442 model 6 and 700 us. for the 1442 model 7. Time from the start read to the first read column request interrupt is 28.4 ms for the model 6 and 23.8 ms for the model 7.

Punch Response (Interrupt): This indicates an interrupt which signals that a column of data must be transferred from the CPU within 300 us. Time from the start punch command to the first punch column response interrupt varies from 1.22 ms to 12.5 ms on the model 6 and 1.56 ms to 6.25 ms on the model 5 and 7.

Level 4 Interrupt Indicator

Operation Complete (Interrupt): This indicates an interrupt which occurs after a card has been processed. For reading, 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.

For punching, this interrupt occurs after the last column to be punched has been punched and checked and the punch drive has stopped. This occurs 12.5 ms after the terminating end-punch has been detected for the model 6 and 6.25 ms after the terminating end-punch for the model 5 and 7.

The operation complete interrupt is forced if a hopper check, feed check, transport error, or feed clutch error occurs while the 1442 is busy. This interrupt is also forced by a read registration check or punch check. No subsequent reading or punching can be done in the card that caused the error. In most cases, intervention by the operator is necessary to clear the error condition before card processing may resume.

There is no time limit on the request for service of the operation complete interrupt. However, to maintain rated speed, the model 5 and 7 must be serviced within 25 ms; the model 6 must be serviced within 35 ms if reading and 25 ms if punching.

Non-Interrupt Indicators

Not Ready: This indicator shows that the 1442 is either busy or not ready. When the 1442 is not ready, manual intervention is required. The following conditions must be met to place the 1442 in a ready condition.

1. Power on.
2. Card registered at the read (sense for Model 5) station (initially).
3. Cards in hopper or last-card sequence in progress.
4. Stacker not full.
5. Feed-check light off (no card jam or feed failure).
6. If the stop key has been pressed, the start key must have been subsequently pressed.
7. Chip box not full or removed.

Busy: This indicator shows that a command cannot be started 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. This indicator will be on when the operation complete interrupt occurs.

Error Check: Indicates that any of several error conditions exist on the 1442. Error conditions such as card feed failure are indicated by lamps on the 1442 console.

Programming Note: The error indicator does not turn 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, the error forces an operation complete interrupt although no reading or writing has taken place. A start punch requiring an automatic feed cycle is treated as two operations: (1) feed cycle, and (2) punch operation.

1442 Usage Meter

This meter will run when both of the following conditions are present:

1. The unit is selected for operation by program control.
2. The 1131 usage meter is running.

The meter will run simultaneously with the 1131 meter until either a program controlled stop or manual nonprocess runout is performed on the machine.

IBM 2501 Card Reader

The IBM 2501 Card Reader (Figure 42), Model A1 and Model A2, provides card input for the IBM 1130 Computing System. Card reading is under direct program control.

Functional Description

The IBM 2501 model A1 reads cards at a maximum rate of 600 cards per minute (cpm); the model A2 reads at a maximum rate of 1000 cpm.

Cards are read serially – that is, column by column – beginning with column 1. Each column is read twice and the two readings are compared to check reading accuracy. Thus, off-punched and mispositioned cards are detected.

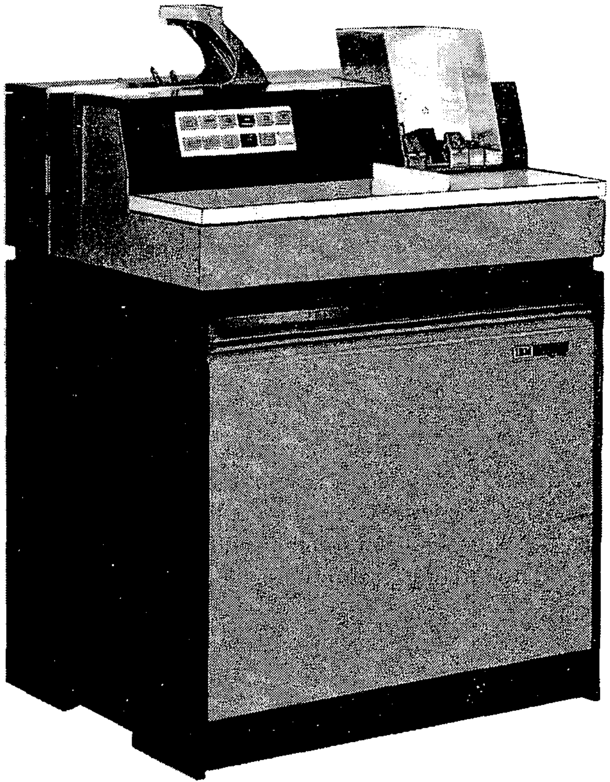


Figure 42. IBM 2501 Card Reader

Data Coding

The 2501 reads punched cards in card image only. Any code translation required must be done by the stored program in the CPU. As shown in Figure 39, the twelve rows (12–9) in a card column correspond to the 0–11 bits, respectively, of a core storage word.

A special load mode is initiated by pressing the program load key on the 1130 console. In the load mode, data is split as it enters core storage to form the load program. Refer to Figure 40.

Card Feeding

After the initial feed cycle (run in), card reading may begin. Card feeding is initiated by an initiate read command. This command causes the card to begin moving. If the data is to be ignored (as in card feeding), the word count must be zero.

Card movement is as follows:

1. The card at the read station moves through the read station to the stacker.
2. A card moves from the hopper to the read station.

When the hopper is emptied, the 2501 leaves the ready condition. The operator may reload the hopper and press the start key to continue processing or the operator may press the start key without reloading the hopper to initiate the last card sequence.

Program Load

Program load may be initiated by pressing the program load key on the 1131 console. This causes the load card data to be loaded into the first 80 core storage locations. After the card has been loaded the instruction address register is reset to 00000 and the CPU goes to this address for its next instruction.

Card Reading

An initiate read (110) command causes the card to be read in one continuous motion. The number of columns actually transferred to core storage depends upon the word count in the first word of the data table.

The data is read into a buffer register where it is checked. Then a cycle steal request is given for each column to be transferred. The checking is accomplished by reading the data a second time and comparing it to the data previously read into the buffer. After the last column (column 80) has been read, an operation complete interrupt is requested.

Last Card Sequence

When the hopper becomes empty during a feed cycle, the 2501 is taken out of the ready status. Intervention by the operator is required to continue processing cards. The operator may reload the hopper and press the 2501 start key to continue processing or the operator may initiate the last card sequence by pressing the 2501 start key with the hopper empty.

The last card sequence places the 2501 in the ready condition for one more feed cycle and turns on the last card indicator. An operation complete interrupt is given at this time. The last card indicator remains on until a sense device command (111) is issued with bit 15 on.

Programming

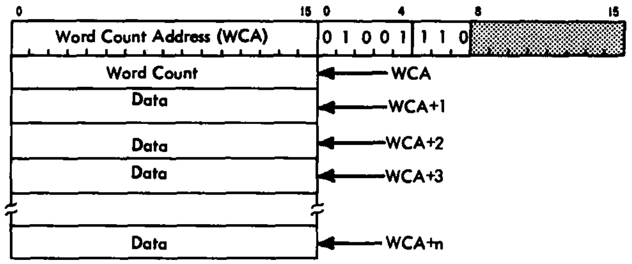
The IBM 2501 Card Reader operates under the control of the stored program in the CPU.

I/O Control Commands

The 2501 is addressed by the five-bit device code 01001.

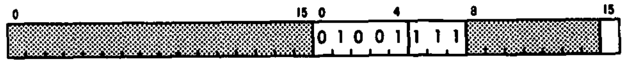
The address portion of the IOCC specifies the location in core storage of the data table. The first word of the data table designates the number of words to be read. This word is called the word count. The word count is located in bit positions 9 through 15 and should never exceed 80 (50 hexadecimal). If the word count exceeds 80, information in succeeding core locations is destroyed.

Initiate Read (110)



This code provides the ability to start a read operation, which subsequently makes data transfers to core storage via a data channel by means of cycle stealing.

Sense Device (111)



This command sets the accumulator with the device status word (DSW) of the 2501. The DSW bits 3 and 4 (last card and operation complete) are reset if bit 15 is on when this command is executed.

DSW Indicators

Refer to Figure 43.

Not Ready or Busy: This indicates that the 2501 is not in a ready condition, or that it has received an instruction and is in the process of executing it.

Busy: This indicates that a card read is in progress and therefore another read card cannot be initiated. This indicator turns off when the operation complete interrupt occurs.

Operation Complete (Interrupt): This is the only interrupt associated with the 2501. This interrupt occurs after column 80 has passed the read station and feed checking has been completed. The operation complete interrupt is independent of the word count and terminates further cycle steal requests. The number of characters actually transferred to the CPU depends upon the word count. The 2501 is assigned to interrupt level 4. Bit 3 in ILSW 4 is turned on if the 2501 caused the interrupt. The sense interrupt (011) command causes ILSW 4 to be loaded into the accumulator if level 4 is being serviced.

Last Card: This indicates that the last card has been fed from the hopper, and the operator initiated a last-card sequence. The indicator may be turned off by a sense device command with reset (bit 15) on.

Error Check: This indicates a feed check or a read check.

Reader and System Timing

There are two basic timing considerations of importance to the user of a 2501 Card Reader attached to an IBM 1130 Computing System:

1. Card throughput in cards per minute (cpm).
2. Time available for other system operation.

System Operations

The 1131 is capable of performing operations (such as reading, processing, and punching) simultaneously. After an operation is initiated, the CPU is busy for only 288 microseconds. The remainder of the card read cycle is available for other use.

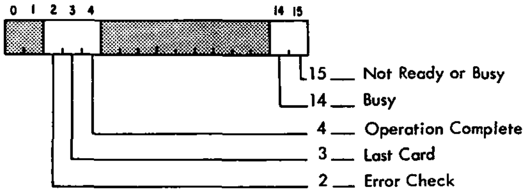


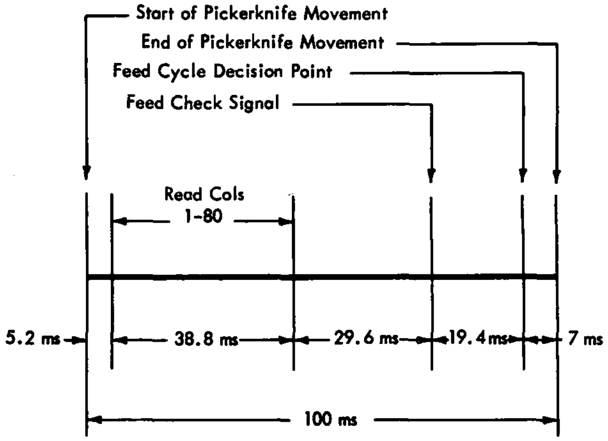
Figure 43. 2501 Device Status Word

Card Throughput

The 2501 model A1 has a 100-ms card feed cycle; the model A2 has a 60-ms card feed cycle. To maintain the rated speed, an initiate read command must occur every 100 ms for the model A1 and every 60 ms for the model A2. A basic timing consideration of importance to the user of the 2501 Card Reader is the time between the feed check signal and the feed cycle decision point. This timing is shown in Figure 44. In order to maintain rated throughput, the read instruction must be received with 18.3 ms (A1) and 3.0 ms (A2) following an interrupt (feed check signal).

If an initiate read command misses the feed cycle decision point, the card reader waits until the feed cycle decision point of the next cycle before starting to execute the command. The result in this case is the throughput is about one-half of the maximum.

Model A1 - 600 CPM (All times shown are nominal at rated thruput.)



Model A2 - 1,000 CPM (All times shown are nominal at rated thruput.)

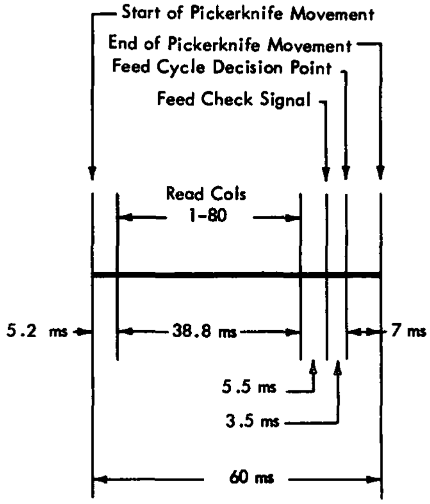


Figure 44. 2501 Timing Schematic

2501 Usage Meter

This meter runs when both of the following conditions are present:

1. The unit is selected for operation by program control.
2. The 1131 usage meter is running.

The meter will continue to run simultaneously with the 1131 meter until either a last card routine is initiated by program control or a manual nonprocess runout is performed.

The IBM 1055 Paper Tape Punch (Figure 45) and the IBM 1134 Paper Tape Reader (Figure 46) provide paper tape I/O for the 1130.

The 1134 and 1055 operate under direct program control.

The 1134 reads one-inch, eight-channel paper tape at a maximum rate of 60 columns per second.

The 1055 is capable of punching eight-channel chad paper tape or edge-punched documents that have prepunched feed holes. The 1055 punches at the rate of 14.8 characters per second.

Tape Specifications

Both the 1055 and the 1134 are capable of using paper tape, Mylar* laminated paper tape, and Mylar coated aluminum tape that meet the specifications in Figure 47. Continual punching of Mylar tape causes excessive wear in the tape punch unit; therefore, Mylar tape should not be used exclusively.

Character Code

The 1134 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.

Figure 48 indicates which bits of the word correspond to the respective holes in the paper tape read by the 1134.

The 1055 punches data as an image of the data contained in positions 0–7 of the core storage word as shown in Figure 48.

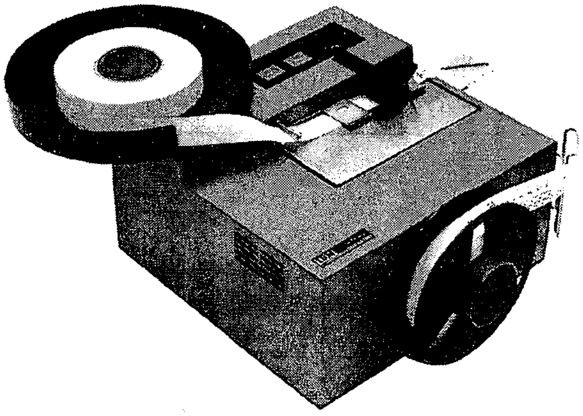

BR2687

Figure 45. IBM 1055 Paper Tape Punch

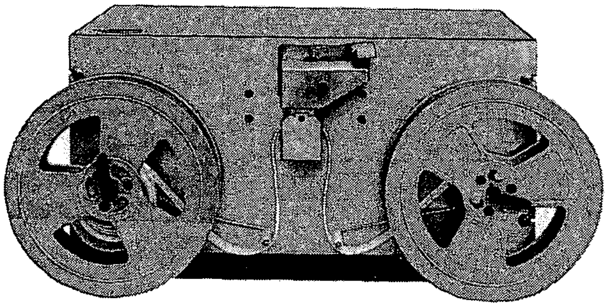
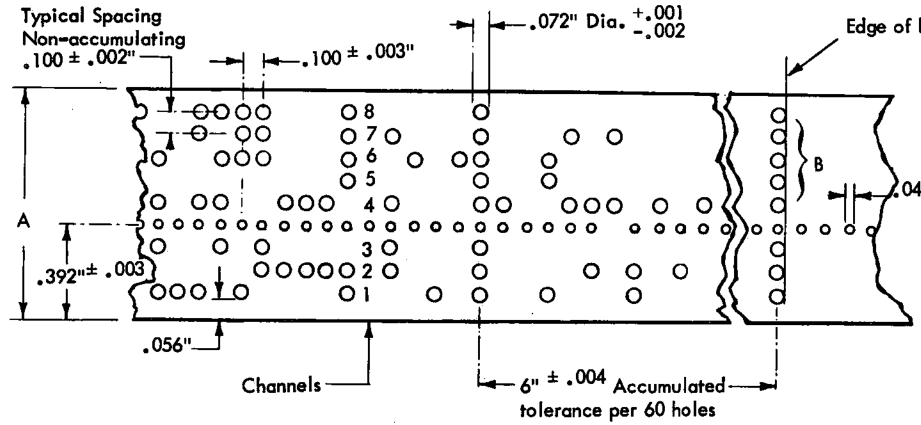

BR2686

Figure 46. IBM 1134 Paper Tape Reader

---

*Trademark of E. I. du Pont de Nemours & Co. (Inc.)



Tape Width A	Number of holes B
1 ± .003 inch	5
7/8 ± .003 inch	4
7/8 ± .003 inch	3
11/16 ± .003 inch	2

Figure 47. Tape Specifications

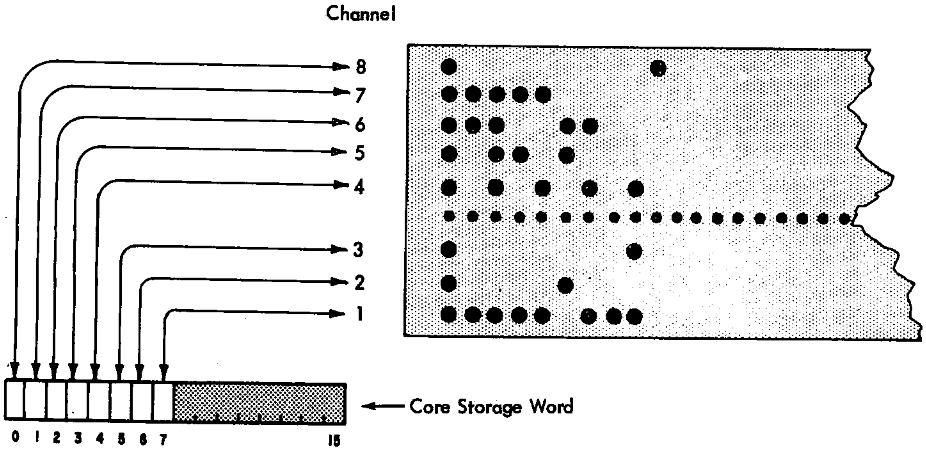


Figure 48. Paper Tape/Core Storage Format

Program Load from 1134

An 1130 system that does not have card I/O will have the program load feature added to the 1134. This feature operates by means of design logic rather than program control. Four-bit paper tape characters are automatically assembled into four-character groups to form 16-bit data words. The program load feature then loads these words into core storage beginning at location 00000.

Only the first four (1-4) tape channels are used. When a channel 5 punch is encountered, program loading stops; the IAR is reset to zero; and program control begins at 00000. Delete characters are permitted at the beginning, but once the program begins to load, the channel 5 in a delete character will end the load.

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's)

The 1134 and 1055 are addressed by the same five-bit device code, 00011.

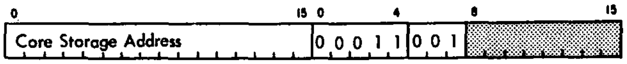
Read (010)

0 15	0 4		8 15
Core Storage Address	0 0 0 1 1	0 1 0	▨

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

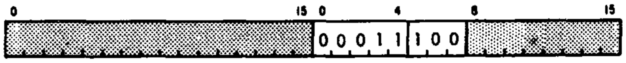
The address word specifies the location in core storage where the tape character is to be stored.

Write (001)



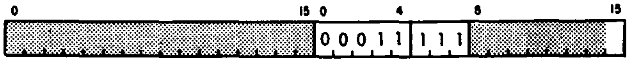
This command writes one character from core storage to the paper tape punch. The address word specifies the location in core storage where the tape character is stored.

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 49) into the ACC. Modifier bit 15 on indicates that the responses are to be reset.

DSW Indicators

Reader Response (Interrupt): This indicates an interrupt which occurs on level 4 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 (Interrupt): This indicates an interrupt which occurs on level 4 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.

Punch Not Ready: This indicator is on when the 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.

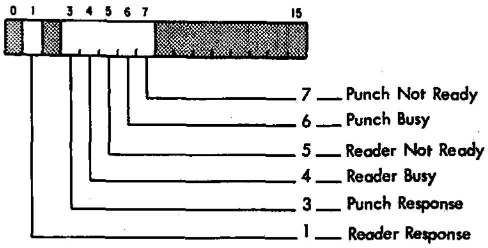


Figure 49. Paper Tape Device Status Word

Printers

Two on-line printers are available for attachment to the 1130 system. A system may include an IBM 1132 Printer Model 1 or Model 2 and/or an IBM 1403 Printer Model 6 or Model 7. The 1403 attachment also requires the attachment of the IBM 1133 Multiplex Control Enclosure.

IBM 1132 Printer

The 1132 Printer Model 1 (Figure 50) provides printed output for the 1130 system at maximum rates of 80 lines per minute (lpm) for alphameric printing and 110 lpm for all-numerical printing. The 1132 Printer Model 2 provides maximum printing rates of 40 lpm for alphameric printing and 55 lpm for all-numerical printing. The print line is 120 print positions long; horizontal spacing is ten characters per inch. Vertical spacing, which is preselected by the operator, is six or eight lines per inch.

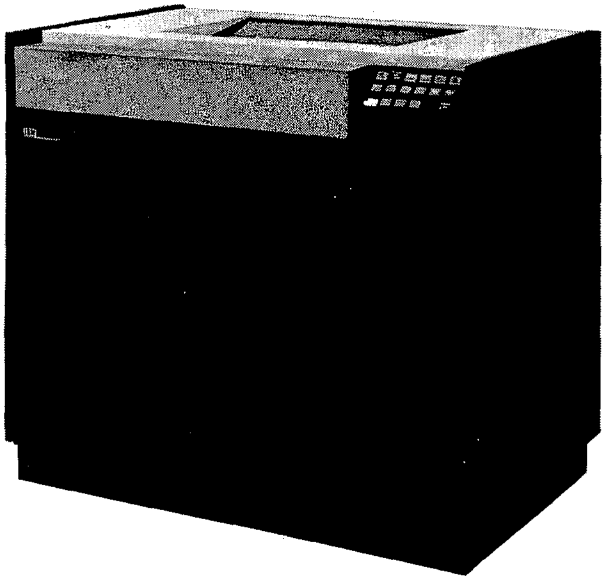


Figure 50. IBM 1132 Printer

FUNCTIONAL DESCRIPTION

Both models of the 1132 contain 120 printwheels, one for each print position. Each printwheel contains a 48-character alphabet consisting of 26 alphabetic characters, 10 numeric characters, and 12 special characters. Special (FORTRAN) characters are as follows:

& – / . $, * ( ) ' + =

All printwheels rotate continuously and in synchronization with each other. Each wheel moves forward to print when the data in the output record specifies that the character to be printed is in position. Thus, all similar characters for the entire line are printed on the same cycle. Forty-eight cycles (one for each character possible) are required to print a complete line.

The 1132 uses interrupt circuitry and responds on level 1.

Forms Control

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.

Spacing is always performed one line at a time under control of the stored program in the CPU.

Carriage skipping is initiated by the stored program and stopped by the program when the predetermined line is reached. Skipping speed is 10 inches per second.

Note: A skip operation must not be less than four lines.

Data Format

The 1132 character code is shown in the appendix. Each character occupies the first eight bits of a core storage word. 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, all by the CPU program. For each equal comparison, the program places a 1-bit 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 32 through 39. The 16 bits of each of the first seven words and bits 0 through 7 of the eighth word represent the 120 printwheels.

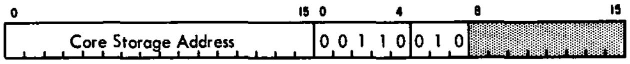
PROGRAMMING

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

Printer I/O Control Commands

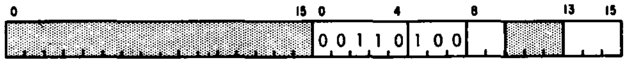
The 1132 is addressed by the binary device code of 00110.

Read Emitter (010)



This command causes the eight-bit EBCDIC code of the next character emitted by the printer to be read into bits 0-7 of the core storage location specified. Bits 8-15 are reset to zero.

Control (100)



This command causes the execution of the function specified by the modifier bit. A 1-bit in the position indicated in parentheses after each command 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 as required 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 51.) 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:

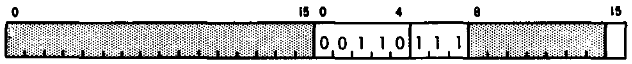
- Eighteen scan cycles have been completed after the command to print the last character.
- The carriage has stopped after a skip operation.
- The interrupt response from the last space command has occurred.

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, a stop carriage command should be given.

Space (Bit 15): This command is given to space the carriage one line. 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 51.

If bit 15 contains a 1, the interrupt responses in the DSW are reset.

DSW Indicators

Three interrupts are associated with the 1132, each on level 1. The associated indicators are turned on in the DSW.

Read Emitter Response (Interrupt): After a start printer command has been executed, the 1132 will interrupt the CPU program each time the printwheels are aligned to print another character. The read emitter command must then be executed to determine the character to be printed.

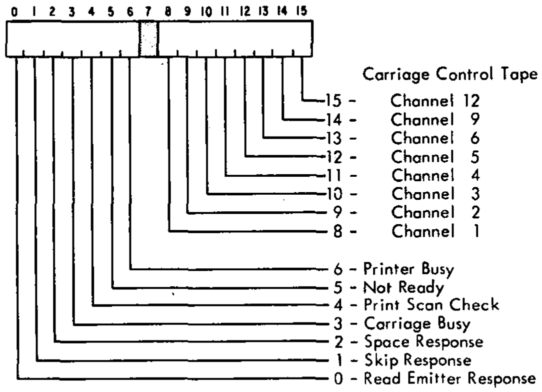


Figure 51. 1132 Device Status Word

Skip Response (Interrupt): This indicates an interrupt which is initiated by the 1132 each time the carriage brushes detect a punch in the carriage tape while a skip operation is in progress. The CPU program must test the DSW bits to determine if the carriage is at the proper channel.

Space Response (Interrupt): This indicates an interrupt which turns on at the completion of a space operation to signal the CPU program.

Note: After an interrupt has been serviced, the level must be reset by a BOSC instruction.

Carriage Busy: This indicator turns on when the 1132 begins carriage movement. It turns off when movement stops.

Print Scan Check: This indicator is turned on when the printer attachment addresses word 39 and there is not a 1-bit in position 15. A 0 in bit 15 indicates that the printer subroutine did not finish setting up the print scan field.

Not Ready: This indicator is turned on by an out-of-forms condition, motor power off, or at the end of the operation in progress if the stop key is pressed.

Printer Busy: This indicator is turned on when an XIO start printer instruction is executed for the 1132. It is turned off by an XIO stop printer instruction to the 1132, by a system reset, or by a CPU stop.

Carriage Control Channels: As each hole in the carriage tape is read after a start carriage control command, the associated indicator in the DSW is turned on.

Programming Notes

The status of the 1132 indicators should be checked before a line is printed. This is accomplished by transferring the printer DSW into the ACC with a sense device command. The modifier bit (bit 15) of the sense device command should be set to 0 to prevent reset of the DSW responses and indicators. Bits 3, 5, and 6 of the DSW are tested and if all three positions are 0, the printer is ready to print the next line. A start printer control command is then given to start the sequence. A scan field transfer, using cycle steal cycles, takes place under control of the printer. Therefore, the scan field must be clear and have a 1-bit in position 15 of core storage word 39 before the start print command is given.

After the code of the next character has been emitted by the printer, a level 1 interrupt is given and the character is read into core storage by a read emitter command. There are 11.2 ms available to test each position of the output record with the character read and set up the 1 bits in the printer scan field. At the end of the 11.2 ms, the 1132 attachment begins its scan and fires each printwheel with a corresponding 1-bit in the printer scan field. If the program has been interrupted for a considerable period by higher levels, the scan may not have been completed. To insure that the program detects this condition, the first steps of the printer subroutine for each character should clear the printer scan field to 0's and, upon completion of the programmed scan, place a 1 bit in position 15 of the eighth word (39). When the printer attachment scans the field it checks this position. If it is 0, the print scan check indicator (bit 4 of the DSW) is turned on. The program can test this indicator and branch to an error routine that provides 47 idle scan cycles and resumes programmed scanning at the point where the scanning was interrupted. This results in overprinting of the characters that were printed unless the error routine keeps track of the positions that were printed and does not set them up again on this scan.

After the final scan cycle for a line of printing, 16 idle scan cycles must be taken before spacing or skipping is started to allow time for completion of the mechanical operation of printing the last character. If the operation is a single or double space, the next scan cycle can be started two scan cycles after the last space command is given.

During an idle scan cycle the printer scan field should be set to 0's, except for bit 15 of the eighth word (39), to prevent the print scan check indicator from being turned on.

1132 Usage Meter

This meter starts when both of the following conditions are present:

1. The unit is selected for operation by program control.
2. The 1131 usage meter is running.

The 1132 meter continues to run simultaneously with the 1131 meter unless a manual space or manual carriage restore is initiated by the operator.

IBM 1403 Printer

The IBM 1403 Printer (Figure 52) greatly increases the output capabilities of the IBM 1130 Computing System while reducing the time that the CPU is required to print, thus leaving more time for other functions. The 1403 is available in two models for attachment to the 1130:

1. The model 6 has a maximum printing speed of 340 (or 210) lines per minute (lpm), depending upon the attachment feature specified in the 1133 Multiplex Control Enclosure.
2. The model 7 has a maximum printing speed of 600 lpm.

Each printer can print 48 different characters in 120 positions. There are 26 alphabetic, 10 numeric, and 12 special characters.

Vertical spacing and skipping are initiated by the stored program. Horizontal spacing is ten characters per inch. Standard vertical spacing is six and eight lines per inch, controlled manually by the operator. Skipping is about 33 inches per second.

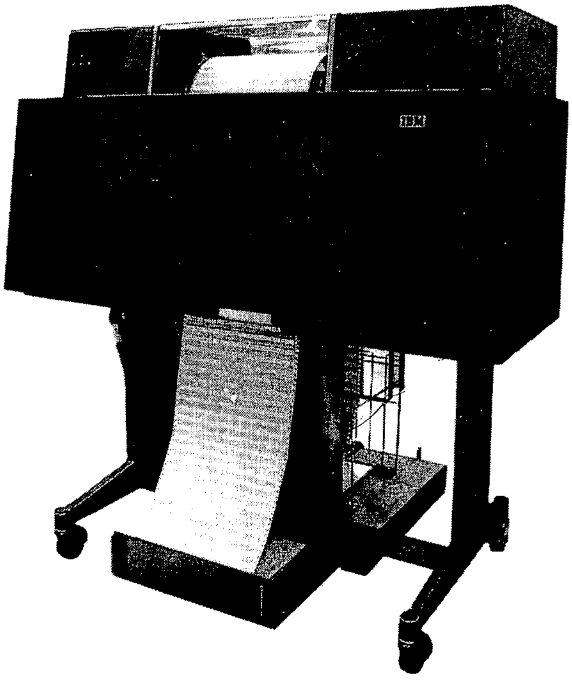


Figure 52. IBM 1403 Printer

FUNCTIONAL DESCRIPTION

Printing

The alphabetic, numeric, and special characters are assembled in a chain. As the chain travels in a horizontal plane, each character is printed as it is positioned opposite a magnet-driven hammer that presses the form against the chain.

Data to be printed must first be edited, translated to the 1403 binary code (see Appendix), and arranged in core storage in exactly the form that it is to be printed. The data format in core storage is two seven-bit characters per word (Figure 53).

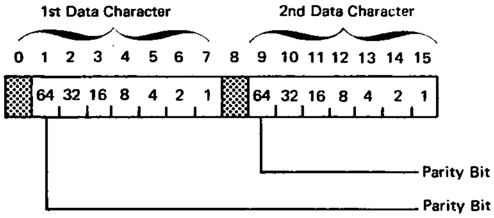


Figure 53. 1403 Data Format Hexadecimal Bits

Spacing and Skipping

Spacing is always performed one line at a time under control of the stored program in the CPU.

Carriage skipping is controlled by prepunched holes in a paper or plastic tape that corresponds in length to the length of one or more forms. Holes punched in the tape stop the form when it reaches any predetermined position.

Note: A skip operation must not be less than two lines.

Control Tape

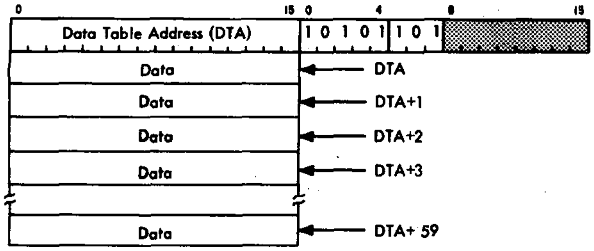
The control tape has 12 columns, indicated by vertical lines. These positions are called channels. Holes can be punched in each channel throughout the length of the tape. A maximum of 132 lines can be used to control forms although, for convenience, the tape blanks are slightly longer. Horizontal lines are spaced six to the inch for the entire length of tape. Round holes in the center of the tape are prepunched for the pin-feed drive that advances the tape in synchronization with the movement of a printed form through the carriage. The effect is exactly the same as though the control holes were punched along the edge of each form.

Programming

All operations of the 1403 are under control of the stored program in the CPU.

Data to be printed must be edited, translated to the 1403 binary code, and arranged in core storage in exactly the form that it is to be printed. The 1403 has a fixed length data field of 120 characters or 60 words.

Initiate Write (101)



An initiate write command transfers data from core storage to the print buffer using the cycle steal method.

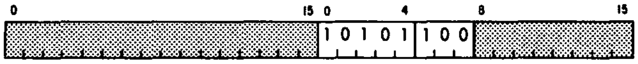
During data transfer each core location to be printed is addressed twice. During the first cycle, bits 1-7 are transferred to an even address of the print buffer. During the second cycle, bits 9-15 are transferred to the next higher odd address of the print buffer.

The total time demand on the processor is dependent on the core storage cycle time. Approximately 432 us. is required for the 3.6-us. core storage, and approximately 264 us. is required for the 2.2-us. core storage.

The printer does not interrupt the CPU until after the 120-position buffer is filled. It then initiates a transfer complete interrupt on level 4.

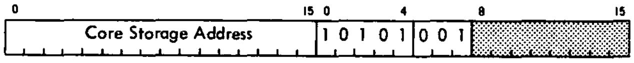
After completion of printing the line a level 4 interrupt is initiated to signal print complete.

Control (100)



An XIO control command initiates a single line space.

Write (001)



An XIO write command controls carriage skipping. This command causes the carriage to skip even if it is at the specified channel. It skips until that channel is detected again. The carriage may be controlled to skip to any channel 1-12 by placing a 1-bit in positions 4-15 of the core location specified by the address.

The carriage will stop only on an exact compare of all corresponding bits. Therefore, if more than one channel is punched on a line, the corresponding bits must be set in bits 4-15 of the address word for the carriage to stop on that line. Likewise, if no bit is present in bits 4-15, the carriage will stop on the next line that has no channel punches.

A carriage control command given prior to loading of the print buffer causes immediate execution of the command. If the command is given during loading of the buffer, the command is not executed until after the line is printed. The programmer must check to insure that the carriage is not busy when the command is given.

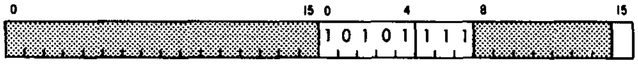


Figure 54. Sense Device (1403)

An XIO sense device command (Figure 54) causes the 1403 DSW (Figure 55) to be placed into the accumulator. If bit 15 is on when the command is executed, the DSW interrupt indicators and channel 9 and 12 indicators are reset.

DSW Indicators

Transfer Complete Interrupt: The 1403 requests this interrupt when the 1403 buffer is full.

Print Complete Interrupt: This interrupt indicates the 1403 has completed printing a line.

Carriage Interrupt: This interrupt indicates the 1403 has completed a skip or space operation.

In addition to the preceding interrupts, the following status conditions are also indicated in the 1403 DSW.

Parity Check: This indicates an even bit count of the seven-bit print buffer data word.

Print Check: This indicates an error occurred in modification of the buffer address register.

Sync Check: This indicates that the print chain is not synchronized with the compare counter.

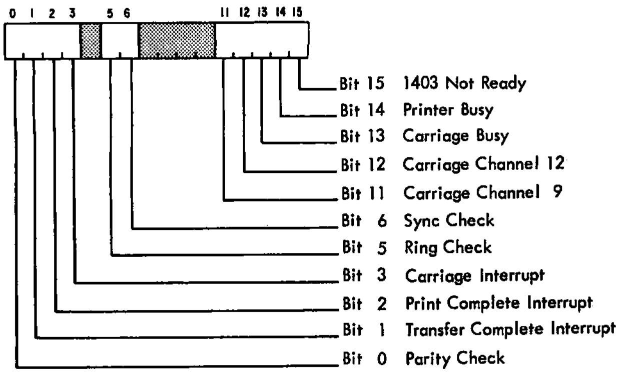


Figure 55. 1403 Device Status Word

Carriage Channel 9: This indicates the carriage passed a channel 9 punch in control tape.

Carriage Channel 12: This indicates the carriage passed a channel 12 punch (normally used for the last printing line on a form) in the control tape.

Carriage Busy: This indicates that the carriage is performing a space or skip operation. This bit goes off when bit 3 comes on to signify completion.

Printer Busy: This indicates that the 1403 buffer is being loaded or a line is being printed.

Not Ready: This indicates the 1403 is not ready. Printing, spacing, or skipping under program control cannot occur until the 1403 is ready.

1403 Usage Meter

This meter will run when the following conditions are present:

1. The 1133 is in an enable status.
2. The 1403 is selected for operation by program control.
3. The 1131 usage meter is running.

This meter will run simultaneously with the 1131 meter until either stopped by the operator or the 1133 is placed in a disable status.

IBM 1627 Plotter

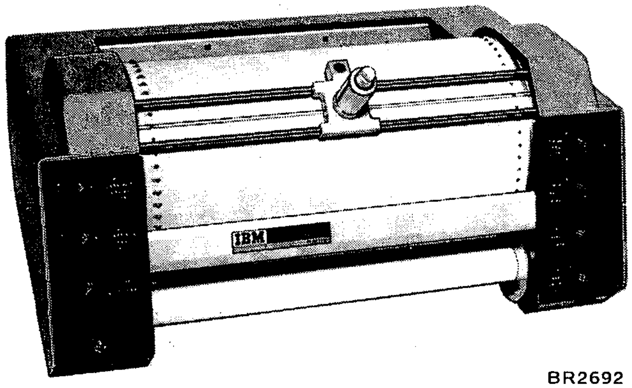


Figure 56. IBM 1627 Plotter

The IBM 1627 Plotter (Figure 56) provides an exceptionally versatile, reliable, and easy-to operate plotting system for the 1130 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 that can be plotted on the 1627.

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, up to 18,000 steps/minute.
Model 2	Plotting area: 29-½ inches by 120 feet, 1/100 inch incremental-step size, up to 12,000 steps/minute.

See Figure 57 for more information. The 1627 is equipped with a ball point pen, which lasts for about five or six hours of continuous plotting. A liquid-flow ink pen is optional.

		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

Figure 57. IBM 1627 Operating Characteristics

FUNCTIONAL DESCRIPTION

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.

The actual recording is produced by incremental movement of the pen on the paper surface (y-axis) and/or the movement of the paper under the pen (x-axis). The pen is mounted in a carriage that travels horizontally across the paper. The vertical plotting motion is achieved by rotation of the pin feed drum, which also acts as a platen (Figure 58)

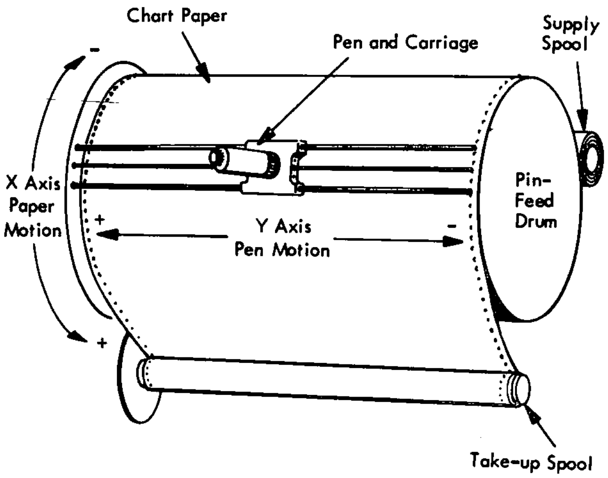


Figure 58. Plotter Paper and Pen Movement

The drum and the pen carriage are bidirectional; that is, the paper moves up or down, and the pen moves left or right. 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.

The drum and the pen-carriage movements and the pen status are controlled by bits transferred to the 1627. Each output word is decoded into a directional signal that causes a 1/100 inch incremental movement of the pen carriage (Figure 59) and /or paper, or a raise-pen or lower-pen movement. The motion or action resulting from each word in the output record is shown in Figure 60.

The time required for execution of raise-pen and lower-pen commands is 100 ms. The time to plot a point is approximately 3.3 ms model 1, or 5 ms model 2.

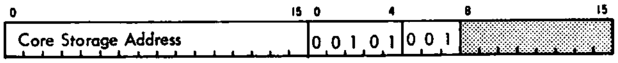
PROGRAMMING

The 1627 Plotter operates under direct program control of the 1130 and responds on interrupt level 3.

I/O Control Commands (IOCC's)

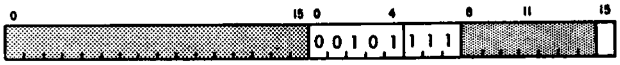
The 1627 is addressed by the five-bit device code 00101.

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. (See Figure 60)

Sense Device (111)



This command causes the 1627 device status word (Figure 61) to be placed into the accumulator. Modifier bit 15 on specifies reset for the plotter response.

DSW Indicators

Plotter Response (Interrupt): This is the only interrupt associated with the 1627. This interrupt occurs when the 1627 has completed the action specified by the last write command. The 1627 is on interrupt level 3.

Not Ready: This indicates the 1627 is not ready to execute commands. A write command issued to the plotter when the not ready condition exists causes a plotter-response interrupt even though the plotter has not executed the command.

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 issue additional 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.

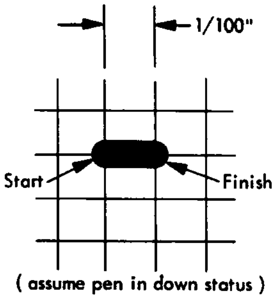


Figure 59. Result of One Horizontal (y-axis) Movement

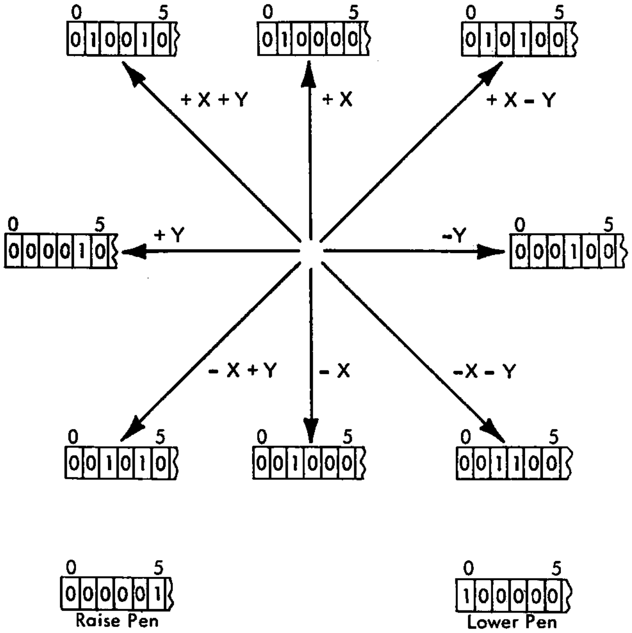


Figure 60. Plotter Command Codes

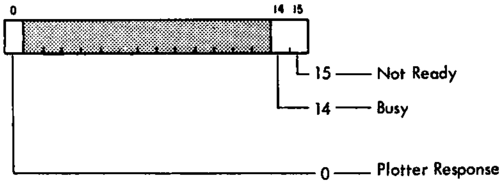


Figure 61. 1627 Device Status Word

IBM 1231 Optical Mark Page Reader

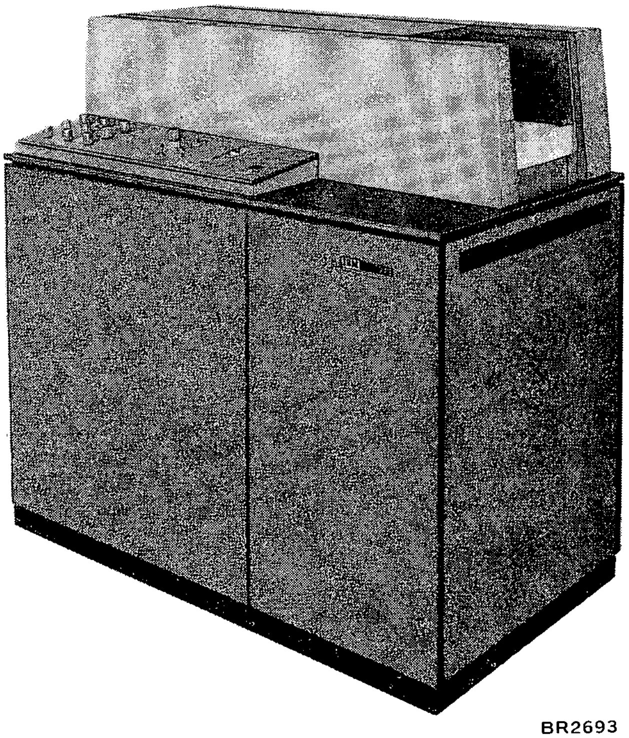

BR2693

Figure 62. IBM 1231 Optical Mark Page Reader

The IBM 1231 Optical Mark Page Reader (OMPR) represents a breakthrough in source recording and data entry. The OMPR provides a facility for recording the data at its source, in a form that can be read directly into the IBM 1130 Computing System.

The 1231 (Figure 62) reads positional marks made by an ordinary lead pencil on paper documents. The positional marks are read directly into the IBM 1130 core storage.

Documents are read at a maximum rate of 2,000 sheets per hour.

DATA SHEET

The document used as input to the Optical Mark Page Reader is an 8-½" x 11" sheet of paper called a "data sheet." The data sheet contains a maximum of 1,000 mark positions. The mark positions are arranged in an many as 50 rows, each row containing a maximum of 20 mark positions.

Each row is divided into two groups of ten mark positions each. The ten mark positions are called "words." Each word is divided into two groups of five mark positions called "segments." Consequently, each data sheet can have a maximum of 50 rows, 100 words, and 200 segments. A data sheet normally contains five rows per inch but may have less.

Timing marks are printed along the right-hand edge of each data sheet. These marks are used to synchronize the motion of the document with the sensing unit of the reader. Each word on the data sheet has an associated timing mark. For forms design information see the System Reference Library publication *IBM 1231, 1232 Optical Mark Page Readers* (Order No. GA21-9012).

Data Sheet Terminology

Timing Mark: A rectangular mark preprinted on the data sheet in non-reflective ink. The timing mark is used to synchronize the motion of the document with the sensing unit of the 1231. Timing marks are located on the right-hand side of the data sheet.

Mark Positions: Areas printed in reflective ink that designate where marks are to be placed. A non-reflective mark in this area is read as a word or bit.

Word: Ten mark positions of a row. Words on the left half of the data sheet are odd words; words on the right half of the data sheet are even words.

Segments: Mark positions 0 through 4 and 10 through 14, or 5 through 9 and 15 through 19, of any word.

Non-Reflective Ink: A type of ink that is sensed by the 1231. Usually, timing marks are the only non-reflective printing on the data sheet. The recommended non-reflective ink is black.

Reflective Ink: A type of ink not sensed by the 1231. Reflective inks are used for headings, data sheet instructions, mark position outlines and any other data that is not to be read.

Marking the Data Sheet

Marks that are to be read by the IBM 1231 must be dark enough for positive machine reading, yet erase easily and completely. For these reasons, a number 2 pencil is recommended.

Marks made with a number 1 pencil, or an IBM ELECTROGRAPHIC ® pencil are difficult to erase. Even after an erasure is made a residue remains that could be read as a mark by the machine.

Erasures should always be made carefully and completely. Any incomplete erasure could be read as a mark.

When response positions are marked, the mark should be made the full length of the mark position, and should fill at least two-thirds of the space between the top and bottom of the guide lines. A mark that extends no more than 1/16 inch past the ends of the response position is acceptable in all but the last even-word position (next to the timing marks). In this position, a mark must not extend beyond the right end of the guide lines or it could be read as a timing mark. This would result in erroneous reading of the rest of the data sheet.

FUNCTIONAL DESCRIPTION

The 1231 uses sonic delay lines for storing controls and data. Controls are marked on the regular data sheet and entered into delay line storage during the program load cycle. This data sheet is referred to as a program control sheet. The program control sheet is automatically placed in the select (upper) stacker during the load cycle.

As data sheets are read, data is stored in the delay lines according to instructions from the program control sheet. Each word to be stored on the delay line must be programmed by the program control sheet.

When a data sheet passes under the photoelectric read head, each word is tested for conditions, such as no-mark, multi-mark, or other-than-one. Switches on the 1231 control panel in conjunction with the program control sheet control the test of these conditions. Any word that does not pass the requirements of the switch settings causes the data sheet to be routed to the select stacker.

Document Path

The data sheet begins its movement through the optical mark page reader when it is fed from the hopper by CPU program control. The document then passes under a read head and is next transported through the transport area, past a selection station, and on into one of the two gravity stackers.

Message Format

Each word transferred from the 1231 to the 1130 reads into a single position of core storage. Words are transferred one segment at a time to the A buffer and the B buffer in the attachment; all odd segments (A buffer) enter positions 0-4 and 14, and all even segments (B buffer) enter positions 5–9 and 15 (Figure 63). If the 1231 is programmed for only one segment, all segments enter positions 0-4 and 14. Words with marks in positions 0, 1, 2, 3 or 4 transfer to core storage as an odd segment and marks in positions 5, 6, 7, 8, or 9 transfer to the 1130 as an even segment. Combinations of the bits make up a valid character which must be translated by the 1130 stored program. Any or all of the marking positions on the data sheet may contain marks.

Data is read by the 1231 from left to right, top to bottom, a row at a time. Information from a data sheet is stored in the following sequence:

1. Segment 1 of the first word programmed to read.
2. Segment 2 of the first word.
3. Segment 1 of the second word programmed to read.
4. Segment 2 of the second word.

If only one segment of any word is programmed to read, then each segment goes into a separate core storage word.

Mark Recognition and Discrimination

During the reading of data sheets, the Optical Mark Page Readers categorize marks according to their degree of light reflectance (Figure 64). A mark falls into one of the following categories:

1. Good
2. Poor
3. Uncertain

A good mark is recognized as a positive indication of a mark; a poor mark (or good erasure) is not recognized as a mark, and an uncertain mark (light mark or poor erasure) is one whose light reflectance level comes somewhere in between a good mark and a poor mark but cannot be positively identified as either. The reading or rejection of uncertainties can be customer-controlled.

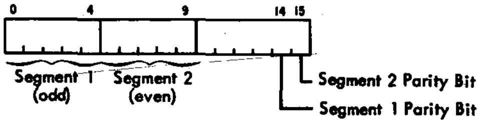
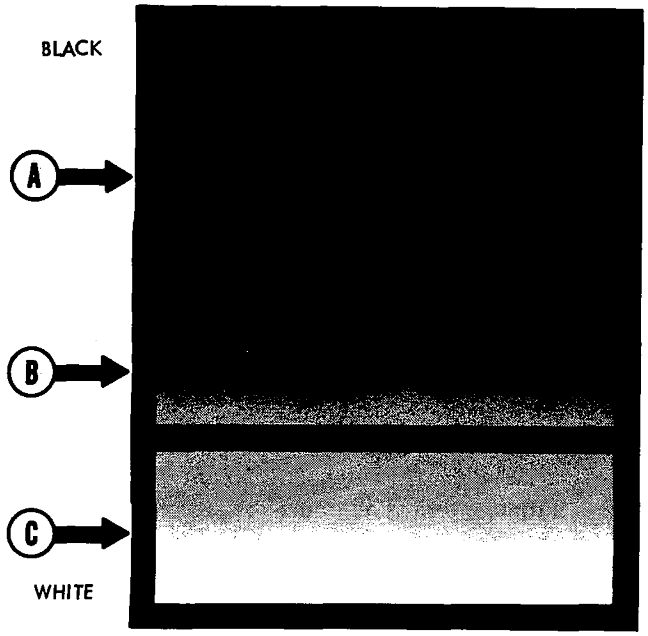


Figure 63. 1231 Data Format



A: Good Mark
B: Light Mark or Poor Erasure
C: Poor Mark or Good Erasure (Marks in this area are not read)

Figure 64. Mark Reflectance Relationship

Three read-mode switches, each associated with a set of field-checking switches, allow operator control of mark discrimination on a field-by-field basis. Documents containing uncertainties can be selected for a visual check if desired.

Each of the three read-mode switches has four settings: SING RESP (single response), MULT RESP (multiple response), SING RESP SEL UNC (single response select uncertainties), MULT RESP SEL UNC (multiple response select uncertainties).

The setting of each read-mode switch affects mark discriminaton as follows:

1. SING RESP:
    a. Marks in area A are accepted.
    b. Marks in area B that are not accompanied by a mark in area A of the same word or segment are accepted.*
    c. Marks in area B that are accompanied by a mark or marks in area A of the same word or segment are not accepted.
2. SING RESP SEL UNC:
    a. Marks in area A are accepted.
    b. Marks in area B that are not accompanied by a mark in area A of the same word or segment cause the data sheet to be selected.
    c. Marks in area B that are accompanied by a mark in area A of the same word or segment are not accepted as marks.
3. MULT RESP:
    a. Marks in area A are accepted.
    b. Marks in area B are accepted.
4. MULT RESP SEL UNC:
    a. Marks in area A are accepted.
    b. Marks in area B cause the document to be selected.

Whenever a data sheet is selected by the 1231, storage is cleared and data from that data sheet is prevented from being transferred to the computer.

Data Flow

Before the 1231 can act as an input device to a data processing system, the controls for the internal functions must be loaded and switches must be set to establish the conditions required for the particular run.

Two storage devices (sonic delay lines) are used to store and control the data as it is read from the data sheets. One of these storage devices, the "master" line, is used to store all the controls from the program control sheet. If the 1231 is equipped with the master mark special feature then master-mark data and controls associated with master-mark data are also stored.

The other storage line, the data line, is used to store information from the data sheet. As the data sheet is read, the two storage lines work concurrently and in synchronism. The master line, which contains the program instructions, determines which information from the data sheet is to be retained.

The following sequence is used for entering data into a fully equipped 1231 and for making this information available to the processing system (Figure 65).

1. Line mark and word mark bits are generated by internal circuitry to establish the starting point of the data on the delay lines. These bits go into the data delay line.

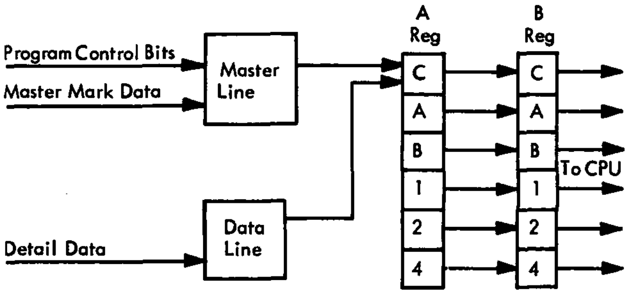


Figure 65. IBM 1231 Data Flow

---

*Number of mark positions included in any one mark discrimination test is determined by the setting SEGMENT or WORD of the check-length switch.

2. Program control bits are loaded into the 1231 from the program control sheet and go into the master line.
3. Master-mark information (if master-mark special feature is installed and being used) is transferred the same as detail data. It is up to the system to store this information and merge it with each succeeding data sheet until new master data is transferred.
4. Detail data reads into positions 12 through 111 of the data line.
5. The first word of data (master or detail) is sent to the 1231 attachment buffer in the 1131. When this buffer is loaded it causes an interrupt to be generated. The 1131 program must then issue a read command to transfer this word into core storage. This process is repeated until all the data is transferred.

Field Checking

In field checking, each word programmed to be read is checked for mark conditions which may indicate invalid data. Three switch-controllable mark conditions, each of which will cause the document to be selected, can be checked. These conditions are: multi-marks, no marks, or other-than-one. The switch also has an off position.

Data sheets are usually designed with "fields" of similar data in vertically consecutive words on the left or right sides of the data sheet.

A field checking field differs from a data sheet field primarily in functional grouping. A data sheet field groups similar information for ease of marking and reference. A field checking field may contain part of, or several, data sheet fields. The primary requirement of field checking is that all mark data within a field's area of coverage be checked for the same conditions (multi-mark, no mark, etc.).

The field checking fields on the data sheet are defined by special codes (start-of-checking codes) which are entered into 1231 storage from the program control sheet. A field checking field can be from 1 to 100 words in length.

Three start-of-checking control codes allow any specific area of the data sheet to be checked according to one of three groups of field checking switches. The three groups of field checking switches are labeled, field I, field II, and field III.

The checking of a field checking field by a particular group of switches begins on the word in which the field checking control code is recognized. On the program control sheet, a mark in position 6 designates the start of data checking according to conditions set up on field I switches; a mark in position 7 designates the start of data checking according to conditions set up in field II switches; marks in positions 6 and 7 designate the start of data checking according to field III switches.

Three switches are assigned to each field: (1) a read mode switch, which determines how uncertain marks are handled, (2) a check length switch, which determines whether information is to be checked on a word or segment basis, and (3) a select condition switch, which determines the conditions for which a data sheet will be selected.

Because the data sheet is read from left to right, top to bottom, row by row, field checking becomes an important factor when a new data sheet is to be designed. If, for instance, the data sheet is to be used for a yes or no survey, the yes and no mark positions should be within one segment or word in order to allow checking for both, either, or neither answer.

The field checking feature can be summarized as follows:

1. Each of the three field-checking switches can be set to one of the select conditions, or to the off position.
2. Unless programmed for another field, checking always returns to field I at the start of a new data sheet.
3. Field checking by a given set of field checking switches begins in the word programmed and continues (in all words programmed to be read) until a new field checking command has been recognized.
4. A field can begin or end on either the left or the right word.
5. A field can be from 1 to 100 words in length.
6. Three conditions can be checked: other-than-one (multi-mark and/or blank detection), no-mark (blank detection only), and multi-mark (multi-mark detection only).

When the data sheet is selected because of a field checking condition, or because of an uncertain reading, the 1231 causes the delay line to be reset. Some data may have been transferred. This data is considered to be invalid.

Alphabetic Coding

An alphabetic coding capability is necessary and desirable in many applications. Three schemes of alphabetic coding are illustrated in Figure 66.

Scheme 1: To code an alphabetic character, a mark must appear in the appropriate marking position of both the odd (left hand) and even (right hand) words of the same horizontal row. For example, to indicate the letter K, one mark must be made in the marking position immediately below the caption "J" through "R" in the odd word, and one mark must be made in the marking position immediately below the caption "BKS" in the even word of the same horizontal row. The odd word in this scheme represents the zone portion of the character.

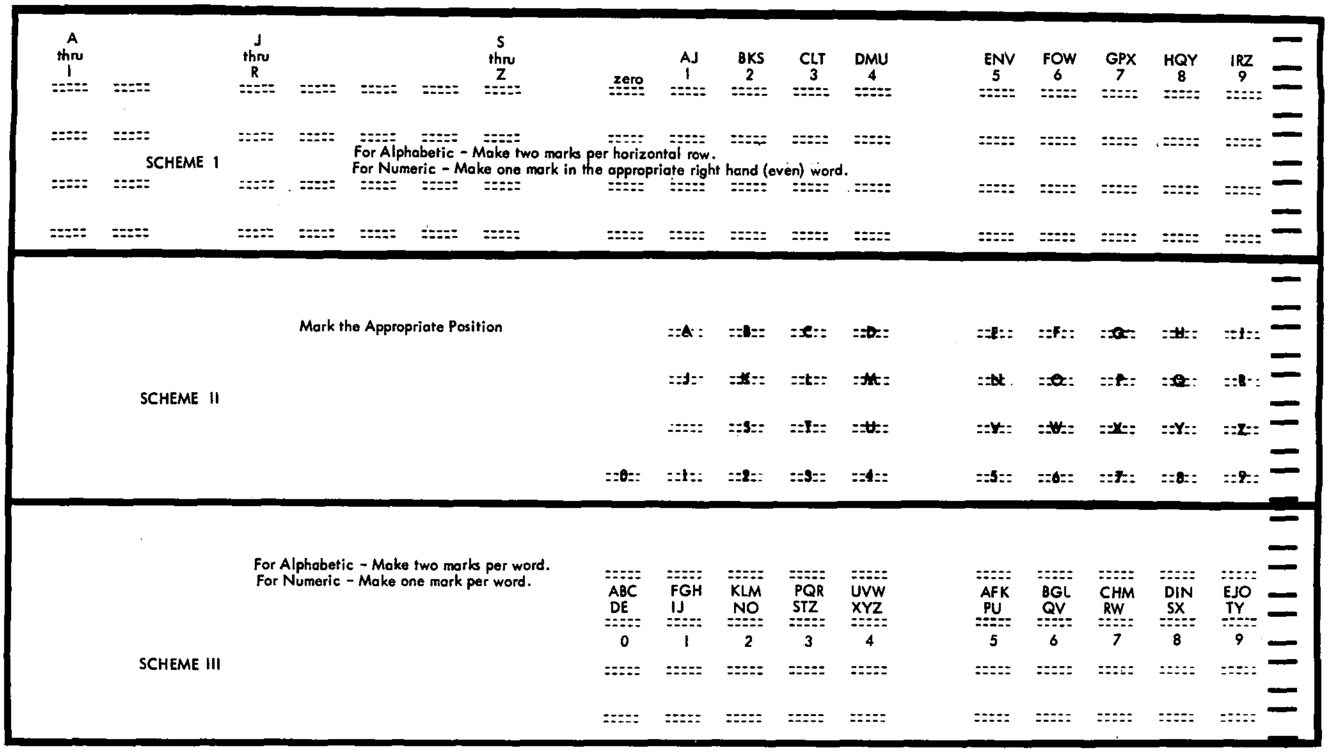


Figure 66. Alphabetic Coding Schemes

Scheme 2: Each letter of the alphabet can be preprinted on the data sheet in reflective ink. The letters and/or numbers may be printed above, on, or below the mark positions. In this approach, the identity of each character is determined by its position in the matrix, which is programmable.

The entering of the marks is simple; however, considerable space is required on the data sheet to represent all the alphabetic and numeric characters.

Scheme 3: Each letter of the alphabet, or digits 0 through 9, can be represented by using only one word. An alphabetic character must be represented by a mark in each segment of the word selected for this purpose. (Z is an exception.) To indicate a K, marks in the 2-position of the left-hand segment and in the 5-position of the right-hand segment are required.

PROGRAMMING

Programming for the IBM 1231 Optical Mark Page Reader depends upon two sources of control: controls stored within the 1231 and controls received from the 1130. Controls stored within the reader are entered into storage from a program control sheet.

Program Control Sheet

A program control sheet is a data sheet with certain operational controls marked in the data areas. Each word from the data sheet consists of ten positions. Each word in the delay line storage consists of 16 positions: ten for the positions on the data sheet, and six for storing operational and internal controls generated by circuitry of the 1231.

Every word that is to be retained for transferral to the 1130 must have an operational control marked in that word on the program control sheet. When operational control information is entered into storage, the controls go into some of the six control positions associated with each word.

During the program load cycle, the mark positions used as control positions are:

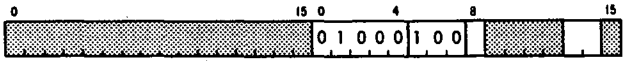
1. A mark in position 8. This designates that a word is to be stored. This word is available later for readout to the 1130 system. There must be two timing marks for each row on the detail data sheet programmed to read by the program control sheet.
2. A mark in position 0 and a mark in position 8. This stores data from segment 1 only (bits 0-4 or 10-14 of the data sheet).
3. A mark in position 5 and a mark in position 8. This stores data from segment 2 only. (Bits 5-9 or 15-19 of the data sheet).
4. A mark in position 6. This indicates the start of field checking according to the settings of field I switches.
5. A mark in position 7. This indicates the start of field checking according to the settings of field II switches.
6. A mark in position 6 and a mark in position 7. This indicates the start of field checking according to the settings of field III switches.

System Programming

Three I/O commands are used with the 1231: control (100), read (010), and sense device (111). In addition the control command uses three modifier bits to expand the number of commands.

The 1231 is addressed by the five-digit device code (01000 = area code 8).

Control (100)



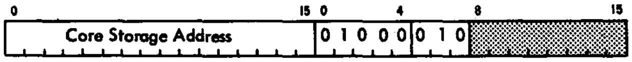
This control command uses three modifier bits:

Read Start (bit 13): Causes the document to move through the read station. Data is collected and placed on the delay line controlled by the control sheet and switch settings. As soon as the first word programmed for output is placed on the delay line, it is made available to the processor.

I/O Disconnect (bit 14): Terminates the read operation from the document and clears the delay line storage by signaling the 1231 that no more data is desired. This command should be given to prevent a read (overrun) error on the next document if all data from the previous document is not cleared from the delay line storage.

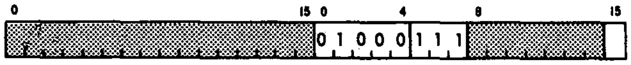
Select Stacker (bit 8): Causes the document just read to enter the select stacker. This command can be given within 40 ms after the last word is placed on the delay line. Indicator bit 5 in the device status word is on if it is permissible to select the document. This bit should be tested prior to issuing the select command. If the bit is off and a command is issued the command will be ignored.

Read (010)



This command causes the next word in the 1231 attachment buffer to be loaded into the core storage location specified by the address.

Sense Device (111)



The sense device command causes the 1231 DSW (Figure 67) to be placed in the accumulator. Modifier bit 15 resets the responses.

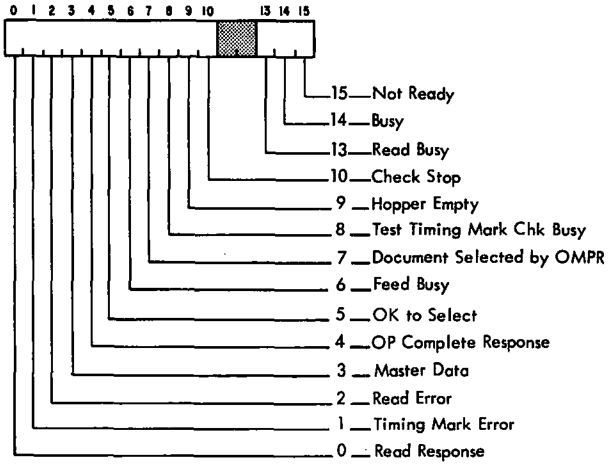


Figure 67. 1231 Device Status Word

DSW Indicators

It is possible to read the last character of a document and receive the end of transmission signal from the 1231 before the timing mark check is made if the timing mark switch is set to yes. This would turn on the operation complete interrupt, which would then be serviced by the processor. If the timing mark switch were set to YES, a timing mark error could occur after the operation complete routine. Were this to occur, the timing mark error indicator would be turned on. This indicator would remain on until the 1231 was placed in the ready state. The operator would be aware or this error either by program analysis or by visual inspection of the 1231 control panel.

Read Response (Interrupt): Bit 0 signals that a word (one or two segments) has been loaded in the 1231 attachment buffer and can be accepted by the CPU. An XIO sense DSW command with bit 15 turns off the read request and loads the 1231 DSW into the accumulator. An XIO read command is needed to turn off the read busy indicator and transfer the word to core storage.

Timing Mark Error (Interrupt): Bit 1 indicates there is a timing mark error. Timing mark error turns on the operation complete interrupt. The error is dependent upon the settings of two switches: timing mark check switch and control timing mark switch. The timing mark check switch is an 11-position rotary switch. It has an off position and ten positions labeled 0 through 9. In the off position no checking is performed. If checking is desired, the switch is placed on the digit corresponding to the units value of the number of timing marks on the documents. The control timing mark switch has two positions labeled YES and NO. The no position is used for documents having the normal number of timing marks (100 or fewer); the yes position is used for documents having 106 timing marks.

Read Error (Interrupt): Bit 2 indicates that a parity error (even count) occurred, an overrun condition occurred, or that no bits were entered onto one of the delay lines with either a data sheet or a control sheet. The not ready and the operation complete interrupt is turned on. The delay line storage is cleared so that it is not necessary to give an I/O disconnect.

Master Data: Bit 3 indicates that data to be transferred is master data. A master data subroutine should place master data in a reserved area. The indicator does not come on if the master-mark switch is off.

Operation Complete (Interrupt): Bit 4 indicator is turned on by the end of transmission and signifies that the last word has been read by the CPU. It is also turned on by the timing mark error or read character error. It is turned off by an XIO sense device with bit 15 on.

Okay to Select: Bit 5 indicator comes on when a document read is initiated and remains on for 40 ms after the document has been read.

Feed Busy: Bit 6 indicator comes on when the XIO control command with modifier bit 13 is executed. It remains on until the first 1231 interrupt is turned off. Another start feed command should not be given while this indicator is on.

Document Selected: Bit 7 is caused by either a mark count reject or data uncertainty, according to the setting of the field checking switches and the program control sheet. It is turned off by an XIO sense device with bit 15 on. Document feeding is not inhibited. When this indicator is turned on the delay line storage is cleared of data and the transfer to the processor is terminated. The operator must be flagged by the program that this has happened. If the next feed command has been issued it will be necessary to refeed two documents. If the document selected indicator is turned on and serviced before the next feed command, only the top sheet in the select stacker should be processed. Data should not be processed if this indicator is on.

Test Timing Mark Check: Bit 8 indicator comes on just before the first character interrupt from the 1231 attachment and remains on for 90 ms after the last word has been placed on the delay line. If the control timing mark switch is set to YES, the timing mark check is not made until the end of the 90 ms period. If this indicator is to be tested, data processing should not take place until the indicator goes off.

Hopper Empty: Bit 9 indicates that the 1231 hopper is empty and turns on not ready.

Check Stop (Interrupt): Bit 10 indicates the 1231 has encountered: a double feed, a transport jam, a feed command given but the documents are not being fed, or the 1130 program has issued a start read command and a not ready condition occurs before the 1231 responds.

Note: Not ready and hopper empty bits may appear in the DSW when a read response interrupt is being serviced during processing of the last document. Read response interrupts will continue and data will be available after not ready and hopper empty occur. Check stop interrupt is blocked until the document has been read.

Read Busy: Bit 13 indicates that the 1231 attachment buffer is full. It is turned off when the XIO read command for the 1231 is given.

Busy: Bit 14 indicator comes on after the read start command has been issued and remains on until an operation complete interrupt is received or an XIO disconnect command is issued. This indicator being on indicates that a document is being read.

Not Ready: Bit 15 indicator is off if the 1231 is ready to accept instructions from the CPU program. The following conditions are necessary for the 1231 to remain in a ready condition:

1. Power on.
2. Off line/on line switch set to ON LINE.
3. Control sheet loaded.
4. Hopper loaded.
5. No read or feed errors.
6. Start key is depressed after the program load light on the 1231 goes off after loading the control sheet.

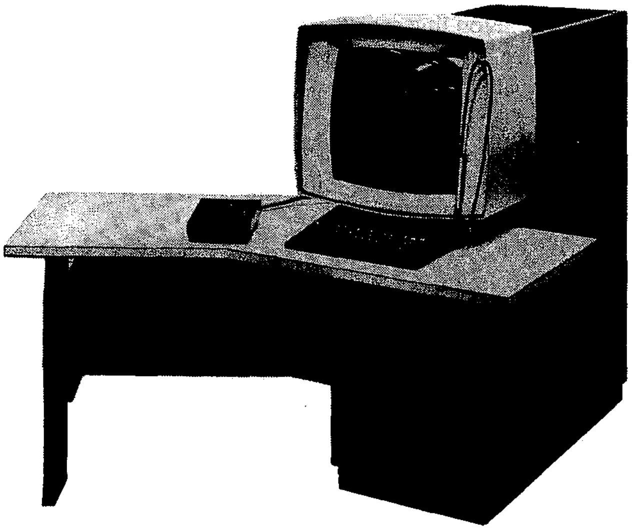
1231 Usage Meter

This meter runs when both of the following conditions are present:

1. The unit is selected for operation by program control.
2. The 1131 usage meter is running.

The meter continues to run simultaneously with the 1131 meter until either the 1231 hopper becomes empty or it is stopped by the operator.

IBM 2250 Display Unit



BR2691

Figure 68. IBM 2250 Display Unit Model 4

The IBM 2250 Display Unit, Model 4 (Figure 68) is a cathode-ray tube display unit that attaches to the IBM 1130 Computing System. The 2250 Display Unit operates under the control of a display order program and input/output (I/O) control commands. The program and commands are sent from the 1131 via the 1131 storage access channel (SAC) or 1133 storage access channel II (SAC II).

The 1130 system and 2250 model 4 operate asynchronously. The display program orders can be sent to the 2250 at a rate up to 40 frames per second (25 millisecond frame time); however, the 2250 operation can be delayed while the 1130 is servicing a device with a higher cycle-steal priority.

This section provides a brief description of the IBM 2250 Display Unit, Model 4. For a more complete description of the 2250 model 4 refer to the Systems Reference Library publication *IBM 1130 Computing System Component Description – IBM 2250 Display Unit, Model 4,* Order No. GA27-2723.

FUNCTIONAL DESCRIPTION

As soon as a 2250 operation is started by an I/O control command, the 2250 addresses CPU storage to execute the display program – stealing core storage cycles from the CPU. Core storage cycle demands by the 2250, therefore, always have higher priority than those of the CPU program. Units that operate synchronously with the CPU are assigned higher cycle-steal priority than the 2250, eliminating 2250 interference with synchronous operations.

The 2250 can generate images of vectors (straight lines), points, and characters on the 21-inch cathode-ray tube. (The usable display area is 144 square inches – 12 inches by 12 inches.) A visible display is produced when the electron beam in the cathode-ray tube strikes the phosphor-coated cathode-ray tube screen. The screen area struck by the beam glows briefly. Normally, the glow fades within a fraction of a second – too rapid for human eye perception or recognition. For this reason, the display is continuously regenerated at a discernible rate.

As soon as regeneration is started by an 1130 I/O control command, the 2250 channel interface section – where storage addressing is performed – continuously fetches orders and data from the display program in storage. Orders are decoded in this section. Deflection information is transferred to the 2250 display section, where it is used to draw the appropriate display. Regeneration is accomplished by continuously repeating the display program. Orders and data in the display program can be modified during regeneration – as directed by the CPU program or by the display program itself – to update or change the display.

The 2250 display section, furthermore, performs various nondisplay services by providing the interface between the user and the problem program with three devices: the programmed function keyboard, the alphameric keyboard, and the fiber-optics light pen.

The programmed function keyboard provides communication between the user and a CPU program. The keyboard consists of keys, indicators, and sensing switches for use with replaceable descriptive overlays. The function of each key and indicator is defined by the CPU program. Punches in the top edge of each overlay identify the overlay to the CPU program. To identify the key and indicator functions to the operator, the key or indicator labels (or both) can be placed on the overlay. Each key can be used by the program to initiate a subroutine associated with the respective overlay, thereby performing the indicated function. For example, depression of a key might result in the enlargement, reduction, or deletion of the display image.

The alphameric keyboard makes it possible for character displays to be created, edited, or changed by the user. With the typewriter-like keyboard, alphameric messages can be entered into the display program for displaying and editing. The alphameric keyboard key codes can be interpreted by the CPU program and used for control purposes in a manner similar to operations with the programmed function keyboard.

The light pen provides the means by which the display program and the CPU program can identify the storage address of the order that initiated the display of a vector, point, or character. This information can be used for operations determined by the display program, by the alphameric keyboard, or by the programmed function keyboard. The user can identify a displayed image simply by pointing the light pen at the image or by pressing the tip switch (the point at the end of the light pen) against the image. The method of identification is determined by the display program.

DISPLAYS

Information positioning on the 2250 display area is controlled by a display program in the 1131 core storage. This program is sent to the 2250 (by a 16-bit word) via the 1131 storage access channel or 1133 storage access channel II. Orders in this program specify electron beam deflection to horizontal (X) and vertical (Y) coordinates on a square reference grid. This grid is composed of the 1,024 possible electron-beam-deflection end points.

Information can be displayed in the 2250 in either the graphic mode or character mode.

Graphic Mode

Either vector or point operations can be performed by the 2250 in graphic mode. If no specific graphic mode has been set previously by an order from the display program, the vector mode is set automatically. In graphic mode, the 2250 can receive (from the display program) either electron-beam positioning orders or an order to establish a different mode of operation, such as to set point mode from vector mode or to enter character mode from graphic mode.

Character Mode

The set of characters that can be displayed by the 2250 in character mode is defined by the programmer. This character set resides in 1130 storage as a subroutine of the display program. The character set can comprise any number of characters in any font. These characters can be modified at any time during execution of the display program. Characters in this set can be displayed in either of two sizes – basic or large, as determined by the character-mode order.

In character mode, the current X, Y position of the beam on the 1,024-by-1,024 position display area becomes the center of a basic- or large-size character area. This area is maintained throughout one character mode operation. The program places the beam at a starting position on the display area (using a blanked point or vector) before a character display operation is started.

CHANNEL INTERFACE SECTION

The 2250 channel interface section interfaces the storage access channel (SAC) and the 2250 display section. It decodes and executes orders and commands, addresses CPU storage, and handles data transferred to or from CPU storage. Information transfer across the SAC/2250 interface is by a 16-bit word.

An address register in the 2250 channel interface section specifies (to CPU storage) the location at which information will be stored or from which it will be retrieved for 2250 operations. This address register is initially loaded by an initiate write (start regeneration) command from the CPU program. It can then be stepped automatically by the 2250, altered by the display program, or reloaded by the CPU program. Thus, display regeneration can be performed without CPU intervention.

The display program consists of display orders, associated data for image generation, and control orders for various nondisplay functions. Undefined order codes received by the 2250 are treated as no-operation orders or are interpreted as data if in the appropriate format.

The CPU program initiates 2250 operation by issuing an execute input/output (XIO) instruction. The I/O control command at the effective storage address specified by the XIO is then sent to the 2250. If the I/O control command is initiate write (start regeneration), the 2250 fetches display program information from core storage starting at the address specified by the I/O control command.

Display program information consists of orders and data. Orders either initiate a 2250 operation or establish a mode. Order-initiated operations include point and vector plotting, branching, and CPU interrupt generation.

Data is defined as information that does not contain an operation code. Character-stroke words are the only data received by the 2250. Even though a character-stroke word may contain one or more control bits, these bits are used directly to perform an operation.

PROGRAMMING

The 1131 Central Processing Unit (CPU) uses input/output (I/O) control commands to control 2250 operations.

Input/Output Control Commands

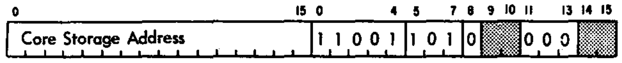
The 2250 is selected by the five-bit device code 11001. The three-bit function code specifies primary I/O functions. The modifier portion of the command provides additional information for the device and function specified. Command modifier bits 11, 12, and 13 must be 0's. Unassigned modifier bits are not decoded. Unassigned functions codes are treated as no-operation commands by the 2250. The execution time of each command is equal to the XIO instruction time plus one core storage cycle time for each cycle steal required for data transfer. When an XIO instruction is executed, the odd word of the I/O control command is sent to the 2250 via the storage access channel before the even word.

The I/O control commands associated with the 2250 are initiate write, initiate read, control, sense interrupt, and sense device.

Initiate Write

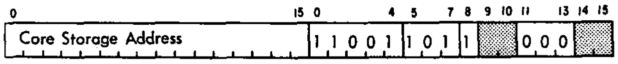
The two modifiers of the initiate write command are start regeneration (bit 8 = 0) and set programmed function indicators (bit 8 = 1). Both modifiers cause the corresponding even I/O control command word to be loaded into the 2250 address register. Words are then accessed from CPU storage by cycle stealing, starting at this address. An initiate write command can be executed only when the 2250 is not busy (not regenerating), and is treated as a no-operation command when the 2250 is busy. A control (reset display) command can be used to stop regeneration.

Start Regeneration Command: Starts execution of the display program at the address specified in the even command word.



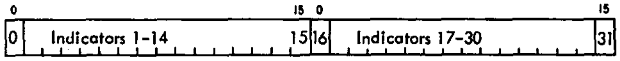
Regeneration continues under control of orders in the display program until terminated by a control (reset display) command or by a 2250 interrupt. The busy bit in the device status word is set during regeneration. (The device status word contains one bit of information for each indicator within the device.) The start regeneration command also clears the interrupt status indicator (device status word bits 0-2); if the keyboard interrupt bit is set, the command unlocks the 2250 keyboards, resets the data available bit, and clears initiate read command response word 4 and 5.

Set Programmed Function Indicators Command: Is used to load the programmed function keyboard indicators with the contents of two consecutive words in CPU storage.



The first of these two words is specified by the address word of this command. Two cycle-steal operations are performed.

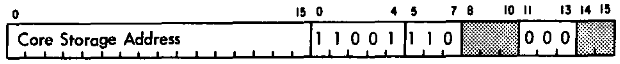
Each bit in the two indicator words corresponds to one programmed function keyboard indicator.



All 1-bits cause their associated indicators to light, and all 0-bits cause their associated indicators to be turned off. No interrupts are generated.

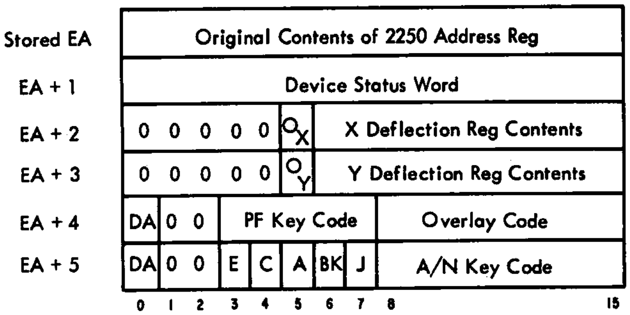
All programmed function indicators are turned off by a power-on reset (generated when 1130 power is turned on) and by a manual reset (generated when the 1131 reset pushbutton is pressed).

Initiate Read



The initiate read command causes six words of 2250 status information to be placed, by cycle stealing, into CPU storage starting at the address specified in the command. The original contents of the 2250 address register are saved (as the first word of status information) before the command address word is loaded but are not restored after execution of the command.

An initiate read command is normally issued immediately after a sense interrupt command in response to a 2250 interrupt; however, it can be executed any time the 2250 is not busy. Interrupts are not generated by the initiate read command, and the 2250 interrupt request is reset (if set). The six words of status information read by this command are as follows:



Legend: O_X = X Overflow
O_Y = Y Overflow
DA = Data Available
E = End Key
C = Cancel Key
A = Advance Key
BK = Backspace Key
J = Jump Key
PF = Programmed Function Keyboard
A/N=Alphameric Keyboard

The words reflect the status of the address register, device status word, X and Y deflection registers, programmed function keyboard, and alphameric keyboard at the time of the preceding interrupt. If a keyboard is not attached to the 2250 or does not have data available, the appropriate data available bit (bit 0) will be a 0. The device status word contents are defined in the sense device command description. The address register contents in the first word of this response, to be meaningful, may require modification as specified by address displacement bits 14 and 15 in the device status word.

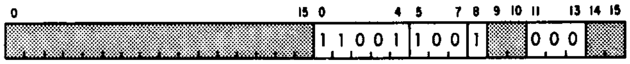
Control

During control command execution, the 2250 address register is not loaded by an address from the I/O control command. Cycle steals are not used, and interrupts are not generated. The two control modifiers are no-operation (bit 8 = 0) and reset display (bit 8 = 1).

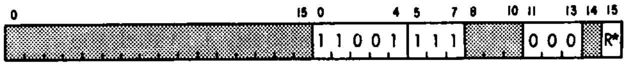
No-Operation Command: Is ignored by the 2250. It is reserved as a no-operation, and will not be assigned a function in the future.

Reset Display Command: Stops regeneration immediately and generates a unit reset in the 2250, causing all registers, controls, and keyboards to be reset. Zero is the reset state of all registers except the X and Y deflection registers, which are reset to 512 each (the center of the reference grid). The display mode is reset to graphic mode (vector), light pen control is reset to the disable detects and defer interrupts condition, all pending interrupts are cleared, and the 2250 is made not busy.

In addition, the bit configuration in the even word of the control (reset display) command (effective address) is imaged twice in the programmed function indicators, once in indicators 0-15 and again in indicators 16-31. Each 1-bit lights two indicators, and each 0-bit clears two.



Sense Device



*Reset (R): If set to 1, causes interrupt request to be reset.

This command causes the 2250 to send a device status word (Figure 69) to the 1131, where it is loaded into the accumulator. Cycle steals are not used and interrupts are not generated. If the 2250 is regenerating (is busy), only bit 8 of the device status word will be set. When the 2250 is not busy, the device status word contents describe the control status of the 2250, as follows:

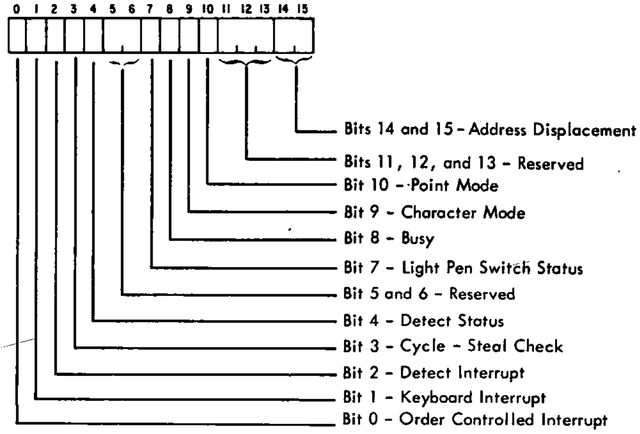


Figure 69. 2250 Device Status Word

DSW Indicators

Order-Controlled Interrupt: Is generated when the 2250 is executing either the unconditional or conditional interrupt variation of the long branch/interrupt order. The conditional interrupt variation can cause an interrupt only when the light pen detect and/or light pen switch status bits are tested successfully by the order. A 1 in device status word bit 0 indicates the occurrence of an order-controlled interrupt.

Following execution of an initiate read command, the address in the first word of status information points to the second word of the long branch/interrupt order, which may contain an address or other interrupt identification data. Bits 4 and 7 of the device status word indicate the light pen detect and light pen switch status at the time of interrupt. However, 4 is reset after it is tested successfully.

Keyboard Interrupt: Is set when a key has been pressed either on the alphameric keyboard or on the programmed function keyboard, and the next start timer order is decoded. An initiate read command reads the appropriate keyboard (response word 4 or 5). Both keyboards are locked, and light pen detects are inhibited at the time of interrupt and remain in this condition until an initiate write (start regeneration) command is executed. A 1 in device status word bit 1 indicates the occurrence of a keyboard interrupt.

If both keyboards are simultaneously activated, the programmed function keyboard is given priority by the 2250 and causes the interrupt; in this case, the alphameric keyboard is locked out. Bits 4 and 7 of the device status word indicate the light pen detect and light pen switch status at the time of interrupt.

Detect Interrupt: Is generated when the 2250 is enabled for light pen interrupts and when a detect has occurred. This interrupt is indicated by a 1 in device status word bit 2.

If the 2250 is enabled for light pen detects when a detect occurs, the address in the first initiate read response word depends on the type of data detected. Bits 9 and 10 of the device status word identify the display mode as character, vector, or point. Bits 14 and 15 of the device status word specify a displacement. This displacement should be subtracted from the initiate read response word 0 contents to obtain the address of the graphic positioning order causing display of the detected image, or the branch order to the detected character. Light pen switch status at the time initiate read was executed is indicated in device status word bit 7. In addition, the contents of the X and Y deflection registers (initiate read response words 2 and 3) might be significant. Detect Status indicates that the light pen has detected a point, vector, or character with interrupts deferred. This bit is reset whenever it is tested successfully or when device status word bit 2 is set.

Cycle-Steal Check Interrupt: Indicates the 2250 has stolen 32 consecutive cycles. This interrupt detects and stops a runaway cycle-steal situation in the 2250 caused by either 2250 malfunction or program error. If 32 consecutive cycles are stolen during any 2250 command, order, or character generation, regeneration is stopped, the busy bit is reset, cycle-steal requests are blocked, and an interrupt is requested. If a read status command is issued in response to this interrupt, no data is read since the cycle-steal request is blocked. The sense DSW command must be used following an unsuccessful read status command to examine the DSW and identify this interrupt. A reset display command or a manual reset must be given to clear the 2250 and prepare it for restarting the display.

Light Pen Switch Status: Indicates that the light pen switch was closed when last Start Timer order was executed.

Busy: Indicates that the display is currently regenerating in cycle-steal mode. This bit is always 0 if interrupt has occurred or display is not regenerating (or both).

Character Mode: Is equal to 1 when in basic or large-character mode; is equal to 0 when in graphic mode.

Point Mode: Is significant if bit 9 is equal to 0. Bit 10 is equal to 1 for point mode, or 0 for vector mode.

Address Displacement: Indicates the number of locations the address register (in first word of read status response) is ahead of address of the order being executed when detect interrupt occurred. Contains indeterminate value at any other time. Reset to 01.

IBM 2285 Display Copier

The 2285 (Figure 70) attaches directly to any 2250 Display Unit Model 4 that is equipped with the Display Copier attachment feature. The 2285 provides an 8½-by-11-inch paper copy output of the associated 2250 display upon request by the 2250 operator. The 2285 obtains analog signals and power from the 2250 to which it is attached and requires *no* programming. For operation procedures for the 2285, refer to the *IBM 2285 Display Copier Component Description,* Order No. GA27-2730.

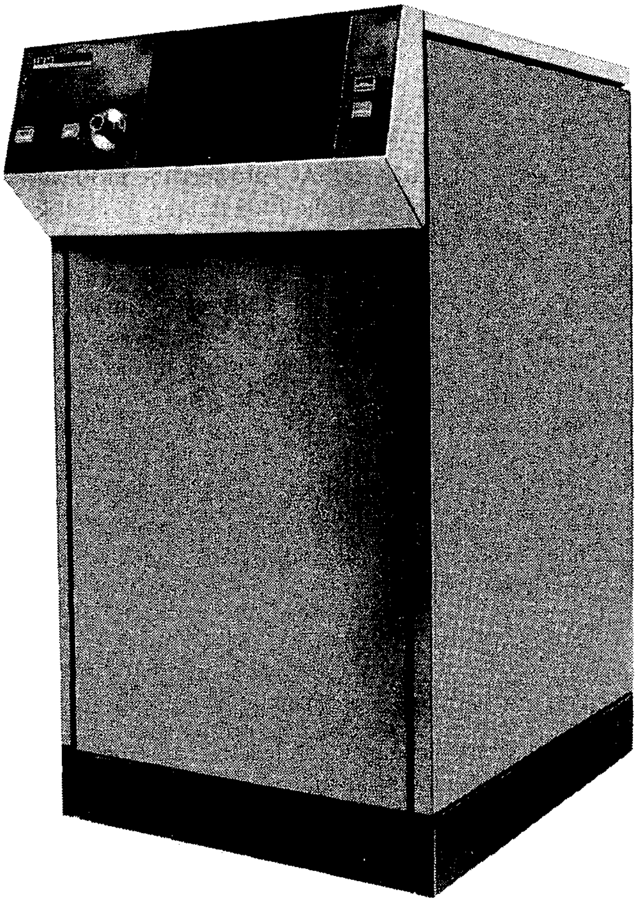


Figure 70. IBM 2285 Display Copier

USAGE METERS

2250 Model 4 Usage Meter (SAC)

This meter will run when the following conditions are present:

1. The 2250 enable/disable switch is in the enable position.
2. The 1131 usage meter is running.

The meter will continue to run simultaneously with the 1131 meter until the enable/disable switch is placed in the disable position. The condition required to change the status of the 2250 is: the CPU clock must not be running when the switch position is changed.

2250 Model 4 Usage Meter (SAC II)

This meter will run when the following conditions are present:

1. The 1133 in an enabled status.
2. The 2250 enable/disable switch is in the enable position.
3. The 1131 usage meter is running.

The meter will continue to run simultaneously with the 1131 meter until the enable/disable switch is placed in the disable position, or the 1133 is disabled. The condition required to change the status of the 2250 is: the CPU must be in the stopped or wait state when the switch position is changed.

1133 Usage Meter

The 1133 meter runs when the following conditions are present:

1. The enable/disable switch is in the enable position.
2. The 1131 usage meter is running. This meter will continue to run simultaneously with the 1131 meter until the enable/disable switch is placed in the disable position. The condition required for honoring the switch in either the enable or disable position is that the 1131 clock must be stopped. The status of the 1133 cannot be changed (enable to disable or disable to enable) while the 1131 clock is running.

Storage Access Channel

The storage access channel (SAC) provides the 1130 with additional I/O capability. If the 1403 Printer, 2311 disk storage, or 2310 Disk Storage is included in the system, it is necessary to attach the 1133 Multiplex Control Enclosure to the SAC. However, an additional channel (SAC II) is provided by the 1133 as a special feature.

Through the facilities of SAC or SAC II, the user may attach his own device or the IBM 2250 Display Unit. The customer device may interrupt on any level from 2 through 5. Any bit within ILSW's 2 through 5 that has not been previously assigned may be used. This is also true for the assignment of area codes for the customer device. The customer device may be assigned any area code that has not been previously assigned.

FUNCTIONAL DESCRIPTION

The storage access channel feature allows external devices or systems to communicate directly with the 1131 core storage unit. The transfer of data to or from core storage and the SAC takes place in one of two modes.

1. *Cycle Steal Mode:* An XIO instruction, initiate read or initiate write, gives control of the data transfers to the SAC. When the SAC transfers a word or words to or from core storage, CPU cycles are stopped and a cycle steal cycle or cycles are taken. The CPU program has no control of or awareness of the cycle steal cycles.
2. *Interrupt Mode:* The external device can cause an interrupt of the CPU program by bringing up an interrupt-request-level 2, 3, 4, or 5 line, which is serviced by the CPU in the same manner as the basic interrupts.

Because of the SAC's ability to interrupt on levels 2, 3, and 5, interrupt level status words for these levels, as well as for level 4, are provided so that the CPU program may determine which device caused the interrupt.

When an interrupt is caused by a basic device, the CPU program must give an XIO sense interrupt command. The attachment for the device places the ILSW bit for that device on the I/O input bus, and reads the bit into the B-register and transfers to the accumulator. If a device on the SAC causes an interrupt, the CPU program must give an XIO sense interrupt command, and the device must decode the command and place its ILSW bit on the I/O input bus to be read into the B-register.

When an XIO sense device command is given to the SAC, the device must decode the command and set the status bits on the I/O input bus.

The customer must provide his own interrupt routines and controlling programs.

The customer may assign to devices on the SAC any area codes that are not already assigned to a basic device on his system. The decoding of the area codes is done in the devices on the channel.

The customer may assign any bit in the ILSW to a device on the SAC that is not assigned to a basic device on his system.

No change is made to the 1131 or the SAC attachment in the assignment of area codes, interrupt levels or ILSW bits.

Cycle-Steal Priority

There are four cycle-steal priority levels. The CPU Disk Storage is on level 0; SAC is on level 1; the 1132 is on level 2; and the 2501 is on level 3. There is no polling of cycle steal requests. That means the SAC, by keeping its request active, may completely block the 1132 printer and other lower priority devices.

PROGRAMMING

The storage access channel (SAC) operates on the IBM 1130 system under direct program control or cycle-steal control.

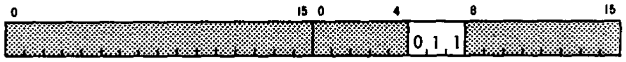
An XIO instruction addresses an I/O control command (IOCC) word, which is placed on the I/O output bus.

The devices or systems on the SAC must decode the IOCC area code to select one device or system for the operation.

The device or system selected must decode the function field and control the transfer of data to or from core storage.

I/O Control Commands

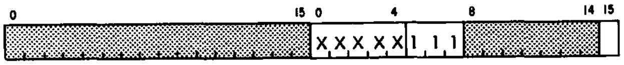
Sense Interrupt (011)



The sense interrupt IOCC is placed on the I/O output bus, and the interrupt level being serviced is sent to SAC. The device then sets its assigned bit on the I/O input bus. The CPU program then analyzes the ILSW and branches to the subroutine for the device.

The customer assigns interrupt status bits for the devices on the channel in his programs. The devices may being up an interrupt status bit assigned by the customer. The interrupt status bits may be any bits not used by a basic device.

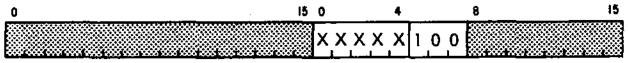
Sense Device (111)



This command sets the IOCC on the I/O output bus. The devices decode the area code and the selected device decodes the sense device function and sets the device status word (DSW) bits on the I/O input bus to read into the accumulator.

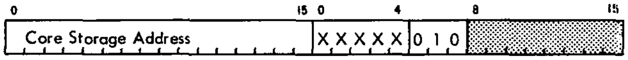
The conditions causing the interrupt are turned off by setting the modifier bit 15 to 1. If the device interrupts on more than one level, the conditions are turned off by modifier bit 15 for the highest level, bit 14 for the next highest level, etc.

Control (100)



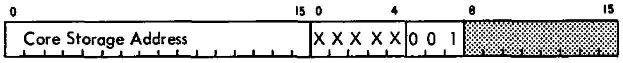
This command sets the IOCC on the I/O output bus. The devices decode the area code. The selected device decodes the control function and sets controls in the device to perform the action specified by the modifier bits (8-15) of EA + 1 or EA (address word). The device and the customer – provided programs control the function to be performed.

Read (010)



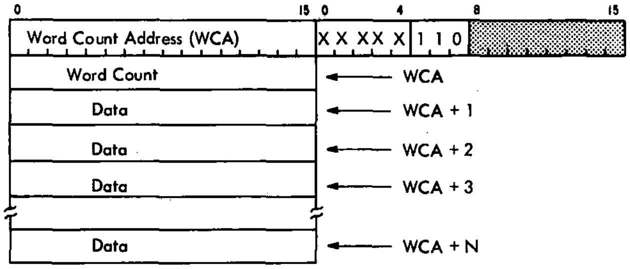
This command sets the IOCC on the I/O output bus. The devices decode the area code. The selected device decodes the read function and sets a single word on the I/O input bus.

Write (001)



This command sets the IOCC on the I/O output bus. The devices decode the area code. The selected device decodes the write function and takes the word from the I/O output bus.

Initiate Read (110)

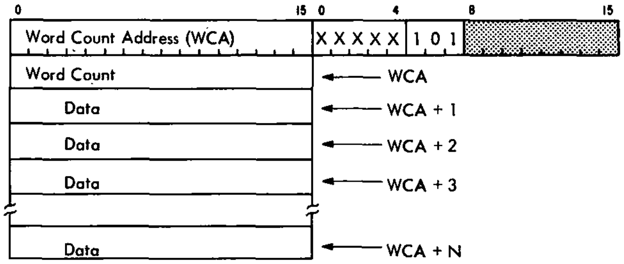


This command sets the IOCC on the I/O output bus. The devices decode the area code. The selected device decodes the initiate read function and sets the controls in the device for cycle-steal operation.

The word count address (WCA) is sent to the device.

The first cycle-steal cycle is taken and the word count is transferred to the device. The device then controls the transfer of data to the CPU core storage by cycle-steal level 1 cycles until the number of words specified by the word count has been transferred.

Initiate Write (101)



This command sets the IOCC on the I/O output bus. The devices decode the area code. The selected device decodes the initiate write function and sets the controls in the device for cycle-steal operation.

The word count address (WCA) is sent to the device.

The first cycle-steal cycle is taken and the word count is transferred to the device. The device then controls the transfer of data from the CPU core storage to the device by cycle-steal level 1 cycles until the number of words specified by the word count has been transferred.

For additional information regarding SAC, refer to *IBM 1130/1133/SAC, Original Equipment Manufacturers' Information,* Order No. GA26-3645.

Special Power Sequencing Considerations

Since no power sequencing is provided by the CPU, one of the following procedures must be followed when powering up or down of the I/O device attached to the SAC to avoid possible loss of data:

1. Apply power to the I/O device. Then power up the CPU. Reverse procedure for powering down.
2. If CPU has power applied, hold dc reset depressed while powering up or down the I/O device.
3. CPU must be in a halt condition. Turn console mode switch to SS. I/O device may now be powered up or down without affecting CPU operation. When desired status of I/O device has been achieved, program operation may resume. (Halt is defined as the CPU stopped or in a wait condition with the run light out and no I/O devices operating.)

The synchronous communications adapter special feature enables the IBM 1130 Computing System to function as a point-to-point station or multipoint data transmission terminal, using either private or commercial common-carrier (switched or non-switched) line transmission facilities. The adapter sends data to or receives data from the line transmission facilities under control of the stored program in the 1130. It operates on an interrupt request basis similar to that used by other input/output devices in the IBM 1130 Computing System.

The synchronous communications adapter (SCA) provides data interchange between remote locations and a central data-processing location. The mode of communication may be either binary synchronous or synchronous transmit-receive and requires its own program. The mode is switch-selected by the operator. IBM supplies subroutines to support both modes.

The term "synchronous transmission" is used to describe continuous bit-stream transmission, without start-of-character identification. Thus, synchronous transmission is more efficient than start/stop transmission because fewer control bits are transmitted.

Binary Synchronous Communications (BSC)

The binary synchronous mode of data transmission provides for point-to-point and multipoint operation. The 1130 may be the primary station in a communication network or it may serve as a secondary station to a larger computing system. IBM programming systems provide primary and secondary station support for point-to-point operation and secondary station support for multipoint operation.

The capability of BSC mode to operate with any six-, seven-, or eight-bit level code provides the 1130 with the ability to communicate with a greater variety of devices. It is no longer necessary for a device to adhere to an eight-bit level code in order to communicate with the 1130 system.

Certain factors should be considered in selecting a character set if the user does not use the IBM-supported character sets. The six- and seven-bit codes provide a faster and more efficient type of communication because the data sets are rated in bits per second. Thus, the fewer number of bits to make a character, the more characters may be transmitted in any given segment of time. However, the number of separate characters that can be contained in a code is decreased, proportionately, as the number of bits used to make a character is decreased.

IBM programming systems support for the SCA in the BSC mode includes a subroutine for point-to-point operation and a subroutine for multipoint operation of a secondary station. Using these programs, text may be transmitted in either normal text (extended binary-coded-decimal interchange code, System/360 and 1130 internal code) or full-transparent text. Full-transparent text uses EBCDIC communication control characters. In normal text, data may not have the same bit configuration as any control character. In full-transparent text, control character recognition is handled by a special procedure, thus making it possible to have data with the same configuration as control characters. Full-transparent text permits unrestricted coding of data within messages, and is useful in transmitting binary data, decimal data, and other data configurations.

A 2701 or 2703 Data Transmission Unit with the binary synchronous feature (SDA-2) must be attached to System/360 Models 30, 40, 50, 65, and 75 for communication in the BSC mode.

Synchronous Transmit-Receive (STR)

All synchronous transmit-receive (STR) devices use the four-of-eight line transmission code shown in Figure 71. The STR mode provides only point-to-point communication. It is used to communicate with the IBM 1009 Data Transmission Unit, the IBM 7701 and 7702 Magnetic Tape Transmission Terminals, the IBM 1013 Card Transmission Terminal, the IBM 7710 and 7711 Data Communication Units, and other STR devices.

The SCA provides the 1130 system with the ability to communicate with the communications adapter (#2073) of the Model 20 and with other System/360 configurations which have the IBM 2701 Data Transmission Unit attached. System/360 (other than Model 20) in the STR mode requires a 2701 (with the SDA-1 feature) attached to System/360.

LINE ATTACHMENT

The synchronous communications adapter is attached to either private or commercial line transmission facilities through a common-carrier data set. In the United States the interface for this data set is defined by EIA (Electronic Industries Association) Standard RS-232-B (voltage mode) and requires a Western Electric data set model 201A3, 201A4, 201B1, 201B2, 202C1, 202D1, or equivalent. Outside the United States the data set is defined by the CCITT (Consultive Committee on International Telephone and Telegraph) Standard and requires an IBM 3977 Modem or equivalent.

Graphic	4 of 8 Code		Graphic	4 of 8 Code	
	N X O R	8 4 2 1		N X O R	8 4 2 1
blank	1 1 1 1	0 0 0 0*	F	0 1 1 0	0 1 1 0
¢	0 1 1 0	1 0 1 0	G	1 0 0 0	0 1 1 1
.	1 0 0 0	1 0 1 1	H	0 1 1 1	1 0 0 0
<	0 1 1 0	1 1 0 0	I	0 1 1 0	1 0 0 1
(	0 1 0 1	0 1 1 0	J	1 1 0 1	0 0 0 1
+	0 0 1 1	0 1 1 0	K	1 1 0 1	0 0 1 0
|**	1 0 0 0	1 1 0 1	L	1 1 0 0	0 0 1 1
&	1 0 0 0	1 1 1 0	M	1 1 0 1	0 1 0 0
!	1 1 0 0	1 0 1 0	N	1 1 0 0	0 1 0 1
$	0 1 0 0	1 0 1 1	O	1 1 0 0	0 1 1 0
*	1 1 0 0	1 1 0 0	P	0 1 0 0	0 1 1 1
)	0 1 0 1	1 1 0 0	Q	1 1 0 1	1 0 0 0
;	0 0 1 1	1 1 0 0	R	1 1 0 0	1 0 0 1
¬	0 1 0 0	1 1 0 1	none#	1 0 1 0	1 0 1 0
-	0 1 0 0	1 1 1 0	S	1 0 1 1	0 0 1 0
/	1 0 1 1	0 0 0 1	T	1 0 1 0	0 0 1 1
,	0 0 1 0	1 0 1 1	U	1 0 1 1	0 1 0 0
%	1 0 1 0	1 1 0 0	V	1 0 1 0	0 1 0 1
—	0 1 0 1	1 0 1 0	W	1 0 1 0	0 1 1 0
>	0 0 1 1	1 0 1 0	X	0 0 1 0	0 1 1 1
?	0 0 1 0	1 1 0 1	Y	1 0 1 1	1 0 0 0
:	0 0 1 0	1 1 1 0*	Z	1 0 1 0	1 0 0 1
#	0 0 0 1	1 0 1 1	0	1 0 0 1	1 0 1 0
@	1 0 0 1	1 1 0 0	1	1 1 1 0	0 0 0 1
'	0 0 0 0	1 1 1 1	2	1 1 1 0	0 0 1 0
=	0 0 0 1	1 1 1 0	3	1 0 0 1	0 0 1 1
"	0 0 0 1	1 1 0 1	4	1 1 1 0	0 1 0 0
A	0 1 1 1	0 0 0 1	5	1 0 0 1	0 1 0 1
B	0 1 1 1	0 0 1 0	6	1 0 0 1	0 1 1 0
C	0 1 1 0	0 0 1 1	7	0 0 0 1	0 1 1 1
D	0 1 1 1	0 1 0 0	8	1 1 1 0	1 0 0 0
E	0 1 1 0	0 1 0 1	9	1 0 0 1	1 0 0 1

*This is correct for System/360 Programs, but is not consistent with certain other STR devices. See the specific device manual.

**Group Mark

#Record Mark

Figure 71. STR 4-of-8 Line Transmission Code

The SCA can operate in half-duplex mode using either two-wire or four-wire line transmission facilities. Data rates, selected by the machine operator, are 600, 1200, 2000, or 2400 baud (bits per second) in STR or BSC mode. In BSC mode only, operation can be at 4800 baud.

The adapter can be jumper wired to allow the program to control the data terminal ready condition in the data set interface. This selection will allow the program to control the disconnect of a switched data link. If the program-controlled disconnect feature is used, any such disconnect will prevent further operations with the data set until the communication adapter is restored to transmit mode, receive mode, or auto-answer enabled condition.

Half-Duplex Operation

Half duplex is a mode of operation wherein either terminal can transmit or receive in conjunction with the remote terminal, but neither terminal can transmit and receive data simultaneously. In effect, the operation is quite similar to a normal telephone conversation; that is, one party talks while the other party listens. During the course of the conversation, each party may alternate between talking and listening as often as necessary.

Two-Wire Operation

Synchronous transmit-receive or binary synchronous operation with a two-wire half-duplex transmission system requires a delay of approximately 200 milliseconds when the adapter switches from receiving to transmitting data. This turnaround delay allows the data set and the communication lines to reverse the direction of transmission and line echo to settle. The amount of delay is therefore related to the character of the line and data set. Line turnaround time is controlled by the data set. When this turnaround is completed, the data set signals the adapter. The adapter does not transmit until the data set signals the completion of line turnaround by activating the clear-to-send (CTS) line.

Four-Wire Operation

The adapter operates in four-wire mode with either half- or full-duplex communications facility. Four-wire, half-duplex operates the same as two wire. That is, request-to-send is controlled by the adapter. The advantage of this facility is that the 200-ms delay on turnaround is saved.

STR operation in a four-wire, full-duplex facility requires that idle characters be transmitted on the pair of wires that is not passing intelligent data. This allows the STR adapter or STR device to maintain character phase and receive-clock synchronism.

BSC operation on a four-wire, full-duplex facility eliminates turnaround time. Unlike STR, the pair of wires that is not being used at any given time does not pass idle on SYN characters. BSC mode does not maintain continuous clock synchronization but requires that the clocks be re-synchronized each time the adapter is turned around.

FUNCTIONAL DESCRIPTION

The entire synchronous communications adapter is contained within the 1131 Central Processing Unit. The adapter functions as an input/output control unit between the 1130 system and the transmission line. All data transfer is character-synchronous. This means that once an initial synchronous idle character is recognized, each subsequent character is recognized as a group of incoming data bits timed by an internal electronic clock for data terminal clocking or by the data set clock for data set clocking. Continuous regulation of the receiver's clock is provided in the case of data terminal clocking.

Incoming data from the transmission line is serial by bit and serial by character. As the data comes in, it is stored, one bit at a time, in the receive deserializer. When a complete character has been assembled, the character is transferred into the buffer register. Then the adapter initiates an interrupt request to notify the CPU that a character is ready to be read into core storage. When the interrupt request is serviced, the character is read in parallel into the high-order eight positions of a 16-bit word in core storage.

Outgoing data, from core storage to the transmission line, is taken in parallel from the high-order eight positions of the address location in core storage. The adapter initiates an interrupt request to notify the CPU that the adapter is ready to accept a character from core storage. When the interrupt request is serviced, the character is transferred in parallel to the adapter buffer register. Data from the register is subsequently sent to the transmission line one bit at a time.

Data transfer to or from the transmission line begins with the low order position. Each eight-bit character is located in bit positions 0-7 of a 16-bit core storage location as follows:

Bit Transfer Sequence	Bit Position in Core Storage
First	7
Second	6
Third	5
Fourth	4
Fifth	3
Sixth	2
Seventh	1
Eighth	0

The seventh and eighth (bit 6 and 7) bits are ignored when using a six-bit level code. The eighth bit (bit 7) is ignored when using a seven-bit level code (Figure 72).

Timers

There are three electronic timers in the SCA. Each timer is adjustable. One timer is set for 3 seconds and another is set for 1.25 seconds. The third timer (0.35 seconds) is available for sync insertions in transparent mode, BSC.

In the STR mode the three-second timer is designated as the receive timer and causes an interrupt and turns on DSW bit 3 when in the receive mode to signal the end of the listening period while establishing synchronization. This interrupt also occurs in the transmit mode if a clear to send is not received from the data set within a three-second period. Clear to send is a signal from the data set when it is ready.

The 1.25-second timer is used in the synchronize mode to signal the end of the transmission of idle characters for synchronization in the STR mode. It also causes an interrupt with DSW bit 3 on. This timeout is always coincident with a write response.

The third timer is inhibited in STR operation.

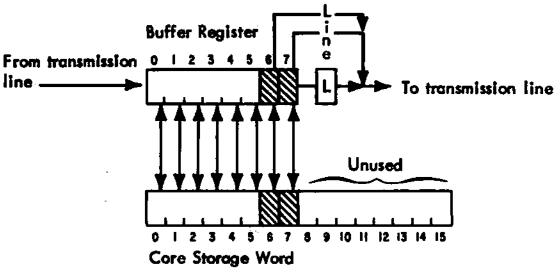
An XIO control (100) command with bit 10 on turns on a timer trigger which inhibits the 1.25- and 3-second timers when it is first issued. Issuing the command a second time removes the inhibited status, leaving the timers free to run. This command reverses the status of the timers each time it is issued.

The timers may be restarted at any time by issuing a sense device (111) command with bit 14 on if they are not inhibited.

In the BSC mode, the timers are set the same as for STR, but they have a different function. The receive timer (3 seconds) starts to run when the program enters the receive mode. The program should restart this timer when it detects the synchronous idle sequence (Figure 76). The sending station must transmit this sequence every 1.25 seconds. The 3-second timer also interrupts in the transmit mode if a clear to send is not received from the data set within 3 seconds. In either case DSW bit 3 is turned on.

The 1.25-second timer is used in the synchronize mode to signal the program that it is time to transmit the synchronous idle sequence.

The third timer (designated the program timer) will interrupt in either transmit or receive mode if it is allowed to run by the timer trigger. The IBM-supplied subroutines use this timer and therefore it is not available for customer use when these subroutines are used.



Shaded areas show unused bit positions 6 and 7 for 6-bit and 7-bit codes respectively.

Figure 72. Communication Data Flow

An XIO control (100) command with bit 10 on inhibits the 1.25- and 3-second timers and starts the program timer. If the program timer is allowed to time out it resets the timer trigger and removes the inhibit condition from the other timers. Issuing another control command with bit 10 on also resets the timer trigger. A sense device (111) command with bit 14 on will restart any timer that is running.

Synchronous Transmit-Receive (STR) Operation

In order to communicate with a STR device, the STR/BSC switch msut be placed in the STR position, and the 1130 must contain a program to control the communication. The program must use the four-of-eight code and must use STR line-control conventions. IBM provides a subroutine to control STR communication. This program is described in *IBM 1130 SCA Subroutines,* Order No. GC26-3706.

STR line-control conventions are described below. Most of the operations described are performed automatically when the IBM subroutine is used. These operations are described here for the user that wishes to write his own routines, and to provide a general understanding of STR communication.

IBM programming systems support for the SCA in the STR mode of operation uses the four-of-eight code. Two types of characters are used:

1. *Control characters* are used to control line functions; i.e., to acknowledge receipt of a message, to acknowledge synchronization, to signal the start of a message or the end of a transmission. The four-of-eight code, used by STR devices, contains special characters used to control line functions.
2. *Data characters* contain the information to be transferred to or from the adapter. The four-of-eight code contains 64 valid data characters; however, some STR devices do not utilize all of the 64 data characters. The 1130 system can recognize any or all of the 64 data characters as directed by the stored program, but the programmer should determine the character set recognized by the remote STR to avoid sending invalid characters.

Control Operations – STR

The four-of-eight code contains special characters which are reserved for control functions. These control characters and their bit structures are shown in Figure 73. Control sequences are initiated by the 1130 program and are transmitted to the remote terminal as data. The remote terminal then has the responsibility of recognizing the control sequence and responding appropriately.

All operations of the adapter are controlled by the 1130 program. The program places the adapter in either the synchronize, transmit, or receive mode. In addition the program must initially store the idle character in the sync/idle register and must generate the longitudinal redundancy check (LRC) character, which is transmitted at the end of each record.

The idle character is a special character which the adapter transmits automatically to the receiving terminal when no other data or control characters have been transferred to the adapter for transmission. This condition occurs during the synchronization mode at the start of each transmission, and when the program responds too slowly to the adapters request data. The idle character is not included in the LRC character. At least one idle character must be transmitted before each block of records. The adapter makes this transmission automatically on line turnaround.

Control characters are used generally in two-character sequences (Figure 74). Each sequence is made up of a leader character and a trailer character. Two of the control characters can be used as leaders of a control sequence. These are the transmit leader (TL) character and the control leader (CL) character. The special characters used as trailers each have two possible meanings depending on whether the TL or the CL character precedes them. For example, the INQ/ERR character is interpreted as an INQ character when preceded by the TL leader and is interpreted as an ERR character when preceded by the CL leader. The end-of-transmission sequence and the telephone sequence consist of one control character followed by one of two data characters. These data characters are interpreted as being part of a control sequence only when they are preceded by the CL character. When not preceded by the CL character, they are interpreted as data.

The inquiry control sequence is used by a terminal when it wishes to transmit a message. The terminal that is in control status may at any time send the inquiry control sequence, which notifies the other terminal of the desire to transmit and asks for permission to do so. If the other terminal is able to receive a message, it acknowledges the inquiry control sequence with an acknowledge sequence.

The start-of-record control sequence is transmitted immediately before each block of data. The start-of-record 1 (SOR 1) control sequence is transmitted before the first, third, fifth, etc., record of each message, while the start-of-record-2 (SOR 2) control sequence is transmitted before the second, fourth, sixth, etc., record of each message. This odd-even labeling of each record is used to ensure that no records of a message are lost or duplicated.

The end-of-transmittal record control sequence is sent immediately after each record of a message. The end-of-transmittal record control sequence contains the LRC character, which is used to check the validity of the transmission.

	4 of 8 Code							
Control Characters	N	X	O	R	8	4	2	1
Buffer Positions →	0	1	2	3	4	5	6	7
Idle	0	0	1	1	1	0	0	1
Start of Record 1 or Acknowledge 1 (SOR 1 or ACK 1)	0	1	0	1	0	0	1	1
Start of Record 2 or Acknowledge 2 (SOR 2 or ACK 2)	0	0	1	1	0	0	1	1
Transmit Leader (TL)	0	0	1	1	0	1	0	1
Control Leader (CL)	0	1	0	1	0	1	0	1
End of Transmission (EOT)*	0	1	0	1	1	0	1	0
Inquiry or Error (INQ or ERR)	0	1	0	1	1	0	0	1
Telephone*	0	1	0	1	1	1	0	0
Group Mark	1	0	0	0	1	1	0	1
Longitudinal Redundancy Check (LRS)**	—	—	—	—	—	—	—	—

* Also used as a data character

** This character has a 0 bit in each bit position that contained an even number of 1 bits for that bit position in the data record. If that bit position in the record had an odd number of 1 bits the LRC character ranges from all 0s to all 1s and thus, is not in the 4 of 8 code.

Figure 73. STR Control Characters

One of the acknowledge control sequences is sent by the receiving terminal after it correctly receives each block of data. This control sequence indicates to the transmitting terminal that it may proceed to send another record. The acknowledge record 1 control sequence should be sent after a record that began with the start-of-record-1 control sequence is received, while the acknowledge record 2 control sequence should be sent after a record that began with the start-of-record-2 control sequence is received. This assures the sender that the receiver has not lost a record. The last acknowledgment is always sent in response to an inquiry.

The repeat last record (error) control sequence is sent by a receiving terminal if it receives a block of data that is in error. This sequence notifies the transmitting terminal that it should repeat the transmission of the last record.

The end-of-transmission control sequence is sent by the transmitting terminal after it has sent the last record of a message. This indicates that the message has been sent completely. A receiving terminal answers the end-of-transmission control sequence by sending back an end-of-transmission control sequence, thereby notifying the transmitting terminal that the receiving terminal has received the full message. After the transmission of these two end-of-transmission control sequences, the two terminals return to synchronize mode of operation and exchange end-of-file idle sequence (handshake).

The telephone control sequence can be sent by either terminal and indicates that the terminal operator desires voice communication, via the handset, with the other terminal operator.

Synchronize Mode – STR

The synchronize mode (entered by a synchronize IOCC, with bit 11 = 1) provides a means of synchronizing the transmitting and receiving terminals to ensure the proper recognition of data bits and characters as they are transmitted between terminals. The synchronize mode consists of the transmission of a series of idle characters for 1.25 seconds, followed by a control sequence and then turning around and listening for a similar series of characters from the other terminal for 3 seconds. The time intervals for transmit (1.25 seconds) and receive (3 seconds) are controlled by timers in the adapter. The timers are under control of the program. The character used in the synchronization sequence is called an idle character.

	Control Character Sequence	
Control Sequence	Leader Character	Trailer Character
End of IDLE (EOI)*	CL	1 IDLE
Inquiry (Synchronized ?)*	TL	INQ
Acknowledge (Synchronized)	CL	ACK 2
Telephone Sequence *	CL	TEL
Acknowledge Telephone *	CL	TEL
Start of Record 1 (SOR 1) 1st or odd numbered record	TL	SOR 1
Start of Record 2 (SOR 2) 2nd or even numbered record	TL	SOR 2
End of Transmittal Record (EOTR)	TL	LRC
Acknowledge Record 1	CL	ACK 1
Acknowledge Record 2	CL	ACK 2
Repeat Last Record (ERROR)	CL	ERR
Intermediate LRC**	GM	LRC
End of Transmission (EOT)*	CL	EOT
Acknowledge EOT *	CL	EOT

*These sequences are always preceded by a 1.25 second transmission of IDLE characters.

** This sequence may be required on some terminals i.e. 1013, 7701, 7702

Figure 74. Control Sequences

At the end of the 1.25-second transmission time, the transmitting terminal sends an end-of-idle control sequence – a control leader (CL) followed by an idle character (Figure 73). This control sequence signals the receiving (remote) terminal to change from receive mode to transmit mode. When the turnaround is completed (200 ms for two-wire half-duplex) the remote terminal transmits the idle character for 1.25 seconds. At the end of this time the remote terminal sends the end-of-idle sequence. If neither terminal has a message to transmit, the synchronization sequence continues.

Transmit Mode – STR

When a terminal has a message to transmit, that terminal sends 1.25 seconds of idle characters (caused by a synchronize IOCC, with bit 11 = 1) followed by the inquiry sequence. This sequence informs the remote terminal that a message is about to be transmitted. The remote terminal, if it is in synchronization and is ready to receive, sends an acknowledge control sequence (ACK2). On receipt of the acknowledge sequence, the transmitting terminal transmits its message.

The first two characters of a message are the start-of-record-1 sequence. This sequence is preceded by one or more idle characters. This sequence is followed by the message data characters for this record. Some terminals may use or require an intermediate block check. (This sequence is GM-LRC.) At the end of the record, the end-of-transmittal-record (EOTR) sequence is sent. This sequence consists of a TL character and a longitudinal redundancy check (LRC) character. Two functions are performed by this sequence: it indicates the end of the record, and provides (via the LRC character) the receiving terminal with a method of checking for a complete message. The receiving terminal acknowledges the EOTR by sending the acknowledge 1 or 2 sequence (if LRC compares) or by the error sequence (if LRC does not compare).

Messages which contain more than one record indicate the start of the second record by sending a start-of-record-2 sequence. The start-of-record-1 sequence is used each time an odd-numbered record is transmitted, and the start-of-record-2 sequence is used each time an even-numbered record is transmitted. The use of the two different start-of-record sequences enables detection of lost or duplicated data records from a terminal.

When the receiving terminal has acknowledged the correct receipt of the last record of a message, the transmitting terminal sends the end-of-transmission sequence. This sequence consists of a CL character and an end-of-transmission (EOT) character. The receiving terminal acknowledges the EOT sequence by returning the same sequence. The terminals, if so programmed, return to the synchronize mode.

Receive Mode – STR

In the receive mode, the adapter accepts data from other line devices and transfers it to the 1130 core storage. Prior to the transfer of data, the transmitting and receiving terminals must be synchronized.

In the receive mode, the adapter compares the incoming data to the character in the idle register. After at least one idle character has been recognized, the first non-idle character detected and all subsequent characters including idles are transferred into core storage. Idle characters and control sequences are not included in the LRC. When the transmitting terminal signals the end of a record, the 1130 program checks the transmitted LRC character with the one compiled from the received record. If the two LRC characters are the same, the 1130 program generates the appropriate acknowledgment, which is then sent from the adapter to the transmitting terminal. If the LRC characters are not the same, the 1130 program responds with an error sequence which is then sent from the adapter to the transmitting terminal. Normally, the 1130 program requests that the previous record be transmitted again. The number of transmission attempts is controlled by the programmer and may vary.

Special Programming

Special programming techniques are required in STR when an 1130 is used to communicate with a hardware device such as a 1013, 1009, or 7702. The special technique is required when either a 201 Data Set or an IBM 3977 Modem is used in a two-wire operation. No idles are received from these devices before the control leader (CL) or transmit leader (TL). Since the 1130 SCA requires at least one recognizable character before interrupting the CPU, the following special technique should be used:

1. If the 1130 is the slave, it will be receiving records. After writing the acknowledgment character (ACK 1, ACK 2, or ERR), the program should load the sync/idle register with the TL. Since the TL is now the recognizable character, it is not loaded into the buffer for the CPU to read. The first character which interrupts the CPU is the trailer. The program must indicate to itself that the TL has already been received. This should be done when the first read interrupt occurs. If the 1130 times out, the remote station may send a message beginning with a TL or it may begin "handshaking" beginning with an idle character. To cope with either possibility, after a time-out, the 1130 program loads the sync register with a TL, and if another time-out occurs, the sync register is then loaded with an idle character. Once character phase is reestablished, the alternating of TL or idle characters ceases.

2. If the 1130 is the master, it is sending records. After writing an INQ, the LRC character of an EOTR, or the last character of an abort sequence (idle), the program should load the sync/idle register with the CL. Since the CL is now the recognizable character, it is not loaded into the buffer for the CPU to read. The first character which interrupts the CPU is again the trailer. The program must indicate to itself that the CL has already been received. This can be done either at the time the sync/idle register is loaded with the CL or when the first read interrupt occurs.

   For both cases (1 and 2), the sync/idle register should be reloaded with the idle character prior to each transmission. An idle character should also remain in the sync/idle register after the program writes the idle of an end-of-idle sequence, or the TEL character, or the EOT character.

   Since the above technique works for all data sets and STR devices, it is recommended that it be followed.

Binary Synchronous Communications (BSC) Operation

In binary synchronous operation the receiving terminal's ability to interpret the data it receives is the prime consideration in selecting the code to use for communication.

A variety of codes for communication is available. The user may select any code of six, seven, or eight bits. IBM programming systems for the 1130 use the extended binary-coded-decimal interchange code (EBCDIC) communication control characters for all BSC operations. Figure 75 shows the control characters and Figure 76 shows the sequences in which they are used. In full-transparent text, control character recognition should be handled by a special procedure, thus making it possible to have data with the same configuration as control characters. All characters are transferred to core storage in the CPU for program interpretation. (Refer to *IBM 1130 Synchronous Communications Adapter Subroutines*, Order No. GC26-3706, for information concerning terminals and character codes supported.)

In the selection of a code, care must be taken in selecting the proper SYN character. If only two characters are used for synchronization, the first bit of the SYN character must be 0. A minimum of two characters are required for character-phase synchronization. The first bit of the character must be 0. The bit configuration must not be a repeating bit pattern (that is, the first of each character must be recognizable).

If the data set uses business machine clocking, several preceding characters must be transmitted to establish bit phase in the data set before the SYN characters mentioned above can be transmitted. These characters can be of the same bit configuration as the character-phase SYN character. The prime requirement for the characters used to establish bit phase is that the total line transitions must be a minimum of 16 before the data set is ready to accept the SYN characters for character phase.

Character	Bit Configuration		Meaning
	0 1 2 3 4 5 6 7	Hex	
SYN	0 0 1 1 0 0 1 0	32	Synchronous Idle
DLE	0 0 0 1 0 0 0 0	10	Data Link Escape
ENQ	0 0 1 0 1 1 0 1	2D	Enquiry
SOH	0 0 0 0 0 0 0 1	01	Start of Heading
STX	0 0 0 0 0 0 1 0	02	Start of Text
ETB	0 0 1 0 0 1 1 0	26	End of Transmission Block
ETX	0 0 0 0 0 0 1 1	03	End of Text
EOT	0 0 1 1 0 1 1 1	37	End of Transmission
ITB	0 0 0 1 1 1 1 1	1F	End of Intermediate Block
NAK	0 0 1 1 1 1 0 1	3D	Negative Acknowledgement
*ACK 0	0 1 1 1 0 0 0 0	70	Positive Acknowledgement (even record)
*ACK 1	0 1 1 0 0 0 0 1	61	Positive Acknowledgement (odd record)
*RVI	0 1 1 1 1 1 0 0	7C	Reverse Interrupt
*WACK	0 1 1 0 1 0 1 1	6B	Wait Before Transmit Positive Acknowledgement
PAD	1 1 1 1 1 1 1 1	FF	Transmission Trailer

* Control characters when preceded by DLE

Figure 75. Binary Synchronous EBCDIC Control Characters

Characters	Meaning
ENQ	Enquiry
SOH	Start of Heading
STX	Start of Text
DLE STX	Start of Transparent Text
ETB CRC-16*	End of Block
DLE ETB CRC-16	End of Transparent Block
ETX CRC-16	End of Text
DLE ETX CRC-16	End of Transparent Text
DLE ACK 1	Acknowledgement of Odd Record
DLE ACK 0	Acknowledgement of Even Record
NAK	Negative Acknowledgement
EOT	End of Transmission
DLE EOT	Disconnect Signal
SYN SYN	Synchronous Idle (Normal)
DLE SYN	Synchronous Idle (Transparent Text)
ITB CRC-16	End of Intermediate Block
DLE ITB CRC-16	End of Intermediate Transparent Block
DLE WACK	Wait Before Transmit Positive Acknowledgement
DLE RVI	Reverse Interrupt
STX ENQ	Temporary Text Delay
DLE DLE	Data DLE in Transparent Mode

* CRC-16 is a 16-bit cyclic check character accumulated from text and heading data.

Figure 76. Binary Synchronous Control Sequences

Control Operation — Binary Synchronous

The binary synchronous communications control procedures are generally independent of the transmission code. Any code having a fixed number of bits (six, seven, or eight) per character may be used if the ten control characters are set aside and a proper choice is made for the synchronous idle character. The EBCDIC control sequences are presented in this manual (Figure 75).

The control sequences are initiated by the 1130 program and transmitted to the remote terminal as data. The remote terminal then has the responsibility of recognizing the control sequences and responding appropriately.

All operations of the adapter are controlled by the 1130 program. The program places the adapter in either the synchronize (transmit) or the receive mode. In addition the program must initially store the synchronous idle (SYN) character in the sync/idle register. The program also accumulates the block check character (CRC-16), which is transmitted at the end of each record. Because the CRC is 16 bits long, two 8-bit characters must be transmitted and received.

In BSC, data may be transmitted in two modes: normal (EBCDIC) text and full-transparent text. In normal text mode, data may not have the same bit configuration as any control character. In full-transparent text, data may contain any bit configuration since control character recognition is handled by a special procedure. Full-transparent text is quite useful in transmitting machine language and other codes that may contain control characters.

In full-transparent mode, the DLE STX sequence is a special sequence that is transmitted prior to transmitting full-transparent text. When a receiving terminal receives this sequence it will stop checking for control characters and treat all subsequent characters as transparent text. The only control character that is recognized is another DLE character. The detection of another DLE character switches the mode back to normal text mode, and the receiving terminal will start checking for control characters. If the next character is DLE or SYN the receiving program will treat the character as data or as synchronous idle and will return to the transparent mode. Therefore, in full-transparent text mode, all control characters, including SYN, must be preceded by the DLE character to be recognized by the receiving terminal. In full-transparent mode the program must store the DLE character in the sync/idle register. The SYN character must be stored after leaving full-transparent text mode.

Line Turnaround

When a terminal wishes to transmit, it sends two SYN characters followed by the ENQ character. Then the terminal goes to the receive mode and waits for an acknowledgment from the receiving terminal. The receiving terminal detects the ENQ character as a request from the transmitting terminal, goes to the transmit mode, and replies with a positive acknowledgment (ACK0) if it is ready to receive. When the transmitting terminal receives the positive acknowledgment, it may start to transmit its record. If the receiving terminal is not ready to receive, it should respond with the NAK character (negative acknowledgment). If the terminal is unable to respond, the transmitting terminal will time out in 3 seconds.

Several of the control characters when detected by the program should cause line turnaround; that is, the transmitting terminal switches to the receive mode and the receiving terminal switches to the synchronize (transmit) mode.

Within the program, the end-of-block and the end-of-text (ETB and ETX) characters (when transmitting without checking), also cause line turnaround. IBM subroutines always use the block check character. The acknowledgments alternate: ACK1 for the first record and all succeeding odd records, and ACK0 for the second record and all succeeding even records. If the block check character is used, the line turnaround follows it. When a station is through transmitting, it may relinquish its right to transmit by sending the end-of-transmission (EOT) character. The EOT character does not require an acknowledgment. The right to transmit reverts back to the master station or to contention if a master station is not designated.

Multi-Point Operation

In multi-point, centralized operation, IBM programming systems include subroutines that permit the 1130 to operate only as a slave station. Programs to support noncentralized operation must be supplied by the user. A slave station is one that may respond to a call from the control (master) station but cannot initiate the call. Initialization is performed when the control station sends polling or selection addresses. A particular polling address gives a unique station on the line an opportunity to transmit to the control station. The polled station responds with a positive response (data transmission) or a negative response (EOT). Selection addresses are used to request a particular station to receive data transmission. A selected station responds with its status, ready to receive (ACK0) or not ready to receive (NAK).

A nonselected terminal should restart the timers and reset character phase on recognition of all turnaround sequences seen on the line.

In noncentralized operation, the operation is similar to centralized operation except the selected station (after being polled) must respond with its address and the address of the station to which it wishes to transmit. The selected station must reply with its address and a positive acknowledgment if it is ready to receive or a negative acknowledgment if it cannot receive.

Receive Mode — Binary Synchronous

In the receive mode, the adapter accepts data from the transmission line and transfers it to the 1130 core storage. Prior to the transfer of data, the adapter must be synchronized with the transmitting terminal. An initiate read command (110) with all modifier bits (8-15) set to 0 places the adapter in a receive mode.

In the receive mode, the adapter compares the incoming data to the character in the sync/idle register. After at least two SYN characters have been recognized, the first non-SYN character detected and all subsequent characters including SYN characters are transferred to core storage. The receive mode is terminated by the program when it detects a valid turnaround sequence.

For a slave station, if a receive time-out occurs, an end operation command should be used to reset the clock and character phase. The slave should issue an initiate read command immediately after the end operation command. If a receive time-out occurs, the master should issue an initiate write command to send ENQ.

Data Transmission – Binary Synchronous

Data sets (with business machine clocking) require a minimum of 16 line transitions to establish bit phase. The IBM-written program uses the normal SYN character (hexadecimal 32) for this. However, any character may be used. The 16 line transitions must precede the two SYN characters used to establish character phase in the SCA adapter. If data-set clocking is utilized, these preceding line transitions are not required.

In full-duplex operation, the SYN characters must be preceded by a pad character. The pad character cannot be another SYN character, but can be a marking line character (hexadecimal FF).

When a terminal has a message to transmit, the terminal sends the synchronous idle sequence followed by the enquiry control character. The enquiry character informs the remote terminal that a message is about to be transmitted. The remote terminal, if it is ready, sends the SYN characters and acknowledges by sending the acknowledge control sequence (DLE ACK0). Upon receipt of the acknowledge control sequence, the transmitting terminal transmits its message. The entire message, including control characters and check characters, is generated and transmitted from core storage under control of the stored program in the CPU.

Synchronize Mode – Binary Synchronous

The synchronize mode in binary synchronous communication is a transmit mode which allows a timeout to occur if the transmission is longer than 1.25 seconds. The program must insert the synchronous idle sequence after this timeout to ensure that the receiving terminal remains synchronized. Data transmission may continue after the synchronous idle sequence. The receiving terminal will time out if it does not receive the synchronous idle sequence within 3 seconds. A control command (100) with bit 11 set to 1 places the adapter in the synchronize mode.

Transmit Mode – Binary Synchronous

The transmit mode may be used in binary synchronous operation in lieu of the synchronize mode where a time-out is not required or desired. An initiate write command (101) with bit 9 set to 0 places the adapter in the transmit mode.

Special Programming

Most binary synchronous communications equipment with which the 1130 communicates generates block check characters (CRC-16) and expects block check characters following each line turnaround character. For this reason, the 1130 must generate and transmit these characters any time it communicates with such equipment. The block check characters must be accumulated under program control in the 1130.

The block check characters are formulated based upon the division algorithm for polynomials over the field of integers modulo two. In this field, addition and multiplication are performed according to the following rules:

Addition (Exclusive OR)	Multiplication
1 + 1 = 0	1 • 1 = 1
0 + 0 = 0	0 • 0 = 0
0 + 1 = 1	0 • 1 = 0
1 + 0 = 1	1 • 0 = 0

Operating under these rules, addition and subtraction are the same; in other words, $a + b = a - b$.

The block check characters are formed as a 16-bit remainder $[R(x)]$ from the polynomial division:

$$\frac{B(x) \cdot x^{16}}{x^{16} + x^{15} + x^2 + 1}$$

where $B(x) = b_n x^n + b_{n-1} x^{n-1} + \ldots b_1 x^1 + b_0 x^0$.

The following equations illustrate the computation of the block-check characters for the two-character transmission G4:

Bits as transmitted: 1100 0111 1111 0100

$$B(x) = x^{15} + x^{14} + 0x^{13} + 0x^{12} + 0x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + 0x^3 + x^2 + 0x + 0$$

$$B(x) \circ x^{16} = x^{31} + x^{30} + 0x^{29} + 0x^{28} + 0x^{27} + x^{26} + x^{25} + x^{24} + x^{23} + x^{22} + x^{21} + x^{20} + 0x^{19} + x^{18} + 0x^{17} + 0x^{16} + 0x^{15} + 0x^{14} + 0x^{13} + 0x^{12} + 0x^{11} + 0x^{10} + 0x^9 + 0x^8 + 0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x + 0$$

$$\frac{B(x) \circ x^{16}}{x^{16} + x^{15} + x^2 + 1} = x^{15} + x^{10} + x^8 + x^6 + x^4 + x^3 + x + 1 \text{ with a remainder } R(x)$$

$$R(x) = 0x^{15} + 0x^{14} + 0x^{13} + x^{12} + 0x^{11} + 0x^{10} + 0x^9 + 0x^8 + 0x^7 + 0x^6 + x^5 + x^4 + 0x^3 + x^2 + x + 1$$

Block check character = 0001 0000 0011 0111

This 16-bit block check character is transmitted as two 8-bit characters by the transmitting station and compared at the receiving station with block check characters accumulated by the same algorithm. If the block check characters are equal, the transmission is acknowledged as correct by the receiving station.

Characters are included in the block check accumulation under the following rules. In nontransparent mode:

1. The first SOH or STX character clears the count and is not included in the accumulation.
2. All other characters after the initial SOH or STX are included in the block check character accumulation, except SYN.
3. An ITB character encountered in the message is included in the accumulation. The block check characters are transmitted (or received and compared) immediately following the ITB character. The block check characters are cleared and a new accumulation starts with the next non-SYN character received.
4. An end of block line turnaround character (ETB or ETX) is included in the block check character summation. The block check characters are transmitted (or received and compared) immediately following the ETB or ETX. The block check summation is ended by either of these line turnaround characters and will not resume until a new SOH or STX character is encountered.

In transparent mode:

1. The block check summation is initiated by the first appearance of an SOH or an XSTX character. This character clears the summation and is not included in the summation.
2. All characters transmitted after the initial SOH or XSTX are included in the block check summation up to and including the first end of block line turnaround character (XETB, XETX in transparent blocks, ETB or ETX in nontransparent blocks). An XSTX following an XITB is included as the first character of the summation following clearing of the summation after checking is performed.
3. The idle character (XSYN) is not included in the block check summation.
4. The DLE character used to designate that the control character or transparent DLE character transmitted is to be used as a control character is not included in the block check summation except for the DLE STX sequence following heading information, ITB, or XITB.

PROGRAMMING

All adapter operations are programmed using the 1130 XIO instruction (see execute I/O description in this manual). The effective address position of the XIO instruction specifies the address of the two-word IOCC which is required for the desired operation.

The adapter interrupts the 1130 system program on interrupt level 1. Bit position 1 of this interrupt level status word (ILSW) indicates that the interrupting device is the adapter. The program then senses the device status word (DSW).

The DSW is generated by the adapter to indicate the cause of the interrupt (Figure 77). The DSW bit positions indicate the following conditions:

Bit 0 – The adapter is in receive mode (or diagnostic mode) and the buffer register in the adapter contains a data character which should be transferred into the 1130 core storage.

Bit 1 – The adapter is in transmit mode (or diagnostic mode) and requires a data character from the 1130 core storage for transmission.

Bit 2 – This bit indicates an error condition: data overrun or character gap.

Data overrun indicates that a character was still in the buffer when another character came, either from the transmission line (receive mode) or from core storage (transmit mode). This condition results in a loss of data. In the transmit mode, data overrun is the result of a program sending another character to the adapter without an interrupt request from the the adapter. In the receive mode, this condition is the result of a program operating too slow; that is, a character is received from the transmission line before the preceding character has been transferred to core storage (interrupt not yet serviced).

Character gap indicates that the data characters are being received by the buffer too slowly for correct adapter operation. In the transmit mode, the program is operating too slowly. (Note: The adapter automatically transmits the character in the sync/idle register.) In the receive mode, the program requested another character from the adapter without an interrupt request from the adapter.

Bit 3 – In STR this bit indicates the end of the 1.25-second timeout for transmission of idle characters, or the end of the three second listening time for synchronization. In BSC this bit indicates it is time to insert the synchronous idle sequence in synchronize mode, or a receive time-out occurred in the receive mode or a sync insertion is required in transparent mode.

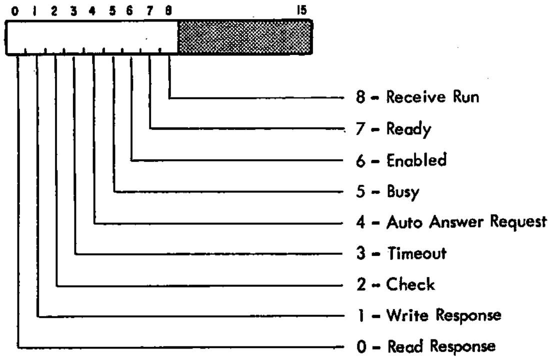


Figure 77. Device Status Word

Bit 4 – This bit indicates that the data-set phone is ringing. Bit 4 is used only if the auto-answer feature is in the data set.

Bit 5 – This bit indicates that the adapter is in either the receive or transmit mode.

Bit 6 – This bit indicates that the adapter has been enabled for an auto answer request interrupt. (See Bit 4.)

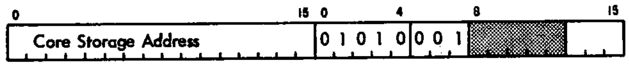
Bit 7 – This bit indicates that the data set is connected and ready to receive, synchronize, or transmit data.

Bit 8 – This bit is used with two-wire half-duplex STR systems only. It indicates that the adapter is in the "slave" mode. In slave mode the adapter transmits the normal acknowledge responses but does not transmit data records. Receive and transmit clocks are tied together. The transmit clock is corrected with the receive clock when correction is required.

I/O Control Commands (IOCC)

The adapter is addressed by the five-bit (bits 0 through 4) device (area) code in the IOCC. This code is 01010.

Write (001)



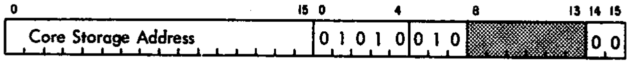
A write command without a modifier bit instructs the 1130 to transfer the contents of the specified core storage address to the adapter buffer. The adapter then serializes the contents of the buffer register onto the transmission line.

If modifier bit 13 is set, this command is used to set the sync/idle register. The 1130 transfers the data to the sync/idle register in the adapter. The idle character is transmitted during the synchronize mode or when the adapter is in transmit mode and has not received a data character for transmission.

Modifier bit 14 turns on the audible alarm trigger in the adapter.

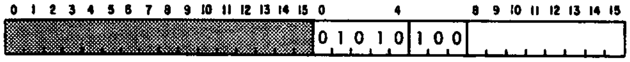
Modifier bit 15 turns off the audible alarm trigger.

Read (010)



The read command instructs the 1130 to transfer the contents of the adapter buffer to the core storage location specified in the address portion of the command. Modifier bits 14 and 15 must be 0's in application programs. When on, they are used for reading diagnostic words.

Control (100)



The control command is always used with a modifier bit. This command causes the adapter to accomplish the functions specified by the modifier.

Modifier bit 8, when set to 1, enables the adapter for auto answer operation. Auto answer allows the adapter to interrupt the 1130 program in response to a telephone ring from the remote terminal.

Modifier bit 9, when set to 1, disables the auto answer operation and does not allow a telephone ring from the remote terminal to interrupt the 1130 program.

Modifier bit 10 reverses the status of the timers from run to inhibit or from inhibit to run.

Modifier bit 11 sets the adapter to the synchronize mode. This is used to establish and maintain synchronization in the STR mode with minimum program interruption. Idles are transmitted without the program being interrupted until transmit time-out occurs.

In binary synchronous mode, modifier bit 11 allows the adapter to transmit in the synchronize mode. Write responses occur normally. A transmission longer than 1.25 seconds causes a time-out interrupt. The program must transmit the synchronous idle sequence before continuing to transmit data. The synchronous idle sequence is the only synchronization necessary in BSC. This usually consists of two sync characters.

If the SCA is not already in transmit mode when this command is given, a turnaround occurs. The turnaround, with a 1 in position 0, 1, 3, or 4 of the address word in the IOCC, resets the corresponding position of the DSW.

The on condition of modifier bit 12 places the adapter in a diagnostic condition. Bit 12 should be off for all application programs. Because of the short time between interrupts in this condition, the diagnostic program should be run alone.

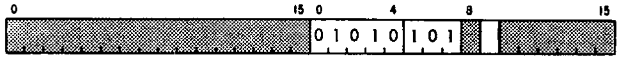
Modifier bit 13 is the end operation command. This command resets the adapter regardless of the mode of operation. If the adapter is in the transmit mode, the reset is delayed until a character gap of one character is detected. This allows the last character to get through the data set before the adapter is reset. It then resets the adapter and also resets the timers used in the synchronization mode and disconnects the adapter from the communication line if a switched network is used. In binary synchronous mode, this command should be issued after a receive time-out to reset the clock.

Modifier bit 14 is used to set the adapter for a six-bit character frame. Setting bit 14 automatically resets the seven-bit character mode.

Modifier bit 15 is used to set the adapter for a seven-bit character frame. Setting bit 15 automatically resets the six-bit character mode.

Both frame size modes are reset when the adapter leaves both the receive and transmit mode. Thus it becomes necessary to reenter the proper mode after each line turnaround. Attempting to set both bit 14 and bit 15 with the same instruction is ambiguous and may result in an error.

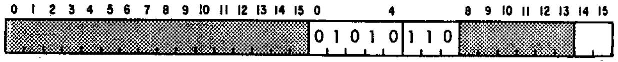
Initiate Write (101)



The initiate write command places the adapter in the transmit mode of operation.

If the SCA is not already in transmit mode when this command is given, a turnaround occurs. The turnaround, with a 1 in position 0, 1, 3, or 4 of the address word in the IOCC, resets the corresponding position of the DSW (i.e., read response, write response, timeout, or auto answer request). Initiate write with modifier bit 9 on resets all conditions in the adapter.

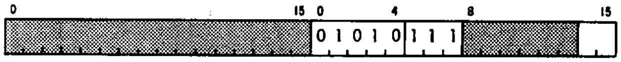
Initiate Read (110)



The initiate read command places the adapter in the receive mode of operation.

An initiate read command with modifier bit 14 on sets the send/receive run trigger and places the adapter in slave mode operation for STR operation. This mode of operation is used with two-wire half-duplex systems. In the slave mode the adapter should not be programmed to transmit data records to the master. The only transmissions that the adapter will make are the normal responses to the inquiry from the master and the normal acknowledgments. The start-read command and a modifier bit 15 clears the send/receive run trigger and removes the adapter from slave mode operations. This clearing places the adapter in the master mode, which is used for the transmission of data.

Sense Device (111)



The sense device command instructs the 1130 to sense the device status word (DSW). The DSW is generated by the adapter to indicate the cause of the interrupt. The DSW for the adapter is shown in Figure 77.

Sense DSW with modifier bit 14 on will restart the timers. If the synchronous idle sequence is received while in BSC receive mode, the program should restart the timer. If the timer is not reset within 3 seconds, the adapter will cause a time-out interrupt. Sense DSW with modifier bit 15 on resets the device status word responses.

TIMING FOR SCA PROGRAMMING

In order to prevent an overrun on receive a character must be sent from the SCA buffer following a read response interrupt within the period shown in Figure 78. Also to prevent a character gap on transmission, a character must be written to the SCA buffer following a write response within the period shown in Figure 78.

Time Between Characters

Char. Size Baud	6 Bit	7 Bit	8 Bit
600	10.0 ms	11.6 ms	13.3 ms
1200	5.0 ms	5.8 ms	6.6 ms
2000	3.0 ms	3.5 ms	4.0 ms
2400	2.5 ms	2.9 ms	3.3 ms
4800*	1.25 ms	1.45 ms	1.65 ms

Character Rate

Char. Size Baud	6 Bit	7 Bit	8 Bit
600	100 cps	85.7 cps	75 cps
1200	200 cps	171 cps	150 cps
2000	333.3 cps	286 cps	250 cps
2400	400 cps	343 cps	300 cps
4800*	800 cps	686 cps	600 cps

*BSC mode only.

Figure 78. Transmission Timing

Overlapping Input/Output Operations And Throughput Considerations

The 1130 system permits input/output devices to operate simultaneously; that is, to overlap their operation with other functions of the CPU. Overlapping I/O operations provides increased data throughput and more efficient utilization of the central processing unit.

This section will aid the programmer wishing to maximize I/O throughput in the 1130 system. A primary concern, however, is the possible loss of data if the capabilities of the system are saturated by overlapping too many operations. Loss of data can occur only on an 1130 system which has either a 1442, 1132, or synchronous communications adapter, and then only if too many I/O operations are overlapped with these devices.

The material provided here may be used by the programmer to calculate the maximum throughput for his system without loss of data.,

Cycle-Stealing Concept

The cycle-stealing concept of the 1130 permits the CPU program to start an operation on an I/O device and then continue the mainline program while the I/O device is performing its operation. Each I/O device that operates in this manner takes (steals) a cycle from the CPU when it is needed.

The CPU is "tied up" only one cycle while a data character is being transferred. The frequency at which devices steal cycles depends on the type of device.

Since the CPU is much faster than any I/O device on the system, the CPU may be performing another function, such as arithmetic, at the same time an I/O operation is being performed. In fact, several I/O operations may be overlapped with each other and with other CPU functions. For example, the data transfer rate of a single disk storage drive is 27.8 us. per word. Thus, each disk storage drive read/write operation requires one CPU cycle (3.6 or 2.2 us.) out of each 27.8 us., leaving 25.6 or 24.2 us. of CPU time available for other functions. If two single disk storage drives are transferring data at the same time, then 23.4 or 20.6 us. is available for other CPU functions.

Direct Program Control (via Interrupt)

Direct program control applies to I/O devices that are totally dependent upon the CPU program. These devices interrupt the mainline program by requesting service. Once the service request is honored, the actual transfer of data also requires programmed commands. Servicing interrupt requests generally requires several CPU instructions (a subroutine). The system programmer must calculate the time of the I/O subroutines to determine the maximum data throughput (without data loss) of his system configuration. See *IBM 1130 Subroutine Library*, Order No. GC26-5929, for the execution times for IBM-supplied subroutines.

Conditions causing I/O interrupt requests are preserved in the device status word (DSW) of the I/O devices until the interrupt is accepted by the CPU.

The sequence of events after an interrupt request is received is:

1. Instruction in progress is allowed to continue to completion.
2. Interrupt request is accepted if higher level interrupt is not in progress.
3. Branch to an appropriate interrupt subroutine to service the request.
4. Housekeeping program routines must store all registers and linkage addresses to allow mainline program to continue after the interrupt is serviced.
5. Examine the interrupt level status work (ILSW) to determine the interrupting device.
6. Examine the DSW of the interrupting I/O device to determine the action required to service the request and reset the interrupt response bit in the DSW.
7. Service the request and restore the necessary register and address information to resume the mainline program operation or to service other interrupts. Turn off the interrupt level with a BOSC instruction (a BSC instruction with a bit 9 set to 1).

Exposure to Loss of Data

Some direct program control devices and all non-buffered cycle-steal devices are time-dependent (require service within a specified time). Time-dependent devices are subject to losing data if not serviced within specified times. These times vary depending upon the device type and function being performed.

A significant factor that must be considered by the programmer is the priority levels of the devices in his system configuration and the times in which these devices must be serviced.

Device Priority

Overlapping I/O operations in a computing system requires that a priority sequence be established. In the 1130 system, the priority levels for both cycle-steal and interrupt are established for each device that can be attached to the system.

Cycle-Steal Priority

A cycle-steal request may be honored at the end of any core storage cycle. Cycle stealing allows an external device to intervene during the processing of a CPU operation and use one or more core storage cycles in order to communicate directly with CPU core storage. At the completion of the cycle-steal operation, CPU operation is resumed at the point where the cycle-steal request occurred.

CPU cycle-steal level 0 – Single Disk Storage (in the CPU)
CPU cycle-steal level 1 – SAC/2250 (see the multiplex levels below)
CPU cycle-steal level 2 – 1132 Printer
CPU cycle-steal level 3 – 2501 Card Reader

Cycle-steal level 1 is subdivided by the channel multiplexer (when the 1133 is attached to SAC) as follows:

Multiplex level 0 – 2310 Disk Storage, drive 1 or 2311 Disk Storage Drive, drive 1
Multiplex level 1 – 2310 Disk Storage, drive 2 or 2311 Disk Storage Drive, drive 2
Multiplex level 2 – 2310 Disk Storage, drive 3
Multiplex level 3 – 2310 Disk Storage, drive 4
Multiplex level 4 – Reserved (RPQ)
Multiplex level 5 – Reserved (RPQ)
Multiplex level 6 – SAC II/2250
Multiplex level 7 – 1403 Printer
Multiplex level 9 – Reserved
Multiplex level 10 – Reserved
Multiplex level 11 – Reserved (RPQ)

The preceding assignments are given in consideration of expansion for the user who may wish to expand his system at a later date. The cycle-steal levels listed as "Reserved (RPQ)" are for RPQ (Request for Price Quotation from IBM) activity.

Interrupt Priority

Interrupts are caused by a request for service from an I/O device or by the termination of an I/O operation. The interrupt facility provides an automatic branch (to one of core storage locations 8–13) from the normal program sequence in order to react to an external request or conditon.

At the completion of any program instruction, any pending interrupt requests are serviced if no higher level interrupt is in progress.

Interrupts are assigned priority levels to allow the most efficient use of all attached I/O devices in the system.

Level	Device
0	1442 Card Read Punch (column read, punch)
1	1132 Printer, synchronous communications adapter
2	Disk storage, storage access channel (SAC)
3	1627 Plotter, SAC, 2250 Display Unit, or System/7
4	1442 (operation complete), keyboard, console printer; 1134 Paper Tape Reader, 1055 Paper Tape Punch, 2501 Card Reader, 1403 Printer, 1231 Optical Mark Page Reader, SAC
5	Console (program stop switch and interrupt run), SAC

Service Request Limitations

The I/O devices in the 1130 system are subject to loss of data or extremely reduced throughput if service request, either cycle-steal or interrupt, is not honored within times given next. (Refer to Figure 79 for a summary of service request times.)

Cycle-Steal Devices

Single Disk Storage Drives require one CPU cycle every 27.8 us. while an XIO initiate read/write operation is in progress. The end of operation interrupt request may wait indefinitely without losing data but should be completely serviced within 500 us. to gain maximum throughput. The disk storage drives are assigned to the highest cycle-steal priority levels in the system because of their fast data transfer rate.

2311 Disk Storage Drives require one CPU cycle every 16 us. while an XIO initiate read/write operation is in progress. The end of operation interrupt request may wait indefinitely without losing data but should be completely serviced within 0.25 ms. to gain maximum throughput. The disk storage drives are assigned to the highest cycle-steal priority levels in the system because of their fast data transfer rate.

1403 Printer requests one CPU cycle every 11 us. (model 7) or every 18 us. (model 6) while an XIO initiate write is in progress. However, the 1403 is fully buffered and is not subject to losing data if its request remains unhonored. In fact, the 1403 is designed to prevent it from interfering with time-dependent devices on lower priority levels.

The programmer does not need to consider the 1403 regarding loss of data but should consider it regarding throughput.

In order to maintain 340 lines per minute with model 6, the space command (XIO control) must be issued within 117 ms following the transfer complete interrupt. Also, the print complete interrupt must be serviced and the buffer loaded for the next print line within 32 ms (the time required to space one line). If the 1403 received all of its cycle-steal requests without interference, 3 ms would be required to load the model 6 buffer. Therefore, approximately 29 ms is available to service the print complete interrupt.

In order to maintain 210 lines per minute with model 6, the space command (XIO control) must be issued within 187 ms following the transfer complete interrupt. Also, the print complete interrupt must be serviced and the buffer loaded for the next print line within 72 ms (the time required to space one line). If the 1403 received all of its cycle-steal requests without interference, 3 ms would be required to load the model 6 buffer. Therefore, approximately 69 ms is available to service the print complete interrupt.

Device	Time allowable to service I/O request without data loss		Time allowable to service I/O request to maintain rated speed		Time allowable to service end of operation interrupt to maintain rated speed	Frequency of request		
	Interrupt	Cycle-steal	Interrupt	Cycle-steal		I/O Interrupt	I/O Cycle-steal	End Operation Interrupt
Single Disk Storage Drives	-	27.8 usec	-	27.8 usec	500 usec	-	27.8 usec	9 ms
2311 Disk Storage Drives	-	16 usec		16 usec	6.25 ms		16 usec	12.5 ms
1403 Printer Model 6 (340 lpm)	-	-	-	3 ms out of 32 ms	29 ms	-	18 usec	176 ms
1403 Printer Model 6 (210 lpm)	-	-	-	3 ms out of 72 ms	69 ms	-	18 usec	285 ms
1403 Printer Model 7	-	-	-	2 ms out of 19.9 ms	17 ms	-	11 usec	100 ms
1132 Printer Model 1	1.5 ms	16 consecutive cycles within 300 usec	1.5 ms	16 consecutive cycles within 300 usec		11.2 ms	16 cycles every 11.2 ms	-
1132 Printer Model 2						22.2 ms*	16 cycles every 22.2 ms*	-
2501 Card Reader Model A1	-	466 usec	-	466 usec	18.3 ms	-	482 usec	100 ms
2501 Card Reader Model A2	466 usec	-	466 usec	-	3.0 ms	-	482 usec	60 ms
2250 Display Unit	-	-		-				
1442-6 Card Punch	300 usec	-	300 usec	-	25 ms	12.5 ms	-	1216 ms
1442-5/7 Card Punch	300 usec	-	300 usec	-	25 ms	6.25 ms	-	663 ms
1442-6 Card Read	800 usec	-	800 usec	-	35 ms	2.5 ms	-	200 ms
1442-7 Card Read	700 usec	-	700 usec	-	25 ms	1.87 ms		150 ms
SCA (8-bit, 2400 baud)	3.3 ms	-	3.3 ms	-	200 ms (depends on line turn-around)	3.3 ms	-	Depends on number of characters per record
1231 OMPR	-	-	13.2 ms	-	130 ms	13.2 ms	-	2000 ms
1134 Paper Tape Reader	-	-	500 usec	-	16 ms	16.7 ms	-	16.7 ms
1055 Paper Tape	-	-	8 ms	-	8 ms	66.7 ms	-	66.7 ms

*See description under "Cycle Steal Devices."

Figure 79. Table of I/O Timing Requirements

In order to maintain 600 lines per minute with the model 7, the space command must be issued within 72 ms following the transfer complete interrupt. Also, the print complete interrupt must be serviced and the buffer loaded for the next print line within 19.9 ms. The model 7 takes about 2 ms to load the buffer. Therefore, approximately 17.9 ms is available to service the print complete interrupt.

Note: If the space command is not issued until after the print complete interrupt has occurred, the 1403 will not maintain rated speed.

1132 Printer operates in both the cycle-steal mode and the direct program control (interrupt) mode. The 1132 requires 16 consecutive CPU cycles within 300 us. following a read emitter instruction. The 1132 Model 1 also requests an interrupt (direct program control) every 11.2 ms. This request must be honored within 1.5 ms. The 1132 Model 2 presents interrupt requests every 22.2 ms except:

1. When the read emitter instruction reveals that the next character to be printed is X, the next interrupt request is presented in 11.2 ms.
2. When the read emitter instruction reveals that the next character to be printed is Z, the next interrupt request is presented in 33.3 ms.

Any of these interrupt requests should be honored within 1.5 ms.

2501 Card Reader requires one CPU cycle every 466 us. while an XIO initiate read operation is in progress. The end operation interrupt request may wait indefinitely without losing data but should be serviced within 18.3 ms (model A1) or 3.0 ms (model A2) to maintain rated speed.

2250 Display Unit is designed to prevent interference with the 1442, 1132, 2501, or synchronous communications adapter by inhibiting cycle-steal request by the 2250 while these devices are being serviced. In effect, this inhibiting causes the 2250 to be on a priority level lower than any device except the 1403. Because the 2250 is not subject to losing data (actually, the brilliance of the screen could fade on a 3.6-us. system), it may be overlapped with any or all devices in the system.

The time demand from the 1131 CPU varies depending on the mode (character or vector), the number of characters displayed on the screen, the actual characters displayed, and the state of the CPU (wait or processing). The greatest time demand (CPU in the wait state) could be almost every CPU cycle. The least time demand could be about two CPU cycles every 25 ms.

The average interference with CPU processing is:

For 3.6 us. core storage —	character mode 80% of CPU cycles; vector mode 20% of CPU cycles
For 2.2 us. core storage —	character mode 66% of CPU cycles; vector mode 18% of CPU cycles

Program Control Devices

1442 Card Punch requires the punch interrupt request be serviced within 300 us. (model 5, 6, and 7) to prevent loss of data. The end operation interrupt should be serviced within 25 ms to obtain rated speed.

1442 Card Read requires the read interrupt request be serviced within 800 us. (model 6) or 700 us. (model 7) to prevent loss of data. The end operation interrupt must be serviced within 35 ms (model 6) or 25 ms (model 7) to obtain rated speed.

Synchronous Communications Adapter operates at one of several transmission speeds. The time between character transfer interrupts depends on the speed selected by the speed selection switch and the number of bits per character. The times between characters (interrupts) for the various combinations of bit speed and character size are shown in the following chart:

Baud \ Char. Size	6 Bit	7 Bit	8 Bit
600	10.0 ms	11.6 ms	13.3 ms
1200	5.0 ms	5.8 ms	6.6 ms
2000	3.0 ms	3.5 ms	4.0 ms
2400	2.5 ms	2.9 ms	3.3 ms
4800*	1.25 ms	1.45 ms	1.65 ms

*BSC mode only.

Data may be lost if the SCA read request is not honored within the times shown. If the 1130 is transmitting, data will not be lost but fill characters will be automatically inserted, thus reducing actual effective baud rate.

1231 Optical Mark Page Reader requests a read response interrupt each time the one-character buffer in the attachment is loaded. If the request is honored within 13.2 ms, the requests occur every 13.2 ms until the entire data sheet has been read. However, the 1231 has a sonic delay-line buffer capable of storing all characters from a single data sheet. Therefore, data will not be lost if the read response interrupt request remains unhonored longer than 13.2 ms.

To maintain the rated throughput of 2,000 data sheets per hour, the 1231 read response interrupt request should be honored within 13.2 ms, and the read start command (XIO control with bit 13 on) for the next data sheet should be issued within 130 ms after the first read response interrupt has been serviced. The programmer must be aware of the possibility of a read error causing a data sheet to be selected and the operation terminated before all characters have been read and transferred to core storage. If data from data sheets is directly related to the previous sheets (and assuming the previous sheet has been read correctly), then the read start must not be issued until after the operation complete interrupt has occurred and the error indicators have been tested.

The 1231 is on interrupt level 4 and will not impact the throughput of other devices on the system.

1134 Paper Tape Reader requests a read response 500 us. after a feed command. A read command should be given to accept the character stored in the paper tape attachment buffer before the next feed command is issued. The continuous read rate is 16.7 ms per character. A feed command must be issued 16 ms after the response interrupt to maintain rated speed. The 1134 is not subject to losing data unless two feed commands are issued consecutively. The 1134 is on interrupt level 4 and will not impact the speed of the other devices on the system.

1055 Paper Tape Punch requests a punch response interrupt every 66.7 ms if punching continuously and should be serviced within 8 ms following the interrupt request to maintain rated speed. The 1055 is on interrupt level 4 and will not impact the speed of other devices on the system.

IBM System/7

The 1130 system can provide expanded facilities to the IBM System/7 (a sensor based system described in *IBM System/7 System Summary,* Order No. GA34-0002). To the 1130, the sensor based system (Figure 80) appears to be another I/O unit attached to the storage access channel (SAC or SAC II). Communication between the System/7 and the 1130 is by means of 1130-initiated storage-to-storage data transfers with a mutual interrupt system. The mutual interrupt system allows either system to signal the other that a transfer of data is either desired or completed. Facilities are provided to alert the 1130 in case of interface error or System/7 malfunction and to enable the 1130 to determine the nature of the error or malfunction.

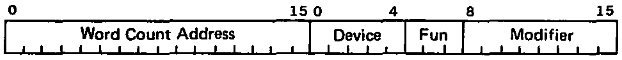
Standard 1130 execute I/O instructions are used to control the System/7. XIO control instructions are used to transmit the beginning System/7 storage address and the word count of the number of words to be transmitted. (See "Storage Access Channel.") An XIO initiate read or initiate write instruction is used to transfer data between the 1130 core storage and the System/7 main storage. An XIO sense interrupt instruction is used to identify that the System/7 has interrupted the 1130. The System/7 interrupts on interrupt level 3, interrupt level status word bit 4. An XIO sense device instruction is used to obtain status indications from the System/7. For details of the instruction formats and programming considerations, refer to *IBM System/7 Functional Characteristics,* Order No. GA34-0003.

PROGRAMMING

The IOCC specifies the operation to be performed and the device to which the operation is directed. For data transfer operations, the IOCC also specifies the word count or address of the data to be transferred.

I/O Control Commands (IOCC)

An IOCC must start at an even storage address in the 1131 processor and has the following format:



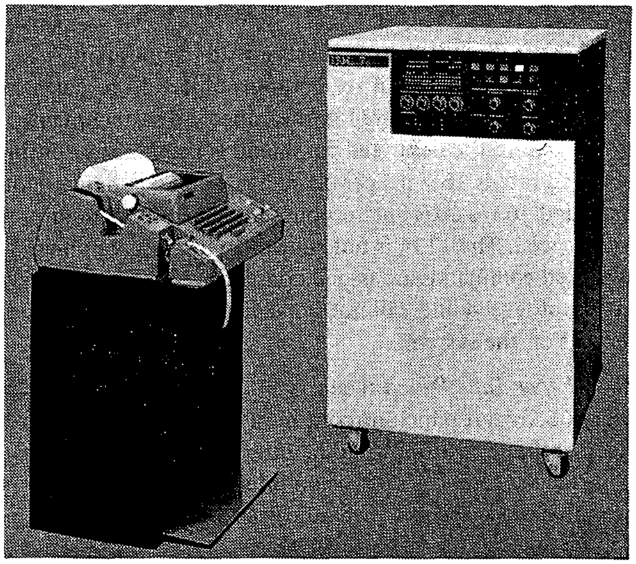


Figure 80. IBM System/7

The IOCC fields are described as follows:

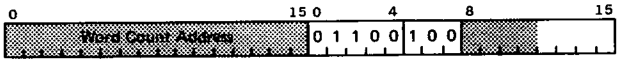
Word Count Address: This field is used to pass the three values required by the System/7 1130 host attachment in order to perform a data transmission: (1) a transmission word count, (2) a System/7 storage address, and (3) an 1130 storage address. The count specifies the number of data words to be transferred. The addresses establish the beginning of the data tables between which the data transfer is to occur. The count and one of the addresses are transferred to System/7 storage by two separate control IOCC's. The third value, an address, is transferred by the 1130 initiate read or initiate write IOCC.

Device: This 5-bit field identifies the I/O device to which the IOCC is directed. In this case, a binary 01100 is the code identifying the System/7.

Function (Fun): The 3-bit function code determines the specific I/O operation to be performed.

Modifier: This 8-bit field provides additional information, when necessary, for the function specified.

Control (100)



An XIO instruction with a control IOCC loads the System/7 1130 host attachment with one of two parameters necessary before starting a data transfer operation between the System/7 and the 1130. The two parameters are:

1. The System/7 storage address at which the data transfer is to begin.
2. The count of the number of 16-bit data words to be transferred.

The 1130 storage address involved in the data transfer is indicated subsequently in either the 1130 initiate read or initiate write IOCC.

Since only one of the two parameters can appear in a single control IOCC, two control IOCC's are required (and, hence, two XIO instructions) prior to performing the actual data transfer.

Modifier bits 14 and 15 signal which, if either, of the two parameters is contained in the control IOCC as follows:

Bit 14 = 0 – Neither parameter being transferred (ignore bit 15)
Bit 14 = 1 – Perform function specified by bit 15
Bit 15 = 0 – Word count being transferred to attachment
Bit 15 = 1 – System/7 storage address being transferred to attachment

Modifier bits 12 and 13 of the control IOCC determine the basic interruption controls to be established in the attachment. To establish the basic control status, modifier field bits 12 and 13 are set as follows:

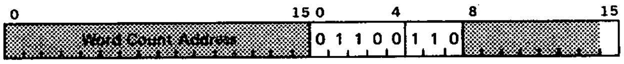
Bit 12 = 0 – Ignore bit 13
Bit 12 = 1 – Perform function specified by bit 13
Bit 13 = 0 – Permit attention and power/thermal interruptions
Bit 13 = 1 – Prevent attention and power/thermal interruptions

Modifier field bit 14 serves a dual purpose in a control IOCC. This bit also establishes a temporary interruption control status as follows:

Bit 14 = 0 – No effect; return to basic control status
Bit 14 = 1 – Prevent attention and power/thermal interruptions

The temporary control status is temporary only in the sense that an initiate read command, an initiate write command, or another control command, immediately following such a control IOCC, can return the interruption control status to the basic control status.

Initiate Read (110)

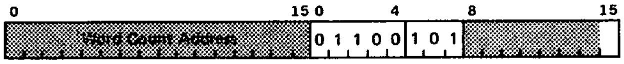


An XIO instruction with an initiate read IOCC sends a block of contiguous data from System/7 to 1130 storage. The starting location of the System/7 data and the number of words transferred must have been established by previously executed control IOCC's. The address field of the initiate read IOCC contains the starting location in 1130 storage for the data to be received.

Modifier field bit 15 is used to control interruptions as follows (refer also to "Interruptions to System/7"):

Bit 15 = 0 – Do not interrupt System/7 for operation end
Bit 15 = 1 – Interrupt System/7 for operation end

Initiate Write (101)

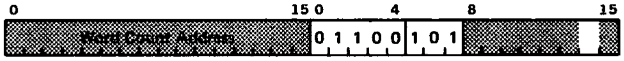


An XIO instruction with an initiate write IOCC sends a block of contiguous data from 1130 storage to System/7 storage. The number of words transferred and the System/7 starting location into which they are stored must have been established by previously executed control IOCC's. The address field of the initiate write IOCC contains the starting location in 1130 storage for the data to be transmitted.

Modifier field bit 15 is used to control interruptions as follows (refer also to "Interruptions to System/7"):

Bit 15 = 0 – Do not interrupt System/7 for operation end
Bit 15 = 1 – Interrupt System/7 for operation end

Electronic Initial Program Load (101)



An XIO instruction with this special initiate-write IOCC is used to IPL the System/7 from the 1130. Modifier bit 14 in this IOCC serves a dual purpose for the electronic initial program load (EIPL) function. This bit also establishes a temporary "prevent" status for attention and power/thermal interruptions. Thus, the 1130 will not recognize an attention or power/thermal warning interruption during a EIPL to the System/7. The basic interruption status must be re-established by a control command with modifier field bit 14 = 0 or by an initiate read or initiate write command.

The host attachment switch on the System/7 console must be in the enable and IPL position for this command to perform the IPL. When the 1130 host attachment recognizes the EIPL command, the System/7 does a system reset and enters the wait state. The host attachment sets the System/7 address to 0 and proceeds with the EIPL as if it were a normal initiate-write from the 1130.

For an error-free termination, the System/7 instruction address register is set to a 0 value, priority level 3 is activated, and System/7 begins to execute instructions starting at location 0. A standard operation-end interrupt is also generated in the 1130.

For an error termination, the System/7 is not informed of the IPL termination, a standard operation-end interrupt is generated in the 1130, and the error condition is indicated by setting a corresponding status bit in the DSW for the 1130.

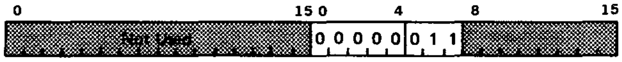
A word count must be established in the 1130 host attachment prior to attempting an EIPL operation from the 1130. This word count is established by an XIO instruction with a control IOCC.

A System/7 address need not be established, since the host attachment sets this address to 0.

Interruptions to System/7

The program in the 1130 can request an interruption to the System/7 processor module. The interrupt priority level, sublevel, and device address are fixed for the 1130 host attachment. As directed by a bit in an initiate read or initiate write IOCC, the request to the System/7 processor is made on priority level 3 with a sublevel of 0. The interruption request presented to the System/7 by the 1130 is handled the same as any other priority interruption request.

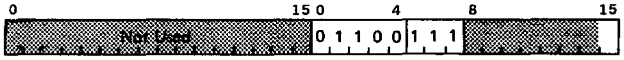
Sense Interrupt (011)



An XIO instruction with a sense interrupt IOCC loads the 1130 accumulator with the interrupt level status word (ILSW) associated with the highest priority level that is on, in order to determine the interrupting device. The sense interrupt command is common to all 1130 I/O devices; therefore, no device-code field is needed.

A System/7 interrupt request to the 1130 sets on ILSW bit 4 for interruption level 3. (The System/7 and the IBM 2250 Display Unit are mutually exclusive on the 1130 SAC.)

Sense Device (111)



An XIO instruction with a sense device IOCC loads the 1130 accumulator with the DSW from the 1130 host attachment in the System/7. The 1130 can determine the cause of an interruption by analyzing these DSW bits.

Modifier field bit 15 has the following meaning in this IOCC:

Bit 15 = 0 – Do not reset DSW bits
Bit 15 = 1 – Reset DSW bits

DSW bits can also be reset by turning on System/7 power or by the reset line in the 1130 SAC.

Device Status Word (DSW): The 16-bit device status word associated with the 1130 host attachment has various bits set on to indicate program operating status and detected errors.

The significant bits in the DSW presented to the 1130 processor and their meanings are:

Significant Bits	*Meaning*
0	Attention. A System/7 set interrupt command is directed to the 1130. This is not an error.
1	Operation end. The word count equals zero, or a power/thermal warning or error was detected during a data transfer operation.
2	Invalid storage address. The 1130 has attempted to address a storage location outside the installed capacity of the System/7. This is an error.
3	Data check. A System/7 parity error was detected by the 1130 host attachment when fetching words from System/7 storage. This is an error.
4	Count = 0. The word count register in the 1130 host attachment is equal to 0. This is not an error.
5	Power or thermal warning. A power failure or thermal warning condition has occurred. This bit is not reset by the sense-device IOCC.

Significant Bits	*Meaning*
6	Storage control check. The 1130 host attachment detected that System/7 storage has not responded to the attachment's request for a storage cycle, sometimes referred to as an "overrun" condition. This is an error.
14	Ready. The System/7 is on line and power is good. This bit is not reset by the sense-device IOCC. This is not an error.
15	Busy. The 1130 host attachment is performing a data transfer operation between the attachment and System/7 storage. This bit is turned off as soon as the operation-end bit (bit 1) is set on. Any 1130 command (except sense-device) to the 1130 host attachment is ignored if the busy bit is on. This is not an error.

Interruptions to 1130

The System/7 1130 host attachment presents interruptions to the 1130 on its interruption level 3. Interruption requests are made to the 1130 under any of the following conditions:

1. The DSW attention bit is set on because a System/7 set interrupt command is directed to the 1130 (if the interruption controls do not inhibit the interruption request).
2. The DSW operation-end bit is set on. (The count = 0 DSW bit, the power/thermal warning bit, or an error bit is also on for this interruption request.)
3. The DSW power/thermal warning bit is set on by the corresponding condition (if the interruption controls do not inhibit the interruption request). The DSW ready bit remains on until the power actually fails. If a data transfer operation is in progress when the power/thermal warning bit comes on, the operation-end bit is also set on, requesting an interruption to the 1130.

The interruption that occurs at the operation-end time of a data transfer can be directed to the System/7 as well as to the 1130. However, if an error is detected during the data transfer, the operation is terminated and the interruption is directed to the 1130 only (count = 0 DSW bit may or may not be set on).

When the System/7 directs a set interrupt command to the 1130 attachment, the DSW attention bit (bit 0) is set on unless it is already on. If already on, condition code 2 is returned to the System/7. If the 1130 attachment is busy, the attention bit is set on and indicated to the 1130 at the time an operation-end interrupt occurs for the operation in progress. If the 1130 attachment is not busy, the attention bit is set on and an interruption request is made to the 1130 on its priority level 3 unless attention interruptions are inhibited by the host attachment interruption controls.

The host attachment interruption controls, which permit or prevent attention and power/thermal interruptions to the 1130, are determined by the setting of modifier field bits 12 to 14 in an 1130 IOCC. Modifier bits 12 and 13 are the *basic* interruption control; bit 14 is a *temporary* interruption control. Attention and power/thermal interruptions are permitted only if the basic status is "permit" and there is no temporary "prevent." If the basic status prevents attention and power/thermal interruptions, these interruptions are not permitted again until another control IOCC that alters the basic interruption control status is executed to permit the interruptions.

If interruptions are temporarily prevented as directed by bit 14 in a control IOCC, they are not permitted again until one of the following occurs:

1. The DSW is reset.
2. Another control IOCC is executed with bit 14 changing the previous temporary status.
3. An initiate read or initiate write IOCC is executed.

Following any of these events, the 1130 host attachment returns to the basic interruption control status.

Attention and power/thermal interruptions to the 1130 are also prevented by either a System/7 power-on reset or an 1130 storage access channel reset. These interruptions are not permitted again until a control IOCC is executed to establish a basic interruption control status.

Appendix A. Character Codes

Ref. No. ①	EBCDIC		IBM Card Code		Graphics and Control Names	Console Printer Hex	PTTC/8 Hex ②	1132 Hex ③	1403 Hex
	Binary	Hex	Rows	Hex ④					
0	00000000	00	12,0,9,8,1	8030	NUL				
1	0001	01	12,9,1	9010	SOH				
2	0010	02	12,9,2	8810	STX				
3	0011	03	12,9,3	8410	ETX				
4	0100	04	12,9,4	8210	PF Punch Off				
5*	0101	05	12;9,5	8110	HT Horiz Tab	41 ⑤	6D(U/L)		
6*	0110	06	12,9,6	8090	LC Lower Case		6E(U/L)		
7*	0111	07	12,9,7	8050	DEL Delete		7F(U/L)		
8	1000	08	12,9,8	8030					
9	1001	09	12,9,8,1	9030					
10	1010	0A	12,9,8,2	8830	SMM				
11	1011	0B	12,9,8,3	8430	VT				
12	1100	0C	12,9,8,4	8230	FF				
13	1101	0D	12,9,8,5	8130	CR				
14	1110	0E	12,9,8,6	8080	SO				
15	1111	0F	12,9,8,7	8070	SI				
16	00010000	10	12,11,9,8,1	D030	DLE				
17	0001	11	11,9,1	5010	DC1				
18	0010	12	11,9,2	4810	DC2				
19	0011	13	11,9,3	4410	DC3				
20*	0100	14	11,9,4	4210	RES Restore	05 ⑥	4C(U/L)		
21*	0101	15	11,9,5	4110	NL New Line	81 ⑦	DD(U/L)		
22*	0110	16	11,9,6	4090	BS Backspace	11 ⑨	5E(U/L)		
23	0111	17	11,9,7	4050	IDL Idle				
24	1000	18	11,9,8	4030	CAN				
25	1001	19	11,9,8,1	5030	EM				
26	1010	1A	11,9,8,2	4830	CC				
27	1011	1B	11,9,8,3	4430	CUI				
28	1100	1C	11,9,8,4	4230	FLS				
29	1101	1D	11,9,8,5	4130	GS				
30	1110	1E	11,9,8,6	4080	RDS				
31	1111	1F	11,9,8,7	4070	US				
32	00100000	20	11,0,9,8,1	7030	DS				
33	0001	21	0,9,1	3010	SOS				
34	0010	22	0,9,2	2810	FS				
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	03 ⑩	3D(U/L)		
38*	0110	26	0,9,6	2090	EOB End of Block		3E(U/L)		
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	SM				
43	1011	2B	0,9,8,3	2430	CU2				
44	1100	2C	0,9,8,4	2230					
45	1101	2D	0,9,8,5	2130	ENQ				
46	1110	2E	0,9,8,6	2080	ACK				
47	1111	2F	0,9,8,7	2070	BEL				
48	00110000	30	12,11,0,9,8,1	F030					
49	0001	31	9,1	1010					
50	0010	32	9,2	0810	SYN				
51	0011	33	9,3	0410					
52	0100	34	9,4	0210	PN Punch On				
53*	0101	35	9,5	0110	RS Reader Stop	09 ⑧	0D(U/L)		
54*	0110	36	9,6	0090	UC Upper Case		0E(U/L)		
55	0111	37	9,7	0500	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	CU3				
60	1100	3C	9,8,4	0230	DCA				
61	1101	3D	9,8,5	0130	NAK				
62	1110	3E	9,8,6	0080					
63	1111	3F	9,8,7	0070	SUB				
64*	01000000	40	No Punches	0000④	blank (space)	21	10(U/L)	‡	7F
65	0001	41	12,0,9,1	8010					
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	8820④	¢	02	20(U)		
75*	1011	4B	12,8,3	8420④	. (period)	00	6B(L)	4B	6E
76*	1100	4C	12,8,4	8220④	<	DE	02(U)		
77*	1101	4D	12,8,5	8120④	(	FE	19(U)	4D	57
78*	1110	4E	12,8,6	80A0④	+	DA	70(U)	4E	6D
79*	1111	4F	12,8,7	8060④	|(logical OR)	C6	3B(U)		
80*	01010000	50	12	8000④	&	44	70(L)	50	15
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	4820④	!	42.	5B(U)		
91*	1011	5B	11,8,3	4420④	$	40	5B(L)	5B	62
92*	1100	5C	11,8,4	4220④	*	D6	08(U)	5C	23
93*	1101	5D	11,8,5	4120④	)	F6	1A(U)	5D	2F
94*	1110	5E	11,8,6	40A0④	;	D2	13(U)		
95*	1111	5F	11,8,7	4060④	¬(logical NOT)	F2	6B(U)		
96*	01100000	60	11	4000④	- (dash)	84	40(L)	60	61
97*	0001	61	0,1	3000④	/	BC	31(L)	61	4C
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	2420④	, (comma)	80	3B(L)	6B	16
108*	1100	6C	0,8,4	2220④	%	06	15(U)		
109*	1101	6D	0,8,5	2120④	_ (underscore)	BE	40(U)		
110*	1110	6E	0,8,6	20A0④	>	46	07(U)		
111*	1111	6F	0,8,7	2060④	?	86	31(U)		
112	01110000	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	0820④	:	82	04(U)		
123*	1011	7B	8,3	0420④	#	C0	0B(L)		
124*	1100	7C	8,4	0220④	@	04	20(L)		
125*	1101	7D	8,5	0120④	' (apostrophe)	E6	16(U)	7D	0B
126*	1110	7E	8,6	00A0④	=	C2	01(U)	7E	4A
127*	1111	7F	8,7	0060④	"	E2	0B(U)		

‡ Any code other than those defined for the 1132 will be interpreted by the PRNT 1 subroutine as a blank.

① Codes identified by * are recognized by all Monitor System conversion subroutines. Codes that are not asterisked are recognized only by the SPEED subroutine.

② U = Upper Case, L = Lower Case; ③ EBCDIC subset

④ Hexadecimal codes identified by ④ can also be entered from the console keyboard.

Console Printer Codes: ⑤ Tabulate, ⑥ Shift to Black, ⑦ Carrier Return

⑧ Shift to Red ⑨ Backspace ⑩ Line Feed

Ref. No. ①	EBCDIC		IBM Card Code		Graphics and Control Names	Console Printer Hex	PTTC/8 Hex ②	1132 Hex ③	1403 Hex
	Binary	Hex	Rows	Hex ④					
128	10000000	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	10010000	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	10100000	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	10110000	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					

Ref. No. ①	EBCDIC		IBM Card Code		Graphics and Control Names	Console Printer Hex	PTTC/8 Hex ②	1132 Hex ③	1403 Hex
	Binary	Hex	Rows	Hex ④					
192	11000000	C0	12,0	A000	(+ zero)				
193*	0001	C1	12,1	9000 ④	A	3C/3E	61(U)	C1	64
194*	0010	C2	12,2	8800 ④	B	18/1A	62(U)	C2	25
195*	0011	C3	12,3	8400 ④	C	1C/1E	73(U)	C3	26
196*	0100	C4	12,4	8200 ④	D	30/32	64(U)	C4	67
197*	0101	C5	12,5	8100 ④	E	34/36	75(U)	C5	68
198*	0110	C6	12,6	8080 ④	F	10/12	76(U)	C6	29
199*	0111	C7	12,7	8040 ④	G	14/16	67(U)	C7	2A
200*	1000	C8	12,8	8020 ④	H	24/26	68(U)	C8	6B
201*	1001	C9	12,9	8010 ④	I	20/22	79(U)	C9	2C
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	11010000	D0	11,0	6000	(- zero)				
209*	0001	D1	11,1	5000 ④	J	7C/7E	51(U)	D1	58
210*	0010	D2	11,2	4800 ④	K	58/5A	52(U)	D2	19
211*	0011	D3	11,3	4400 ④	L	5C/5E	43(U)	D3	1A
212*	0100	D4	11,4	4200 ④	M	70/72	54(U)	D4	5B
213*	0101	D5	11,5	4100 ④	N	74/76	45(U)	D5	1C
214*	0110	D6	11,6	4080 ④	O	50/52	46(U)	D6	5D
215*	0111	D7	11,7	4040 ④	P	54/56	57(U)	D7	5E
216*	1000	D8	11,8	4020 ④	Q	64/66	58(U)	D8	1F
217*	1001	D9	11,9	4010 ④	R	60/62	49(U)	D9	20
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	11100000	E0	0,8,2	2820					
225	0001	E1	11,0,9,1	7010					
226*	0010	E2	0,2	2800 ④	S	98/9A	32(U)	E2	0D
227*	0011	E3	0,3	2400 ④	T	9C/9E	23(U)	E3	0E
228*	0100	E4	0,4	2200 ④	U	B0/B2	34(U)	E4	4F
229*	0101	E5	0,5	2100 ④	V	B4/B6	25(U)	E5	10
230*	0110	E6	0,6	2080 ④	W	90/92	26(U)	E6	51
231*	0111	E7	0,7	2040 ④	X	94/96	37(U)	E7	52
232*	1000	E8	0,8	2020 ④	Y	A4/A6	38(U)	E8	13
233*	1001	E9	0,9	2010 ④	Z	A0/A2	29(U)	E9	54
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*	11110000	F0	0	2000 ④	0	C4	1A(L)	F0	49
241*	0001	F1	1	1000 ④	1	FC	01(L)	F1	40
242*	0010	F2	2	0800 ④	2	D8	02(L)	F2	01
243*	0011	F3	3	0400 ④	3	DC	13(L)	F3	02
244*	0100	F4	4	0200 ④	4	F0	04(L)	F4	43
245*	0101	F5	5	0100 ④	5	F4	15(L)	F5	04
246*	0110	F6	6	0080 ④	6	D0	16(L)	F6	45
247*	0111	F7	7	0040 ④	7	D4	07(L)	F7	46
248*	1000	F8	8	0020 ④	8	E4	08(L)	F8	07
249*	1001	F9	9	0010 ④	9	E0	19(L)	F9	08
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					

Glossary

Area Code. See Device Code.

Assembler Language. A programming language that is more closely related to actual machine language than either RPG or FORTRAN.

Baud. A communications term that specifies bits per second. For example, 600 baud is the same as 600 bits per second.

Binary Synchronous Communication (BSC). A mode (of the synchronous communications adapter) that provides for point-to-point or multipoint operation.

BSC. See Binary Synchronous Communications.

Byte. An increment of information that is made up of eight bit positions (0 1 2 3 4 5 6 7). Each 1130 word location (in core storage) is made up of two bytes.

Central Processing Unit (CPU). The central machine unit (the 1131) in the 1130 System. Core storage is housed in the CPU. The logical circuitry that causes execution of program instructions is also in the CPU.

Core Storage. Core storage (also called storage) contains programs being executed and input data to be processed. Processed data is set up (by the program) in output areas and then moved to one or more output devices.

CPU. See Central Processing Unit.

Cycle-Steal. The method of data transfer between certain I/O devices and main storage. The I/O device "steals" a CPU cycle, when necessary, to transfer a word to or from main storage. The CPU program is slowed only to the extent of the amount of cycles "stolen."

Device Address. See Device Code.

Device Code. The binary field in input/output control commands that specifies the I/O device involved in the operation. (Also called area code or device address.)

Device Status Word (DSW). A 16-bit increment of information that specifies the status or condition of an I/O device. Each I/O device has one or more associated DSW's.

Direct Address. A method of forming the effective address from values in an instruction or from values in an instruction in a register.

Direct Program Control. Refers to the need for program control for each operation performed. Specifically, some I/O devices require instruction and input/output control command execution for each data character transferred to or from main storage. Contrast with cycle steal.

Displacement. A field in short-format instructions that is usually added to the contents of a register to obtain the effective address. The displacement has other uses in certain instructions.

Double Precision Format. A binary number format of 32 bits (two words). The arithmetic sign is the leftmost bit. See sign bit.

DSW. See Device Status Word.

Effective Address (EA). The actual address of data or an instruction in main storage. The effective address is derived in various ways, depending upon the instruction and the manner in which that instruction is executed.

FORTRAN. FORTRAN (FORmula TRANslation) is a high-level programming language designed specifically for engineering and scientific data-processing applications.

Four-Wire Operation. A communications arrangement of terminals that use two data paths. The paths are arranged so that signals can be transmitted on one path only and received on the other path. Four physical wires may or may not make up the data paths.

Half-Duplex. A communications method of operation in which each terminal can transmit or receive information signals, but only one of these (transmit *or* receive) at a time.

ILSW. See Interrupt Level Status Word.

Indirect Address. In indirect addressing, the effective address is the contents of a core-storage location which itself is located by direct addressing.

Input/Output Control Command (IOCC). A 32-bit increment of information that specifies the operation, data address, I/O device, etc. during I/O device operations. An IOCC is to an I/O device what an instruction is to the CPU.

Interrupt. The temporary stopping of an operation in order to perform some higher priority operation. Interrupts are used to transfer data to or from I/O devices, handle unusual I/O device conditions, and terminate I/O device operations.

Interrupt Level Status Word (ILSW). A 16-bit increment of information that specifies the I/O device(s) causing an interruption. There are six levels of interruption (0 through 5) but only five ILSW's (for levels 1 through 5).

Interrupt Vector. There are six interrupt-vector locations in main storage. These locations point to the beginning of the interrupt-handling subroutine for the associated interrupt level. (The interrupt vectors must be program-loaded with the desired addresses at program-loading time.)

IOCC. See Input/Output Control Command.

Multipoint. A private line arrangement in which more than two communication terminals are capable of communication among themselves but on the same line.

Point-to-Point. The transmission of data directly from one terminal to another without the use of any intermediate computer or terminal.

RPG. RPG (Report Program Generator) is a high-level programming language that is mainly applicable to commercial data-processing applications.

SAC. See Storage Access Channel.

SCA. See Synchronous Communications Adapter.

Sign Bit. The leftmost bit of a single- or double-precision binary operand. When this bit is at a value of zero, the binary operand is positive; when this bit is at a value of 1, the binary operand is negative.

Single Precision Format. A binary number format of 16 bits (one word). The sign is specified by the leftmost (high-order) bit. See Sign Bit.

Storage. See Core Storage.

Storage Access Channel (SAC). This channel provides for attaching certain I/O devices to the 1130 system.

STR. See Synchronous Transmit-Receive.

Synchronous Communications Adapter (SCA). A feature in the 1130 that enables the system to function in a communications network in either point-to-point or multipoint operation. The term "synchronous" signifies that signal transmission is continuous rather than start-stop for each character.

Synchronous Transmit-Receive (STR). A mode (of the synchronous communications adapter) that provides for point-to-point operation only.

Two-Wire Operation. A communications operation of terminals that can transmit in either direction (from terminal A to terminal B or from terminal B to terminal A) but not both at the same time.

Word. The amount (16 bits) of bit positions available for data at each core-storage location. The positions of a word are numbered 0 to 15, left to right.

Wraparound. Going sequentially from the highest core-storage location to core-storage location 0000.

Index

Where more than one page reference is given, the major reference is first.

GA26-5881-6

IBM 1130 Functional Characteristics (1130-01) Printed in U.S.A. GA26-5881-6

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM 1130 Functional Characteristics
GA26-5881-6

READER'S COMMENT FORM

Your views about this publication may help improve its usefulness; this form will be sent to the author's department for appropriate action. Using this form to request system assistance or additional publications will delay response, however. *For more direct handling of such requests, please contact your IBM representative or the IBM Branch Office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Index Figures Examples Legibility

Cut or Fold Along Line

What is your occupation? ____________________

Number of latest Technical Newsletter (if any) concerning this publication: __________

Please indicate your name and address in the space below if you wish a reply.

____
____
____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

102

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut Along Line

Fold Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold Fold

IBM 1130 Functional Characteristics (1130-01) Printed in U.S.A. GA26-5881-6

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]