IBM® Customer Engineering

Manual of Instruction

7030 Data Processing System

Introduction and Programming, Volume III

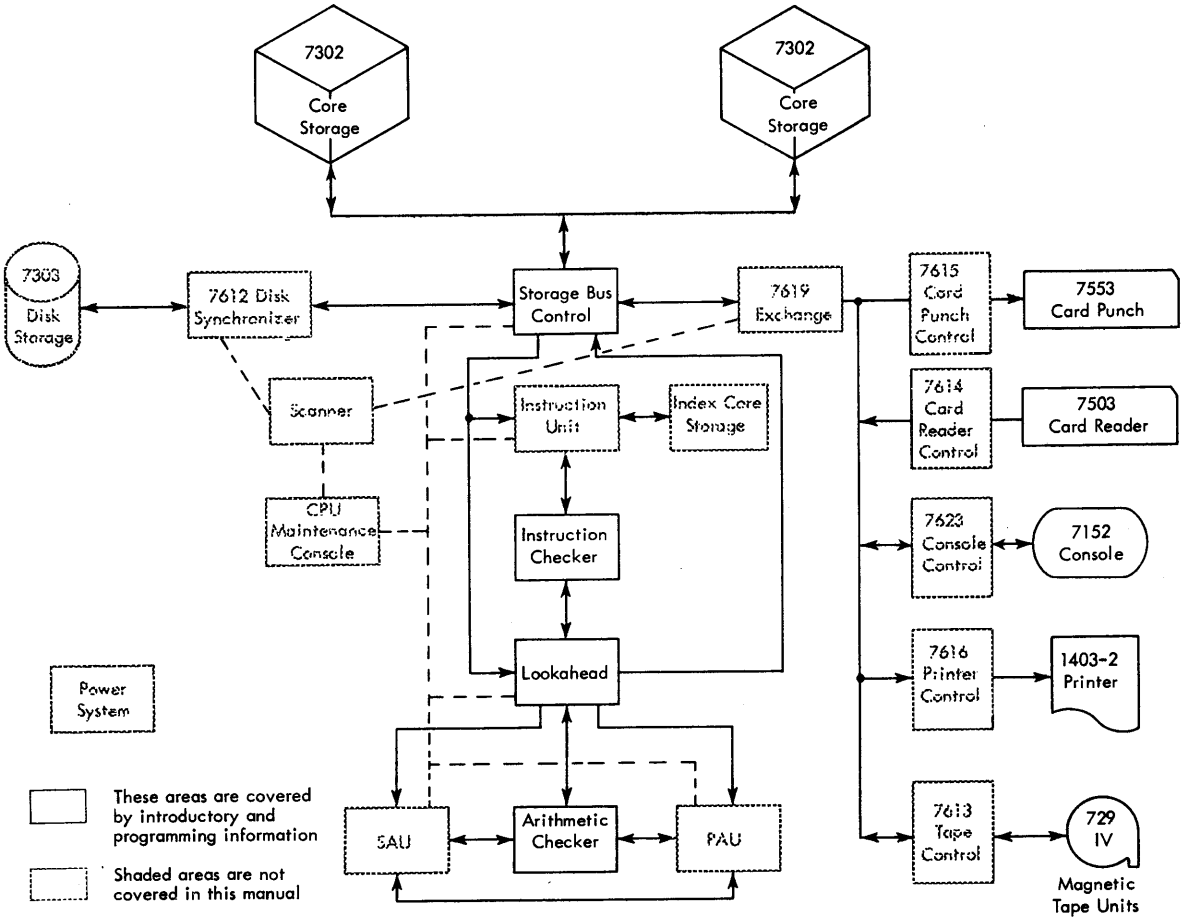
PREFACE

This is Volume III of the Customer Engineering Manual of Instruction, 7030 Introduction and Programming. The material is based on the information available on the commercial 7030 system as of October 17, 1960. Areas of the system are covered as shown in the frontispiece.

The manual contains:

1. Introduction to 7030 input-output units and features.
2. Descriptions of I-O programs and instructions for each I-O unit.
3. Explanation of STRAP notations and use as a symbolic program system.
4. Appendix with charts and tables used in the introductory course of instruction.

This Volume III, Form R23-9695, obsoletes sections of the Preliminary Instruction Text on 7030 Introduction and Programming, Books A through G.



IBM 7030 DATA PROCESSING SYSTEM

CONTENTS

APPENDIX

7 INTRODUCTION TO 7030 INPUT-OUTPUT

A SIMPLIFIED version of the 7030's input-output section is shown in Figure 7-1. Note that all input-output data flow is accomplished through the exchange. Input to the system passes from the input devices to core storage through the exchange. The exchange assembles complete 64-bit words from the flow of input information and stores the assembled words in core storage locations without tying up the central processing unit. The CPU specifies the starting location and the number of input words to be read. While the CPU proceeds with computation, the exchange completes the input operation and signals the CPU when the transmission is finished.

The exchange operates similarly for output operations, fetching core storage words and disassembling them for the output devices independently of the CPU.

The exchange can operate eight independent input-output units. This eight-channel exchange can be enlarged by adding other eight-channel groups. The exchange can address up to 32 channels. Each channel handles nine bits, an 8-bit byte and one parity bit, used to check transmission over the channel. Each of the exchange channels has a nominal information rate of 500,000 bits per second. However, for special requirements, higher information rates are possible. The exchange can reach a peak rate of 100,000 words or 6,400,000 bits per second.

A wide variety of input-output units can be operated by the exchange. These include card readers and punches, printers, magnetic tape units, and operator's consoles. The individual units are connected to the exchange through control units. Most control units permit only one input-output device to be attached to a channel. The magnetic tape control unit is designed to attach up to eight tape units to a single exchange channel; only one of these units can be operated at any one time.

7.1 CHARACTERISTICS OF 7030 I-O DEVICES

To completely understand I-O programming, some characteristics of the I-O devices must be considered. This manul section explains these characteristics, but only as they affect programming and data flow.

7.1.1 IBM 7503 Card Reader

The IBM 7503 Card Reader operates at 1,000 cards per minute. Cards are loaded into the hopper face down, 9-edge inward. They travel past a first reading station for checking, past a second reading station for data entry and into a random stacker. A clutch permits intermittent starting and stopping up to the second reading station; from there the cards travel continuously into the stacker.

The card contents read at the first and second read stations are transferred to core storage buffers in the card reader control unit. The control unit contains three buffers. The record buffer, used for actual data entry, receives information from the second

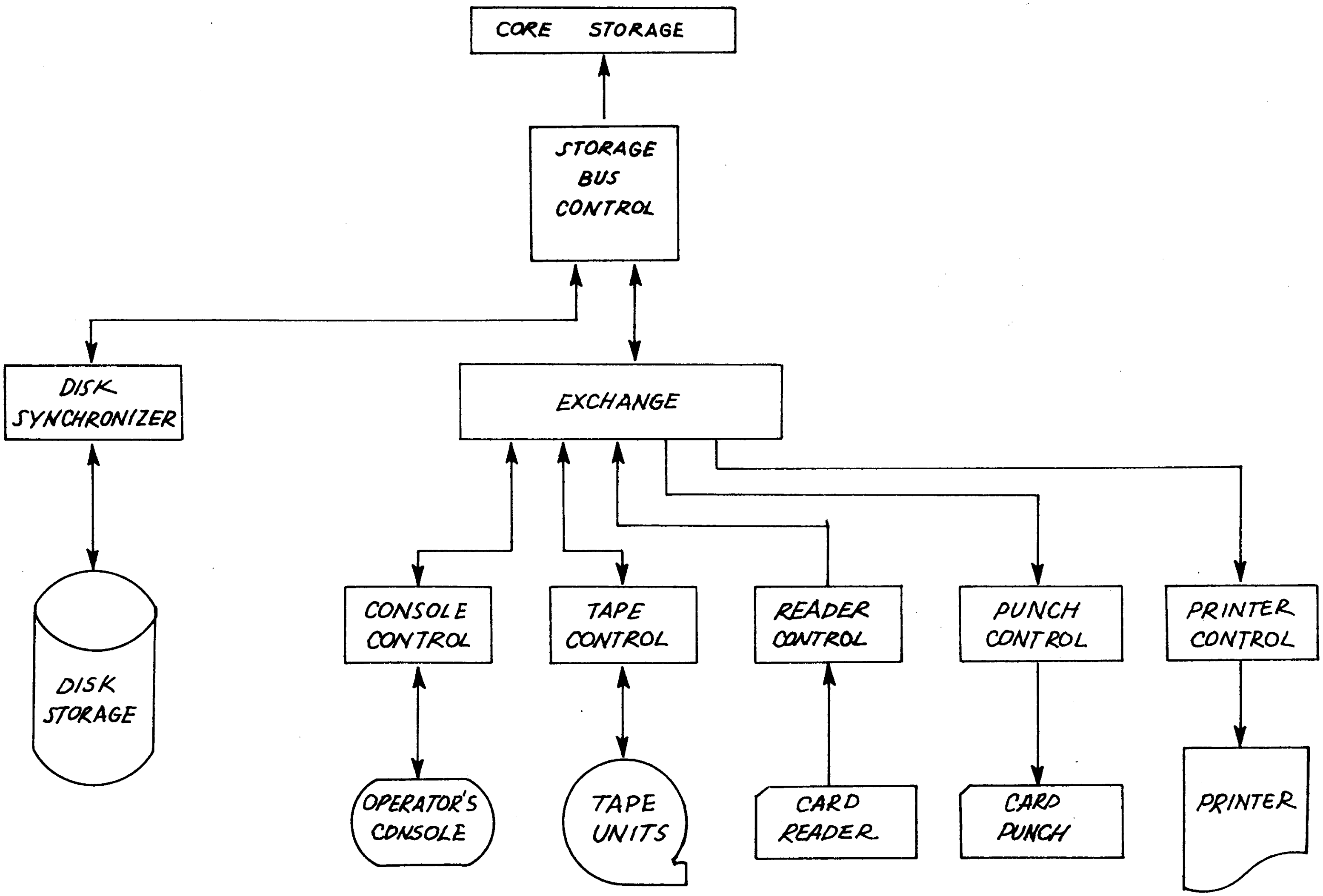


FIGURE 7-1. 7030 INPUT - OUTPUT SECTION

read station. The remaining two buffers, used for checking purposes, receive information from the first read station alternately. Information from one card is delivered to check buffer A, from the next card to buffer B, and so on.

The card reader is loaded by placing cards in the hopper and depressing the start key, to initiate three card feed cycles. The three cycles deliver the first card past the first and second read stations, into the stacker, and the second card through the first read station. The record buffer and check buffer A now contain the information read from the first card, while buffer B contains the information read from the second card.

When the card reader is in the described condition, it is said to be "ready", as indicated by a ready light located on the front of the card reader. When the reader is ready, a READ from the computer causes the record buffer contents to be delivered, via the exchange, to main core storage. At the time the record buffer is read out, the appropriate check buffer is read out and its contents compared against those of the record buffer. This comparison is for checking purposes, and only the contents of the record buffer are delivered to the exchange. As soon as the record buffer is emptied, a new card cycle places the next card's contents in check buffer A and the record buffer. When the record buffer is next read out, it is compared against buffer B which received the contents of the current card at the time it passed through the first read station. Any disagreement in the comparison causes a unit check indication to be given.

The cards are read row-by-row in the card reader. Once in the buffer, however, the information is transferred in 8-bit bytes to the exchange as if it had been read column-by-column. The transfer of the card, column-by-column, is shown in Figure 7.1-1. The exchange assembles eight consecutive 8-bit bytes into a 7030 word. Therefore, five and one-third columns of the card are assembled as one word. This scheme gives the punched card a capacity of fifteen 64-bit words.

The 64-bit word, does not have parity check bits associated with it. When words of this type are sent to core storage from cards, all of the information from the cards can be used as data. This information is transferred when the card reader is operating in no-ECC mode, that is, without parity check bits.

In the ECC mode, 72 consecutive bits, or six adjacent columns, are read from the cards as one word. This word consists of 64 data bits preceded by eight ECC, or parity check, bits. The ECC mode is used, primarily, to punch binary data for re-use in this system.

In the no-ECC mode, the entire contents of a card appear in core storage as 15 full words. In the ECC mode, a card is treated as 13 full words. These 13 words correspond to 78 card columns, with the last two columns ignored.

The card punch punches cards in the described fashion. Therefore, much information read by the card reader is binary and in a columnar format. Other information formats, such as the normal IBM card code, are read in the same manner as the columnar binary data. The method of handling the information once it is in main core storage is determined by the programmer. Notice that VFL instructions, with a field length of 12 specified, facilitate the handling of standard IBM coded information, one character at a time.

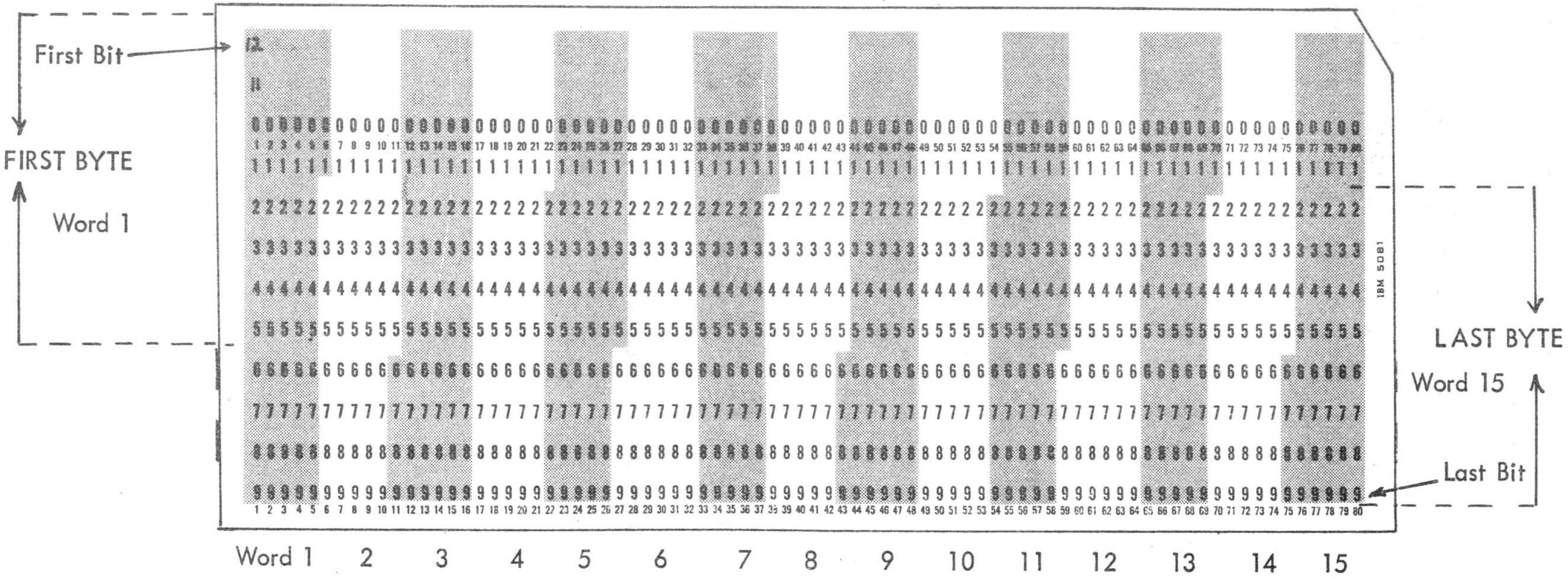


FIGURE 7.1-1. CARD FORMAT

7.1.2 7553 Card Punch

The IBM 7553 Card Punch operates at 250 cards per minute. Cards are loaded in the hopper face down, 9-edge inward. They travel past a punching station, past a reading station for checking, and into a random type stacker. A blank station precedes the punching station so that, when the punch is ready, two blank cards are positioned between the hopper and the punching station. A clutch permits intermittent starting and stopping up to the offset station; from there the cards feed continuously to the stacker.

The punch is made ready in the same manner as the card reader. After placing cards in the hopper, depression of the start key results in three card feed cycles. At the completion of these three cycles, the first card is positioned ahead of the read station and the second card is positioned ahead of the punch station. The first card placed in the stacker is blank. Ready status is indicated by the ready light located on the front of the punch.

The card punch receives information row-by-row from the punch control unit. The control unit, in turn, receives information from the exchange. The data received by the control unit are stored in two core storage buffers, each with a capacity of 15 words (960 bits). The contents of the two punch buffers, A and B, are routed to the punch magnets on alternate card cycles. One card is punched with the contents of buffer A, the next card is punched with the contents of buffer B, and so on.

At the time buffer B is being punched, the contents of buffer A are being compared against the card passing through the read station. This card was previously punched with the contents of buffer A. Any differences in this comparison cause a unit check indication.

From the punch viewpoint, a card cycle is divided into phase 1 and phase 2. During phase 1, data are brought from the exchange to core storage in the punch control unit. During phase 2, these data are punched into a card and the previously'punched card is read for punching errors. If an error occurs at this point, (a discrepancy between card data and core array data) a "unit check" occurs.

"Unit check" may be caused by "data error" or by "parity error" within the punch control. Data error occurs in phase 2. Parity error may occur in phase 1 or phase 2. Phase 1 parity error implies that wrong parity was sent to the punch from the exchange within the last three incoming bytes. Phase 2 parity error implies error in data readout of the array to be punched and then regenerated back into the array.

Unit check does not, therefore, always indicate data error. No distinction is made between phase 1 and phase 2 unit check. The offset station follows the read station. To offset the error card, the required CONTROL must follow the unit check and occur in the next punch card cycle if a phase 2 unit check occurs. If a phase 1 error occurs and the offset occurs in the next cycle, the card that is offset in the stacker is not in error. The following card is the error card.

After checking, buffer A is loaded with new data to be used for punching the next card. While buffer A is punching out, buffer B is being compared against the card at the read station.

Information is delivered to the control unit, from the exchange, in 8-bit bytes. The control unit places the bytes in the punch buffers in the form of a card image as in Figure 7.1-1. The information is then delivered to the punch magnets row by row.

When the punching of a set of cards is finished, the last card has not been checked. To check and stack the last card, which is in the checking station, two additional card cycles are needed, one to get it past the checking station and another to get it past the offset station. Absence of unit check, after these two cycles, then verifies the correctness of the last card actually punched.

The card punch also provides a means of offsetting a punched card under program control. The offset is initiated with a CONTROL. The offset card may be used to indicate a card that was erroneously punched or the end of a particular program deck.

The punch can operate in either ECC more or no-ECC mode. In ECC mode, each word punched is 72 bits in length: 64 data bits and eight ECC bits. Therefore, each card can contain 13 words. These words would occupy the first 78 card columns with the last two columns not used. It is important that the last two columns be blank because they are actually checked at the check read station, and any prepunching in these columns causes the parity checking to indicate an error.

7.1.3 IBM 1403-2 Printer

The 7030 uses a modified IBM 1403-2 Printer that operates through a 7616 Control Unit. The Model 2 printer has 132 printing positions. Each 7616-1403-2 combination will be able to handle three different basic chains:

48-character ECS (48 characters + blank)
48-character BCD (48 characters + blank)
120-character ECS (119 characters + blank)

Chains are interchangeable and mode of operation is selected from the operator's panel on the 1403-2. The 1403-2 operates at a speed of 600 lines per minute with 48 characters or 275 lines per minute with 120 characters; printing density is ten characters per inch. Line spacing of six or eight lines per inch can be selected by the operator.

Information to be printed is coded in an 8-bit code, one line occupying up to 17 words. WRITE transfers the data from core storage to a buffer in the printer that stores one complete line of printing. When all the information to be printed on one line has been transferred, printing is initiated.

The actual printing is done one position at a time. One character on the chain is positioned for printing at any one time. A magnetic timing drum rotates as the print chain revolves and emits pulses to identify the chain's relative position at all times. When the "home pulse" is emitted from the drum, it signifies that when the next pulse is emitted the first character of the chain will be positioned at print position 1. The drum pulses are used to step a comparing counter. The comparing counter always contains the bit configuration which represents the character that is currently positioned for printing. As any character becomes aligned with a particular position, the eight bits which represent the character to be printed in that position are read out of the print buffer and compared against the comparing counter. If the result of the comparison is equal, meaning that the character aligned is to be printed, the hammer for that position is fired, driving the paper and ribbon against the chain.

As shown in Figure 7.1-2, the distance between adjacent print positions is less than the distance between adjacent characters on the chain. Using the figure as a guide, note that the next chain character to be aligned with a print position, after position 1, is character 3. It is aligned with print position 4. The time between the alignment of characters 1 and 3 is 11 microseconds. Thus, if the characters shown in positions 1 and 3 on the chain are to be printed in positions 1 and 4, they are printed consecutively, 11 microseconds apart. Also, every eleven microseconds one chain character becomes aligned with one print position. As each alignment occurs, the 8-bit character code representing the character to be printed is read out of the buffer and compared against the comparing counter. After every third printing position has been tested to determine printing, the process is repeated starting at position 2. This testing and possible printing of every third position is called a subscan. Because only every third position is tested during a subscan, three subscans are required to test all 132 print positions. At the completion of three subscans each position has been tested for possible printing of only one character. Therefore, in order to print any one of 48 significant characters, 48 complete scans, consisting of three subscans each, must be performed for each line of printing.

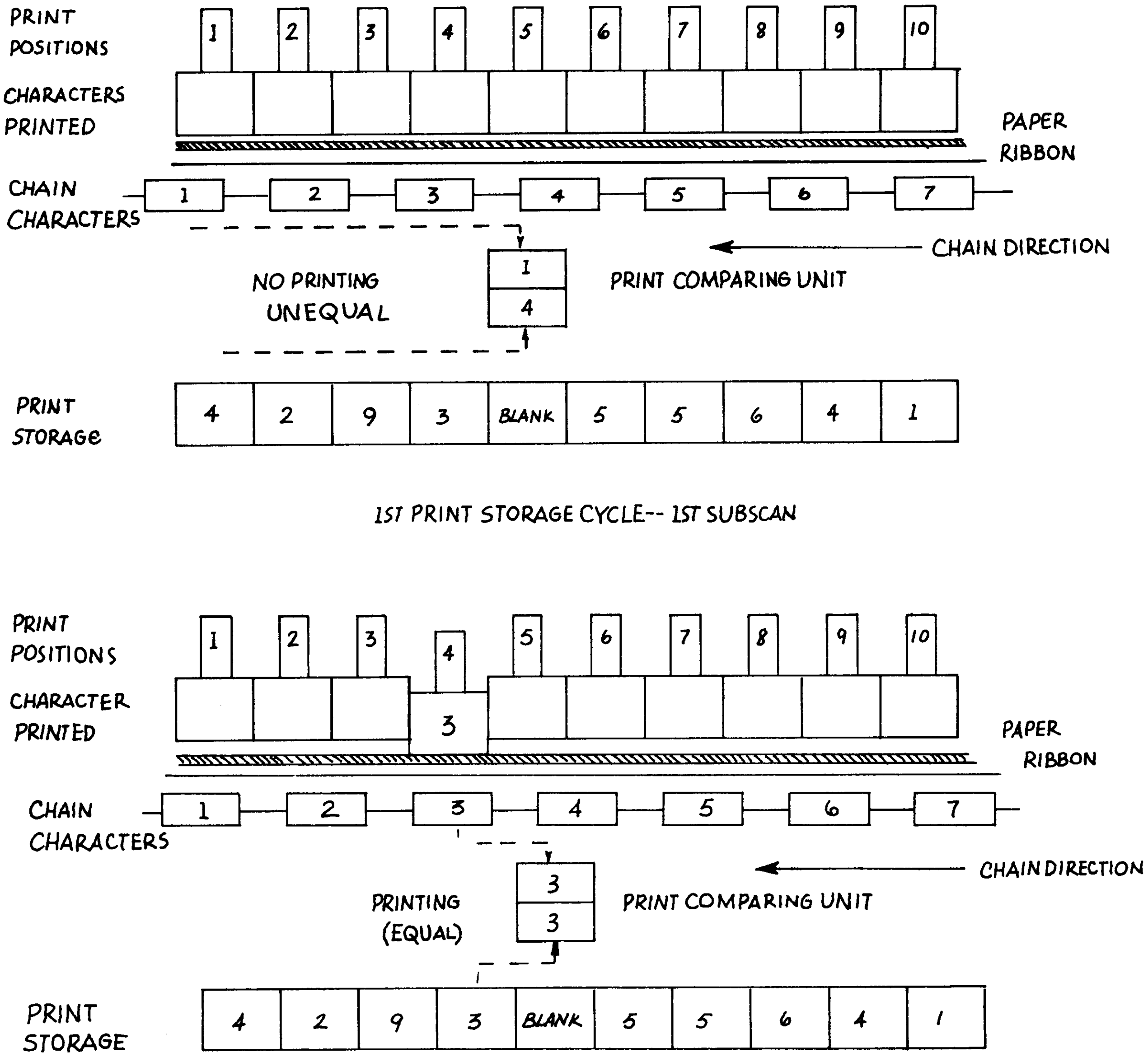
Checking of the printer operation is automatic and concurrent with printing. Each character code sent to the printer is associated with an odd parity bit. These bits are used to detect errors introduced in the transmission of the character codes from the exchange to the printer control and to check the storing and handling of these codes within the printer control. An echo caused by the regenerative part of the hammer tripping pulse is used to verify that a hammer has been tripped at the proper instant and that it has not been tripped at any other time.

Operation of the carriage is under program control and is determined by a control field that must always precede the information to be printed on one line. One of two carriage speeds is selected automatically, depending upon the distance the carriage is to advance. The printer has no control panel and no carriage control tape, thus providing the greatest flexibility and simplicity of operation. Tape channel 1 is used for restore or reference. Tape channel 12 has two uses:

1. It is always used for end of forms, so that the last complete form is printed.
2. From a switch on the operator's panel, channel 12 can be used for an automatic overflow or restore feature.

The operation to be performed by the carriage is specified by a special control code called "carriage control field" (CCF). The carriage control field must be sent to the printer along with the data for each line of printing. This field consists of one or more 8-bit bytes. The low-order byte always specifies the number of lines over which the printer must skip before printing the data associated with the control byte.

After printing, the carriage automatically advances one line position. This post-spacing feature saves time because it permits the carriage to advance before the next carriage control field has arrived. If the next carriage control field specifies skipping additional lines, the carriage continues to move to the desired line. If the next carriage control field specifies to skip 0 lines, the previous post-spacing operation results in



1ST PRINT STORAGE CYCLE-- 1ST SUBSCAN

2ND PRINT STORAGE CYCLE -- 1 SUBSCAN

FIGURE 7.1-2. 1403-2 PRINTER SCHEMATIC

single spacing. If the carriage control field is delayed, the carriage waits at the next line. The post-spacing can be suppressed when the next line is to be printed on the same line position as the present line. A "suppressed post-spacing carriage" control function is provided for this purpose.

The carriage control field permits skipping up to 223 lines in one operation. This skipping command is encoded from one 8-bit byte, and if only normal skipping is desired, the carriage control field needs only one 8-bit byte. In addition to the skipping codes, the carriage control field can specify a number of other functions by using more than one byte.

Code			
Binary	Decimal	Meaning	
0000 0000	0	Skip 0 lines (single space)	
0000 0001	1	Skip 1 line (double space)	
0000 0010	2	Skip 2 lines (triple space)	
⋮	⋮	⋮	
1101 1111	223	Skip 223 lines	
1110 0000	224	Select No Report	And suppress post-spacing if printing is suppressed
1110 0001	225	Select Report 1	
1110 0010	226	Select Report 2	
1110 0011	227	Select Reports 1 or 2	
1110 0100	228	Select Report 3	
⋮	⋮	⋮	
1110 1000	232	Select Report 4	
⋮	⋮	⋮	
1110 1111	239	Select Reports 1,2,3 or 4	
1111 0000	240	Suppress Post-Spacing	
1111 0001	241	Not Used	
⋮	⋮	⋮	
1111 1101	253	Not Used	
1111 1110	254	End	
1111 1111	255	No Operation	

FIGURE 7.1-3. CARRIAGE CONTROL CODES

Figure 7.1-3 shows the coding used in the 8-bit bytes of the carriage control field. Carriage control bytes with decimal codes 0 through 223 specify skipping up to 223 lines in one operation. A skip byte is always the last byte of the carriage control field, and the printer interprets the information that follows as data to be printed. The remaining carriage control bytes with decimal codes 224 or higher provide for special functions. These codes, except for the end code, cause the following byte to be interpreted as another carriage control byte, thus making it possible to append line selection information to the control bytes specifying special functions. Carriage control bytes that initiate functions other than skipping are:

1. Codes 224 through 239 operate in conjunction with the report selection keys on the printer and permit selective printing of data.
2. Code 240 suppresses post-spacing; it allows normal carriage operation before printing, but the automatic spacing after the printing cycle is omitted. The suppression of post-spacing makes it possible to print two blocks of information on one line position. For the next line to print in the same line position as the present line, it must have the skip 0 lines carriage control byte.

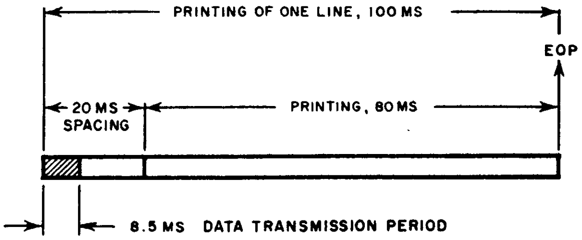
3. Code 241 has been assigned for programmed restore. This is treated like any other skipping CCF except that instead of skipping a definite number of lines, the carriage skips to the next 1 hole (restore position). This code switches the mode from CCF to LOAD.
4. Codes 242 through 253 do not have any functions assigned to them and are ignored by the printer. They have the same effect as the no-operation code. For compatibility with other devices, it is recommended that these codes not be used.
5. Code 254 normally is not used in the CCF mode. It is intended as a coded early end or early disconnect. Its main use is for ending records that are less than 132 characters long. An end code received during CCF turns on the stop trigger and stores the end code. A valid skip instruction must follow to take the printer from CCF to load mode, at which time the stored coded end causes blanks to be loaded in all print positions. A normal print operation follows and post-spacing takes place if it has not previously been suppressed.
6. Code 255, no operation, is ignored by the printer and may be used as a filler preceding another carriage control byte.

The line spacing can be one-sixth or one-eighth of an inch, depending upon the manual setting of the feed clutch. The carriage control fields apply to either case, the line space always being defined by the current setting of the clutch.

When a selective printing key is on and a report selection code is encountered that does not match the key, the remainder of the block, including the following carriage control bytes, is ignored, and printing does not take place. Transmission of data to the printer continues, however, until the end code is encountered or the buffer is filled up. When printing is suppressed, post-spacing of the carriage is also suppressed. Normally end of operation is given when printing of the line is completed. When printing is suppressed, end of operation is sent at the completion of data transfer. No indication concerning the suppression is ever sent to the computer.

Printer Timing

The time required to print and space one line is about 100 milliseconds. Data transmission from the computer to the printer's buffer always follows immediately after WRITE is issued and takes place at the rate of one 8-bit byte per 64 microseconds. A delay circuit has been put in to slow the printer' s request cycle to one in each 64 usec as specified. A complete line of printing information is thus transmitted in approximately 8. 5 milliseconds. The time required to space over up to three lines is approximately 20 milliseconds, of which the first 8. 5 milliseconds are overlapped with the data transmission time. The 20 millisecond period can be occupied either by post-spacing caused by the preceding print cycle, by skipping for the present cycle, or by a combination of both. When spacing is completed, the contents of the buffer are printed in approximately 80 milliseconds, as shown below.



Selective Printing

Selective printing keys, in conjunction with report selection carriage control bytes, provide for selective printing of data sent to the printer. This feature makes it possible for the operator to assemble reports out of selected lines of print information without modifying each time the program that provides the data.

Selection of data to be printed is accomplished by matching the four rightmost bits of report selection codes with a bit map generated by depression of the selection keys. Assuming the bits in each byte to be numbered from 0 through 7, starting with the leftmost bit, a depression of select 1 key corresponds to a 1-bit in bit position 7, depression of select 2 corresponds to a 1-bit in position 6, depression of select 3 corresponds to a 1-bit in bit position 5, and depression of select 4 corresponds to a 1-bit in bit position 4. Whenever a carriage control byte has bits 1110 in the high-order positions, the printing of the line is suppressed unless there is a 1-bit in the low-order four bits to match the selective printing key that has been depressed. For example, code 225 (1110 0001) permits printing of the current line only if the select 1 key has been depressed while code 227 (1110 to 0011) allows printing for keys select 1 or select 2. On the other hand, when the key select 1 is depressed, only lines with the report selection codes 225 (1110 0001), 227 (1110 0011), 229 (1110 0101), 231 (1110 0111), 233 (1110 1001), 235 (1110 1011), 237 (1110 0010), and 239 (1110 1111) can be printed.

When all print selection keys are off, the report selection codes are ignored. If there is no report selection code in the carriage control field, the key setting is ignored. In either case no selection takes place, and all data are printed.

7.1.4 IBM 7152 Operator's Console

The IBM 7152 Operator's Console provides communication between operator and computer. For maximum flexibility, the console is treated as an input-output device which is connected to the computer through the exchange. Thus, all functions on the console are under stored program control. More than one console may be attached to the system.

The console contains a number of toggle switches, columnar switches, rotary switches, and a keyboard for input. The switches are scanned to furnish the first three input words when the console receives a READ. Further input words come from the keyboard.

For output, the console contains lights, a numerical display, and an I-O printing mechanism. The lights and digital displays are set by the first three words of the output message sent by a WRITE. Any further words in the output message control the printing. The physical arrangement of the various input and output sections of the console is shown in Figure 7.1-4.

The keyboard and the printing mechanism serve for input and output, and together they also constitute a typewriter. All information entered into the computer via the keyboard is also printed. By making the console "not ready," the operator can use the typewriter for making notes and for adding remarks to the information entered into the computer or typed out by it.

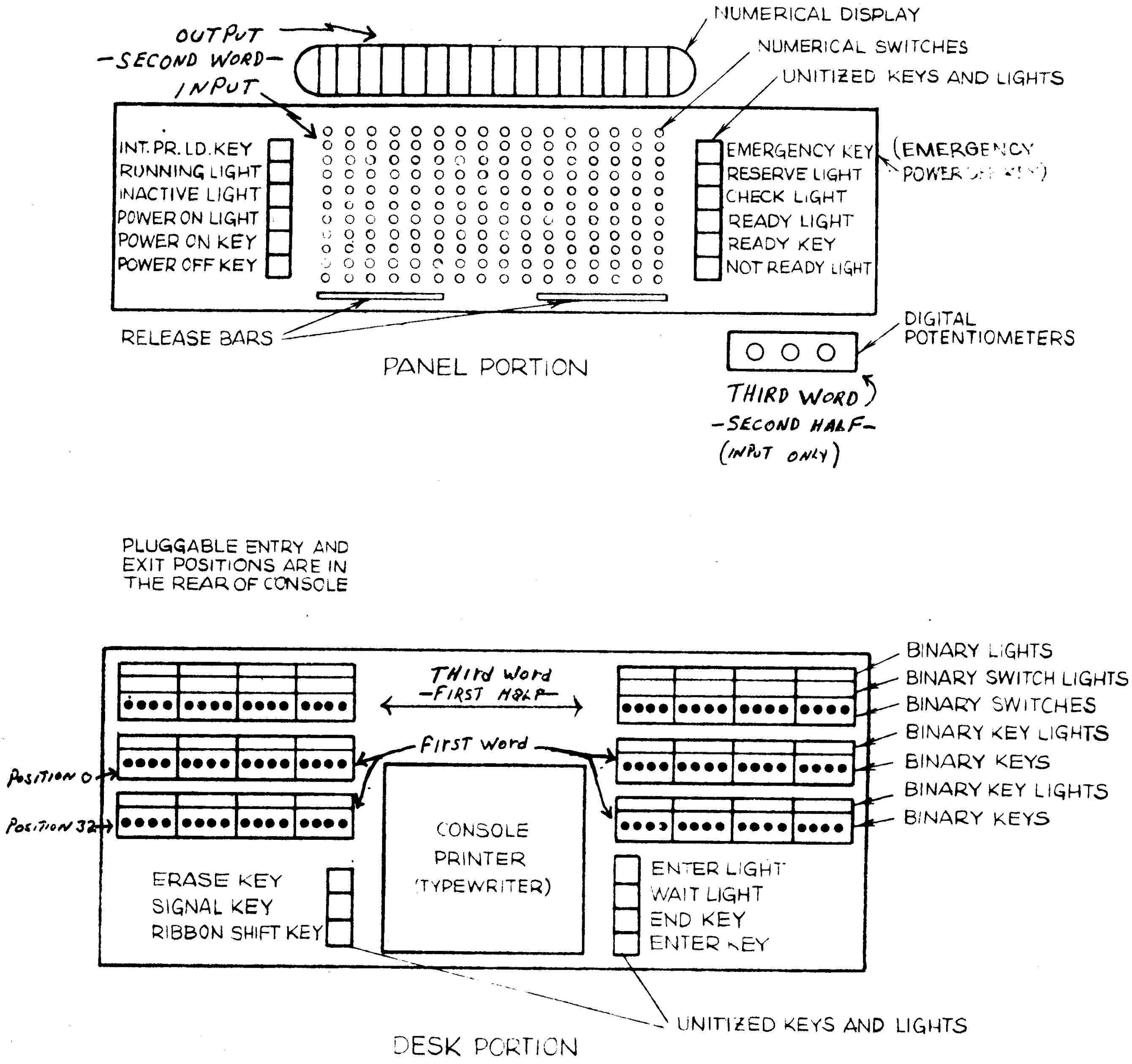


FIGURE 7.1-4. CONSOLE ENTRY AND DISPLAY DEVICES

Two keys, signal and enter, cause a channel signal indication to be sent to the computer. As a result of this signal, the program may issue a READ. The enter key also prepares the keyboard for entering information. A READ following depression of the signal key is terminated by the console when the words from the switch inputs have been read. A READ following depression of the enter key is not terminated automatically, but accepts characters from the keyboard until the operation is complete. The completion of the operation may be determined by the computer, or may be the result of pressing the erase key or end key.

Keys

Power On, Power Off, Emergency Power Off: These keys are standard.

Signal: Depression of this key when the unit is ready generates a channel signal and disables the keyboard. The key is inoperative when the unit is in the not-ready status.

Ready: If all the conditions necessary for operation are satisfied, depression of this key puts the unit in the ready status. The ready key has the same function as the start key on standard units, except that it does not automatically generate a channel signal indication.

Not Ready: This key is the same as the stop key on standard units; it places the unit in not-ready status. If the console is reading or writing at the time the not-ready key is pressed, the operation is interrupted. The operation is resumed again when the unit returns to the ready status.

Enter: Depression of this key enables the keyboard and generates a channel signal indication. If the keyboard has been enabled, it becomes active when the subsequent READ has completed reading of the three words from the console switches. The active state of the keyboard permits information to be entered and is indicated when the enter light is on and the keyboard is unlocked. A READ received by the console with the keyboard not enabled is automatically terminated when the three words scanned from the switches have been sent to the exchange.

The enter key enables the keyboard and generates a channel signal regardless of any reading or writing currently taking place at the console. The key is ineffective when the console is in the not-ready status.

End: Depression of this key when the keyboard is active (the enter light is on) enters the end function code, terminates the operation, locks and disables the keyboard, and turns off the enter light. The end function code (8 bits) is placed in main storage as if it were data. During a write operation the end code does not cause printing; rather, it terminates the operation. The end key does not terminate read operation in multiple block mode.

Erase: Depression of this key when the keyboard is active enters an erase function code, terminates the read operation, locks and disables the keyboard, and turns off the enter light. An end exception signal is also sent to the computer to indicate that the data just read are invalid. The erase key does not terminate operation in multiple block mode.

Ribbon Shift: Depression of this key causes ribbon shift. If the console printer is equipped with a two-color ribbon, all characters that are entered through the keyboard while the ribbon shift key is depressed are printed in the color associated with the ribbon shift. In addition, when a character is entered through the keyboard with the ribbon shift key depressed, the character code sent to core storage is modified to cause ribbon shift when the code is subsequently sent to the console printer for writing.

Initial Program Load: Refer to Section 7.11 for a description of this key.

Lights

Power On, Ready, Check: These lights are standard.

Enter: This light indicates that the keyboard is active. The light goes off when the keyboard becomes inactive.

Wait: This light goes on when the signal or enter key is depressed and is turned off by completion of the first three words of a READ. It indicates to the user that the computer has not yet accepted the information set up on the switches, which therefore should not be changed. An audible click is generated each time the wait light is turned off.

Reserved: This light is turned on and off by program control as an indication to the operator. Reserve lights are also located on the card reader, card punch, and the printer.

Binary Keys and Key-Lights

The console has 64 keys, arranged in two banks of 32 each, and grouped by four within each bank. Associated with each key is a binary key-light that can be set by the output word from the computer or by the key. Each group of keys and key-lights has a frame for a single labeling strip which can be readily changed. The key has a toggle action with a center neutral position and two extreme latching positions. In one extreme position the key turns the light on; in the other extreme position it turns the light off. When latched in either position, it holds the light on or off so that it cannot be changed by the computer. When the key is in the neutral position, the corresponding light can be set on or off by a WRITE.

The bits corresponding to the states of the 64 binary key-lights make up the first input word when a READ is issued and the first output word when a WRITE is issued. The upper row corresponds to bit positions 0 through 31, from left to right, the lower row to bit positions 32 through 63. When a light is on, it corresponds to a one bit; when it is off, to a zero bit.

Numerical Switches

Each of the two banks of decimal columnar switches consists of eight columns, each column containing keys for the digits 0-9 and a reset key. Each digit key is illuminated when depressed. The depression of any key releases the other in the same column. There is also a general release bar for each of the two banks. When a READ is issued, the setting of the 16 numerical switches makes up the 64 bits of the second input word. The four bits of the leftmost column occupy bit positions 0 through 3. The keys in each column are arranged from bottom to top in the order: reset, 0-9. Reset and digits 8 and 9 are distinguished in appearance from digits 0-7 to faciliate entry of octal data. The rightmost two columns in each bank are similarly distinguished from the first six. The banks are physically grouped together.

Numerical Display

A group of sixteen positions is used to display numbers in decimal or octal form. Each position consists of a device for displaying one of twelve characters, or a blank. The twelve characters that can be displayed are the digits 0-9, minus sign, and decimal point. The character to be displayed at any position is selected by decoding four bits sent from the computer. Codes 0000 through 1001 are decoded as 0-9; 1010 is decoded as a point; 1011 is decoded as minus (-): 1100, 1101, 1110, and 1111 are all decoded as blanks.

When a WRITE is issued, the contents of the second output word are displayed in the numerical display, from left to right, starting with bits 0 through 3 for the leftmost digit. A character remains displayed on the numerical display until it is changed by a subsequent WRITE. The numerical display is aligned digit by digit with the sixteen numerical switches. There is no direct physical or electrical connection, however.

The following list describes the code set used in decoding the information sent to the numerical display and in encoding the keys on the numerical switches.

BINARY CODE	CHARACTER DISPLAYED ON NUMERICAL DISPLAY	CORRESPONDING KEY ON NUMERICAL SWITCHES
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	. (decimal point)	- - -
1011	— (minus sign)	- - -
1100	Blank	- - -
1101	Blank	- - -
1110	Blank	- - -
1111	Blank	- - -*

* This code is generated when no digit key is depressed.

Binary Switches

The console has 32 binary toggle switches, arranged in two sets of 16 each and grouped by fours within each set. The switches have labels that can readily be changed. Each label is illuminated by the associated binary switch light when the corresponding switch is on. When a READ is issued, the settings of the binary switches make up the first four bytes of the third input word, from left to right, with bit 0 reflecting the state of the leftmost light. In contrast to the keys, the switches have only latching action but no neutral position.

Pluggable Entry Positions

To permit other switching and selection devices to be attached, there are 32 pluggable entry positions, each consisting of an input jack. Each pluggable entry corresponds to one of the binary switches described in the preceding section. When a plug is in the jack for any pluggable entry, the corresponding binary switch is ineffective.

When a pluggable entry is in use, it is represented by a binary zero if there is no connection between the terminals of its jack, and by a binary one if there is a closed connection. The bits from the pluggable entries substitute for the input bits from the corresponding switches; they constitute the first four bytes of the third input word.

A thirty-third jack causes a channel signal to be sent to the computer whenever a connection is made between its terminals. This jack does not disable the signal key.

Binary Lights

For display of machine conditions, flags, indicators, and other binary information, there are 32 independent binary lights. They are located immediately behind the binary switches and are so grouped that there is visual correspondence between the binary lights and binary switches, even though there is no physical connection. Each of the sets of lights has a frame to hold a labeling strip.

When a WRITE is issued, the contents of the first four bytes of the third output word are displayed in the binary lights, with those lights illuminated that receive binary ones. The lights remain on until turned off by another WRITE.

Pluggable Exit Positions

To permit attachment of other display devices, the console has 32 pluggable exit positions, each consisting of an output jack. Each pluggable exit corresponds to one of the binary lights described in the previous section. Insertion of a plug in the jack does not disable the corresponding binary light. When a pluggable exit is in use, it is active (provides current) when it is furnished a one-bit by means of a WRITE. The exit remains active until another WRITE is given. Because the pluggable exits parallel the binary lights, the information addressed to the exits occupies the first four bytes of the third output word. A thirty-third output jack furnishes a momentary current whenever the gong is sounded. A thirty-fourth output jack parallels the wait light and provides current when the wait light is on. The insertion of a plug in this jack does not disable the light.

Digital Potentiometers

To facilitate the entry of analog information, there are three digital potentiometers. Each of these is a rotary switch with 128 positions encoded 0000000 to 1111111 in clockwise order. Each switch has a knob with a pointer that indicates the approximate setting of the potentiometer. When a READ is issued, the settings of the digital potentiometers constitute the rightmost seven bits of the fifth, sixth, and seventh bytes of the third input word. The leftmost bit in each byte is zero. No provision is made to display the contents of the second half of the third word during a WRITE.

Keyboard

The keyboard arrangement is a standard typewriter keyboard, with 44 character keys and a shift key for lower or upper case. As a character key is depressed, the associated character is automatically typed on the console printer. An 8-bit character code is associated with each character for entry to and exit from the computer. The space bar causes the printing mechanism to skip one printing position and enter a blank character consisting of all zeros.

The keyboard has several function keys. In addition to causing the indicated action on the printing mechanism, depression of each function key generates a corresponding function code for entry into the computer. Receipt of the function code at the console printer during a WRITE operation again causes the indicated action. The function codes are listed below:

Binary Code	Interpretation of Code During Writing	Key that Generates Code During Reading
1111 0111	Blank	Erase
1111 1011	Tabulate	Tabulate
1111 1100	Backspace	Backspace
1111 1101	Carriage Return	Carriage Return
1111 1110	End	End

The carriage return key causes the printing mechanism to advance to the first printing position on the next line. The backspace key causes the printing mechanism to return to the previous printing position on the same line. The tabulate key causes the printing mechanism to skip to the next preset tabulate position to the right, on the same line.

Entry of Information

To enter information through the console, the console must have received a READ from the computer. The operator can request a READ at any time by causing a channel-signal indication, provided the program so interprets the channel signal indication.

A channel signal indication can be caused in two ways: by depressing the signal key or by depressing the enter key. The enter key, in addition to generating a channel signal, also enables the keyboard. The keyboard is disabled by the termination of the subsequent READ.

In order to use the keyboard for entry of information, the keyboard must have been enabled by depressing the enter key. The subsequent READ, after scanning the three words from the console switches, makes the keyboard active so that a message can be keyed. When the keyboard is active, the enter light is on and the keyboard is unlocked. As the message is keyed, a copy is always typed on the console printer.

When a READ is received by the console with the keyboard not enabled, the message length is limited by the console to three words, and normally is automatically terminated when the three words from the console switches have been read.

Writing under Computer Control

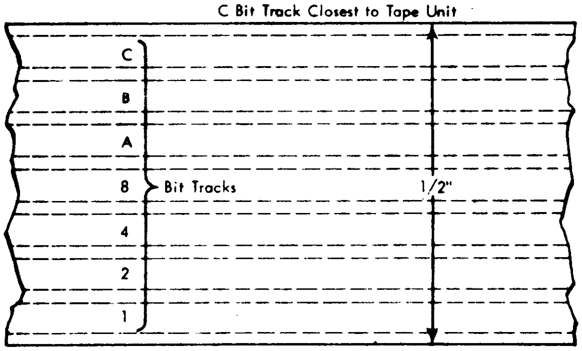
Information can be displayed on the console by means of a WRITE. The first three words of the output message addressed to the console are interpreted by the binary lights and the numerical display. The remainder of the message, regardless of its length, is always used to actuate the console printer mechanism.

The information sent to the console can have any character and function codes in any order. The part of the message applying to the binary lights and the numerical display is decoded according to the corresponding display devices. All the subsequent bytes are interpreted by the console printer and are decoded as control functions or characters. The three control functions carriage return, backspace, or tabulate initiate the same action as the corresponding keys, and the function codes are the same as those generated by these keys during manual entry. The keyboard is locked if printing under computer control is in progress.

The console printer message can consist of any number of lines of printing, but a carriage return function code must be given on every line at or before the end of the printing line. Otherwise, overprinting of characters occurs at the extreme right. No indication concerning the end of the printing line is given to the computer. The maximum length of the printing line is 83 characters. The actual length at any time depends upon the manual setting of margin controls.

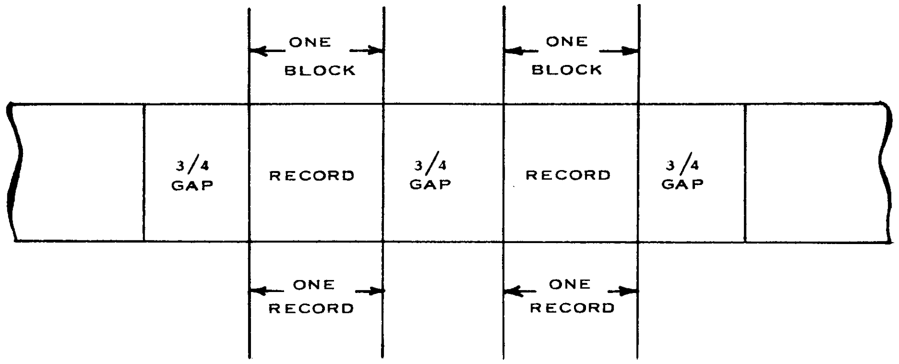
7.1.5 IBM 729 IV Magnetic Tape Unit

The IBM 729 IV Magnetic Tape Unit runs at a speed of 112.5 inches per second and uses half-inch tape with seven tracks: six data tracks and one parity check track. As shown below, the seven tracks on tape are usually designated 1248ABC from front to rear of the unit, where C is the parity track and 1248AB are information tracks. When these bits are stored, bit B is in the lowest-numbered, or leftmost, bit position. Successive bytes are stored in the order in which they are read, with the B bit adjoining the 1 bit of the preceding byte: BA8421 BA8421 BA....



Tapes are usually written and read with the odd-parity mode of checking where the C-bit provides an odd number of 1-bits in the seven tracks. Other systems use the same physical tapes with an even parity method of checking; therefore, the mode of writing and reading even-parity tapes is provided.

Records on tape are not restricted to any fixed length of characters, fields, words, or blocks. Records may be of any practical size within the limits of available storage capacity. Records or groups of data are separated on tape by a record gap, a length of blank tape about 3/4 inch long. During writing, the gap is automatically produced at the end of a record. During reading, the record begins with the first character sensed after a gap and continues without interruption until the next gap is reached. The blank section also allows for starting and stopping the tape between records. A single unit or block of information is, therefore, defined or marked by an inter-record gap before and after the data as illustrated below.



An inter-record gap, followed by a special single-character record, is used to mark the end of a file of information. A file usually consists of a number of records. The character designating an end of file is called a tape mark and consists of one bits in tracks 1, 2, 4, and 8.

Because writing automatically destroys any previous information on the tape, a file protection device is used to prevent accidental erasure of information. The file protection device is a plastic ring that fits into a round groove molded in the tape reel. When the ring is in place, reading or writing can occur. When the ring is removed, only reading can take place; writing is suppressed.

Photosensing markers called reflective strips are placed on the tape by the operator to enable the tape unit to sense the beginning and the end of the usable portion of tape. Photoelectric cells in the tape unit sense the markers as either the load point marker (where reading or writing is to begin) or as the end-of-reel marker(where writing is to stop). The markers are small pieces of plastic, 1 inch by 3/16 inch, coated with vaporized aluminum on one side and adhesive on the other. They are fastened to the base (uncoated) side of the tape. The marker designating the beginning of the usable tape area is called the load point marker. At least ten feet of tape must be allowed between the beginning of the reel and the load point marker as a leader for threading the tape on the tape unit. More than ten feet may be allowed by placing the marker at any desired distance from the beginning of the reel. To indicate the load point, the 1-inch dimension of the marker must be parallel to, and not more than 1/32 inch from, the channel 1 edge of the tape (the edge nearest the operator when the reel is mounted). About 18 feet of tape are usually reserved between the end-of-reel marker and the end of the tape. This space includes at least ten feet of leader and enough tape to hold a record after the end-of-reel marker is sensed. Any usable length over the ten feet may be allowed to permit additional records to be written after the marker is sensed. To indicate end of reel, the marker must be placed parallel to, and no more than 1/32 inch from, the C track edge of the tape (the edge nearest the tape unit when the reel is mounted).

The tape may be recorded at one of two densities. The higher density is 556 bits per inch. The lower density is 200 bits per inch and permits the units to handle tape that is fully compatible with that used on the IBM 727 units. By using one of the two densities, it is thus possible to communicate with any system or auxiliary equipment designed to work with the IBM 727 tape units or any units of the IBM 729 series. One of the two densities is selected under program control. Except for this difference, both kinds of tape units are controlled and programmed the same way. The information rate of the tape depends upon the recording density. Because the tape can be read and written at two different densities, the following two information rates are available:

Recording Density	Information Rate in 6-Bit Bytes per Sec.	Avg. Word Period in usec per 64-Bit Word
556 bits/in.	62,550	170
200 bits/in.	22,500	474

The maximum word rate of the tape, corresponding to 62,550 six-bit bytes per second, is 5,864 words (64-bit) per second.

All tape units are connected to the exchange through a tape control. Each exchange channel can accommodate one tape control and as many as eight tape units may be attached to the tape control. Only one of the units attached to a channel can be operated at a time. Simultaneous operation requires more than one tape control, each attached to a different channel on the exchange. Tape units connected to different controls can operate independently of one another. The tape control contains a "byte converter" that converts four successive 6-bit bytes on reading into three successive 8-bit bytes for entry into the exchange, or three 8-bit bytes from the exchange into four 6-bit bytes on writing. Incomplete 6-bit bytes at the end of a word are filled with the first bits of the next word so that no gaps are left on tape. Conversely, successive 6-bit bytes on tape appear in consecutive 6-bit groups in storage without gaps, and may straddle storage word boundaries. As a result, the information rate at the exchange in terms of 8-bit bytes per second is 75% of the tape rate in terms of 6-bit bytes per second; the bit or word rates are the same.

If, during writing, the number of bits in the words to be written as one block is not divisible by six, the remaining four or two information bits are combined with two or four zeros, respectively, to form the last 6-bit byte to be written. On subsequent reading these zeros are treated as information bits. Similarly, during reading, if the number of information bits in the tape block is not divisible by 8, the remaining 6, 4, or 2 bits are combined with zeros to form the last 8-bit byte to be sent to the exchange.

Keys, Lights and Switches

Power On, Power Off, Master Power: These keys are located on the tape control and control the power for all tape units connected to one exchange channel.

Load-Rewind: If the unit is in not-ready status, depression of this key causes loading of the tape into the vacuum columns and searching for the load point in a reverse direction. If the tape is more than 450 feet from its load point, a high-speed rewind is initiated first, with the tape out of the columns, before low-speed searching for the load point. The key is inoperative if the unit is ready.

Start: If the reel door is closed and the tape has been loaded into the vacuum columns, depression of this key places the unit in ready status. A channel signal indication is then given. If the start key is pressed when the unit is in ready status already, nothing happens.

Change Density: Depression of this key changes the density mode in the unit. If the unit was set to read and write in the low density mode, depression of this key changes the unit to the high density mode, and vice versa, provided the unit is not ready.

Unload: If the unit is not ready, depression of this key removes the tape from the columns and raises the upper head assembly, regardless of the distribution of tape on the two reels. In addition, the unload key turns off the tape-indicator-on light if the light is on at the time the key is depressed. If the tape is to be unloaded, the tape should first be brought to the load point by pressing load-rewind before pressing unload. The unload key is inoperative when the unit is ready; in order to make use of this key, the unit must first be made not ready by depressing the reset key.

Reset: Depression of this key immediately stops any tape operation in progress and places the units in not-ready status. If a high-speed rewind is in progress, pressing reset once causes the unit to change to a low-speed rewind; a second depression of reset stops the unit. Note that, unlike the stop key on other units, reset stops any reading or writing operation immediately without waiting for the end of the block.

Address Selection Switch: This is a rotary switch for assigning a selection address to the unit. It is mounted horizontally and is operated by a large knurled disk. Only a small section of the disk protrudes from the panel. The switch has ten positions marked with the numbers 0 through 9, out of which the positions 0 through 7 represent assignable addresses. Positions 8 and 9 are invalid and a unit that has one of these addresses cannot be selected. The address assigned to a unit is displayed on an illuminated translucent band which rotates with the knurled disk.

Lights

Power On, Ready, Select: These lights are standard. (The power-on light is located on the tape control.)

High-Low Density: This light indicates the density mode of the tape unit. The high or low portion of the light is illuminated.

File Protect On: This light is on when the unit is unloaded or is loaded with a reel that does not have the file-protect ring attached. The light is on regardless of the presence of the file-protect ring during rewinding.

Tape Indicator On: This light is turned on whenever the end-of-tape reflective strip is sensed during writing or when the tape breaks. It is turned off by the initiation of rewinding and by the remove-end-of-tape-condition-control instruction. When this light is on, every WRITE is terminated by end-exception after writing one block. The light can be turned off manually by pressing the unload key.

Fuse: This light is on when a circuit breaker opens a circuit in the tape unit. When this light is on, the unit is not ready and customer engineer intervention is required.

7.2 IBM 7619 EXCHANGE

The IBM 7619 Exchange directs information flow between input-output units and core storage. The exchange relieves the central processing unit of the task of communicating with input-output units and enables processing of data to proceed simultaneously with the operation of a number of these units.

The exchange provides a general method of connecting many different kinds of units to the computer system. It contains the common control facilities to be time-shared among the external units, thus keeping these units as simple as possible, yet maintaining fully overlapped operation. The exchange also does the necessary housekeeping of addresses and the assembly or disassembly of information without taking time away from the computer or core storage. The only computer time involved is that needed to start and interlock the operations. The only core storage cycles required during external operations are those needed to transfer the data to or from the final locations in core storage. These cycles are interspersed between computing operations without interfering with the computer program except for the slight delay involved in the actual core storage references.

A common method of program control applies to all external units. When an input-output instruction is given, the computer executes all address modification. It then sends the addresses and the decoded operation to the exchange, which completes the execution of the instruction by obtaining the operand (control word) from core storage and starting the external unit. This procedure permits the exchange, before it accepts an instruction, to determine from its stored status indication bits whether the unit is ready, and to sandwich into available time periods the extra cycles necessary to start an operation. If the unit is not ready (for instance, if the unit is out of material and waiting to be reloaded), the exchange rejects the instruction and sends a signal to the indicator register located in the CPU.

The computer waits until the exchange has signalled that it has accepted or rejected the instruction. If the exchange is operating near full load, the computer may have to wait some microseconds because it is more flexible in its operation than the external units, most of which cannot wait. It usually does not wait for the external unit to respond or to finish the operation, which may take milliseconds to minutes. The exchange takes over full control and signals the computer when the operation is completed. This signal is termed end of operation (EOP). The computer can in the meantime proceed with the program.

The exchange may operate several I-O devices simultaneously. The amount of simultaneous reading and writing depends upon the number of exchange channels and upon the data transmission rates of the units attached. As to the data transmission rates, the limitation can be imposed either by the transmission of bytes of data between the exchange and the units or by the transmission of data words and control words between the exchange and core storage. If either of these two data paths is operated beyond its capacity, information is lost.

7.2.1 Reading and Writing

Information is transferred between external units and core storage by read and write operations. A READ or WRITE is first given by the program, to specify the external unit (channel address) and the location of a control word, which defines the beginning and size of the core storage area to be used for the data transfer. These two pieces of information are sent to the exchange and the computer then proceeds to the execution of the next instruction in the program. The control word defines the core storage area to be used in data transfer by specifying the initial data word address in the storage and a count of the number of words to be transferred. After READ or WRITE is received, the exchange obtains the specified control word from core storage and stores it in the exchange. The desired unit is then started, and reading or writing proceeds, using the storage area indicated by the control word.

As each data word is read or written, the control word in the exchange is modified; The data word address is advanced by one, and the word count is decreased by one. The control word in the exchange always contains the data word address of the next core storage reference and count of the number of words remaining to be transferred to or from the external unit. The control word in core storage, however, is not affected during the data transfer. Thus, the control word in the core storage unit retains the initial data word address and count, and it can be used repeatedly to transfer data to or from the same storage area. The four basic variations of a read or write operation, depending on the setting of flag bits in the control word, are:

1. The operation may be terminated by the exchange upon reaching a count of zero or by the unit upon reaching the end of the block, whichever happens first (single block operation without chaining). A block of data is defined for each type of unit as the amount of information recorded in the interval between adjacent starting and stopping points of the unit. The length of a block depends upon the type of unit used; it can be a card, a line of printing, or the information between two consecutive gaps on tape.

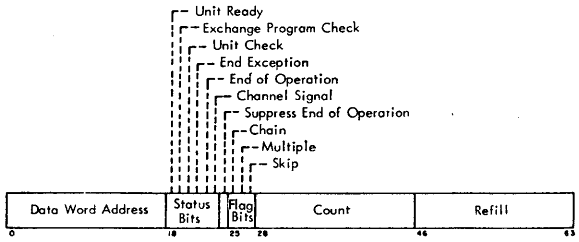
2. After the core storage area defined by a control word is exhausted, it is possible for the exchange to substitute another control word and continue data transfer without stopping, using the core storage area defined by the new control word (chaining).

3. More than one block may be read or written with one instruction, until the specified core area has been used (multiple block operation).

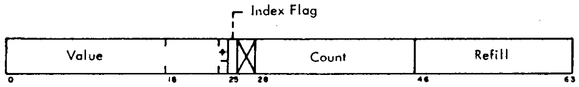
4. During reading, it is possible to suppress entry into storage of selected portions of the information (skipping).

The Control Word

The control word format follows that of an index word: the data word address occupies the word address portion of an index word value field while the count and refill fields of control and index words are the same. This facilitates the use of control words for indexing. In addition, the control word contains a number of bits which describe the status of the external units and specify the mode of reading or writing. The control word and the index word are illustrated on the next page.



CONTROL WORD



INDEX WORD

Bits 0 to 17, Address

Bits 0 through 17 of the control word contain the data word address. This address is the location of the first word in core storage.

Bits 18 to 24, Status

Bits 18 through 24 of the control word are called status bits. When obtaining a control word from core storage, the exchange ignores these bits and treats them as if they were zeros. Inside the exchange, however, these same bit positions are used to retain the status bits which indicate the status of the unit. A brief description of these positions is presented below. A more complete description of the status bits can be found under "I-O Programming."

Bit 18, Unit Ready: This bit is on whenever the unit is in condition to accept an instruction from the computer or is executing an operation. When the bit is off, the unit is in the not-ready status and cannot be operated by the computer.

Bit 19, Exchange Program Check (EPGK): This bit is turned on when an operation concerning an external unit is terminated prematurely by a programming error.

Bit 20, Unit Check (UK): This bit is concerned with data errors and malfunctioning of the recording medium or equipment that can be identified with a particular channel. Among the causes of unit check are data transmission errors and I-O unit malfunctions. A data transmission error may be in the form of dropped bits. A unit malfunction may be in the form of a card jam or a broken magnetic tape.

Bit 21, End Exception (EE): This bit is used to signal a number of exceptional conditions usually associated with the recording medium or some subdivision of the data to be transmitted. It indicates that an operation has been terminated.

Bit 22, End of Operation (EOP): When no uncorrectable data errors are discovered in a READ or WRITE, the EOP bit in the control word is turned on at the end of the block in which data transfer is completed.

Bit 23, Channel Signal (CS): This bit signals operator request for attention or indicates that a unit has changed from not ready to ready status.

Bits 19 through 23 have corresponding indicators in the CPU indicator register and, when certain conditions are satisfied, cause the indicator to be turned on.

Bit 24, Suppress End of Operation (SEOP): This bit is turned on by a SEOP (suppress end of operation) instruction to suppress the end of operation indication upon a normal completion of the operation.

Bits 25 to 27, Flag Bits

These are control bits which permit certain variations in reading and writing operations.

Bit 25, Chain Flag: If on, another control word is obtained by the exchange when the count in the current control word reaches zero.

Bit 26, Multiple Flag: If on, more than one block of data may be read or written with a single instruction.

Bit 27, Skip Flag: If on during reading, transfer of data to storage is suppressed. The skip flag has no effect on writing.

Bits 28 to 45, Count

This field contains the number of words that can be transferred under control of the corresponding control word. (This field has a similar meaning in an index word.)

Bits 46 to 63, Refill Address

This field contains the address of the next control word to be obtained by the exchange when the count reaches zero and the chain flag is on. (This field has an analogous meaning in an index word.)

7.3 EXCHANGE DATA FLOW

Data flow is presented as the result of a specific instruction. The instruction format and the STRAP notation for the instruction are included in each example.

Before considering the actual data flow, note the physical characteristics of the exchange. See Figure 7.3-1. The exchange channels are numbered 32 through 63. A channel receives 9-bit bytes from the I-O device during a read operation. Eight of these bits are data and the ninth bit is a parity bit for checking purposes. Converting the input data to an 8-bit byte for the exchange is a function of the individual I-O control unit. The assembling of bytes to form a word is accomplished by the exchange.

During a READ or WRITE, the partial words are contained in the exchange storage. Two storage locations per channel are used for this purpose. These two data word locations are used alternately by the exchange. During a read operation, when one data word is completely assembled, assembling of the second word and the storage of the first word are started. The exchange storage contains 256 locations.

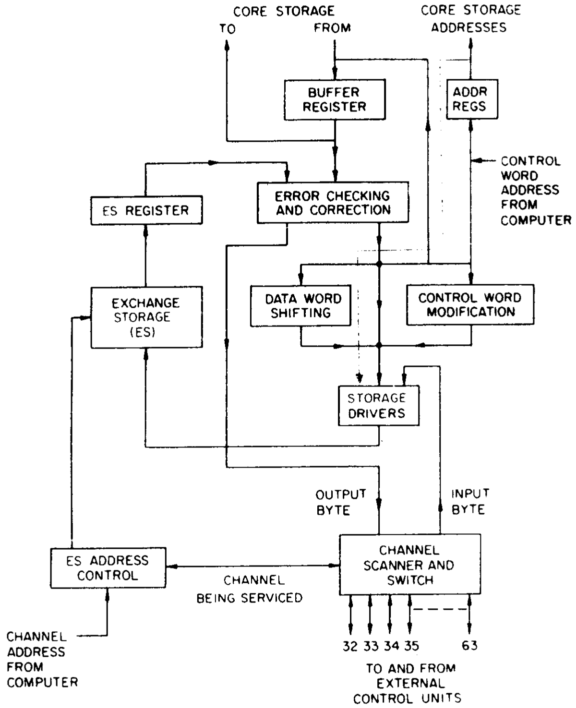


FIGURE 7.3-1. EXCHANGE DATA FLOW

7.3.1 Read Data Flow

READ locates the channel to be read and the control word to be used for the operation. If the exchange determines that the READ can be performed, an accept signal is sent to the CPU. The CPU is then free to proceed to the next instruction. Assume ten words are to be read from the I-O unit connected to channel 36 of the exchange, and placed in storage starting at location 500. To accomplish this objective the exchange must receive a READ (RD) specifying channel 36 and the location of a control word. The control word in turn must specify the starting address (500) and the number of words to be read (10). Assume the proper control word is located in address 7000.

Operation: Read ten words from channel 36; place words in storage starting at location 500.

Instruction Format:

	CHANNEL ADDRESS		1000	I	ADDRESS		OP	10000	I
0	12	18	24	28	32	49	51	55	60 63

Positions 32-49 of the instruction are used to address the control word. This address is subject to modification as called for by the I field in positions 60-63. The channel address located in the left half of the instruction is subject to modification as called for by the I field in positions 28-31. Note that the channel address appears in positions 12-18. Position 18 is normally considered the low-order bit of a half-word address.

The low-order bit of a full-word address is normally located in position 17. In the case of an I-O instruction, positions 12-18 are delivered to the exchange as a channel address. When written in STRAP notation, the channel address may be expressed in either of two ways. First, because the channel address includes instruction bit 18, this address may be expressed as a half-word address. For example, to address channel 36, bit positions 12-18 of the instruction would contain the binary number 0100100. Expressed as a half-word address, this becomes 18.0. If channel 37 is desired, a channel address of 18.32 could be used. The second method of expressing a channel address, in STRAP notation, is by using the actual channel number as a symbolic address. To address channel 36, for example, the number 36 could be used in the STRAP notation. Similarly, to address channel 37, the number 37 could be used. When designating a channel address in this method, the actual half-word address (18.0 or 18.32) would be substituted for the symbolic address when the instruction is processed for assembly by STRAP.

```
STRAP Notation:   RD, 18.0, 7000
                  /     |      \
              Read  Channel  Control word
                    Address    address
```

Operation Code (51-54): 0000

Sequence of Events: Consider the I-O device connected to channel 36 to be a card reader. Assume the card reader is in ready status and the first card to be read contains at least ten words of information. See Figure 7.3-1.

1. The instruction line, the control word address, and the channel address are delivered to the exchange from lookahead.

2. Status tested to determine acceptance or not.

3. The control word address is placed in the address register (clear register and insert byte flag).

4. The fetch of the control word is initiated (storage request).

5. The control word is delivered, via SBC, to the buffer register. The control word is then routed through the error checking circuits to the storage drivers. The exchange storage address control section is, at this time, addressing the proper exchange storage location for this control word. The control word is, therefore, routed through the storage drivers into the proper location of the exchange storage.

6. The card reader connected to channel 36 is started.

7. As mentioned previously, each channel of the exchange has two data word storage locations, referred to as data word A and data word B. At this time the exchange resets channel 36 data word A and places a 1-bit in the eighth from low-order position of the word, by reading out the old data word A and placing it in the word register. The contents of the word register are then dropped and all zeros plus a 1-bit in the eighth position are sent from the storage drivers to the exchange storage. As located in exchange storage, data word A now appears as all zeros with the exception of the eighth bit from the low-order end of the word which contains a one. This one bit is called the byte count

flag, used to indicate that the assembling of a data word is nearing completion. The first 8-bit byte of information from the card reader is routed through the channel scanner to the input byte register.

8. The new data word A is read out of exchange storage and placed in the word register. The contents of the word register are then routed through the error checking circuits and the data shifting circuits, where a left shift of eight positions is performed. After this shift of eight places, the byte count flag is in the sixteenth position of the word.

9. The shifted data word A and the contents of the input byte register are routed through the storage drivers to the exchange storage. The modified data word A now contains the first byte of information from the card reader.

10. When the next byte of information is available from the card reader a signal is sent to the exchange. This signal is called a service request. The exchange in the meantime has been scanning all channels to determine the presence of any service request. During the scanning of the channels, the storage address control section is addressing the corresponding data word location for each channel. Thus, when the service request from the card reader is received, the appropriate storage data word is being addressed by the address control section.

11. Data word A channel 36 is read out of exchange storage and placed in the word register.

12. The input byte from the card reader is routed to the input byte register.

13. Data word A is routed from the word register through the error checking circuits and the data shift circuits, where a left shift of 8 is performed, to the storage drivers. The input byte is routed to the eight low-order positions of the storage drivers.

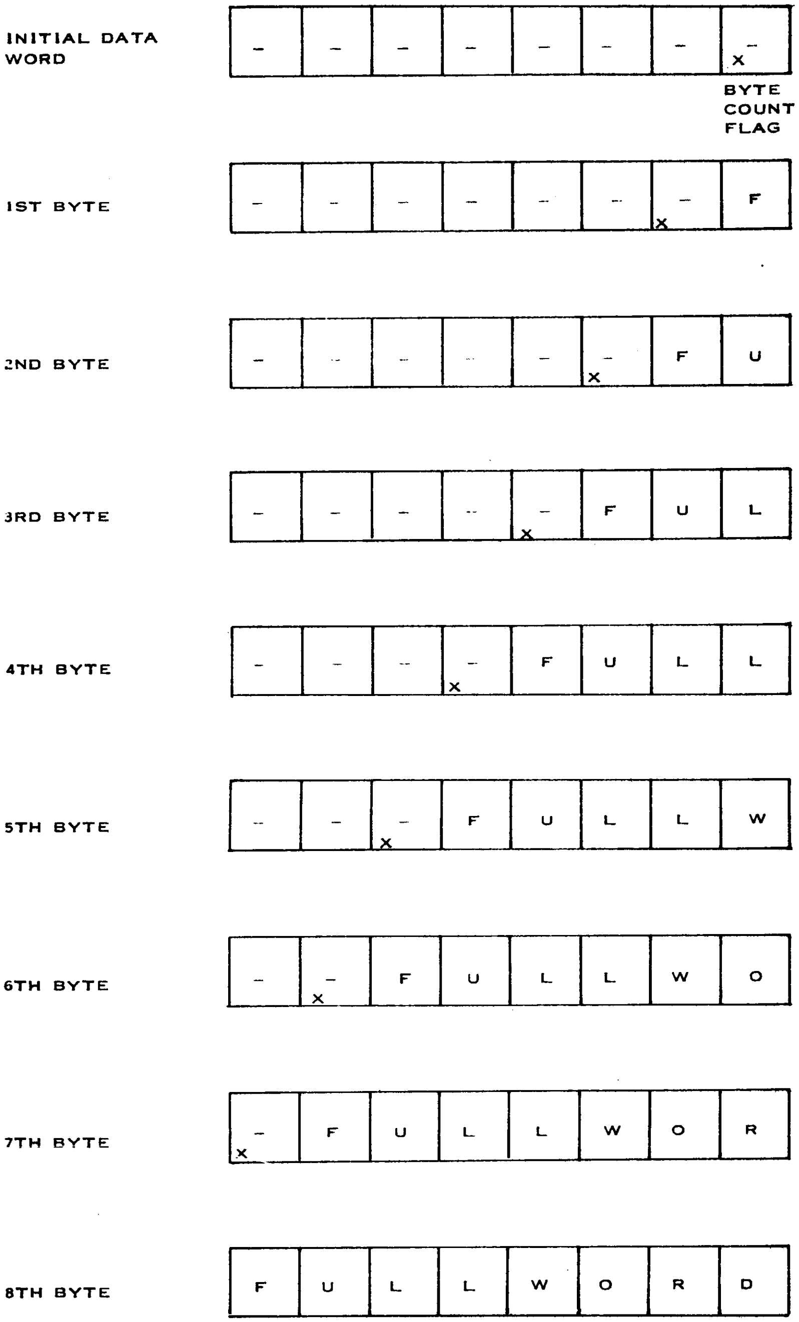
14. The storage drivers set the shifted data word A and the new input byte into the proper exchange storage location. At this time data word A contains two bytes of input data. The byte count flag is, at this time, located in the 24th bit of the word.

Steps 10 through 14 are repeated until, by use of the byte count flag, the exchange determines a full 7030 word has been assembled. Remember that the exchange is continually scanning the channels and, if more than one I-O device is in operation, the assembly of data words for each unit is being interspersed with the assembling of the card reader data word. The assembling of the card reader data word by bytes is shown in Figure 7.3-2.

During the processing of the seventh byte the exchange detects the presence of the byte count flag in the high-order position of the word. This condition indicates that the next byte will be the final byte for this word.

15. After processing the eighth byte, the completely assembled data word A is located in exchange storage.

16. The control word for channel 36 is read out of exchange storage and placed in the word register.



INITIAL DATA WORD	–	–	–	–	–	–	–	– X
								BYTE COUNT FLAG
1ST BYTE	–	–	–	–	–	–	– X	F
2ND BYTE	–	–	–	–	–	– X	F	U
3RD BYTE	–	–	–	–	– X	F	U	L
4TH BYTE	–	–	–	– X	F	U	L	L
5TH BYTE	–	–	– X	F	U	L	L	W
6TH BYTE	–	– X	F	U	L	L	W	O
7TH BYTE	– X	F	U	L	L	W	O	R
8TH BYTE	F	U	L	L	W	O	R	D

FIGURE 7.3-2. ASSEMBLY OF CARD READER DATA

17. From the word register the control word is routed through the error checking circuits to the control word modification circuits. At this time the data word address portion of the control word (500) is routed to the address register.

18. In the control word modification circuits, the count field of the control word is reduced by one and the data word address is increased by one.

19. The modified control word is then routed through the storage drivers back to its original location in exchange storage.

20. Data word A is read out of the exchange storage and placed in the word register. From the word register the word is routed to the error checking circuits where error correction code bits (ECC) are developed.

21. The data word and the eight ECC bits are routed from the error checking circuits to the buffer register. Prior to this time, the exchange may be assembling the next data word using data word B location.

22. Under control of the address register, SBC attends to the storing of the data word located in the buffer register.

The reading and storing of the first word is now complete. The modified control word contains a word address of 501 and a count field of 9. The entire process is repeated until, by detecting a zero count field, the exchange determines that the desired number of words have been read. The exchange is continually scanning the channels to test for service requests. In scanning, the exchange assigns the highest priority to the low-order channels. If more than one service request is received at one time, the lowest channel is honored first. After performing the necessary action required by the first service request, the exchange again scans the channels starting with channel 32. Because of this system, the service request on the lowest order channel will always be the next one honored. If the I-O devices which require the most frequent servicing by the exchange are connected to the low-order channels, they automatically receive the highest priority.

The need for a priority scheme is indicated by certain characteristics of I-O devices. The control unit for a card reader can retain a full card of information. The card reader itself can stop between cards. Because of these two factors, the card reader has a relatively low priority and would normally be connected to one of the higher exchange channels. If the exchange is kept busy by higher priority devices, the card reader merely stops and waits for its service request to be honored. When the exchange finds time to honor the card reader's service request, the reader is automatically restarted and reading continues.

The 729 IV tape unit reads and writes at a high rate of speed and cannot stop arbitrarily without coasting over some information. During a read operation, the tape control unit never contains more than 12 bits of information. Therefore, a tape service request must be honored immediately. For this reason, tape units are normally connected to the low-order channels.

Although the data flow through the exchange is somewhat complex, it is accomplished at very high speed. For example, the time required to read a data word out of exchange storage, shift it eight positions and place the data word and input byte back in storage is approximately one microsecond. This high speed gives the exchange the ability to operate many I-O devices simultaneously.

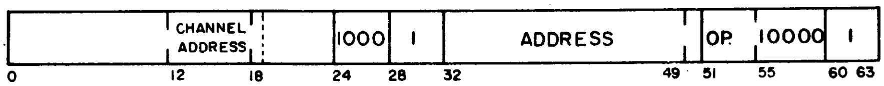
7.3.2 Write Data Flow

WRITE (W) initiates a write operation for the external device specified by the channel address. As in a READ, WRITE must also specify the location of a control word, which in turn must specify the number of words to be written and the location of the first word. If, upon receipt of the instruction information, the exchange determines that the operation can be performed, an accept signal is sent to the CPU. The CPU is then free to proceed to the next instruction.

For the purpose of explanation, assume one card is to be punched with 15 words from storage starting at location 700. The control word in this case must contain a data word address of 700 and a count of 15. Assume that the proper control word is located in storage location 500, and that the punch is in ready status and is located on channel 38 of the exchange.

Operation: Write fifteen words on punch; first word located at address 700.

Instruction Format



Operation Code (positions 51-54): 0001

STRAP Notation: W, 19.0 500 or W, 38, 500

write
channel address
control word address

Sequence of Events: Before transferring any information to the punch, the exchange will fetch the first two data words. See Figure 7.3-1.

1. The control word address is delivered to the address register in the exchange. The channel address is delivered to the exchange storage address control section.

2. Under control of the address register , the control word is delivered from main core storage to the buffer register in the exchange.

3. The control word is routed from the buffer register through the error checking circuits to the control word modification circuits. At this time certain control information to be used by the exchange is inserted into the ECC field of the control word.

4. From the modification circuits the control word is routed through the storage drivers and into the exchange storage under control of the channel address.

5. The control word is read out of exchange storage and placed in the word register.

6. From the word register the control word is routed through the error checking circuits to the modification circuits.

7. The first data word address is routed to the address register.

8. In the modification circuits the data word address is increased by one. The count field is not altered at this time. The control word is then routed through the drivers to the exchange core storage.

9. Under control of the address register, data word A is delivered to the buffer register from core storage.

10. Data word A is routed through the error checking circuits where ECC is checked. Also during error checking a byte count flag is inserted in position 64 (65th position) of the data word. While each location of the exchange storage consists of 76 positions, 64 of these positions contain the data word. During ECC mode of operation eight of the 76 positions contain the ECC bits for the data word. The remaining four bits are used by the exchange itself to retain certain control information.

11. Data word A with the byte count flag is routed from the checking circuits through the storage drivers to the proper location in exchange storage.

12. In preparation for fetching data word B, the control word is read out of exchange storage and placed in the word register. From the word register the control word is routed through the error checking circuits to the modification circuits.

13. Before modification, the data word address is routed to the address register. The address register now contains the address of data word B. The control word is then modified by increasing the address field by one, and the modified control word is routed through the storage drivers and back to exchange storage.

14. A start signal is directed to the punch control unit. This signal starts a sequence of events in the punch control unit. As a result of receiving the start signal, the punch control unit sends a service request to the exchange.

15. Under control of the address register, data word B is delivered to the buffer register.

16. From the buffer register, data word B is routed through the error checking circuits (where the byte count flag is attached to the word) to the storage drivers and exchange storage.

17. If the exchange is not busy servicing some other channel, the punch service request is honored and processing of data word A is begun.

18. Data word A is routed from exchange storage to the word register. From the word register the word is directed to the error checking circuits. From the error checking circuits the eight leftmost bits of the data word are routed to the punch channel as an output byte. The remainder of the word is routed through the shift circuits where a left shift of eight places is performed, and zeros are entered into the low-order positions of the word. The low-order portion of the data word now contains the byte count flag followed by seven zeros. The output byte is directed to the punch control unit where it is placed in a buffer. When the punch buffer capacity of 15 words is filled completely, the punch is started and the contents of the buffer are punched into the card. If less than 15 words are to be punched, the control unit fills the remaining positions of the buffer with zeros before punching.

19. The shifted data word is routed through the storage drivers and back to data word A location in exchange storage.

20. After placing the output byte in the punch buffer, the punch control unit sends another service request to the exchange.

21. Steps 17 through 20 are repeated until the exchange determines that a full data word has been delivered to the punch control unit. The end of the data word is recognized when position 0 of the word contains the byte count flag and all other positions of the word contain zeros.

22. When data word A has been completely processed, steps 17 through 20 are repeated using data word B.

23. During the processing of data word B, the control word is read out and modified and a new data word A is fetched.

24. Steps 17 through 23 are repeated until the count field of the control word equals zero. Because in this example 15 words were written, the punch is at this time started and the punch buffer contents are punched into the card.

The execution of WRITE is now complete. The count in the control word is not altered as a result of fetching the first two data words. Thus, the count field at all times contains the number of words yet to be delivered to the I-O unit and not the number of words yet to be fetched from storage. Because of this, the fetching of data words is terminated when the count reaches two. At the time the fetching of data words is terminated the count field is reduced in the normal manner until the count reaches zero.

7.3.3 Control Instruction, Data Flow

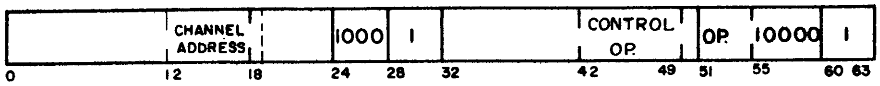
In addition to the basic read and write operations, the I-O devices can perform several other functions. For example, under program control, a tape unit may be directed to rewind the tape or backspace over a certain amount of information. These and several other operations are initiated by CONTROL (CTL) instructions. CTL uses the same word format as other I-O instructions. In the case of CTL the control word address field contains the bits designating the particular control operation to be performed. Only positions 42-49 of the control word address field are used for this purpose. The remaining portion of the address field is ignored.

The eight bits containing the control code are routed through the exchange to the appropriate I-O control unit as called for by the channel address of the instruction. The I-O control unit is also instructed to interpret this byte as control information rather than data.

For the purpose of explanation, assume the tape located on channel 32 is to be rewound. For this example, assume there is only one tape unit connected to the tape control unit.

Operation: Rewind tape located on channel 32.

Instruction Format:



Operation Code (positions 51-54): 0010

Control Code (positions 42-49): 01011110

STRAP Notation: REW, 16.0 or REW, 32

Rewind Channel Address

Sequence of Events

1. The control word address portion of the instruction is delivered to the address register. The channel address is delivered to the exchange storage address control section.

2. Data word A for channel 32 is read out of the exchange storage and placed in the word register. The output of the word register is then blocked and all zeros are routed through the error check circuits to the storage drives.

3. The eight bits of the address register containing the control code are routed to the leftmost eight positions of the storage drivers. These eight bits and the remaining zeros are routed into exchange core storage.

4. Data word A which now contains the eight control bits is again read out and placed in the word register.

5. The word register contents are routed to the error checking circuits. Here, as in the write operation, the leftmost eight bits are routed to the channel scanner as an output byte.

6. Because the tape control unit has been notified that this byte is control information rather than data, the byte is decoded.

7. As directed by the decoded control byte, the control unit causes the tape unit to rewind the tape.

7.4 DISK SYSTEM

The disk system provided with the IBM 7030 is composed of two major units: the IBM 7303 Disk Storage which provides large capacity auxiliary storage to supplement internal core storage, and the IBM 7612 Disk Synchronizer which provides the common control and data transfer paths for the disk system and the computer.

7.4.1 IBM 7612 Disk Synchronizer

The IBM 7612 Disk Synchronizer serves as the major control for the disk system. It controls the execution of data transfer between a disk storage and core storage. Basically, the method of control is the same as that used in the exchange. However, the high rate of data transmission between a disk storage and internal storage causes some differences in operational and program control.

The action of the disk synchronizer (DS) may be summarized as follows: the central processing unit initiates operation by sending an instruction to the DS. When the necessary interlock controls have been satisfied and the instruction accepted, control of the operation is transferred to the DS and the computer proceeds independently until the operation has been completed. All necessary control required for word assembly, word disassembly, and the disk file are provided by the DS, thus enabling it to operate independently of the CPU and exchange portions of the 7030 system. When the operation or operations specified have been completed, the CPU is signalled through the I-O status indicator bits of the interruption system.

The disk synchronizer has provision for addressing a maximum of 32 disk storage units. Each disk has its own channel address. However, only a single data transmission channel is available because of the high data transfer rate. Therefore, only one disk may be used for reading or writing at a time. Concurrently, however, other disks can be accepting and processing instructions for setting the read-write heads in position for subsequent data transmission.

Full words are transferred between the disk synchronizer and core storage. Each word consists of 64 information bits and eight error correction bits.

7.4.2 IBM 7303 Disk Storage

Each IBM 7303 Disk Storage system has a capacity of 2,097,152 (2^{21}) words. Data may be transferred between the disk storage and core storage at the rate of one full word every eight microseconds.

Data transfer between the disk storage and the DS is performed in bytes of 39 bits. The byte consists of 32 data bits and seven ECC bits. All bits (39) of one complete byte (one half-word), including the seven ECC bits are recorded on the surfaces of the disks simultaneously.

Physically, the disk storage contains 78 recording surfaces for the purpose of recording data (and ECC bits). Both sides of each disk are used as recording surfaces. Several additional recording surfaces in a disk unit are used for such things as addressing, timing, and shielding. The disk array is divided into an upper module (A) and a lower module (B). Each module contains 20 recording disks which furnish the 39 surfaces needed to record a single byte. The two bytes that make a full word appear in adjacent recording locations in the same module. That is, the first byte of a word is written in a parallel manner by 39 read-write heads; after a predetermined length of time, the disk has revolved the required distance and the second byte is recorded at the adjacent location on the disk by the same R-W heads. The 39 R-W heads are mounted on a common mechanism and operate together.

To understand how the programmer positions these heads, it is necessary to learn the terms associated with the disk. Figure 7.4-1 illustrates the physical layout of an individual disk surface.

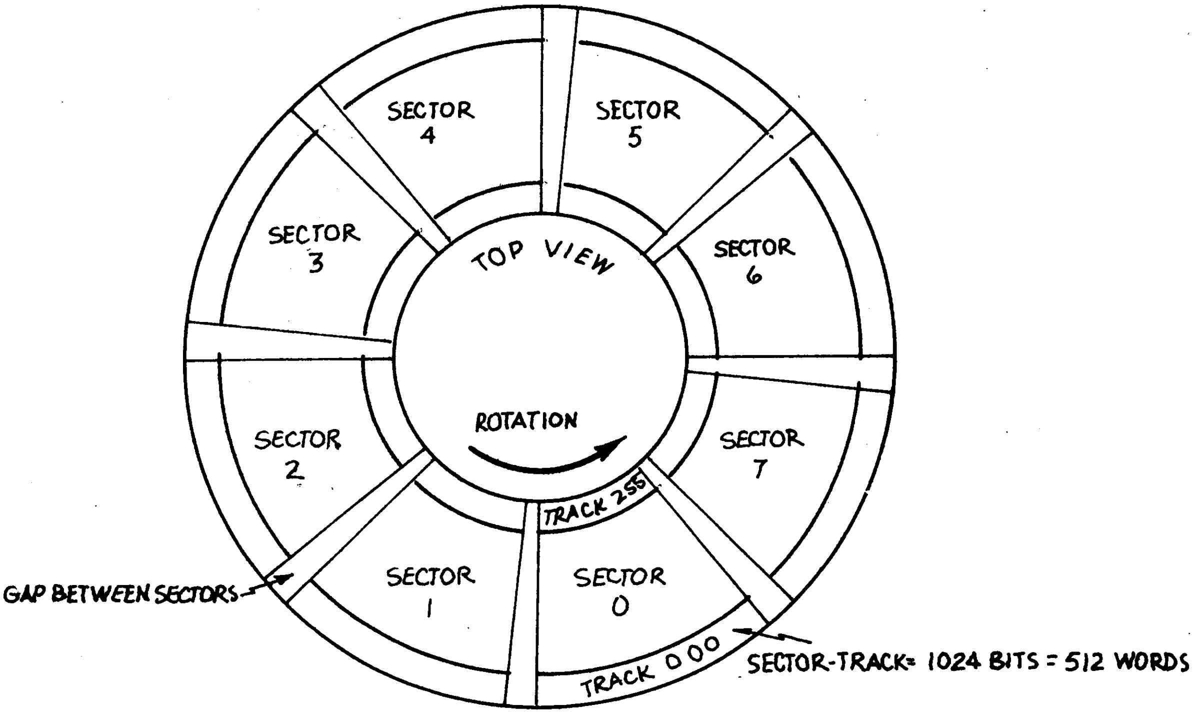
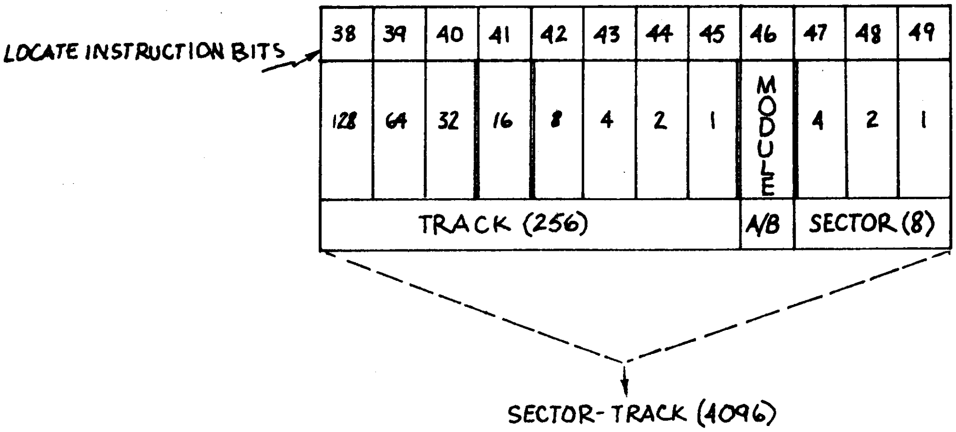


FIGURE 7.4-1. INFORMATION DISK IN IBM 7303



38	39	40	41	42	43	44	45	46	47	48	49
128	64	32	16	8	4	2	1	MODULE	4	2	1
TRACK (256)								A/B	SECTOR (8)		

FIGURE 7.4-2. ADDRESS FORMAT - LOCATE

Referring to Figure 7.4-1, the surface of the disk is divided into eight "sectors" and 256 concentric "tracks." The outermost track on the disk is 000 while the innermost track is track 255. Each track contains 8192 bit positions. However, each pair of bit positions are part of one data word. Therefore, 4096 words are contained on one track, in one module. The portion of a track within each sector is called a "sector-track address." Each sector-track address contains 512 words. The sector-track is the smallest addressable location on the disk. From the above description, the total word capacity of a disk unit can be calculated.

512 words/sector-track x 8 sectors = 4096 words/track
4096 words/track x 256 tracks/disk = 1,048,576 words/disk module
(1,048,576 words/disk module) x 2 modules = 2,097,152 words/disk

There are 4096 addressable units (sector-tracks) in a disk unit.

The addressable scheme employed by the disk unit is shown in Figure 7.4-2. Note that if a large amount of information is transferred, beginning at section 000 of module A, one complete track (8 sectors) of module A is used, and then the data transfer proceeds from sector 000 of module B. After filling track 000 of module B, track 001 of module A is used. The advantage in this addressing scheme is that while one module is reading or writing, the R-W heads of the other module are being positioned to continue the data transfer.

Before transmission of data is started, it is necessary to set the access arms to the track containing the desired starting sector-track. After this initial positioning, the arms are automatically positioned, as successive data transmission requires successive sequential sector-tracks. Programmed repositioning is required only when nonsequential or random sector-tracks are used.

Positioning of the arms under program control requires a maximum of 180 milliseconds (ms) when the access arms are moved from the outermost to innermost tracks. However, when automatically moving from track to track in the same set of disk faces, 33.3 to 50 ms are normally required. This corresponds to the time required for one revolution, amd thus the data flow may be continuous since alternate sets of access arms are used and one set of access arms is reading or writing while the alternate set is in motion.

Data Paths

The Disk Synchronizer contains control features necessary to regulate the information flow over a single data channel linking core storage to disk storage. This is the read and write path for data to and from disk storage. Because data transfers between the DS and core storage are on a full-word basis, the facility for word assembly and disassembly is provided in the Disk Synchronizer. Two words are buffered, one in a full-word register, the second in two half-word byte registers. Through the use of the three registers, the data flow is synchronized between core storage and disk storage. All word and byte transfers are checked for errors through the use of the error check and correction circuits associated with each type of register in the DS. These circuits use the error check and correction bits associated with the data bits during transfer.

When data are assembled or disassembled, the associated ECC bits are not valid for the newly-formed word or byte. A new set of ECC bits must be generated to fit the new word or byte just formed. Through a mathematical relationship existing between old and new correction bits, the continuity of checking is preserved by tests made in the DS before further processing of the data. The disk system and the exchange differ in their operation in the following ways:

1. Only one disk storage may be reading or writing at one time.

2. No use is made of control word flag bits or refill field and they are discarded after checking the control word error correction and check bits. Only the data word address field and word count field of the control word are retained and used by the disk synchronizer.

3. In executing COPY CONTROL WORD, only the data word address and word count fields are returned to core storage. CCW is a data transfer operation and, therefore, cannot proceed if the disk synchronizer is reading or writing.

Roll Mode (Automatic Access-Reduction)

Roll mode is not an instruction but is a method used to shorten access time when reading or writing an entire track of information (4,096 words). The DS tests each R-W instruction for roll mode conditions. If the conditions are met, the roll mode feature is used without need for a specific instruction. The roll mode conditions are:

1. The word count must be 4,096.
2. The data word address must be 4,096 or a multiple of 4,096.
3. The LOC address must be a sector-track address of zero, eight or a multiple of eight.

The test for the first condition is made on the control word count field to determine if a full track of information is specified. The 13 low-order bits of the data word address of the control word are tested for all zeros, but the first data word is not transferred to this address unless transmission starts from the beginning of the sector-track in sector zero. A third test is made to insure that 4,096 words are located in the same track. This is done by testing the three low-order positions of the LOC address in the disk unit for zeros. This specifies the start (sector 0) of any track. Normally the LOC address specifies the exact and only starting point of data transmission. In roll mode, the data transmission may start at the beginning of any one of eight sector-tracks in the track. The data word address is modified to synchronize the starting address of core storage with that of the sector-track position on the disk.

Reading or writing begins at the next sector-track to pass under the R-W heads, and continues for one full revolution (track) without changing tracks. The maximum access time is that required to reach the next sector-track, which is one-eighth of a 33 ms revolution (4.1 ms). The average delay is about 2.1 ms. The address proceeds to all zeros corresponding to the beginning of the sector-track in sector zero, when the disk reaches the end of the track of sector seven. An orderly transfer of data is accomplished by this means and causes the data to appear in core storage in the same order as on the disk, starting at sector zero. After transfer, the data always appear in the correct locations, even though the transfer did not take place in the order specified by the original control word. Therefore, it is unnecessary for the program to alter the data.

7.5 THE 7030 ERROR CORRECTION CODE

A 7030 word in core storage has 72 positions, of which 64 contain data while the remaining eight positions contain error correction code (ECC) bits. The ECC bits provide a means of automatically detecting and correcting single-bit errors. These bits also provide for the detection of double-bit errors. No attempt is made to correct double-bit errors. The detection of a double-bit error during data transmission results in the setting of the machine check indicator. Single-bit errors which occur during data transmission are automatically corrected and no indication is made. Normally, a 7030 word contains ECC bits only when the word is in storage or en route to (or from) storage. As explained in "I Unit Data Flow," after checking, the ECC bits are dropped. Similarly, during a normal WRITE operation, after checking, the exchange drops the ECC bits of the word received from storage. During a "to-memory" operation such as a STORE, new ECC bits are generated in the I Checker and affixed to the data word. Similarly, during a READ, the exchange may generate ECC bits and affix them to the data word before transmitting the word to storage. All generation of ECC bits is accomplished by the exchange, the I Checker, and the disk synchronizer.

Before considering the scheme employed for generating ECC bits, parity checking should first be considered. In parity type checking, one position of a group is reserved for a parity bit. The machine then sets this parity bit to a one or a zero as required to make the total number of ones contained in the group an even number (even parity), or an odd number (odd parity). The exchange uses an odd parity for transmission of data to and from the I-O control units. In a WRITE, the exchange delivers eight bits of data and one parity bit to the I-O control unit. The state of the parity bit is determined by the status of the eight data bits. If the total number of one bits in the data byte is even, the exchange transmits a one bit in the parity position. If the total number of ones in the data byte is odd, a zero parity bit is transmitted. The I-O control unit then tests all nine positions for an odd number of ones. If during the transmission of the data any position changes status, the total parity becomes even and the I-O control unit detects the error. No method of correction is provided by parity checking and even multiple errors in a group may go undetected.

The error correction code bits actually constitute a special type of parity check. In a normal parity scheme as described above, each data bit contributes to the status of the parity bit. Stated another way, all the data bits are combined to determine the status of the parity bit. Note that each data bit exhibits an equal amount of influence on the status of the parity bit. In ECC checking, the parity bit is replaced by a group of bits which record a parity <u>sum</u>. Each data bit contributes to the parity sum. No two data bits contribute the same amount, however; the exact amount contributed by any one bit is equal to the position of that bit in the data word. Thus, position 1 contributes only to ECC bit 1; position 2 contributes to ECC bit 2; position 3 contributes to the parity of both ECC bits 1 and 2. A theoretical example of how individual positions affect the ECC bits for a 7-bit data word with three ECC bits is:

	Data Word							ECC Bits		
Positions	1	2	3	4	5	6	7	4	2	1
ECC bits affected	1	2	2-1	4	4-1	4-2	4-2-1			

Note that ECC bit 1 is affected by every odd-numbered position of the data word, but the only data position that affects ECC bit 1 exclusively is position 1. Furthermore, each data bit affects the ECC value by an amount equal to the position occupied by the data bit. Using odd parity, consider the ECC bits required for the data word shown below.

	Data Word								ECC Bit
Positions	1	2	3	4	5	6	7		
Number =	1	1	1	1	1	1	1		
Positions affecting ECC 4				X	X	X	X	= even; set ECC to 1	4 = 1
Positions affecting ECC 2		X	X			X	X	= even; set ECC to 1	2 = 1
Positions affecting ECC 1	X		X		X		X	= even; set ECC to 1	1 = 1

Note that the total number of 1-bits contributing to ECC bit 4 is even. Therefore, a one is placed in ECC 4 to produce an odd total. Similarly, an even number of 1-bits contribute to ECC bit 2 and a one is placed in ECC bit 2. A one is also placed in ECC bit 1. The ECC bits for the example, therefore, are 111. Assume now that the same data word is transmitted, complete with ECC bits, from storage to the CPU, but that the bit in position 5 is dropped en route to the CPU. By generating new ECC bits and comparing them to the transmitted ECC bits, the position in error is detected.

	Data								ECC		
	1	2	3	4	5	6	7		4	2	1
Word transmitted =	1	1	1	1	1	1	1		1	1	1
Word received =	1	1	1	1	0	1	1		1	1	1
Positions affecting ECC 4				X		X	X	= odd; set ECC to 0	4 = 0		
Positions affecting ECC 2		X	X			X	X	= even; set ECC to 1	2 = 1		
Positions affecting ECC 1	X		X				X	= odd; set ECC to 0	1 = 0		

Transmitted ECC = 111
Generated ECC 1 = 010
Difference 101 = 5 (invert position 5)

Notice that the generated ECC value differs from the transmitted ECC value by an amount of 5. This is logical because the only position (5) which can contribute a value of 5 was dropped during the transmission. Because the 7030 is a binary machine, a single-bit error is corrected by merely inverting the position known to be in error. When the system detects an error it must be determined whether one or two positions have failed. In the above example the same indication would result if both positions 1 and 4 had been dropped. If this was interpreted as a single-bit error, the machine would invert position 5 and three positions would then be in error. The distinction between single and double-bit errors is determined by the eight ECC bit. This bit is actually used as a parity bit for the entire word, including the seven normal ECC bits. An odd parity is used for this purpose in the 7030. An example of a double-bit and single-bit error is illustrated below.

	Data							ECC			Parity
Positions	1	2	3	4	5	6	7	4	2	1	P
Original number	1	1	1	1	1	1	1	1	1	1	1
Single bit error	1	1	1	1	0	1	1	1	1	1	New parity = 0
Generated ECC =								0	1	0	Difference in old and new parity
Position in error =								1	0	1	indicates a single-bit error
Double bit error	0	1	1	0	1	1	1	1	1	1	New parity = 1
Generated ECC =								0	1	0	Same parity indicates
Difference =								1	0	1	double-bit error.

In this example, the new parity bit is generated using the entire transmitted word before the new ECC bits are determined. Thus, an odd number of changes in the word results in a new parity bit status. An even number of changes in the word produces the original parity bit status. The checking system, therefore, recognizes a difference in the ECC's and a difference in the parity bits as a single-bit error. A double-bit error is recognized when the ECC bits differ but the parity bits are alike.

The ECC scheme uses seven ECC bits and one total parity bit. The ECC bits are labeled C0, C1, C2, C4, C8, C16, C32, and CT (parity). The lowest order ECC bit is labeled C0 to coincide with position 0 of the 7030 word.

The actual coverage of the 7030 ECC bits is presented in Figure 7.5-1. Note that the assignment of check bits is very similar to the hypothetical case previously discussed. The major difference in this scheme is the fact that C0 covers the first 33 bits of the data word. This method is used so that each data bit is covered by at least two ECC bits. Therefore, if the generated ECC differs from the received ECC by only one bit, it is reasonable to assume that the transmitted ECC bit, rather than a data bit, is in error. Thus, if every data bit is covered by at least two ECC bits, the correction scheme may also provide single-bit error correction of the ECC bits as well as the data bits. The 7030 does correct single ECC bit errors.

Note in Figure 7.5-1 that position 0 is covered by C32 as well as C0. This provides the double coverage required to distinguish data bit errors from ECC bit errors. Note also that position 32, an even number, is covered by C1. This is a special case to provide a method of distinguishing a data error in position 32 from a data error in position 0.

7.6 INPUT-OUTPUT PROGRAMMING

Once started, the I-O devices complete their operations independently of the CPU, but because the I-O devices are completely under control of the stored program, no action is performed by any I-O device unless the device is placed in operation by an instruction from CPU. This section of the manual deals with the instructions used to control the I-O units that are connected to the exchange.

Information is transferred between the external units and main storage by a READ or WRITE instruction given by the program. This instruction specifies the operation's external unit (channel address) and the location of a control word. These three pieces of information are sent to the exchange, and the computer proceeds to the execution of the next instruction in the program.

The control word defines the storage area to be used in data transfer by specifying the initial data word address in main storage and a count of the number of words to be transferred. After a READ or a WRITE is received, the exchange obtains the specified control word from storage and stores it in the exchange. The desired unit is then started, and reading or writing proceeds using the storage area indicated by the control word.

As each data word is transferred to or from storage, the control word in the exchange is modified: the data word address is advanced by one, and the count is decreased by one. The control word in the exchange always contains the current data word address and a count of the number of words remaining to be transferred. The

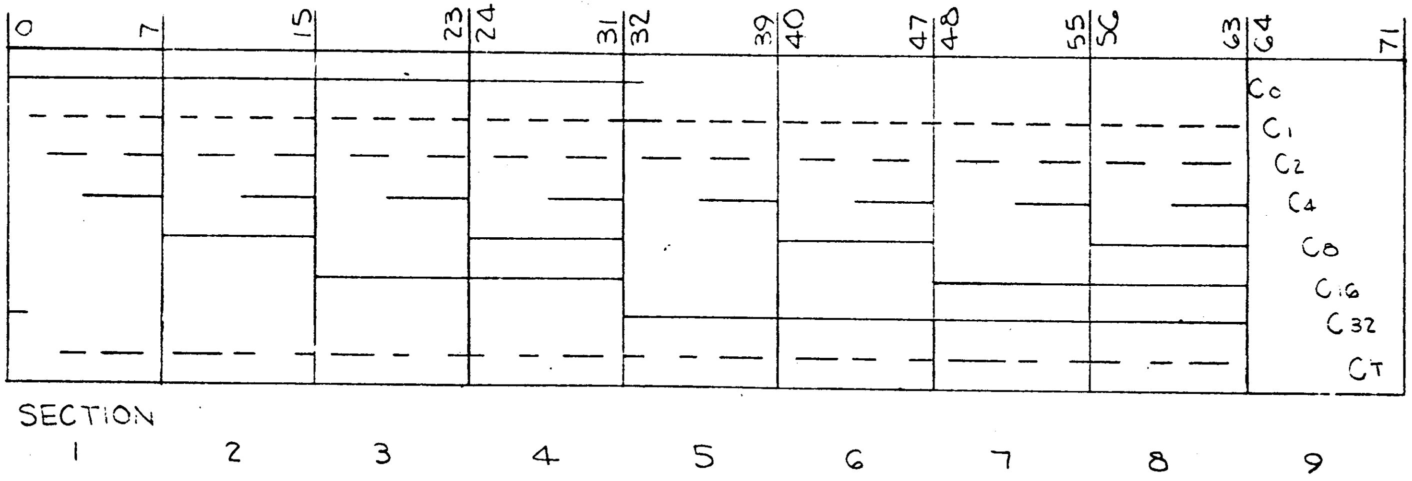


FIGURE 7.5-1. ECC BIT-GENERATION FOR EACH DATA BIT

control word in main storage, however, is not affected during the data transfer. Thus, the control word stored in main storage retains the initial data word address and count, and it can be used repeatedly to transfer data to or from the same storage area. There are four variations of the basic reading or writing operation, depending on the setting of the flag bits in the control word:

1. The external unit may terminate reading or writing if it reaches the end of a block before the entire storage area defined by the control word has been used. ("Single" block operation)
2. More than one block may be read or written with one instruction, until the specified storage area has been used. ("Multiple" block operation)
3. After the storage area defined by a control word is exhausted, it is possible for the exchange to substitute another control word and continue data transfer without stopping, using the storage area defined by the new control word. ("Chaining")
4. During reading, it is possible to suppress entry into storage of selected portions of the information. ("Skipping")

A block of data is defined for each type of unit as the amount of information recorded in the interval between adjacent starting and stopping points of the unit. The length of the blocks depends upon the type of unit used: a card, a line of printing, or the information between two consecutive gaps on tape.

7.6.1 ECC Mode

In a previous explanation of I-O data flow during a write operation, it was stated that, after checking, the exchange ignores the ECC bits and delivers only the 64 data bits to the I-O control unit. This description is true if the I-O unit is operating in the normal (no-ECC) mode. However, an alternate method of writing is also available. This method, called "ECC mode," causes the 8-bit ECC byte and the eight data bytes to be delivered to the I-O unit. In this mode, the first byte delivered to the I-O unit consists of the eight ECC bits associated,with the data word, followed by the eight normal bytes that make up the data word. The I-O unit records the entire 72 bits on the output medium. In this manner, the data word contains the appropriate ECC bits even when the word is recorded as output. Information which has been written in ECC mode may then be read in the same manner. When reading is in ECC mode, the ECC bits and the data bits are delivered to the exchange, which generates new ECC bits and compares them with the ECC bits received from the I-O unit. Automatic error correction may then take place in the normal manner.

When reading is in no-ECC mode, any parity error encountered in the transmission of bytes from the I-O unit to the exchange results in a unit check indication. When reading is in ECC mode, two parity errors in the same word are required to cause a unit check. In ECC mode when only one parity error is encountered in a word, no unit check indication is given, and the exchange is allowed to correct the single error in the normal ECC manner. Because, in ECC mode, the output word contains nine bytes rather than eight, the number of words that can be recorded in the form of a punched card is reduced. Writing in ECC mode gives the punched card a capacity of thirteen 72-bit words. Columns 79 and 80 of the card are not used in ECC mode. The ECC mode of operation does not apply to the printer or the console.

7.6.2 Chaining

The chain flag, position 25, of the control word specifies whether this is the last control word or whether another one follows. If the chain flag is zero, the present control word is the last one, and the refill address is not used. If the chain flag is one, the refill address specifies the address of another control word which the exchange obtains automatically whenever the count reaches zero. Operation then continues. The chaining technique is used to permit scattering portions of a block of information in different areas of storage. A "chain" of control words is, in effect, formed in storage by having each control word specify its successor. The exchange then automatically follows this chain until it encounters a control word with a zero chain flag, or until the external unit stops the process.

7.6.3 Single and Multiple Block Operation

The multiple flag specifies whether a new block of data may be started at the end of the current block or not. When the multiple flag, position 26, of a control word is zero, reading or writing proceeds either until the unit reaches the end of a block or until the control word reaches a count of zero, whichever happens first. If a new control word in a chained operation is fetched before the end of the block, the multiple flag of the new control word determines, according to the above rule, the mode of operation during the time the new control word is in force. It follows that if all control words in a chain have the multiple flags set to zero, only a single block of data may be read or written with one instruction. It is possible, however, to read or write less than a block of data by specifying a count which is less than the number of words in the block.

When the multiple flag of a control word is one, operation always proceeds, if not terminated by exceptional causes, until the count of the control word reaches zero. If the unit meanwhile reaches the end of a block, a new block is automatically started. As before, in a chained operation the multiple flag of the control word in force always determines the mode of operation. With the multiple flag set to one, it thus becomes possible to read or write more than one block of data with one instruction.

All signalling between the unit and the exchange is done one block at a time. This applies to the status indications from the unit to the exchange as well as to the control information sent by the exchange to the unit. To obtain multiple block operation, the exchange merely signals the unit to restart as soon as the end of a block is reached, as if a new READ or WRITE had been given by the computer for each separate block. The distinction between single and multiple block operation is thus retained at the exchange and is not apparent at the unit.

A multiple block operation may, however, be terminated before reaching the last block specified by the control words. If an exchange program check due to inaccessible storage addresses, a unit check for other than data errors, or an end exception occurs, the operation is terminated at the end of the block during which the condition is discovered. A unit check indication due to data errors permits the operation to continue for the remaining blocks.

Whenever the end of a block is reached and a new block is started, data transfer continues with the next full word in storage. If a block is not an integral number of full words long, the remaining bytes, on reading, are filled with zeros.

The case where both the multiple and chain flags are on and the control word count reaches zero anywhere within a block is treated in a special way. When the count reaches zero the exchange sends a disconnect signal to the unit, indicating that it does not expect to transfer any more data to or from the unit until the end of the current block is reached. Upon receipt of this signal, the unit proceeds to the end of the current block with no data transfer taking place and no control word modification being performed. On reading, the remainder of the current block is ignored. On writing, no more data are sent to the unit until a new block is started. When the end of the current block is reached the unit is restarted, the new control word is fetched, and the operation is continued with the new block. Thereafter, chaining and multiple block operation take place in accordance with the flags of the new control word. Thus, in this mode of operation, it is possible to force a gap on tape at the end of any desired control word in a chain. Similarly, it is possible to read the first portion of several blocks as the result of a single READ. The action to be expected for various control words is illustrated in the following chart.

Multiple Flag	Chain Flag	Count 0 = Zero 1 = Not Zero	End of Block 0 = Yes 1 = No	Action
0	0	0	0	End operation
0	0	0	1	Disconnect, then end op.
0	0	1	0	End operation
0	0	1	1	Continue
0	1	0	0	End operation
0	1	0	1	Chain and continue
0	1	1	0	End operation
0	1	1	1	Continue
1	0	0	0	End operation
1	0	0	1	Disconnect, then end op.
1	0	1	0	Restart and continue
1	0	1	1	Continue
1	1	0	0	Chain, restart and continue
1	1	0	1	*
1	1	1	0	Restart and continue
1	1	1	1	Continue

* The action here is: disconnect, wait for end of current block, restart, chain and continue.

7.6.4 Skipping

The skip flag (position 27), when zero, specifies normal reading. During writing, the flag is meaningless and is ignored. When the skip flag is one, no data words are transferred to core storage. The operation of the unit is still continued, and the system goes through all the sequences associated with data transfer between the exchange and the unit. The control word is also appropriately modified by incrementing its address and decrementing the count, and, if necessary, a new control word is fetched in chained operations. The assembled data words, however, are not transferred to main storage. By setting the appropriate control words in chain to skip or not, any selected portion of data may thus be suppressed during reading. Because the testing for data word addresses that exceed the available core storage is performed in the core storage controls, reference to non-existent core storage locations cannot cause an exchange program check indication when reading with the skip flag on. The testing for addresses 0 through 31, however, is performed within the exchange and is not suppressed during skipping. Any reference to core storage locations 0 through 31, therefore, always terminates the operation and causes exchange program check. The exchange program check indication is always given when an attempt is made to fetch a control word from a core storage address that is not accessible to the exchange.

Thus, the program can specify any address above 31 when reading with the skip flag on. If in the process of control word modification the address 2^{18} is reached, the next cycle resets the address to 0. Because address 0 is invalid, the operation is terminated when an attempt is made to store a word at this location.

The presence of the skip flag does not affect the treatment of the unit check and end exception indications. When a unit check or end exception occurs, the operation is terminated in accordance with the general rules.

7.6.5 Examples of Multiple and Chain Flag Operations

The action to be expected during a write operation for various control words is shown in Figure 7.6-1. The symbols used are defined as: CNT = Count; C = Chain Flag; M = Multiple Flag; X = 0 or 1.

Using the same symbols, several examples of reading with multiple and chain flags are presented in Figure 7.6-2. In these examples all control words contain a count of ten. Dashed lines indicate skipping over the remainder of the block with no data transfer and no control word modification being performed.

7.6.6 Byte Weight

Although the exchange can service a large number of units simultaneously, it is possible to exceed the servicing capacity of the exchange.

The limitations on simultaneous data transfer between the exchange and the units is expressed by means of the byte-weight system. The byte-weight of a unit indicates the relative amount of exchange capacity necessary to service the unit. It depends upon the shortest interval in which the exchange may have to respond to the unit's byte request, and is not necessarily proportional to the data transmission rate of the unit.

As far as estimation of the maximum data flow between the exchange and the units is concerned, the external units can be classified into two types: non-buffered units, whose operation cannot be interrupted without loss of information, and units which are buffered or because of some other reason can be delayed in case of conflict.

The latter type of units (card readers, card punches, consoles, and printers) are flexible in timing their data transmission and have a byte-weight of zero. When the exchange cannot service their byte requests, their operation is interrupted. This interruption, however, is not associated with any loss of information. As soon as the excess load is removed and the exchange has time to spare, the mechanism of the unit is automatically restarted and the original reading or writing operation is continued. The interruption does not have any effect on the completion of the operation, and no indication is given to the computer to this effect.

In the non-buffered type of external units, such as tape units, data are transmitted continuously between the external unit and core storage. These units have non-zero weights. When exchange capacity is exceeded, such units cannot interrupt their operation and some data may be lost, causing an automatic unit-check indication to the computer. This indication is identical to that given in case of data errors and can be treated accordingly.

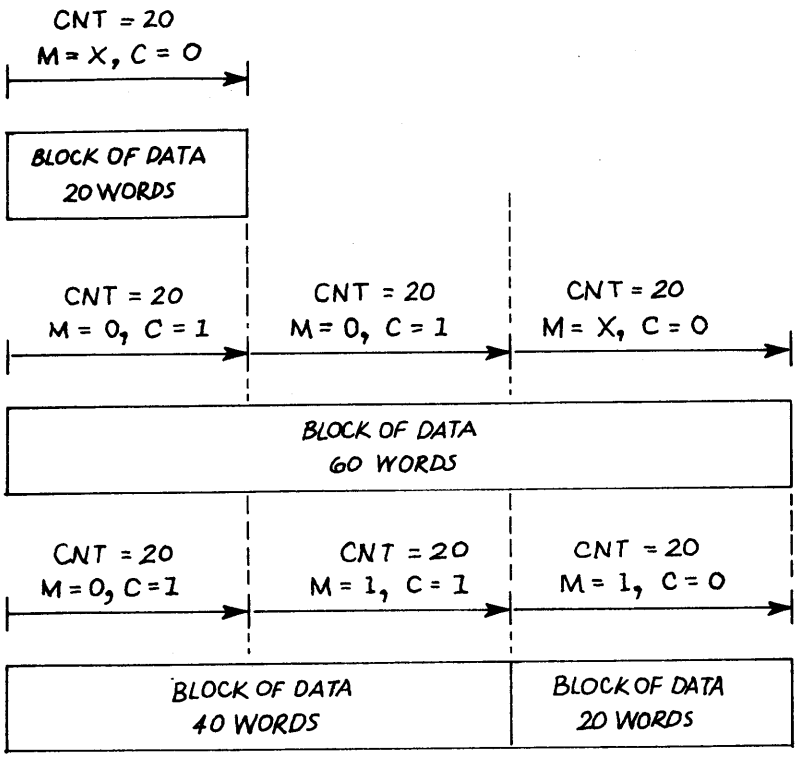


FIGURE 7.6-1. FLAGGED WRITE OPERATIONS

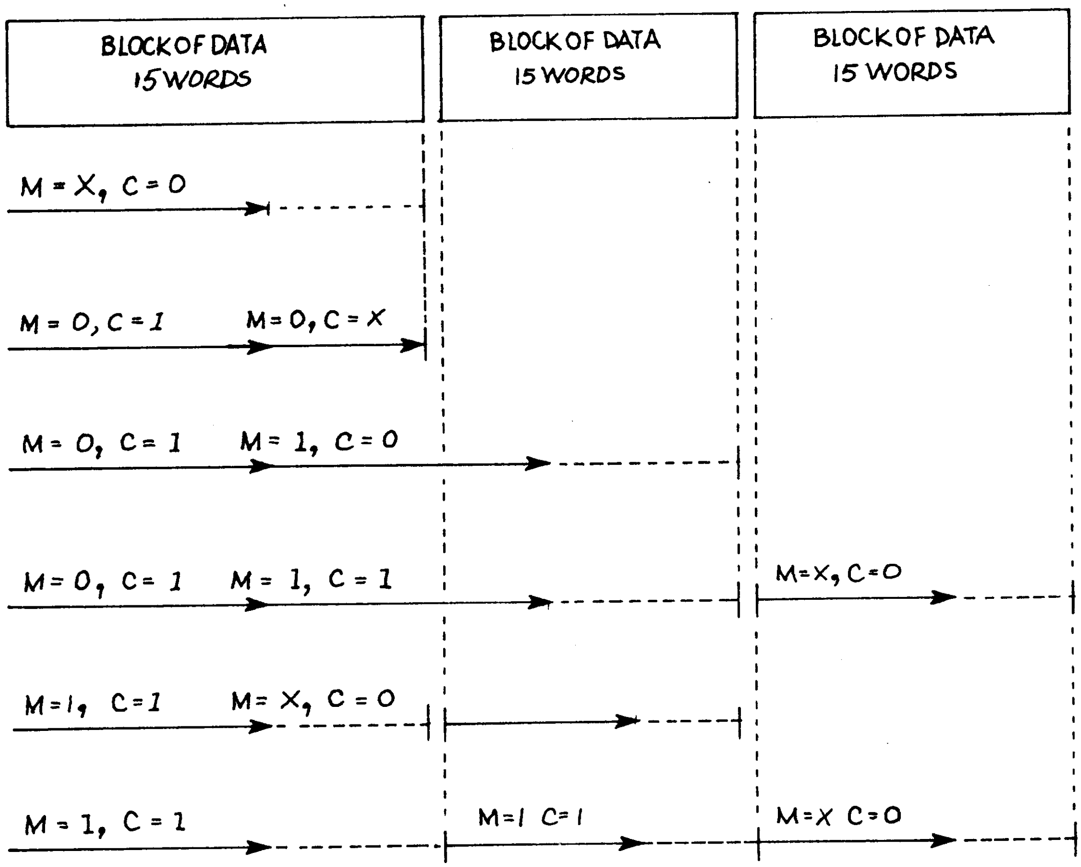


FIGURE 7.6-2. FLAGGED READ OPERATIONS

To avoid exceeding exchange capacity, the program can control the amount of data flow in process by adding the byte-weights, of all units operating, in the form of a byte-weight count. The scale of byte-weights is defined so that a byte-weight count of 256 represents the maximum safe load on the exchange. The total data transmission rate of the exchange corresponding to this byte-weight count is not fixed and depends upon the particular set of units used. The defined byte-weights for the standard I-O units are:

Unit	Byte-Weight
729 High Density	32
729 Low Density	9
Card Reader	0
Card Punch	0
Printer	0
Console	0

7.6.7 Word-Weight

Because the byte-weight of a unit reflects only the rate of transfer of bytes of data between the exchange and the units, it is fixed for a given type of unit and does not depend upon the mode of operation. Normally the sum of the byte-weights indicates when exchange capacity is exceeded. The byte-weight system, however, does not reflect the load imposed on the exchange by the transfer of data words for the zero byte-weight units and the fetching of control words for chained operations. When extensive chaining is employed or many zero byte-weight units are operated, it is possible that the word transfer between the exchange and the core storage may limit the capacity of the exchange.

To facilitate the evaluation of the total instantaneous rate of transmission of information between the exchange and core storage, a word-weight is defined for each external unit, including the zero byte-weight units. The word-weight of a unit is an indication of the unit's data transmission rate. When a unit is engaged in chained reading or writing, its word-weight must be doubled. The word-weight count is the sum of the word-weights of all units reading and writing, and is an indication of the amount of information that is currently being transmitted between the exchange and core storage. As in the case of the byte-weight system, the scale of the word-weight system is defined so that a word-weight count of 256 represents the maximum safe load on the exchange. When the exchange capacity to communicate with core storage is exceeded and a data word or a control word is not transferred, a unit-check indication for that channel is given. The communications between the exchange and core storage limit the maximum instantaneous information transmission rate of the exchange to 100,000 words per second. The assigned word-weights for the various standard I-O units are:

Unit	Word-Weight
729 High Density	17
729 Low Density	6
Card Reader	5
Card Punch	5
Printer	5
Console	3

The two weight count systems provide a programming means for keeping the total input-output data transmission rate below the maximum safe limit. When one of the weight-counts is exceeded, there is danger of losing some information. Note, however, that in specifying the weight systems, the maximum possible instantaneous load on the exchange is considered. In the case of chaining, doubling of the word-weight of a unit affords a further factor of safety in that it permits chaining to occur after each word transmitted to or received from core storage. Thus, on the average, the exchange will not operate as close to capacity as indicated by the weight counts. Unless many channels are installed and units with high data transmission rates are used, in some installations it may not be possible to exceed the exchange capacity.

7.6.8 Weight Count Example

Consider a system at an instant when the following units are executing READ or WRITE:

Type of	No. of Units		Units	Weight	Weight	Counts
Unit	Operating	Chaining?	Byte	Word	Byte	Word
729 High Density	2	No	32	17	64	34
729 Low Density	1	Yes	9	6	9	12
Card Reader	2	No	0	5	0	10
Card Punch	1	Yes	0	5	0	10
Console	1	No	0	3	0	3
			Total weight counts		73	69

Both the byte and word weight counts are well below the maximum limits, and there is no danger of exceeding exchange capacity.

7.7 INPUT-OUTPUT INDICATORS

To permit effective simultaneous operation of the computer and several external units, the input-output instructions are interlocked with the interruption system. Two groups of indicators can be turned on only by input-output instructions: the input-output reject indicators and input-output status indicators. Input-output instructions can also turn on other indicators that are described in this section only as they apply to the input-output instructions.

7.7.1 General Indicators

Exchange Control Check (EK, 3)

The exchange control check is used to indicate that the exchange has failed to function properly in a manner that is not identified with any particular unit. It signals the discovery of errors such as control errors and exchange storage addressing errors. The turning on of the exchange-control-check indicator does not terminate the performance of the current I-O operation in the exchange.

Address Invalid (AD, 16)

The purpose of this indicator is to signal errors in the information contained in an instruction. In input-output instructions, this indicator is turned on if the right effective address of a READ, WRITE or COPY CONTROL WORD is 0 through 31. When an error causing an AD indication is encountered, the operation of the instruction is terminated. The unit is never started and the contents of the channel control word in the exchange are not altered.

Programming Note

The Data Store and Data Fetch indicators are never turned on by an input-output operation. Address monitoring is not effective on input-output operations, and data can be read into and written from the protected storage area defined by the upper and lower boundary registers in conjunction with the boundary control bit. The exchange, however, cannot transfer data to or from locations 0 through 31. Similarly, all control words, including the first one specified by the input-output instruction, can be fetched from any existing storage location, except locations 0 through 31.

7.7.2 Input-Output Reject Indicators

When the computer sends an input-output instruction to the exchange, it usually must wait a few microseconds until the exchange has interrogated the status of the addressed unit to determine that the instruction can be accepted. As soon as the exchange signals the acceptance of the instruction, the computer proceeds to the next instruction. If the exchange cannot accept the instruction, it responds by means of one of the three reject signals, turning on the corresponding indicator in the indicator register. The reject indicators are: 6-EKJ, Exchange Check Reject; 7-UNRJ, Unit Not Ready Reject; 8-CBJ, Channel Busy Reject.

If the program interruption system is enabled, any of the three reject indicators will interrupt the program before the computer has proceeded to the next instruction. If the interruption system is disabled, the reject indicator is set up and the computer proceeds to the next instruction as if the input-output instruction had been accepted. The reject indicators are reset by the next input-output instruction. If more than one condition exists which could reject an input-output instruction, only the reject having the higher priority is given. As an example, if an instruction is given to a unit which is not ready and is busy or is waiting to make an interruption, only the UNRJ indicator is set. When an input-output instruction is rejected, the unit is not started, the control word for the rejected operation is not fetched, and the control word location in the exchange storage is not altered.

Exchange Check Reject (EKJ, 6)

This indicator is turned on when an error is detected by the exchange in the course of testing and setting up the present instruction. It can be caused by:

1. Parity errors in the information contained in the instruction and any malfunction of equipment detected during the initiation of the operation.
2. Specification of channel addresses that are not available to the programmer. An unavailable channel is one that is not installed in the particular system; it is not one installed but inoperative because no unit has been provided for it.

Unit Not Ready Reject (UNRJ, 7)

This indicator is turned on when an instruction is given to a unit that is not ready to be operated (unit-ready status bit in the control word is zero). This happens when the unit accommodated by the channel is in not-ready status or no unit is attached to the channel.

Channel Busy Reject (CBJ, 8)

This indicator is turned on when an instruction is addressed to a channel which is still busy performing a previous instruction or which is waiting to make a program interruption.

7.7.3 Input-Output Status Indicators

When an operation concerning an external unit has been terminated, the status at that time is recorded in the control word. If certain conditions are satisfied, the status bits will turn on the corresponding status indicators to cause program interruption. At the time of such an interruption, the channel address of the external unit is entered into the channel address register. The input-output status indicators are permanent and their mask bits are permanently set to one. The status indicators, in the order of decreasing interruption priority, are: 9-EPGK, exchange program check; 10-UK, unit check; 11-EE, end exception; 12-EOP, end of operation; 13-CS, channel signal.

The status bits in the control word indicating exceptional conditions are turned on as soon as the exchange discovers these conditions. This can happen at any time during the execution of an instruction. The most common type of indications are those associated with reading or writing which are discovered by the external device (data errors, out of material, etc.). These conditions are signalled to the exchange, a block at a time, at the end of the block in which the conditions are discovered, and consequently they can turn on the appropriate status bits only at the end of a block. Other conditions, like channel signal, can cause status bits to be turned on within a block. The corresponding indicators in the indicator register, however, can be turned on and the program can be interrupted only after the termination of the current operation. Because the input-output status is first recorded in the control word and is later set in the indicators only if certain operating conditions are met, the following description of I-O status bits applies to the control word.

Exchange Program Check (EPGK, 9)

This bit is turned on when an operation concerning an external unit is terminated prematurely by a programming error. Among the causes are:

1. An instruction is received which the selected unit is not designed to perform, such as READ given to a printer.
2. The eight low-order bits of the right effective address of a CONTROL contain a code that is not defined for the unit.
3. An instruction is received which the selected unit is unable to perform because of its present condition, e.g., a backspace CONTROL given to a tape unit with the tape at the load point.

4. The exchange attempts to transfer a word to or from a core storage location to which it does not have access (locations 0-31, or locations with addresses above the maximum provided in a particular installation). This can be caused by an invalid data word address, an invalid refill field on chaining, or by specifying a control word address (in an instruction) that exceeds the available core storage. If an instruction specifies a control word address of 0-31, the condition is recognized by CPU and indicator AD is turned on.

Many of the errors causing exchange program check are discovered when the external unit attempts to initiate the corresponding operation. The EPGK status bit, consequently, is turned on a short time after the instruction is accepted. The computer, however, has always proceeded to the following instructions by the time a program interruption is made. The external unit to which the instruction is addressed is never started. In the case of errors which occur later during the execution of an operation, data transmission and control word modification is immediately terminated. This type of error includes addressing of storage locations inaccessible to the exchange. The unit comes to a stop after reaching the end of the current block. The discovery of an error responsible for EPGK always implies that the operation cannot be completed. Hence, EPGK and EOP bits are never on simultaneously. The turning on of the EPGK bit is not associated with any change in the unit-ready status bit.

Unit Check (UK, 10)

This bit is concerned with data errors and malfunctioning of recording medium or equipment which can be identified with a particular unit. Among the causes of unit check are:

1. An uncorrectable data error discovered in any of the following ways.
   a. The input-output unit can discover by means of parity, echo or some other type of checking an uncorrectable error in the data read or written. The error condition is signalled to the exchange at the end of the block in which it is discovered. On reading in the ECC mode, no indication is given for correctable errors.
   b. The exchange can discover an uncorrectable error in the data read or written. This can be a parity error in the byte received from the unit or an error discovered at the error checking and correction station.
   c. The exchange can discover a parity error in the main core storage address when fetching or storing data words, and suppresses reference to that core storage location. On reading, the data word being sent to that address is ignored; on writing, a word consisting of all zeros is written.
   d. Some data may have been lost because exchange capacity is exceeded. This can arise when a unit's byte service or data word transfer has not been performed on time. When a byte or a word of data is missed on reading, the following data are shifted so as to fill the gap. On writing, zeros are written in place of the unavailable data.

2. Malfunctioning of parts of the exchange affecting this channel only, and introduction of errors that prohibit the continuation of the operation. Examples of such errors are parity errors in the main core storage address when fetching a control word, any uncorrectable errors discovered in the control word, after the control word has been fetched, and any parity errors discovered during the control word modification cycle.

3. Malfunctioning of the unit or the control unit, such as card-feed jam, broken magnetic tape, or failure of some components.

4. The operation of certain units has been interrupted by a depression of the stop key. Most units, however, are designed so that a depression of the stop key interrupts the operation without causing unit check. On these units the operation is automatically resumed as soon as the ready condition is restored.

If a unit check is caused by uncorrectable data errors in READ or WRITE, data transfer proceeds normally until the end of the current block, at which time the operation is terminated and both the UK and EOP control word status bits are turned on. Regardless of the setting of the multiple block bit, the operation never proceeds beyond the block containing the error. A data error thus can cause the EOP indication to be given before the last block specified in the instruction has been reached.

When a unit check is caused by any type of malfunctioning in the unit or in the exchange, or when the operator interrupts the operation, data transfer is immediately terminated, and only the UK bit is turned on. If the new control word cannot be fetched because of an error in the core storage address of the control word, the original contents of the exchange control word location are not altered. If the fetching is successful but the control word has an uncorrectable error, the new control word in its incorrect form replaces the old one. Consequently, the exchange control word location contains an incorrect control word when the error is discovered by means of parity checking in the modification cycle. The EOP bit is not turned on regardless of how late in execution of the instruction the error is discovered.

Whenever a READ or WRITE is terminated with a unit check for any of the reasons listed above, the unit has been started and the first block of data has been moved at the unit. This takes place even if the first control word cannot be fetched. In a WRITE, a word consisting of zeros is written if the first data word cannot be obtained.

The state of the unit-ready bit after a UK depends upon the cause of the UK indication. If the indication is due to some malfunctioning in the unit or control unit, or to the depression of the stop key, the unit-ready bit is off. On the other hand, malfunctioning of the exchange and errors introduced in the transmission of data, control codes, or selection addresses do not cause the not-ready status.

End Exception (EE, 11)

This bit is used to signal a number of exceptional conditions usually associated with the recording medium or some subdivision of the data to be transmitted. It indicates that an operation has been terminated because: (1) A unit reached an out-of-material condition such as empty card hopper, full card stacker, or end of tape during writing. (2) A tape mark has been sensed on tape during reading or spacing. (3) The erase key has been pressed on the console.

When an end-exception indication is caused, data transfer is terminated at the end of the current block. At the same time the EE status bit in the control word is turned on. The ready status of the unit may or may not be affected by end exception. If the EE indication is due to a condition which requires operator intervention (e.g., empty card hopper), it causes the unit-ready bit to be turned off. Other causes of the EE indication (e.g., tape mark) do not affect the unit ready status bit.

End of Operation (EOP, 12)

This bit in the absence of the UK bit indicates that an operation initiated for the unit has been completed as specified by the instruction and its control words, if any. EOP in conjunction with UK indicates that an uncorrectable data error has been discovered in the last block. The EOP indication is given for all input-output operations except COPY CONTROL WORD, unless a SEOP (suppress end of operation) instruction is completed without causing a UK or EE indication.

When no uncorrectable data errors are discovered in a READ or WRITE, the EOP bit in the control word is turned on at the end of the block in which data transfer is completed. Data transfer is considered as completed when all the words as specified by the word count in conjunction with the chain and multiple flags have been read or written. When uncorrectable data errors are discovered, both the EOP and UK indications are given at the end of the block in which the error is discovered, regardless of whether or not the operation is completed. As long as the above conditions are satisfied, the turning on of the EOP bit is not affected by any exceptional conditions as indicated by the EE status bit being one.

When any of the five SEOP instructions are given, the SEOP bit in the control word is set to one during the initiation of the operation. As a result of this, the EOP bit in the control word and the indicator register is not turned on upon completion of the operation unless a UK or EE indication accompanies the EOP indication.

Channel Signal (CS, 13)

This bit comes on when the signal key on the unit is depressed,and whenever the unit changes from not-ready to ready status. This change of status can be the result of a manual intervention or can be caused by the completion of rewinding of tape. The main purpose of channel signal is to provide a means of communication from the operator servicing the external units to the computer. The CS indication can be interpreted by means of programming as a request for READ or WRITE. It is also used for initial program loading.

Channel signal is treated in the same way as other input-output status indicators, but is not necessarily related to the initiation, execution, or termination of any external unit operations. It can originate at any time regardless of whether or not the unit is operating. If a CS is sent by a unit while the same unit or another unit connected to the same channel is engaged in data transmission, the CS status bit in the control word is turned on immediately without waiting for the end of the operation or the end of the block. This bit, however, never affects the progress of the current operation, and the corresponding indicator in the indicator register is turned on only after the termination of the operation.

Thus, CS can be the only input-output status indicator turned on in an input-output status report, or it can come on together with any combination of the other status indicators. In the latter case the CS status bit normally is not associated with the operation whose termination is signalled by the other status indicators. When more than one external unit is connected to a common control unit, there is no way of identifying the unit which sends the channel signal. To the program, the CS appears to come from the common control unit regardless of which unit caused it.

7.7.4 Summary of Input-Output Status Indication

When an input-output instruction is terminated, the status bits can appear in various combinations; it is possible to have none of the status bits on (a SEOP type of instruction), any one of the four can be on, or a few of them can be on simultaneously.

If the EOP bit alone is on, the operation has been completed successfully as specified without detecting any errors other than those corrected by the ECC system.

The EPGK, UK, and EE indications provide for exceptions to the normal ending of the operation. Some of the exception indications, however, do not prevent an EOP indication. When a data error occurs, or an exceptional condition is discovered that does not prohibit complete execution of the instruction, the UK and EE indications can be accompanied by EOP. Examples of such cases are a byte parity error (UK and EOP) and spacing over a tape mark (EE and EOP).

EPGK always indicates a type of error that interferes with proper execution of the instruction. Hence, EPGK and EOP are never on together. The same applies to UK if the indication is caused by a condition other than a data error such as a card jam. In the case of EE, EOP is absent if the condition causing the EE indication prevents proper completion of the operation, such as running out of material prematurely.

When more than one exceptional condition is discovered, the exchange turns on all the exceptional indications which apply to the individual conditions. The EOP bit, however, is never actuated when EPGK is on or a UK has been caused by conditions other than data errors. Thus, there is no distinction between data errors and equipment malfunctioning once a programming error has been discovered, as in either case EPGK is accompanied only by UK. Multiple indications can arise in cases such as when the exchange discovers an invalid data word address in the same block in which a data error occurs (EPGK and UK), the unit runs out of material (EPGK and EE), or both conditions occur together (EPGK, UK, EE). Note that when the unit runs out of material and a data error is discovered in the last block, the EOP bit accompanies UK and EE regardless of whether or not the operation is completed.

Figure 7.7-1 lists all the possible causes of termination of input-output operations and the corresponding sets of status bits that signal the termination. The channel signal indication is caused independently of the current operation and can accompany any of the sets of bits.

7.7.5 Input-Output Status Bits not Represented by Indicators

Two of the control word status bits, unit-ready and suppress end of operation, are not represented by indicators in the indicator register, but they affect the turning on of other indicators reflecting the progress of input-output operations.

Unit Ready

The unit-ready bit indicates whether or not the unit currently selected on the channel is in a condition to be operated. The bit is on whenever the unit can accept an instruction from the computer or is executing an operation. When the bit is off, the unit is in the not-ready status and cannot be operated by the computer. A channel can be in the not-ready status because of one of two reasons: (1) The unit accommodated by the channel

CHECK CONDITIONS

Cause of Termination

①	②	③	④		
EPGK	UK Control Error	UK (Data) Error	EE	Operation Complete?	Indication Given
				Yes	EOP, No Indication for SEOP Series
			X	Yes	EE, EOP
			X	No	EE
		X		Yes	UK, EOP
		X		No	UK, EOP
		X	X	Yes	UK, EE, EOP
		X	X	No	UK, EE, EOP
	X			No	UK
	X		X	No	UK, EE
	X	X		No	UK
	X	X	X	No	UK, EE
X				No	EPGK
X			X	No	EPGK, EE
X		X		No	EPGK, UK
X		X	X	No	EPGK, UK, EE
X	X			No	EPGK, UK
X	X		X	No	EPGK, UK, EE
X	X	X		No	EPGK, UK
X	X	X	X	No	EPGK, UK, EE

Examples:

① EPGK, Invalid Address

② UK, Card Jam

③ UK, Data Error, Uncorrectable

④ EE, Out of Cards

FIGURE 7.7-1 CHECK CONDITIONS

cannot operate because of conditions such as out of material, operator stop, power off, control error or mechanical malfunctioning. (2) No unit is attached to the channel.

When a unit is not ready, it remains inoperative until an operator intervenes. An exception occurs during certain operations such as rewinding of tape; the unit may be in the not-ready status while the operation takes place, but the unit-ready bit is turned on again at the completion of the operation.

The unit-ready status bit cannot directly give rise to program interruption, but it can cause other indicators to be turned on. When an instruction is given to a unit which is not ready, the UNRJ indication is sent to the computer. Once the operation is initiated, however, the exchange ignores the status of the unit-ready bit. It depends upon the design of each unit whether or not changing to not-ready status during an operation generates a corresponding indication. On most units the manual stop key will merely remove the ready status at the end of the current block without forcing unit check. Thus, the operator may stop the unit between blocks and hold up a multiple block operation until he presses the start key without cancelling the operation. On the other hand, if the unit becomes not ready because of running out of material it always causes the end exception indication, while not-ready status due to any malfunctioning is always accompanied by UK. If UK or EE accompanies the change to the not-ready status, the current operation is always terminated.

Suppress End of Operation (SEOP)

The suppress end of operation bit is turned on in the control word whenever a SEOP type operation is initiated at the corresponding channel. Its purpose is to suppress the turning on of the EOP status bit in the control word and the corresponding indicator in the indicator register when an SEOP instruction is completed without encountering any exceptional conditions. The SEOP bit is always turned off at the time the operation is terminated. This is the instant when in the case of successful completion the EOP bit in the exchange control word is normally turned on. When a data error is encountered or some other exceptional condition is discovered, the EOP indication is not suppressed; the EOP bit and the bit indicating the exceptional condition (UK or EE) are turned on, and program interruption is permitted.

7.7.6 Settings of I-O Status Indicators

The status of an exchange channel at any time is described by the unit ready, exchange program check (EPGK), channel signal (CS), unit check (UK), end exception (EE), end of operation (EOP), and suppress end of operation (SEOP) status bits in the control word. EPGK, UK, EE, EOP, and CS have corresponding indicators in the indicator register. When an operation concerning an external unit has been terminated and at least one of the five status bits in the control word is on, the exchange attempts to set up the corresponding indicator or indicators and thus cause interruption of the program. All of the status indicators for a particular channel are set up as a group at one time. At the same time, the address of the channel causing the interruption is entered into the channel address register.

The status indicators in the indicator register and the channel address in the channel address register can be set up only if both of the following conditions are satisfied: (1) The five input-output status indicators (EPGK, UK, EE, EOP, and CS) in the indicator register have previously been reset to zero. (2) The interruption system is currently enabled. If either condition is not satisfied, the status bits remain set in the control word, and another attempt to turn on the indicators is made a short time later. While the status bits are in the control word, further instructions for this channel are rejected by means of a channel busy reject. If both conditions are met, the indicators and the channel address are set up, and the corresponding status bits in the control word are reset to zero. As soon as the status bits in the control word have been cleared, a new instruction for the channel can be accepted.

These two conditions permit the program to interrogate the channel address register and make any other tests without interference from other input-output status reports. Condition (1) protects the channel address register and prevents any new input-output status indicators from being set while at least one status indicator due to a previous status report is on. Condition (2) prevents the destruction of the contents of the channel address register after the last indicator belonging to a given set is reset, but before the subroutine whose execution is caused by the indicator has had time to interrogate the channel address register.

Since the input-output status reports cannot be made while the interruption system is disabled, it is not possible to turn the input-output status indicators on without immediately getting an interruption from the highest priority indicator. If, after getting this interruption, successive interruptions due to the same status report are not desired, the program can test and reset the remaining indicators of the set. In any case, regardless of how status indicators are turned off, the enabling of the interruption system after resetting of the last indicator is used as an indication that all the action associated with the status report of a particular channel has been taken, and that a new set of input-output status indicators can be turned on. The system, therefore, should not be enabled until all information associated with the current input-output status report has been acted upon or stored elsewhere for later use.

It follows that only one channel at a time can cause input-output status interruptions. While interruptions due to one channel are being processed, the exchange holds up any other status reports due to the same or other channels. When the computer has cleared the current set of status indicators and the interruption system is enabled, the exchange presents the next set of status indicators as if they had just occurred.

Channel Address Register

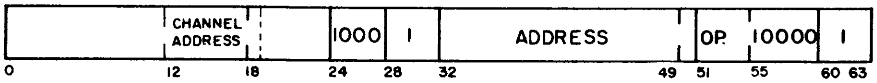
The channel address register identifies the channel responsible for the current input-output status indicators. Whenever input-output status indicators are set up, the channel address of the unit is placed in the channel address register. The register has word address 5 and corresponds to bit positions 12 through 18. These bit positions are the same as those of the channel address in an input-output instruction.

The channel address of a unit is retained in the register until the system is in a condition to accept the next set of input-output indicators, as described in the preceding section. At this time, the channel address is reset to zero. Thereafter, at any time when the interruption system is enabled, the exchange may make a new status report and load a new address in the channel address register. If the register is addressed at an instant when its contents are changed, the instruction check or instruction reject indicator is turned on, depending upon the instruction involved. For this reason the channel address register should be addressed only when handling interruptions. The channel address register can be read out only. The bits are lost whenever information is loaded into it. In this sense the effect of the register is the same as that of location 0.

7.8 INPUT-OUTPUT INSTRUCTIONS

This section describes the various input-output instructions, and lists the input-output reject and status indicators that may be turned on for each. The I-O instructions are contained in the full-word format illustrated below.

Instruction Format:



STRAP Format: OP, A7(I), A18(I)

The effective word address of the first half-word, called the left effective address, of all input-output instructions specifies the channel address. This address is seven bits long, and appears in bit positions 12 through 18. The channel address register in the program interruption mechanism is also seven bits long and corresponds to bit positions 12 through 18 when addressed. The effective word address of the second half-word of the instruction, called the right effective address, contains information peculiar to the particular instruction. The right effective address is 18 bits long, and appears in bit positions 32 through 49. Some instructions use only part of this field.

The addresses of both half-words of the instruction can be indexed. In the first half-word, index arithmetic extends over the full 24 bits of the channel address field of the instruction and the whole value field of the index word, including its sign, but only bits 12 through 18 of the effective address are used. The rest is ignored by the exchange. In the second half-word, index arithmetic extends over bits 32 through 50 of the instruction word and the whole value field of the index word, including its sign, but only bits 32 through 49 are used.

7.8.1 Read (RD)

This instruction initiates a reading operation for the unit specified by the channel address.

STRAP Notation: RD, 16.32, CWD
Operation Code (positions 51-54): 0000

The right effective address specifies the first control word to be used. The control word, in turn, supplies the information defining the data addresses in core storage and the number of words to be read from the external device. An EOP (end of operation) indication is given when the data transfer is completed as specified. The basic reading operation can be modified by means of the chain, multiple, and skip flags.

Indicators Affected: All input-output reject and status indicators, except channel signal.

7.8.2 Write (W)

This instruction initiates a writing operation for the unit specified by the channel address.

STRAP Notation: W, 17.0, CWD
Operation Code: 0001

WRITE uses the control word in the same manner as READ, except that the skip flag is ignored.

Indicators Affected: All input-output reject and status indicators except channel signal.

Programming Notes

The exchange does not obtain the control word for a READ or WRITE from core storage until after the computer has proceeded to the next instruction. Additional control words in a chain are obtained only when the operation has progressed to the point where each control word is needed. The programmer must therefore exercise caution to avoid altering control words in core storage that are being used in current operations. Normally, control words should be set up before an operation starts and remain unchanged during the operation.

Also, some kinds of data errors are not detectable until the end of the block is reached. It is easier and safer not to use portions of a block until the whole block has been read into core storage, or to alter the core storage locations of portions of a block being written until it is certain that the entire block has been written correctly.

The section on data flow in the exchange, stated that the channel address when encoded in STRAP notation must be expressed as one-half the actual channel desired. For example, channel 33 must be encoded as 16.32. In addition to the above method, the assembly program will recognize channel addresses expressed in whole numbers, if no decimal point is used. Thus, channel 32 may be encoded as either (32), or 16.0. If channel addresses are expressed in whole numbers, care must be taken to assure no decimal point appears in the notation.

7.8.3 Locate (LOC)

This instruction is used to select one of several devices accommodated by a single channel address. Once a device is so selected, all operations giving this channel address refer to the same device until a LOCATE is given to select another device on that control unit.

STRAP Notation: LOC, 33, 5
Operation Code: 0011

The right effective address of the instruction contains the selection address for the channel specified by the left effective address. The selection address is three bits long and appears in bit positions 47 through 49 of the instruction. It is sent to and decoded at the addressed control unit. The remainder of the right effective address is ignored. An EOP indication is given when the operation is completed as specified.

In the case of the LOCATE, an exception to the regular instruction rejection occurs. Because it may be necessary to switch from a device that is in the not-ready status to one that is ready, the LOCATE is accepted regardless of the state of the unit-ready status bit in the control word. Thus, a unit-not-ready reject will not be given for a LOCATE when the device originally selected is not ready, or the new device addressed by the right effective address of the instruction is not ready.

Indicators Affected: Exchange check reject (EKJ), channel busy reject (CBJ), exchange program check (EPGK), unit check (UK), and end of operation (EOP).

7.8.4 Control (CTL)

This instruction is used to initiate the performance of certain control functions at the I-O units.

STRAP Notation: CTL, 32, (8) 136: The number (8) denotes to the assembler that 136 is an octal number.

Operation Code: 0010

The right effective address contains control information that is decoded at the device specified by the channel address. If more than one device is accommodated by a channel address, the instruction may apply to the control unit or the device last selected by a LOCATE, depending upon the control function. The code specifying the control function appears in the rightmost byte of the right address, bits 42 through 49. Only this byte is sent to the unit; the rest of the right effective address is ignored. An EOP indication is given when operation is completed as specifed. No distinction is made between instructions that cause some operations to be performed and those that do not. The latter case can arise when the instruction specifies a state or a mode of operation that is already set up in the addressed unit.

An exception to the normal process of the execution of an instruction takes place in the case of rewinding of tape. In order that another tape unit connected to the same control unit may be used while rewinding takes place, the EOP indication is given and the operation at the exchange is regarded as completed immediately after the initiation of the operation. The unit subsequently proceeds with rewinding and is in the not-ready status. When rewinding is completed, a channel signal is given and the unit automatically assumes the ready status. After the EOP indication following the initiation of the instruction is received, another tape unit can be selected by means of a LOCATE.

Indicators Affected: All input-output reject and status indicators.

Functions: The various control functions that may be specified by a CONTROL are described below. The code number is the octal representation of the actual bit configuration used in positions 42-49 of the instruction. The codes are presented as they apply to each I-O unit.

TAPE SYSTEM

Octal Code	Function
016	Remove End-of-Tape Condition. The tape indicator on light is turned off.
036	High Density Mode. The tape unit last selected by a LOCATE is set to read or write at a density of 556 bits per inch. When the power in the tape system is turned on, all units are reset to the high density mode and tape drive 0 is selected.
037	Low Density Mode. The tape unit last selected by a LOCATE is set to read or write at a density of 200 bits per inch.
056	Erase Long Gap. The tape control is set so that the next WRITE (before recording information), places on tape a blank gap of eight and one-half inches.
057	ECC Mode and Odd Parity. The tape control is set to read or write data together with ECC bits and with odd parity.
076	Space Block. The tape is moved forward one block to the next gap, without reading or writing. An end-of-operation indication is given upon completion of the operation. If the block consists of a tape mark, end-exception is given as well as end-of-operation.
077	Space File. The tape is moved forward without reading or writing until a tape mark has been passed. The tape then stops in the gap immediately following the tape mark and an end-exception as well as an end-of-operation indication is given.
117	Write Tape Mark. A standard tape mark (0001111) denoting end of file is written on tape.
136	Rewind. The tape is rewound. If the tape is more than 450 feet from the load point, a high-speed rewind is initiated first, with the tape out of vacuum columns, before low-speed searching for the load point. At the completion of the operation, the load point is located at the read-write head with the tape loaded in the vacuum columns. REWIND differs from other control instructions in that the end-of-operation indication is given as soon as the operation is initiated at the unit. Subsequently, while rewinding takes place, the unit is not ready. When the operation is completed, the unit automatically becomes ready and sends a channel-signal indication.
137	Rewind and Unload. This function is the same as REWIND except that the unit does not automatically become ready upon completion of the operation. When the load point is reached, the unit raises the upper head assembly and removed the tape from the vacuum columns. No channel signal is given upon completion of the operation.
156	No-ECC Mode and Even Parity. The tape control is set to read or write data without ECC bits and with even parity.
157	No-ECC Mode and Odd Parity. The tape control is set to read or write data without ECC bits and with odd parity. When the power in the tape system is turned on, the tape control is reset to the no-ECC mode and odd parity.

TAPE SYSTEM

Octal Code	Function
176	Backspace Block. This is the same as SPACE BLOCK except that the tape is moved one block backward.
177	Backspace File. This is the same as SPACE FILE except that the tape is moved backward until a tape mark has been passed. The head then is positioned on the gap immediately preceding the tape mark, so that the next READ reads this tape mark.

CARD READER

Octal Code	Function
016	Reserved Light Off. The reserved light, which is used as a signal to the operator, is turned off.
017	Reserved Light On. The reserved light is turned on.
057	ECC Mode. The reader is set to read in ECC mode.
116	Check Light On. The check light, which is normally used to indicate a reading error for the benefit of the operator, is turned on. This light may then be turned off by depressing the START key.
157	No-ECC Mode. The reader is set to read in the no-ECC mode.

CARD PUNCH

Octal Code	Function
016	Reserved Light Off. Standard function.
017	Reserved Light On. Standard function.
056	Card Run-Out. The card punch feed goes through one mechanical cycle and feeds one card, without any punching taking place.
057	ECC Mode. Standard function.
116	Check Light On. In addition to turning the write-check light on, this control code causes a card to be offset in the stacker.
157	No-ECC Mode. Standard function.

PRINTER

Octal Code	Function
016	Reserved Light Off. Same as for card reader.
017	Reserved Light On. Same as for card reader.
116	Check Light On.

CONSOLE (CONTROL INSTRUCTIONS)

Octal Code	Function
016	Reserved Light Off. Standard function.
017	Reserved Light On. Standard function.
116	Check Light On. Standard function.
177	Sound Gong. This function causes a clearly audible gong to sound a single stroke.

An alternate method of stating control functions in STRAP notation is provided. In this method, the operation mnemonic states the control function to be performed. Because the control function is contained in the operation mnemonic, no octal code is required when using this method. For example, the STRAP notation resulting in the assembly of a CONTROL which sounds the gong of the console located on channel 37 is: GONG, 18.32. A list of I-O mnemonics is included in the Appendix.

7.8.5 Release (REL)

RELEASE immediately terminates any operation in progress at the external unit specified by the channel address and resets to zero the EPGK, UK, EE, EOP, CS and SEOP status bits in the control word. When the activities at the unit due to the released operation have ceased, the EOP indication is given.

STRAP Notation: REL, 16.32
Operation Code (positions 51-54): 1001

If the addressed channel is reading or writing, data transfer is terminated at the completion of the current word. The unit then proceeds to the end of the block, whereupon all control word status bits except unit ready are reset to zero, and the EOP status bit is turned on. If a CONTROL or LOCATE is in progress at the specified unit, the release operation is executed normally, except that at the completion of the operation all control word status bits indicating special conditions are suppressed and only the EOP bit is turned on. If no operation is being executed by the unit at the time RELEASE is given, all previously set status bits, except unit ready, are reset to zero and the EOP bit is turned on immediately. The address of the second half-word of the instruction is not used and is ignored.

RELEASE is accepted regardless of the status of the unit-ready bit in the control word of the addressed channel. If there is no operation currently in progress at the unit, the status of the unit does not affect the execution of the instruction. The pertinent status bits in the control word are reset to zero, and the EOP indication is immediately turned on. If, however, a LOCATE has been issued to a channel with an inoperative control unit, or an operation has been interrupted because of the not-ready status without cancelling it, the RELEASE cannot free the channel. The completion of the RELEASE in this case depends upon a response from the unit. In order to make the channel available, the unit or control unit must be placed in ready status before the RELEASE is issued. RELEASE is not subject to channel busy reject.

Indicators Affected: Exchange check reject (EKJ) and end of operation (EOP).

Programming Note

If a RELEASE immediately follows a READ, no data transfer takes place, but a block of data is moved at the unit. If RELEASE immediately follows WRITE, a block consisting of one data word is always written before the operation is terminated. If RELEASE is given immediately after a CONTROL or LOCATE, the execution of the subject instruction is not affected except for the suppression of status indications due to exceptional conditions.

7.8.6 Suppress End of Operation

Five SEOP (suppress end of operation) instructions are provided: READ SEOP, WRITE SEOP, CONTROL SEOP, LOCATE SEOP, RELEASE SEOP.

The first four of these five instructions are similar to READ, WRITE, CONTROL, and LOCATE, respectively, except for the use of the end-of-operation indication. At the time and SEOP instruction is given, the SEOP bit in the control word is set to one. As the result of this, the EOP bit in the control word (and therefore in the indicator register) is not turned on unless it is accomplished by unit check or end exception.

STRAP Notation: Add secondary operation (SEOP), enclosed in parentheses, to basic mnemonic. For example, W(SEOP), 33, CW1.

Operation Code: Same as basic operation except that bit 52 contains a one.

The five SEOP instructions provide a means of suppressing program interruption upon the completion of an operation when no exceptional conditions are encountered during the execution of the instruction. When an exceptional condition is discovered, as indicated by the presence of EPGK, UK, or EE status bits, the interruption of the program is not suppressed. In this case the status bit, indicating the exceptional condition, and the EOP bit, are turned on if the operation has been completed, and program interruption is permitted.

The RELEASE SEOP never causes interruptions. It performs all the functions of RELEASE except that it does not cause an EOP indication. Thus, RELEASE SEOP can be used to turn off EPGK, UK, EE, CS and any previously set EOP bits in the control word.

7.8.7 Copy Control Word (CCW)

COPY CONTROL WORD causes the current control word corresponding to the addressed channel to be sent to the core storage location specified by the right effective address. The copy control word is completed before the computer proceeds to the next instruction, and no EOP indication is given.

STRAP Notation: CCW, 35, JOE
Operation Code: 1000

The format of the control word as copied from the exchange is the same as that described previously. The status bits, 18 through 24, reflect the current status of the input-output unit last selected. The data word address and count fields also contain the current values and thus reflect the progress of the operation. The refill address and the three flag bits are never modified and are the same as those originally specified in the control word in core storage.

COPY CONTROL WORD is accepted regardless of the status of the unit and can be executed while an operation is in progress at the addressed channel without disturbing the operation. Consequently, it is not subject to UNRJ or CBJ. The computer program may thus monitor the progress of an input-output operation, if desired for special procedures.

Because COPY CONTROL WORD by definition does not affect the contents of the control word of the addressed channel, it cannot cause any status bits or status indicators to be turned on. As a result, all errors that prevent the completion of the COPY CONTROL WORD are signalled by turning on the EKJ (exchange check reject) indicator. In addition to the regular errors responsible for EKJ, this includes specification of control word addresses that exceed the available core storage and any errors discovered in the process of transmitting the control word to core storage. When COPY CONTROL WORD is terminated by means of EKJ, the control word is not transmitted and the addressed core storage location is not altered.

Caution must be exercised in using the data word address and count information obtained from the exchange if COPY CONTROL WORD is given while another operation is still in progress. During both reading and writing, the data word address of the control word specifies the core storage address to which the next exchange reference will be made. On reading, the address specified by the control word is that of the data word just being processed by the external unit. During writing, the control word always gives an address which is two words ahead of the data word just being written.

Indicators Affected: Exchange Check Reject (EKJ).

7.9 DISK PROGRAMMING

The program set for the disk is similar to that for the other I-O units of the 7030 system, but the differences are important. Some of the material in this section is duplicated in "I-O Programming." The indicators and conditions which turn them on are reviewed in this section, with emphasis on conditions resulting from operation of the disk system.

7.9.1 General Indicators

Address Invalid (AD)

When on, in conjunction with input-output instructions, this indicator signals that the right effective address of READ, WRITE, or COPY CONTROL WORD is 0 through 31. When an error is found which turns on this indicator, the processing of the instruction is terminated in the CPU.

7.9.2 Input-Output Reject Indicators

Exchange Check Reject (EKJ)

This indicator is turned on when an error is detected by the disk synchronizer or disk storage in the course of setting up the present instruction. The error detected may be caused by:

1. Parity errors in the information contained in the instruction and any malfunctioning of equipment which is detected during the initiation of an operation.

2. Specification of disk storage or channel addresses that are not available to the programmer. This includes uninstalled channels or channels that are not operative because no disk storage has been provided for them. When a disk storage is

temporarily physically disconnected from the disk synchronizer, the channel will be in a not-ready status and the disk synchronizer will not set the exchange-check-reject indicator.

Unit Not Ready Reject (UNRJ)

This indicator is turned on when an instruction is given to a disk storage that is not ready to be operated. This is usually caused when a disk storage is temporarily disconnected from its channel.

Channel Busy Reject (CBJ)

Because of the restrictions on addressing and simultaneous operation of disk storage, this indicator can be turned on by a number of conditions that do not exist in the exchange. The conditions which can cause channel-busy-reject in the disk synchronizer are:

1. A READ, WRITE or LOCATE is addressed to a disk which is executing a previous instruction, or is waiting to make a program interruption.

2. A READ or WRITE is addressed to a disk storage at a time when some other disk storage is engaged in a reading or writing operation.

3. A COPY CONTROL WORD is issued at a time when a disk unit is engaged in reading or writing.

7.9.3 Input-Output Status Indicators

When an operation associated with a disk storage has been terminated, the status at that time is recorded. Each channel will retain the status bits associated with termination of the last instruction executed or attempted by the channel or disk synchronizer until the priority sequencing controls permit it to interrupt in the central processing unit. Status information can occur prior to, during, or subsequent to data transmission. Certain checks that occur while data transmission is in progress are held up in the disk synchronizer or disk storage until the end-of-operation is signalled. Other checks can cause immediate termination of the instruction being executed.

Exchange Program Check (EPGK)

This indicator is turned on by a program error. When turned on, it usually implies that the operation cannot be completed. Conditions which set EPGK are:

1. A CONTROL has been issued to a disk storage.

2. The DS attempts to transfer a word to or from a core storage location to which it does not have access (locations 0 through 31, or in locations above the maximum address available in core storage).

3. An attempt is made to locate or to process data to or from a sector-track with address greater than 4095.

Programming errors indicated above are detected after the instruction is sent to the disk system. The EPGK is set soon after the instruction has been accepted by the disk synchronizer. By the time an interruption can be made, however, the computer has proceeded with its instruction sequence. The specified disk storage will not perform the instruction. The EPGK is set, by invalid address detection, when an attempt is made to transfer a word to or from the location specified by the invalid address. The operation proceeds normally until the invalid address is detected.

Unit Check (UK)

This indicator is turned on when an uncorrectable data error or machine malfunction occurs. Two classes of errors that turn on UK are distinguished by the disk system:

1. Those errors which cause immediate termination of the instruction and immediate interruption. These errors are developed by malfunctioning of the disk storage and the disk synchronizer.

2. Those errors which permit the operation to be completed, such as uncorrectable data and addressing parity errors and echo-check error detection during write operations. This class of errors sets the UK and EOP indicator bits.

End of Operation (EOP)

When this indicator is turned on and no other check interruption occurs, it indicates that the instruction has been completed successfully. It is set for all instructions except COPY CONTROL WORD and those instructions specifying SEOP. In this latter case, however, the suppression is ignored if a unit check is also turned on and the operation is completed.

7.9.4 Keys and Lights

The disk units do not have keys or lights other than those required for maintenance control. Power for the disk units is turned on and off with power for the disk synchronizer. The disk units do not require operator access, and can be installed outside of the operating area.

7.10 DISK OPERATIONS

The instructions pertaining to the disk system are expressed in the same instruction format as used for the exchange. The only distinction between an instruction intended for the disk system and one intended for the exchange is the channel address. The disk system uses channel address 0-31. The input-output instructions used for the operation of the disk system are described here, with emphasis on characteristics that are different from the same instructions used with the exchange. They are summarized below:

	Operation	Action	Control Word
RD	Read (SEOP)	Data from disk storage to core storage	Yes
W	Write (SEOP)	Data from core storage to disk storage	Yes
LOC	Locate (SEOP)	Set access mechanism to initial sector-track for later data transmission	No

	Operation	Action	Control Word
REL	Release (SEOP)	Terminate data transmission immediately	No
CCW	Copy Control Word	Data word address field and word count field of control word to core storage	No

When the SEOP modification is used it means "suppress end of operation indication." The normal EOP indication given at the successful conclusion of an operation is suppressed. CONTROL and CONTROL (SEOP) have no meaning for disk storage and set the EPGK indicator.

7.10.1 Read (RD); Read SEOP (RDSEOP)

These instructions initiate a reading operation from the disk specified by the channel address, bit positions 12 through 18, in the left effective address of these instructions. The right effective address of the instruction, bit positions 32 through 49, specifies the core storage location of the control word associated with these operations. Data are transferred to successive locations in core storage beginning with the location specified in the data word address field of the control word. The number of words of data read and stored is given by the value in the count field of the control word.

Used with disk storage, these instructions differ from the operations as used with the exchange in that the control word flag bits and refill fields are ignored. Data will be transmitted from successive sector-tracks of the disk until the count field is reduced to zero. However, the data read need not be a multiple of the number of words contained in a sector-track. If the count reaches zero within a sector-track, the operation is terminated immediately. If at this time another READ is issued to the same channel, the reading will begin at the sector following the last one used by the previous READ. READ SEOP differs from READ in that the EOP indication is suppressed when the operation is completed normally. Otherwise, the operations are the same.

Indicators: Exchange Check Reject (EKJ); Channel Busy Reject (CBJ); Unit not Ready Reject (UNRJ); Exchange Program Check (EPGK); Unit Check (UK); End of Operation (EOP); Address Invalid (AD).

7.10.2 Write (W); Write SEOP (WSEOP)

These instructions initiate a writing operation to the disk specified by the channel address, bit positions 12 through 18, in the left effective address of these instructions. The right effective address of the instruction, bit positions 32 through 49, specifies the core storage location of the control word associated with these operations. Data are transferred from successive locations in core storage beginning with the location specified in the data word address field of the control word. The number of words written is given by the value in the count field of the control word.

Used with a disk storage, these instructions differ from the operations as used with the exchange in that the control word flag bits and refill field are ignored. Data will be written into successive sector-tracks of the disk until the count field is reduced to zero. The data written need not be a multiple of the number of words contained in a sector-track. In this case, however, the remainder of the last sector-track is automatically filled with zeros. WRITE SEOP differs from WRITE in that the EOP indication is suppressed when the operation is completed normally; otherwise, they are the same.

Indicators: Exchange Check Reject (EKJ); Channel Busy Reject (CBJ); Unit not Ready Reject (UNRJ); Exchange Program Check (EPGK); Unit Check (UK); End of Operation (EOP); Address Invalid (AD).

7.10.3 Locate (LOC); Locate SEOP (LOCSEOP)

These operations are used to initiate setting of the access mechanism to the track which contains the specified address. The disk is specified by the channel address, bit positions 12 through 18, in the left effective address of these instructions. The right effective address, bit positions 32 through 49, of the instruction specifies the sector-track address. Only bits 38-49 of this field are used, to address sector-tracks from zero to the maximum of 4095. See Figure 7.4-2 for the format of the sector-track address.

Specification of the sector-track automatically identifies the access mechanism to be used on that sector-track. These instructions cause the proper access mechanism of the pair attached to each disk storage to be positioned at the track containing the specified address. Concurrently, the other access mechanism is positioned on the track immediately following the specified track. When both access mechanisms have been positioned, an EOP indication is given. The starting sector-track address within the track is retained within the disk unit.

A LOCATE or LOCATE SEOP may be given to any disk attached to the data synchronizer at any time if the disk specified is not busy with data transmission or a prior LOCATE. These operations specify an initial sector-track for which data transmission may occur. Because of these instructions, the disk units are randomly addressable by sector-track. Whenever successive sector-tracks are to be read or written, only the initial one need be specified. LOCATE SEOP differs from LOCATE in that the EOP indication is suppressed when the operation is completed normally; otherwise, they are the same.

Indicators: Exchange Check Reject (EKJ); Channel Busy Reject (CBJ); Unit not Ready Reject (UNRJ); Exchange Program Check (EPGK); Unit Check (UK); End of Operation (EOP).

7.10.4 Release (REL); Release SEOP (RELSEOP)

These instructions immediately terminate any data transmission operation currently in process, at the disk unit specified by the channel address, bit positions 12 through 18 of the left effective address. When a disk unit is released, the sector-track address register is left at the address of the last sector-track used unless transmission of that sector-track has been completed. If the disk unit was writing, the remainder of the sector-track, in which the RELEASE operation was accepted, will not be erased. A LOCATE will proceed to its normal end and the sector-track address will be set as specified.

If the disk unit is not ready, the disk system will not accept the instruction. Instead, the unit-not-ready-reject indicator is turned on. In this respect these operations are different from the exchange. RELEASE SEOP differs from RELEASE in that the EOP indication is suppressed when the operation is completed normally. This instruction resets all status indications during its execution.

Indicators: Unit not Ready (UNRJ); End of Operation (EOP).

7.10.5 Copy Control Word (CCW)

The left effective address, bit position 12 through 18, of the instruction specifies the channel address. The right effective address, bit positions 32 through 49, of the instruction specifies the core storage location to which the information is to be sent.

The disk synchronizer does not have control word storage as does the exchange. The data word address and word count fields are retained in registers in the disk synchronizer and are used to control core storage references. Because only one disk unit may transmit at a time, these registers are associated with the last channel which was performing a data transmission operation. The contents of these registers are placed in a full-word format, corresponding to their location in a control word. Because execution of COPY CONTROL WORD uses the transmission channel of the disk synchronizer, the instruction is not executed when a disk storage is reading or writing, and the channel-busy indicator is turned on. The normal completion of this instruction does not give an EOP indication, since when this instruction is used with the disk system it must be completed before the computer proceeds with the next instruction.

Indicators: Exchange Check Reject (EKJ); Channel Busy Reject (CBJ); Unit not Ready Reject (UNRJ); Address Invalid (AD).

7.11 INITIAL PROGRAM LOAD

To load a program into the machine without a programmed READ, an initial program loading technique is provided. The initial program can be loaded from any unit accommodated by the exchange and is based on a special interpretation of channel signal. The loading is initiated by means of the INITIAL PROGRAM LOAD (IPL) key, which sets up the exchange and the computer in a special mode. Once the IPL key is depressed, the first channel signal interruption causes the channel responsible for the interruption to read a specified number of words into a specified core storage area. The IPL key is physically located on the operator's console, although not an integral logical part of it. If the system does not contain an operator's console, it is provided in a separate box, which may be placed at a convenient operation point.

The initial program, as recorded on the medium from which it is to be loaded, must start with a control word that specifies the number of words to be read and the core storage address at which the first word of the remainder of the program is to be stored. This control word is immediately followed by the program itself. When the program has been read, the computer automatically starts execution of the new program. The depression of the IPL key has the following effects:

1. The execution of any program by the computer is terminated.
2. The interruption system is temporarily disabled.
3. All input-output operations of the exchange are terminated.
4. All control words in the exchange are reset to zero.
5. The exchange is set to interpret the next channel signal in a special way.
6. All input-output units are reset to the initial power-on status.

The depression of the IPL key performs these functions even if the computer has hung up within an operation. The disabling of the interruption system prevents any extraneous interruptions, such as the elapsed time clock, from interfering with the initial loading until the program has set up suitable procedures for coping with such interruptions. The initial program should reset the indicator register before enabling the interruption system. The resetting of input-output units to the initial power-on status has the following effects:

1. Where more than one input-output unit is connected to a common control unit, the control unit is reset to select the unit with the selection address 0.
2. Where applicable, the unit is reset to the no-ECC and odd-parity mode of operation.

As a result, a program can be loaded by the initial loading technique only from a unit having the selection address 0. Because the exchange does not differentiate between the channel signals sent by different units connected to the same control unit, the initial program loading is started by a channel upon the depression of the SIGNAL or START key on any unit accommodated by that channel address.

The first channel signal indication received after the depression of the IPL key from a unit accommodated by the exchange initiates the following sequence of events:

1. The exchange automatically stores in the control word location, for the channel signal, a fixed control word as follows:

Data word address = 4, Chain flag = 1, Multiple flag = 0, Skip flag = 0, Count = 1, Refill address = 4. Data word address 4 refers to a special core storage location which is used only during initial program loading. It is normally inaccessible to the exchange and to the computer. This exchange reference to location 4 in no way affects the contents of the register with address 4, which can be addressed by the computer.

2. The exchange initiates a READ for the indicated channel, using the fixed control word.

3. The first word read from the unit is stored in special core storage location 4. Since the fixed control word is then exhausted, its refill address causes the next control word to be fetched from location 4. Location 4 at this time should contain another control word specifying the address at which the word immediately following should be stored, and the number of words to be read from the unit. Reading then proceeds normally.

4. If and when the READ is completed as specified by the control word or words without causing the exchange-program-check or unit check indications, the exchange sends an initial start signal to computer. The encountering of an end-exception condition during the reading does not affect a successful completion of initial loading as long as all the conditions are satisfied that would normally cause EOP to be given. The EOP as well as the EE indications, however, are not turned on upon the completion of the operation. The sending of the initial-start signal to the computer completes the initial program loading cycle in the exchange.

5. The computer, upon receipt of the initial-start signal, fetches and executes the instruction in the location specified by the data word address field of the control word in core storage location 4. The address of this instruction, however, is not placed in the instruction counter. Unless specified otherwise by this instruction, the interruption system after the completion of this instruction returns to the state it was in at the time

the IPL key was depressed. This completes the initial program loading sequence. The first instruction of a program loaded by means of the initial program loading technique must be BRANCH DISABLED. This instruction replaces the contents of the instruction counter and insures that the system does not return to the enabled state.

If the READ as described in 3 and 4 is not successful, the input-output unit comes to a stop. In order to restart initial program loading, the I-O unit must be returned to its original state and the IPL key must be pressed again. The next channel signal then repeats the listed sequence of events. The first channel signal received at the exchange following the depression of the IPL key starts initial program loading, but it does not turn on the CS status bit in the control word. All subsequent channel signals are interpreted in the normal way. If a channel signal is received by the exchange during the initial program loading, the CS status bit is turned on and the computer is interrupted as soon as the system is enabled.

The termination of input-output operations and the resetting of control words are equivalent to issuing RELEASE's to all exchange channels, except that the whole control word instead of only its status bits is reset. When the IPL key is depressed, all data transfer between external units and core storage is immediately stopped, but each unit still continues operating until the end of the block is reached. In tape rewinding, a channel signal is given at the completion of the operation. If this is the first channel signal to be received following the depression of the IPL key, initial program loading is initiated from the c orresponding channel.

7.12 PROGRAMMING EXAMPLES

7.12.1 Example 1

Five thousand words of data from tape unit 3, located on channel 32, are to be read and placed in storage starting at location 1,000. The tape has been previously written in the no-ECC mode. If, during reading, an error occurs, the tape is to be rewound and reading restarted. If again an error occurs, the attempt to read is to be repeated. If after five attempts the reading has not been successful, the desired information is to be read in from the card reader. If the attempt to read the card reader is also unsuccessful, the reader check light is to be turned on and the gong on the operator console is to be sounded. If no successful read operation is performed, indicator PG1 is to be set to one and a BRANCH to the main program is to be performed. If any of the reading operations are successful, the 5,000 words are to be used as data in future operations. In such a case PG1 is to be set to zero before a BRANCH to the main program is performed.

Assume that the card reader is located on channel 34 and the console is located on channel 37. The tape unit is initially at load point and is in ready status. Also assume that the input contains no intentional end exception conditions. In this example, any I-O indicator turned on, with the exception of CS or EOP, is to be considered as a reading error. The desired action is shown in the flow chart, Figure 7.12-1. The program steps are:

```
Name      Instruction           Remarks
BEGIN     LCI, $X4, 5           Place count of 5 in XR4.
          BD, PREP              Disable interrupt system.
```

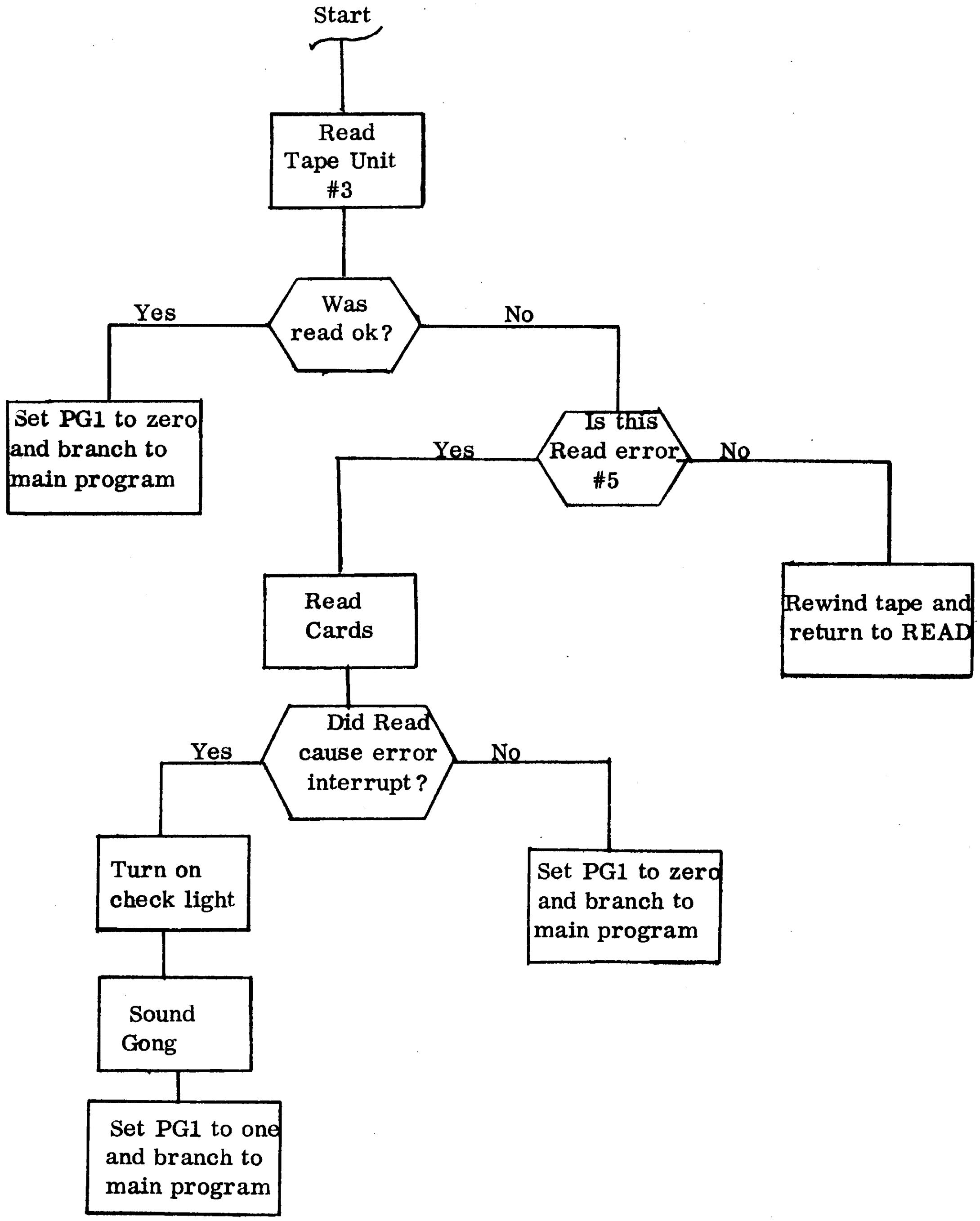

Start
Read Tape Unit #3
Was read ok?
Yes
No
Set PG1 to zero and branch to main program
Is this Read error #5
Yes
No
Read Cards
Rewind tape and return to READ
Did Read cause error interrupt?
Yes
No
Turn on check light
Set PG1 to zero and branch to main program
Sound Gong
Set PG1 to one and branch to main program

FIGURE 7.12-1. PROGRAMMING FLOW CHART

```
Name       Instruction                Remarks
PREP       Z, 11.0                    Set positions 20-63 of indicator register to zero.
           Z, 12.0                    Set positions 20-63 of mask register to zero.
           LVI, $X5, BASE             Prepare to set up interrupt base address.
           SV, $X5, 2.0               Set base address in interrupt address register.
           BE, START                  Enable interrupt system
START      LOC, 32, 3                 Locate tape unit 3.
WAIT 1     BEW, MODE                  Avoid channel busy.
MODE       CTL, 32, (8) 157           Place unit in odd parity, no-ECC mode.
WAIT 2     BEW, TAPE                  Avoid channel busy reject.
TAPE       RD, 32, CWD1               Read tape unit 3 with control word from loca-
                                        tion CWD1.
WAIT 3     BEW, OK 1                  Wait for no-op interrupt.
OK 1       B, MAIN                    Leave PG1 set to zero, and branch to main
                                        program.
CARDS      TI, 1, Too Bad, NoGo       Use TRANSMIT IMMEDIATE to alter interrupt
                                        action (full words).
           RD, 34, CWD1               Read 5,000 words from cards.
WAIT 4     BEW, OK1                   Wait for a no-op interrupt.
NO GOOD    CTL, 34, (8) 116           Turn on card reader check light.
           CTL, 37, (8) 177           Sound gong.
           BZB1, 11.42 Main           Set PG1 to one and branch to main program.
           CNOP                       This is a note to the assembly program.  It
                                        guarantees that the next instruction begins
                                        at a full-word address.  To accomplish this,
                                        the assembler may insert a NOP here.
NO GO      CBZ, $X4, CARDS            After five attempts to read tape, branch to
                                        cards.
           NOP, 0.0                   Use a full word.
           CTL (SEOP),32,(8),136      Rewind tape.
           BEW, START + 2.0           Wait for CS to indicate completion of rewind,
                                        then read the tape.
           CNOP                       "Conditional NOP."
TOO BAD    BD, NO GOOD                For interrupt during card reading.
           NOP, 0.0                   Use a full word.
CWD1       CW (CD), 1000, 5000,       Control word data address = 1000;
             CWD1                     Count = 5000.
```

NOTE: The symbol (CD) is an indication to the assembly program that the control word is to be used to "count disregarding records or blocks." This results in a control word with a multiple flag of one.

```
BASE       BD, FIX1                   Branch to fix-up routine in main program.
BASE +1    BD, FIX2                   Branch to fix-up routine.
BASE +2    BD, FIX3                   Branch to fix-up routine.
BASE+3     B, NO GO                   Exchange control check; consider this a read
                                        error.
BASE +4    BD, WAIT                   Time signal; return to wait.
BASE +5    BD, FIX6                   Branch to fix-up routine.
BASE +6    B, NO GO                   Read error (EKJ).
BASE +7    B, NO GO                   Read error (UNRJ).
BASE +8    B, NO GO                   Read error (CBJ).
```

```
Name         Instruction          Remarks
BASE +9      B, NO GO             Read error (EPGK).
BASE +10     B, NO GO             Read error (UK).
BASE +11     B, NO GO             Read error (EE).
BASE +12     NOP, 0.0             Read OK.
BASE +13     NOP, 0.0             Rewind complete.
BASE +14 }
BASE +19 }   BRANCH DISABLED to proper fix-up routine in main program.
```

The purposes for most of the instructions are explained in the remarks column. The BEW instructions at locations WAIT 1, and WAIT 2 are intended to prevent the following instruction in each case from causing a channel busy reject. For example, the READ at location TAPE cannot be attempted until an EOP is received as a result of the CTL at location MODE.

TRANSMIT at location cards is used to alter the action taken as a result of an interrupt. Because a TRANSMIT involves full-words, the half-word instructions that are being transmitted must start at a full-word address and must be followed by a NOP instruction. This action guarantees that the only significant instruction transmitted is the desired half-word BD. The symbol CNOP is a pseudo-instruction to the assembly program; it guarantees that the next instruction begins at a full-word address. To accomplish this, the assembly program may insert a NOP directly ahead of the instruction that must be at a full-word address.

The purpose of the BEW instruction at locations WAIT 3 and WAIT 4 is to allow the indicators to be set before proceeding with the program. If the indicator set is an EOP only, the effective interrupt address contains a NOP and the program sequence is determined by the BEW. On the other hand, if the indicator set is one of those to be interpreted as a reading error, the effective interrupt contains a BRANCH to NO GO.

The I-O instructions beginning at location NO GOOD are not followed by BEW instructions. Note, however, that the two CTL instructions involve different exchange channels. Thus, the possibility of a channel busy reject does not exist. The BEW following the rewind at NO GO +2.0 is intended to suppress reading the tape until a CS indicates the rewind operation is complete. Because this BEW is to take action only after a CS has been received, the rewind CTL is stated as a SEOP to prevent the EOP indication from activating the BEW.

7.12.2 Example 2

Write a short routine to accomplish the action described below.

1. Read 5,000 words of information from the card reader.
2. Write this information on tape unit 2 of channel 32.
3. Read the information back from the tape unit and compare with the original information.
4. Repeat steps 2 and 3, using tape unit 2 of channel 33.
5. Punch the original information on the card punch.
6. Have cards loaded into card reader and the reader made ready.
7. Read in contents of the new cards and compare with the original information.

8. If any comparison renders other than an equal result, display the channel address of the most recently used I-O device. The display is to be made in decimal on the numeric display section (word 2) of the console. After displaying the channel address, BRANCH to the main program. If all comparisons render display equal results, BRANCH to the main program without performing any display.

The flow chart, Figure 7.12-2, shows the action, obtained through the following program steps. In this example assume channels 32 and 33 accommodate tape control units which have eight tape units each attached to them. The card reader is located on channel 34, the punch on channel 35, and the console on channel 37.

```
Name        Instruction                   Remarks
PREP        LX, $X1, 0.0                  Clear XR1.
            LCI, $X1, 2.0                 Place count of 2 in XR1.
            LX, $X4, 0.0                  Clear XR4.
            LCI, $X4, 3.0                 Place count of 3 in XR4.
            LX, $X5, 0.0                  Clear XR5.
            LCI, $X5, 2.0                 Place count of 2 in XR5.
START       RD, 34, CWD1                  Read in master data.
            BEW, WRITE                    Wait for no-op interrupt to indicate
                                            EOP.
WRITE       LOC, 32($X1),2.0              Locate tape unit 2 of appropriate
                                            channel.
            W, 32($X1), CWD1              Write data from MASTER on
                                            appropriate channel.
            BEW, REWIND                   Wait for no-op interrupt to indicate
                                            EOP.
REWIND      CTL, 32($X1),(8) 136          Rewind tape.
            BEW, READ                     Wait for EOP.
READ        R, 32 ($X1), CWD2             Read information backstore beginning
                                            at NEW.
            BEW, TEST                     Wait for EOP.
TEST        LX, $X2, CWD1                 Prepare to compare.
            LX, $X3, CWD2                 Prepare to compare.
            L (V+I), (BU,64,8),1.0        Load MASTER word.
            ($X2), 0
            K(V+I)(BU,64,8),1.0           Compare NEW word.
            ($X3),0
            BZAE, DISPLAY                 If not equal, branch.
            CB, $X2, TEST +1.0            Compare 5,000 words.
            CB+, $X4, $+.32               Add one to value of XR4 for
                                            indicating channel in use.
            CB+, $X1, WRITE               If only one tape channel has been used,
                                            increment value of XR1 and use
                                            second tape channel.
            CB, $X5, PUNCH                Use the punch one time.
            B, MAIN                       All desired units have been used and
                                            all comparisons are equal; branch
                                            to main program.
PUNCH       W, 35, CWD1                   Punch data from MASTER.
            BEW, SOUND                    Wait for EOP.
```

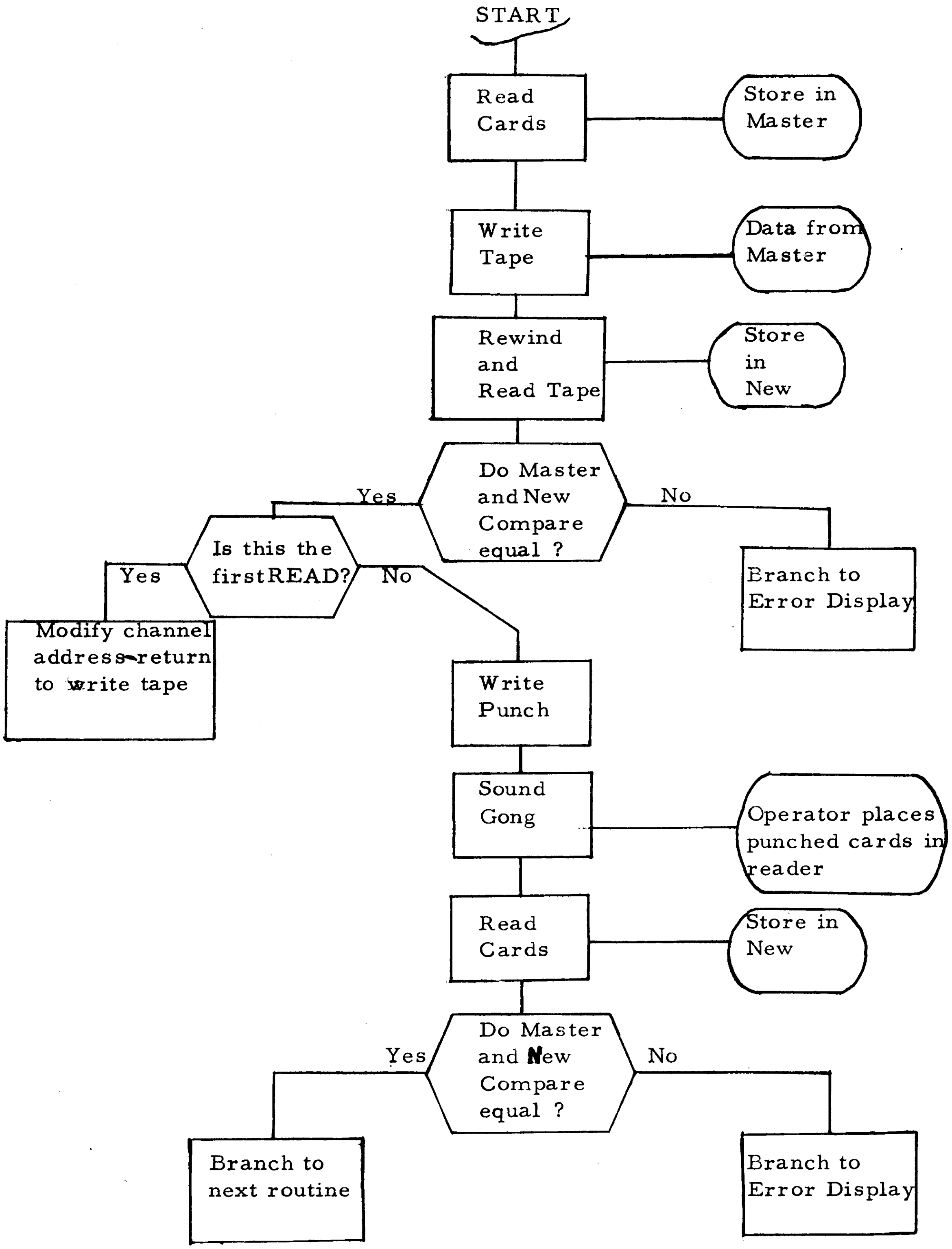


FIGURE 7.12-2. PROGRAMMING FLOW CHART

```
Name        Instruction                       Remarks
SOUND       CTLSEOP, 37,(8)177                Sound gong; operator loads the card
                                                reader
            BEW, CARDS                        Wait for CS to indicate that the reader
                                                is ready.
CARDS       RD, 34, CWD2                      Read data in; store in NEW.
            BEW, TEST                         Wait for EOP.
DISPLAY     B, FIND ($X4)                     XR4 indicates the most recently used
                                                channel.
FIND        LVI, $X6, 32.0                    Channel 32 most recently used.
            B, CONVERT
            LVI, $X6, 33.0                    Channel 33 most recently used.
            B, CONVERT
            LVI, $X6, 35.0                    Channel 35 most recently used.
            Z, OUTWORD +1.0                   Clear second output word.
            Z, TEMP
            SV, $X6, TEMP                     Place channel address in TEMP.
            LCV(BU,8,4),TEMP .10,0            Place converted channel address in B
                                                register.  (Result is decimal-byte
                                                size 4.)
            ST(DU,16,8),OUTWORD               Store channel address in second output
            +1.0,0                              word (byte size 8).
            W, 37, CWD3
            BEW, MAIN                         Channel is displayed;  after EOP,
                                                branch to main program.
            CNOP                              Insure that OUTWORD begins at
                                                full-word address.
OUTWORD     NOP, 0.0                          First output word is meaningless.
            NOP, 0.0
OUTWORD +1.0                                  Stored channel address.
CWD1        CW(CD),MASTER,5000,               Contains multiple flag.
            CWD1
CWD2        CW(CD),NEW,5000,CWD2              Contains multiple flag.
CWD3        CW,OUTWORD, 2,CWD3                Display two words only.
```

8 STRAP

SYMBOLIC programming of the IBM 7030 is accomplished through use of the symbolic programming systems STRAP I and STRAP II. The rules which apply to the symbolic writing of programs are in general the same for both STRAP I and STRAP II. The STRAP I assembly program (compiler) uses an IBM 709 or a 704 with 32,768 words of storage for assembling IBM 7030 programs. STRAP II is a more elaborate assembly program utilizing the 7030 for the assembly process.

The steps involved in writing and compiling a STRAP symbolic program are similar to those involved in other symbolic systems. The program is written in a standard format, using definite rules and mnemonics. This information, the subject program, is punched in cards (and usually transferred to magnetic tape) and loaded into the computer. The information is then operated on as data under the control of the assembly program which converts the symbolic information to the required 7030 binary format. The compiled program and its data are then delivered as output in the form of magnetic tape or punched cards. At this same time, a listing of the program along with certain remarks from the compiler are also delivered as output. The information to be included in the listing is usually recorded on magnetic tape and converted to a printed listing on an off-line printer.

Because a compiler is merely a program used to convert symbolic statements to "machine language" statements, only the mnemonics and formats which the compiler can recognize may be used in writing a program. This section of the manual deals with the particular specifications that must be adhered to when writing symbolic programs to be assembled by the STRAP compiler. The specifications defined here are applicable to both STRAP I and STRAP II.

The STRAP specifications are divided into three categories: Category 1, in which the form that is used in writing the program is defined. Category 2 covers the expression of symbolic machine instructions and defines the various symbols, mnemonics, and symbolic formats. Category 3 pertains to the expression of the compiler's pseudo-instructions, which are not compiled as machine instructions but are used to direct the compiler itself.

8.1 STRAP CODING FORM AND LISTING

The STRAP coding form serves as the source document for the card punch operator. The program to be assembled is initially written on the coding form. The operator then transfers the symbolic information to punched cards. The coding form is directly related to the instruction card form. Both have 80 columns and are divided into four fields, as shown below.

Class 1	2 Name 9	10 Statement 72	73 Identification 80

The purpose of each field is:

1. Class (1 column): This field is unused in the normal STRAP format.
2. Name (6 columns): This field is used in the normal manner to identify the statement. STRAP II provides an 8-position name field.
3. Statement (63 columns): This field is used to express machine instructions or pseudo-instructions.
4. Identification (8 columns): This field is provided as identification for the cards.

Card identification (columns 73-80) is reproduced on the listing but does not contribute any information to the assembly program for translating instructions. The listing provided by the STRAP assembler is contained in the five-field format shown below.

Name	Statement	Remarks	Hexadecimal	Location

Each line of the listing contains one expression. The leftmost field in any line is reserved for the name of the expression. Every expression need not be named, but those names that do exist appear in the name field of the listing. The next field contains the symbolic statement itself. The third field contains any remarks which the programmer wishes to appear on the listing. The fourth field is an octal presentation of the assembled statement. The last two positions of this field are not octal; rather, they represent a special coding and are called hexadecimal characters. The coding of these characters is explained later. The rightmost field of the listing contains the absolute address to which this statement has been assigned. That portion of the compiler that determines the absolute location to which any entry is assigned is called the location counter. Thus, this field of the listing is sometimes referred to as the location counter contents. Separated from the right end of the location field by a single space is a one-bit field which may contain error marks inserted by the compiler. These marks are indications to the programmer that a possible error was encountered in the current expression. A single line of print from a STRAP listing is illustrated below.

Name	Statement	Remarks	Octal-Hexadecimal	Location
EXIT	B, 0. 0 ($X3)	LEAVE	000000. 10 03	005064.00 ?

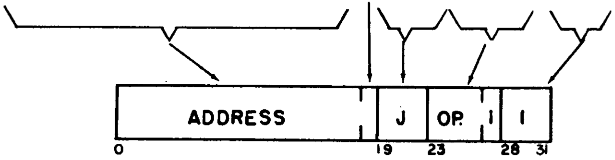
The question mark is not intended here as a valid symbol but merely to designate that portion of the location field in which these marks do appear. The Octal-Hexadecimal field contains numbers and/or characters that represent the assembled binary expression. All of the positions of the field, with the exception of the last two, contain octal numbers and thus represent three binary bits of the assembled instruction. The first octal digit, for example, represents bits 0, 1, and 2 of the instruction. The decimal point between the sixth and seventh octal positions is not significant and is inserted merely to break up an otherwise long list of digits. The last two characters in this field represent four binary bits each, rather than three. The rest of this field contains eight octal digits which represent 24 binary bits. The remaining eight bits of the half-word instruction must be described by the last two positions of the octal-hexadecimal field. Because the value contained in four binary digits may exceed the value which can be represented by a single decimal number, these positions may contain alphabetic characters. The binary values 0-9 are represented by the appropriate decimal numbers.

The binary values 10-15 are represented by alphabetic characters A through F, respectively. The hexadecimal characters used and the binary values they represent are illustrated below:

Hexadecimal	Binary
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

The last hexadecimal character represents instruction bits 28-31. When a full-word instruction appears on the listing the octal-hexadecimal field is followed by an identical field which describes instruction bits 32-63. Note that the statement field of the listing contains the expression as written by the programmer, while the octal-hexadecimal field represents the instruction as it is compiled. Thus, if because of incomplete statements or programming errors, some assumption is required on the part of the compiler, the statement field and the octal-hexadecimal field may not be identical. The conversion of the octal-hexadecimal field to binary is illustrated by the following example:

Statement	SX, $X3, 9.0 ($X15)
Octal-hexadecimal	0 0 0 0 1 1. 0 7 1 F
Compiled Instruction	000 000 000 000 001 001 000 111 0001 1111



8.2 EXPRESSION OF MACHINE INSTRUCTIONS

Machine instructions are written symbolically on the coding form previously described. They are usually entered one per line, according to a prescribed format that varies with the type of instruction, and are written with fixed mnemonic operation codes. A large number of these mnemonics have been used in previous sections of this manual. Symbolic instructions are divided into fields (operation, address, offset, etc.) by commas. The order and manner in which these fields are written is dictated by the various symbolic instruction formats. The punctuation in the various formats is extremely important.

A semicolon, punched as an 11-0 double punch, is used to designate the end of a statement so that multiple statements may be written per line. The number of instructions that may be written on one line is limited only by the number of columns available in the statement field of the card. A symbol in the name field of a card having more than one instruction in the statement field is associated with the first instruction only. The remaining instructions are treated as if they appeared on separate cards having blank name fields. A single instruction cannot be continued from one card to another.

A comment may follow any instruction. A comment is initiated by the apostrophe (an 8-4 double punch) and is terminated either by the end of the card or by a semicolon. Thus, the semicolon may never be used in a comment. An apostrophe in the name field causes the whole card to be treated as a comment; it will be printed on the listing but will not otherwise affect the assembly.

8.2.1 Symbolic Instruction Formats

Symbolic instructions are entered in the statement field. Within this field operation codes and address expressions are separated by commas and form subfields. A modifier to either an operation or an address is enclosed in parentheses and attached to the modified subfield. Blanks have no meanings in any field except to indicate the spacing desired on the printed listing. Blank cards are ignored. The twelve symbolic instruction formats for STRAP I are:

Format Type	Operation
1. OP (dds), A_{18} (I)	Floating point.
2. OP, A_{19} (I)	Miscellaneous, unconditional branch, SIC.
3. OP, J, A_{19} (I) or OP, J, A_{18} (I)	Direct index arithmetic.
4. OP, J, A_{19} or OP, J, A_{18}	Immediate index arithmetic.
5. OP, J, B_{19} (K)	Count and branch.
6. OP, B_{19} (K)	Indicator branch.
7. OP (dds), A_{24} (I), OF_7 (I')	VFL arithmetic, connect, convert.
8. OP_1 (OP_2) (dds), A_{24} (I), OF_7 (I')	Progressive indexing.
9. OP, J, A_{18} (I), A'_{18} (I')	Swap, Transmit (full words).
10. OP, A_{24} (I), B_{19} (K)	Branch on bit.
11. OP_1 (OP_2), A_7 (I), CW_{18} (I')	Input-output.
12. LVS, J, A, A', A'', A''', ...	Load value with sum.

Symbol Definitions	
1. OP and OP_1	Primary instruction operation.
2. OP_2	A secondary operation permitted only in progressive indexing and input-output.
3. A_n	An n-bit data address.
4. B_{19}	A 19-bit branch address.
5. I	A 4-bit index address where (0) signifies no indexing and (1) through (15) signifies indexing by the corresponding index register.
6. K	A 1-bit index address where no modification (0) or modification by index register 1 (1) are the only possibilities.
7. OF_7	A 7-bit offset field.
8. dds	Data description.

9. J	A 4-bit index address that refers to an index register as an operand. Here (0) refers to index register 0.
10. A_7	A 7-bit input-output channel address.
11. CW_{18}	An 18-bit control word address.
12. LVS	Refers to one specific operation: Load value with sum.
13. primes	Used to distinguish otherwise identical fields in a format.

There is a general right-to-left drop-out for all fields separated by commas. For example, a VFL instruction (format above) for which the offset and its index modifier are zero is written as follows:

OP, A (I)

The comma is the major separator for the symbolic instruction types. If there are less than the maximum number of major symbolic fields in a given instruction, the instruction is compiled as if the missing fields contained zeros and had been added to the end of the statement. Such fields whose contents are implied in a standard way by the omission of any explicit specification are called null fields. A null field is usually compiled as a zero. Within a major field, a parenthesized subfield may be made null by omission. Thus, in the VFL example above, if the main index designation were to be zero but the offset and its index modifier were both to be one, the instruction could be written as follows:

OP, A, 1($X1)

A major field may be null even though other non-null fields follow it. Such is the case if nothing but the comma denoting the field termination is written. Thus, in the example just shown, if the offset and its modifier were both to be one but the principal address and its modifier were both to be zero the instruction could be written as follows:

OP, , 1($X1)

8.2.2 Data Description

The small letters (dds) enclosed in parentheses in the special instruction formats stand for the data description field. This field is established by specifying:

1. M	use mode
2. L	field length
3. BS	byte size

These three entries appear in the above order within parentheses and are separated by commas thus: (M, L, BS).

A data word also may have a data description associated with it. When such data are named, the data description may be enforced whenever the name of the data word appears in the address field of an instruction. When both the instruction and the data word have a data description, the dds field of the instruction overrules that of the data word. A description of the method by which a data description may be attached to the symbol that names a group of data is given in section 8.3. The seven use mode designators are:

1. N	Normalized floating point
2. U	Unnormalized floating point
3. B	Binary
4. BU	Binary Unsigned

5.	D	Decimal
6.	DU	Decimal Unsigned
7.	P	Properties mode

The mnemonic P in the mode field of a data description is written as: (P, RIVER).

This use of the mnemonic P in an instruction implies that the data description associated with the symbol RIVER is to be invoked as if it had been written out explicitly. Thus, in an instruction the dds of RIVER overrules anything implied by the symbol in the major address field. The P mode can be used only with legal machine instructions, never with a pseudo-operation.

Within a data description field the usual right-to-left drop-out order and null field conventions hold (except, as indicated, that the mode field may not be null) so that a data description may appear in any of the following four forms:

(M)	field length and byte size are null
(M, L)	byte size is null
(M, , BS)	field length is null
(M, L, BS)	

If the field length is null, a field length of zero (effectively 64, except in the case of immediate VFL operations where it is 24) is compiled. If the byte size is null the compiled byte size is a function of the mode as shown below.

Mode	Standard Byte Size
D or DU	4
B	1
BU	8
N or U	Fixed format of 64 bits; field length and byte size are not appropriate.

Cases can arise from programmer errors in which a data description and the operation are not mutually consistent. In this case the operation overrules. If there is no way to obtain a data description from either a symbolic address or an explicit data field, one of the three following cases can arise.

1. The operation symbol can stand either for floating point or VFL operations (+, -, *, or /). The operation is assembled as an VFL operation with data description (BU, 64, 8).
2. The operation symbol can stand for a VFL operation only (M + 1). It is assigned a data description (BU, 64, 8); if VFL immediate, (BU, 24, 8) is assigned.
3. The operation can stand for a floating point operation only (-A, *NA). The operation is assembled as a normalized floating point except for E + 1 and its modified forms, which are made unnormalized.

An error mark will be printed on the listing in any of these cases.

8.2.3 Mnemonics

A complete list of all machine mnemonics is included in the Appendix. Both operation codes and system symbols are included in the list. System symbols are symbols which represent absolute locations in the system.

8.2.4 Numbers and Symbols

Both integers (whole numbers) and bit addresses may be used in STRAP expressions. Care must be taken when using integer values, as the compiler will interpret the integer as representing a given number of words, half-words, or bits, depending on the field format with which the integer is associated. It is generally considered better programming practice to state locations with a bit address rather than integers. The fixed locations, such as index registers, may be referred to by system symbols such as ($X5).

A bit address is a style of writing a machine address. It consists of a pair of integers separated by a period. The integer to the left of a period specifies a word address and the integer to the right of the period specifies a bit address. Thus, 6. 32 is the decimal equivalent of either a 19- or 24-bit binary address which specifies bit 32 of storage location 6, the bit preceded by exactly 6-1/2 storage words. Note that only the presence of the period distinguishes a bit address from an integer. Example:

505.17 = 500.337 = 0.32337

As the name bit address implies, the two integers are converted to and carried as 24-bit binary integers, such as are appropriate to the address field of VFL instructions. When used in the address field of instructions for which a shorter address is appropriate, a bit address is truncated to the correct length and inserted. The lower-order bits are dropped in such cases. It is well to remember that the period locates the 18th bit of the address.

A symbol is any sequence of six or fewer alphabetic or numeric characters which conform to the following conditions:

1. It contains no special characters.
2. Its first character is specifically alphabetic.
3. It appears in the name field of an instruction by virtue of which it is "defined" and is assigned a value that is either a 24-bit binary address or an integer or, occasionally, both.

A given symbol may appear in a name field only once. The name of an ordinary machine instruction or data entry pseudo-operation is set equal to the value of the assembly program location counter at the point of its appearance in the program. Symbols that identify storage elements in the object program are automatically assigned bit addresses that locate the storage elements.

Symbols that name instructions themselves are automatically assigned data descriptions. Specifically, instruction-naming symbols are given field lengths equal to the length of the particular instruction named (either 32 or 64) and are defined as unsigned binary with bit size 8. Thus, if no (dds) is stated in an instruction which by symbolic addressing refers to another instruction as data, the (dds) is assigned as described above.

Integers in programmer symbolized fields are always converted to binary. They are limited in length to the length of the field in which they are to be inserted. An integer larger than 24 bits cannot be symbolized. Bit addresses and symbols for bit addresses are intended primarily for use in address fields of machine instructions. Integers and symbols for integers are intended primarily for use in fields for which they seem more appropriate: counts, shifts, field length, byte size, and so on.

8.2.5 Arithmetic Expressions

An address may be expressed in a symbolic form as the sum of the symbols, integers, and bit addresses. For example, the address for a floating point instruction could be symbolized in the following manner:

L (N), JOE + GEORGE

In this example the compiler actually computes the address of the instruction by adding the word address of JOE to the word address of GEORGE. A possible use of this feature is illustrated by the case where a control word is desired for writing the first of two blocks of data. The first location of each block is named. A count field of proper size will be generated by the compiler when the following expression is encountered:

CW (CD), FIRST, SECOND - FIRST, HERE

Arithmetic expressions in STRAP I may be composed of addition and/or subtraction of any combination of symbols, integers, and bit addresses. Although integers and bit addresses are generally used in different fields, algebraic addition of the two types of numbers is defined. The result is a function of the type of field into which the sum is to be inserted. Integers add into all fields as integers. That is, the units digit adds into the low-order position of the result field. The number of additive operands in an arithmetic expression is limited only by the space available on the card. Example:

SAM-JOE + FRED - 72.386 + 5

This example, where SAM and JOE are defined as bit addresses and FRED as an integer, is in general a legal address. The data description of the final symbol FRED applies to the whole combination. If the field for which the address is intended is signed (for example, the value field of XW or VF) the sign will be placed in the correct position and the true value will be compiled.

If the final result is negative and the field for which it is intended is unsigned, a twos complement is formed and inserted. For example, in the case of the 7-bit offset field of a symbolic instruction, negative numbers may be used to describe the low-order position of the data field in relation to the left, rather than the right, end of the accumulator. Thus the 128 bits of the accumulator proceeding from left to right bear the offset addresses 127, 126, ...1, 0 or alternatively, -1, -2, ... -127, -128. The programmer is reminded that a twos complement must be used with care on the 7030 in order not to cause the address invalid indicator to be turned on. A positive result is inserted as a true value.

An integer or a bit address, or a combination of the two, may appear in a programmer-symbolized field with the following restrictions.

1. The (I) or (K) index field must contain at least one bit address term. The symbol of an index register is defined as a bit address by the compiler. Thus, ($X1) is defined as 17.0, which is a bit address.
2. No arithmetic may appear in the name field.

3. Arithmetic expressions may not appear in the operation code part of the operation field, the mode subfield, of the data description field, or any entry mode field.

Rules for Combining Integers and Bit Addresses

The following rules describe the method by which bit addresses and integers are truncated and added.

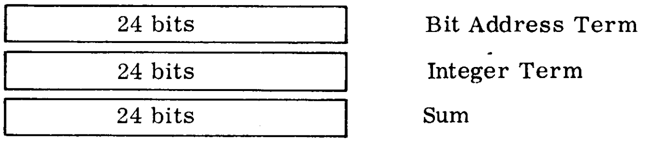
1. The numbers are shifted with respect to each other by the proper amount. This aligns the integer with that portion of the field which is to be the units portion of the result.
2. The numbers are assumed to be signed 24-bit integers before the operation. Addition is algebraic.
3. The result is complemented if necessary (negative result with no sign position).
4. The result is truncated if necessary.
5. The result is inserted into the correct position of the operation word.

Although the following diagrams show the final sum truncated to the appropriate length, the bits are not actually discarded unless they fall outside the address field of the instruction. Some operations do not use all of the space available in their address field (transmit, input-output select) and in these cases bits may be placed in the unused portions.

Truncation occurs for particular fields in the following manner:

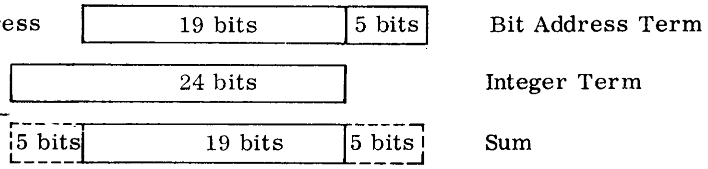
1. A_{24} Bit Address

Rule: No truncation
Note: An integer in a 24-bit field counts bits



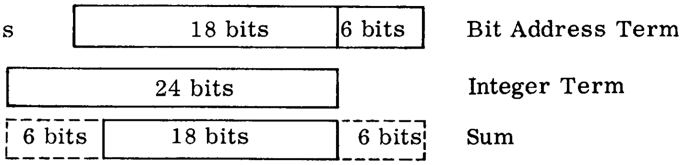
2. A_{19} Half-Word Address

Rule: Leftmost 5 bits and rightmost 5 bits are truncated from sum
Note: An integer in a 19-bit field counts half-words

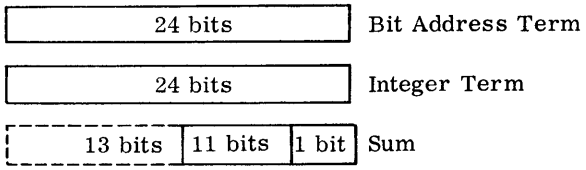


3. A_{18} Full-Word Address

Rule: Leftmost 6 and rightmost 6 bits are truncated from the sum
Note: An integer in an 18-bit field counts full words or unit address, control operation, control word address, and so on, in right I-O address.



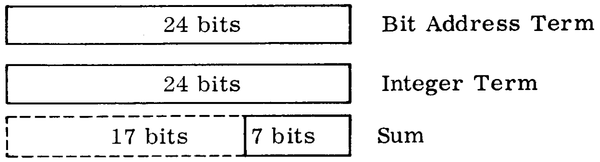
4. $A_{11\pm}$ Signed 11-Bit Address



Rule: Leftmost 13 bits are truncated from the sum. Rightmost 11 bits plus sign are placed in leftmost 12 bits of address field of shift and Add Immediate to Exponent instructions

Note: Integer counts number of bits in shift or number of bits to be added to exponent of floating point word

5. OF_7 Offset

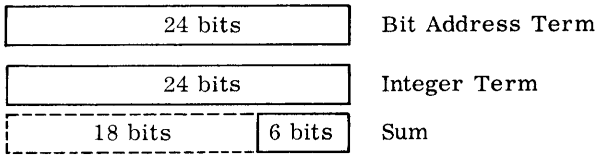


Rule: Leftmost 17 bits of sum are truncated

Note: Integers count number of bits of offset

Bit address 1.32 = .96 = integer 96

6. FL_6 Field Length

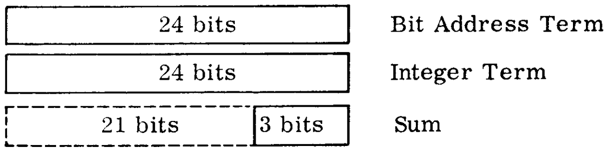


Rule: Leftmost 18 bits of sum are truncated

Note: Integers count length of field in bits

Bit address 1.0 = .64 = 0 not error marked

7. BS_3 Byte Size

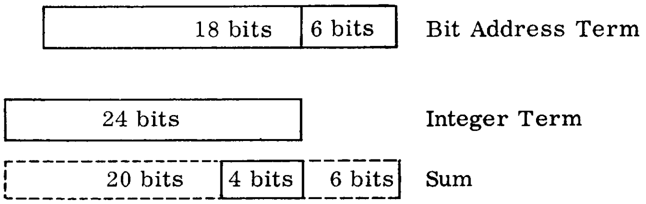


Rule: Leftmost 21 bits of sum are truncated

Note: Integers count byte size in bits

.8 = 8 = 0 not error marked

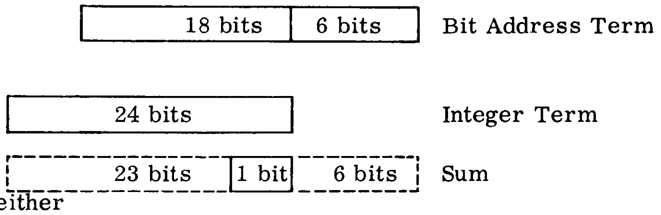
8. I, J 4-Bit Index Fields



Rule: Leftmost 20 bits and rightmost 6 bits of sum are truncated

Note: Integers represent index register number. A "1" in the bit position immediately to the left of the final sum field is discarded with no error indication.

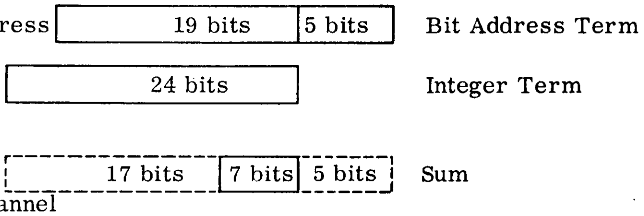
9. K Single Bit Index Field



Rule: Leftmost 23 bits and rightmost 6 bits of sum are truncated

Note: Integers specify either index register 0 or index register 1. A "1" in the bit position that corresponds to "16" in the sum is discarded with no error indication.

10. A_7 I-O Left Effective Address



Rule: Leftmost 17 and rightmost 5 bits are truncated from sum

Note: Integers specify channel address

8.2.6 System Symbols

System symbols are symbols whose value is fixed by the compiler. They are identified in programmer-symbolized fields by the appearance of the special prefix character $ (which as one of the non-alphabetic characters can never appear in a programmer symbol) followed by five or fewer alphabetic or numeric characters. System symbols may appear in arithmetic expressions in programmer-symbolized fields where, if restrictions apply, they can be considered the same as numeric entries, as their values are immediately available to the compiler.

All system symbols that represent the address of special registers in storage ($AOC, the all-ones counter) or special bits in storage ($LC, the lost-carry indicator) are bit addresses. All others are integers or real numbers.

The appearance of the $ character alone makes for a special system symbol that provides the standardized substitute in place of a name for the current statement. That is, the character $ is a bit address which in any statement where it appears functions as if it had been defined by being written in the name field of that statement. Because it represents the value of the location counter when the instruction is encountered by the compiler, the appearance of the dollar sign is interpreted as follows:

```
B, $-2
```

This statement means BRANCH to instruction which begins two full words before this branch instruction.

```
B, $+.32
```

This expression means BRANCH to the next instruction effectively a NOP.

Another special use of the $ character is to prefix any operation code with ($OP). This directs the compiler to suppress any error indications that arise in connection with the compilation of this statement.

The system symbols are described below.

1. Index Registers: $0 through $15, identical to $X0 through $X15, represent index registers 0 through 15, addresses 16.0 through 31.0 in storage. For example, $5 (or $X5) will be correctly replaced by the index field 5 if it appears in an I or J field, or by the address 21.0 if it appears in an address field.

2. Special Registers: The mnemonics for the system symbols that stand for special registers in the 7030 are listed below with the bit address and name for each.

Bit Address	Mnemonic	Name
0.0	$Z	Word number zero
1.0	$IT	Interval timer
1.28	$TC	Time clock
2.0	$IA	Interruption address
3.0	$UB	Upper boundary
3.32	$LB	Lower boundary
3.57	$BC	Boundary control
4.32	$MB	Maintenance bits
5.12	$CA	Channel address
6.0	$CPU	Other CPU
7.17	$LZC	Left zeros count
7.44	$AOC	All ones count
8.0	$L	Left half of accumulator
9.0	$R	Right half of accumulator
10.0	$SB	Sign byte
11.0	$IND	Indicator register
12.20	$MASK	Mask
13.0	$RM	Remainder register
14.0	$FT	Factor register
15.0	$TR	Transit register

3. Indicator Bits: The symbol for any indicator bit may be prefixed with a dollar sign and placed in a programmer symbolized field, where it will represent the correct bit address in word 11. Note that when the indicator symbols are inserted in the "branch on indicator" instructions, the dollar sign prefixed is omitted. System symbols for indicator bits are listed in the Appendix.

4. Input-Output Addresses: Because the actual numeric addresses which are to identify particular I-O units and channels may be chosen arbitrarily, system symbols that represent integers are provided for use in addressing I-O equipment. The numeric values of members of this set of system symbols, unlike the values of all other system symbols, may vary from one installation to another in order that RDR, for example, may represent the card reader channel address independently of what that address, in any particular installation, may be. I-O system symbols are:

Symbol	Meaning
$PCH	Punch (Channel Address)
$PRT	Printer (Channel Address)
$RDR	Reader (Channel Address)
$CNSL	Console (Channel or Unit Address)
$TC0, TC1....etc.	Tape Channels 0, 1, ...etc.

If more than one punch, printer, console or any other input-output unit is attached to the machine, the same numbering system used in tape channel addresses is adopted, where $CONSL = $CONSL0, and so on. Thus, one may have $PRT0, $PRT1, $PRT2, etc.

5. Mathematical Constants: Several mathematical constants are available as symbols. Among these symbols is $PI which equals π.

8.3 PSEUDO-OPERATIONS

In this section are covered a number of mnemonic codes provided for purposes of defining data and controlling and directing the assembly process. Because these codes do not directly produce machine instructions in the object program, they are called "pseudo-operations." The pseudo-operations may be divided into two main categories, the first consisting of those operations that create storage elements, and the second containing the operations that control the assembly process.

8.3.1 Pseudo-Operations That Create Storage Elements

Six pseudo-operations are used for the entry of general data. Several of these pseudo-operations have been used in programming examples in previous sections of the manual. These operations are:

	Mnemonic	Definition	Format
1.	XW	Index Word	XW, VALUE, COUNT, FLAG
2.	VF	Value Field	VF, VALUE
3.	CF	Count Field	CF, COUNT
4.	RF	Refill Field	RF, REFILL
5.	CW	Control Word	CW(OP), ADDRESS, COUNT, CHAIN ADDRESS
6.	DD	Data Definition	(EM)DD(dds), D, D', D'', D''',

The encountering of the mnemonic XW provides the following action by the compiler. The location counter is rounded to the next full word. The contents of the four symbolic fields following the operation are converted and compiled in an index word format. FLAG denotes the machine field comprised of bits 25, 26, and 27. The desired flag configuration is obtained by entering the bit pattern in octal. Thus, an octal 4 in the flag field results in a compiled field which contains a one in the index flag (position 25) of the index word. Position 24 of the index word is the value sign; therefore, the status of the compiled bit 24 is determined by the sign of the number which appears in the value field of the pseudo-operation.

NOTE: When negative amounts are compiled for fields which do not have a sign position, the twos complement of that number is inserted in the desired field. The mnemonic VF provides the following action by the compiler. The location counter is rounded to the next half-word. The contents of VALUE are compiled as a 25-bit signed quantity in positions 0-24 of the next half-word.

The mnemonic CF provides the following action by the compiler. The location counter is rounded to the next half-word. The contents of the count field are compiled as an 18-bit integer in positions 0-17. The mnemonic RF causes the same action as CF. The two operations are labeled differently merely for the convenience of the programmer.

The four pseudo-operations just described are assigned data descriptions by the compiler. Thus, if an instruction which contains no (dds) refers to any location containing an XW, VF, CF, or RF, the data description assigned by the compiler is invoked. The assigned data descriptions for these operations are:

Operation	Data Description
XW	(BU)
VF	(B, 25)
CF or RF	(BU, 18)

The pseudo-operation CW, which utilizes the special format previously illustrated, provides for a secondary operation. These secondary operations, expressed in parentheses following the mnemonic CW, are listed below with the resulting control word flags.

		Multiple Bit	Chain Bit	Skip Flag
CR	"Count Within Record"	0	0	0
CCR	"Chain Counts Within Record"	0	1	0
CD	"Count Disregarding Record"	1	0	0
CDSC	"Count Disregarding Record, Skip, and Chain"	1	1	0
SCR	"Skip, Count Within Record"	0	0	1
SCCR	"Skip, Chain Counts Within Record"	0	1	1
SCD	"Skip, Count, Disregarding Record"	1	0	1
SCDSC	"Skip, Count, Disregarding Record, Skip and Chain"	1	1	1

The location counter is rounded up to insure that the control word compiled will begin at a full-word address. CW is assigned a data description of (BU, 64, 8).

As stated previously, any field which may contain programmer symbols and/or system symbols is called a programmer-symbolized field. The various fields of the pseudo-operations covered above are all programmer-symbolized fields. In any programmer-symbolized field not enclosed by parentheses, numerical integers and bit addresses may be written in any radix from 2 through 10, or 16. The radix is specified by enclosing the appropriate integer, written in decimal, in parentheses at some appropriate point in the subfield. (Usually, but not always, the radix specifier is the first item to appear in the subfield.) Once specified, the radix applies to the entire field unless reset by another radix specifier. If no radix is specified, it is assumed the information being entered is written in decimal. Thus, data to be entered with the pseudo-operation CF could be entered in binary, octal, or decimal as:

```
CF, (2)1000000        Binary
CF, (8)100            Octal
CF, 64                Decimal
```

The pseudo-operation DD, data definition, is the most general method of entering data. As illustrated previously, the format for the data definition operation is (EM)DD(dds), D, D', D'', D''', ... The use of the DD pseudo-operation enables the programmer to enter data into a program in a variety of forms. Among the possibilities that exist are:

1. Decimal to floating binary conversion, either normalized or unnormalized.
2. Conversion of decimal fraction to binary fraction in fixed point.
3. Integer-to-integer conversion from any of the radices 2 through 10 and 16 to a radix of either 2 or 10.
4. Conversion of alphabetic information to binary coded form.

In the general format illustrated above, the (EM) field represents the entry mode, a field which supplies information about the form in which data appear on the card, described in more detail later.

The data description (dds) is identical in form and content to that described under the explanation of instruction formats. Note, however, that the "P" mode is not permitted here or in any other pseudo-operation. It is possible to state a data description with the data at the time they are entered. It is also possible to include a data description when entering the instruction which is to operate on the data. If the instruction contains no data description, it will be assigned the data description which appears with the data upon which the instruction is to operate. If both the instruction and the data have a description associated with them, the (dds) of the instruction overrules. This overruling is strictly local and applies only to the instruction at hand.

To summarize: In general, the (EM) describes the form in which the programmer has written the data which are to be compiled. The (dds) describes the format in which the programmer wishes the compiled data to appear. The fields D, D', D",......are identical to each other. They are compiled sequentially as separate groups of data, each having the data description specified. If the operation is "named," the name applies only to the first D field. To have each compiled field named, each entry must be made on a separate card having its own name.

8.3.2 Entry Mode in Data Definition Statements

As mentioned above, the entry mode defines the language in which the input data are written. The entry mode field may pertain to the entire statement, or only a field of the statement. Some specifications may be used only when the entry mode is used to specify the entire statement. Other specifications may be used only if the entry mode is used to define a field within the statement. When the entry mode is used to describe the entire statement, it is called a "Statement Entry Mode." When used to describe a field within the statement, the entry mode is called a "Field Entry Mode."

Statement Entry Mode

The two entry modes explained here can be used only as statement entries and never as field entry modes. Both of these modes provide a means of compiling alphabetic information. These two entry modes are:

(Ax)	Alphabetic Conversion	(AQ)DD(BU, 60, 6), DO NOT PANIC Q
(IQSx)	Inquiry Station Conversion	(IQSx)DD(BU, 40, 8), WORDS x

The alphabetic conversion entry mode causes the message which follows the (dds) field to be converted to BCD. The end of message is specified by the letter which appears in the entry mode field. Thus, in the example above, the end of the statement which is to be compiled is indicated by the letter Q. Byte sizes of 1 through 12 may be specified in such statements. When the specified byte size is greater than 6, leading zeros are inserted. When the specified byte size is less than 6, the leading bits of each byte are truncated. The statement terminating character may be any legal card code character other than:

)
'(8-4)
;(11-0)
blank

Only one D field is allowed per statement.

The IQS entry mode operates in exactly the same fashion as the alphabetic mode, except that card code characters are converted to the 7-bit console typewriter code. Therefore, leading zeros are inserted where the byte size is greater than seven, and leading bits are truncated if the byte size is less than seven.

Although the IQS (typewriter) code includes a large number of special characters, STRAP-I is limited to those which can be entered by means of IBM off-line card and tape equipment. The typewriter character codes are included in the Appendix.

Statement or Field Entry Modes

Some entry modes may be used either to specify the properties of all fields of a statement or to specify the properties of a specific field or fields in a statement. Statement entry modes and field entry modes may both appear in the same statement. When contradictory properties (for instance, two different radices) are implied by the statement and field entry modes, the field entry mode overrules for the case of the particular field on hand. Entry modes may not appear in a manner that causes parentheses within parentheses.

(Fn): The entry mode (Fn) implies that the information which follows is written in the decimal radix, is to be converted to binary, and may include a decimal fraction portion that is to be converted to a binary fraction of length _n_ bits. The "n" symbolizes a decimal integer that specifies the number of fractional bits desired to the right of the binary point when the number or numbers which follow are converted. (Fn) appears in binary mode statements only.

Radix Specifications: In any programmer-symbolized field not enclosed by parentheses, numerical integers and bit addresses may be written in any radix from 2 through 10, or 16. The radix is specified by enclosing the appropriate integer, written in decimal, in parentheses at some appropriate point in the subfield. (Usually, but not always, the radix specifier is the first item to appear in the subfield.) The radix applies to the entire subfield unless it is reset before reaching the end. If no radix is specified, the base 10 is assumed. If used as a statement entry mode, the radix specified applies to the entire statement unless individual fields contain their own radix specifier, in which case the field entry mode overrules the statement entry mode for that field only.

In the case of data entry, the radix specifier can be used with integers only; a decimal point or floating point notation implies a radix of 10. The entry mode radix specifies the radix in which an integer is written on the card, but says nothing about the one to which it is converted. Some examples of the use of the radix specifier are:

1. (8)573 - 34+50 (All numbers are in octal.)
2. (2) 11011011100011.111100 (Bit address written in binary.)
3. (5) SAM - 342 (The symbol SAM is not affected by the radix, having been previously converted to binary. The integer 342 is written in the number system of the base 5.)
4. (8)7436.(10)60 + 9 (The full-word portion of this bit address is written in octal, whereas the bit address portion and the integer 9 are written in decimal.)

5. (2)DD(B, 16, 8), (10)-972, 111011110 (The first D field is written in decimal; the second one is in binary.)

NOTE: Example 3 contains a symbol, and therefore cannot appear in a DD statement. Examples 2 and 4 are bit addresses and therefore may not appear in a DD statement.

Field Entry Mode

One entry mode in STRAP-I, the parenthetical integer entry, may never appear as a statement entry mode. This particular mode will probably not find wide usage by customer engineers, and is, therefore, not described in this manual. A description of the parenthetical integer entry may be found in the IBM Reference Manual, 704-709-7090 Programming Package for the IBM 7030 Data Processing System, Form C22-6531.

Form of D in a Data Definition Statement

All data fall under the category of one of the six use modes of the data description field: N, U, B, BU, D, DU. The numbers D, D', D", ... are expressed in the general form:

±XX---X. X---X

Decimal numbers are a special case; they may be written in fixed or floating point form, with or without a decimal point. The general form is:

±XX---X. X---XE±YYY

In this form E means that the number which precedes it is multiplied by 10 raised to the power which follows it. That is, 572.34E - 57 means 572.34 x 10^{-57}. Overlapping facilities for specifying an exponent "Ei" are provided in the sense that the decimal point in the number itself also indicates a decimal exponent. If no decimal point is written, the number is assumed to be an integer. Thus, as illustrated below, parts of the general form that are not necessary for writing a number may be omitted.

1.	±XXX	integer
2.	±XXX.XX	decimal fraction
3.	±XXXE±YYY	integer times power of 10
4.	±XXX.XXE±YYY	decimal fraction times power of 10

A plus sign is understood if no sign is specified. The decimal point may be in any position in the number. The portion of the number above symbolized by X is limited in length to 15 digits; that symbolized by Y is restricted to a length of 3 digits. (Recall that floating point numbers in the 7030 are limited to a range of 10^{308} to 10^{-308}.) It should be remembered that the presence of a radix point or exponent indicates that the information is written in decimal.

Data entries may have other quantities following them which are identified and separated from the main number by certain characters, used for the insertion of specific fields:

1. Sign Byte Entry--Si
2. Exponent Entry--Xi

The letter S is used to enter information into the sign byte of data. The letter i represents an octal integer which is evaluated and OR'ed in with any sign byte previously calculated. Thus, if either the sign of the main number or i implies a negative sign bit in the sign byte, the sign byte sign position is made negative. The sign byte generated depends on the byte size in accordance with the following table:

Byte Size	Sign Byte	
1	S	
2	ST	
3	STU	Z = zone bit
4	STUV	S = sign bit
5	ZSTUV	T
6	ZZSTUV	U } flag bits
7	ZZZSTUV	V
8	ZZZZSTUV	

In a data definition statement where byte size 1 is specified, using sign byte entry S1 yields a negative sign, whereas if byte size 4 had been specified, S10 would yield a negative sign with zero flag bits. The letter X may be used to enter any arbitrary information into the exponent of a floating point word. The decimal integer is compiled as the machine exponent of a floating point number. It overrules and replaces the computed exponent, which is completely eradicated by the replacement process.

8.3.3 Rules for Entering Data

The legal formats for entering data can be classified according to the use mode written in the data description field of the DD statement. In general, an element listed in the general format may be omitted if it is not needed to specify the data.

The data entries in a DD statement are restricted to real numbers. Symbols are not permitted. Bit addresses may not be written in a DD statement because the compiler will interpret them as mixed decimal numbers.

Floating point data are always compiled in addressable full-words; the location counter is rounded, if necessary, to the next full-word to accomplish this. This is an example of a general STRAP I principle: if proper use of the data depends on the data's occupying a particular portion of a storage word, the location counter is automatically rounded by the proper amount. Thus:

1. Instructions always start at either half- or full-word bit addresses.
2. Indexing full- and half-word storage formats are forced to begin at full- and half-word addresses, respectively.
3. A floating point data block being reserved through use of a DR operation code (defined in "Pseudo-Operations That Direct the Compiler") is forced to begin at a full-word address. Moreover, when a field from an instruction format requires the truncation of the rightmost bits before compilation, a warning indication is given if significant bits are truncated (which can occur if an instruction addresses a format other than its natural one; e.g., if a floating point instruction addresses a VFL data element).

Normalized Floating Point

Format: Name | DD(N), ±xx···xx.x···xxE±yyySn

The decimal number is converted to a normalized floating binary number consisting of an 11-bit signed exponent, a 48-bit fraction, and a 4-bit sign byte. If no sign byte has been entered by means of an S, the sign preceding the number is used with the flag bits set to zero. If a different binary exponent is desired, it can be entered following an X, as follows:

Format: Name | DD(N), ±xx ···xx.x ··· xxE±yyySnXzzz

Examples:

1. DD(N), 54, 73 E 4
   54.73 x 10^4 is converted to floating binary. The sign bit is zero (= plus), and the flag bits are zero (i.e., entire sign byte is zero).

2. DD(N), -54.73 E 4, or DD(N), 54.73 E 4 S 10
   In this case the sign bit is set to one (negative) and the flag bits are zero.

3. DD(N), -54.73 E 4 S 5
   The sign bit is one, since the number is negative, and flag bits T and V are one. U is zero.

4. DD(N), 1, 3E-5, -45.7, 12 S 17
   This example illustrates the multiple entry feature. This single DD statement compiles four 64-bit floating point words and advances the location counter accordingly.

NOTE: (EM) is not allowed with normalized entries.

Unnormalized Floating Point

Format: Name | (Fn)DD(U), ±xx---x.x---xE±yyySn X±n
or | DD(U), (Fn) ±xx···xx.x···xE± yyySnX±n, (Fn)±xx···etc.

The number is converted to binary with the correct number of binary fractional places as specified by the (Fn) entry mode, and a correct exponent is computed and entered. This exponent is overruled and replaced by that following the X if X is used (necessary only if, for some reason, the programmer desires an incorrect exponent). The entry mode (Fn) can come before the DD, in which case it applies to all D fields of the statement, or it may form the first element of a D field, in which case it overrules the one given before the DD. Either the X or the S, or both, may be omitted or their order may be interchanged. Omitting S has the same effect here as in the normalized case. Omitting X simply allows the correct exponent to remain as computed. Leaving out the sign, decimal point, or E is permitted as in normalized numbers.

Examples:

1. DD(U), (F21) - 343.7, (F10) 432
   Two numbers are compiled. In the first, 343 is converted as an integer and .7 is converted to a 21-bit fraction. They are joined and placed in the rightmost

bits of the fraction portion of the floating point word, and the correct exponent (in this case 27) and sign are supplied. In the second D field, 432 is converted to a binary integer. Because ten fractional bits are specified, but no decimal fraction is written, the ten rightmost bits (50-59) of the fraction field are set to zero and the number is entered with its rightmost bit in position 49.

2. (F15)DD(U), 767.52, 767.52 X-12 S11
   The (F15) applies to both D fields. In the second, the computed exponent is overruled by the specified one and the number is made negative by means of the specified sign byte.

3. (F15)DD(U), 767.52, (F20) 767.52 S11 X-12, 398
   This is identical to example 2 except that in the second field the operation entry mode (F15) is overruled by a field entry mode (F20), and the order of S and X is interchanged, which makes no difference. (F15) still applies to 398.

If the entry mode is omitted, two cases arise: 1) If the number entered is an integer, (F0) is understood. 2) If the number entered is a decimal fraction, it is converted to a normalized floating point number. The data, however, retains an unnormalized dds.

Examples:

1. DD(U), 17, 17X-35
   In the first case 17 is converted to binary and placed in the fraction with its rightmost bit in position 59 and an exponent of 48 supplied. In the second field the same thing is done except that the exponent is set to -35.

2. DD(U), 17.5
   In this example 17.5 is converted to normalized floating binary and stored as such. However, instructions whose normalization bits depend on the symbol in the name field of this pseudo-operation will have them set to unnormalized.

NOTE: 17 E 5 is an integer and will be recognized as such.
17 E-5 is a decimal fraction and will be normalized.
17.5 E 5 is an integer but will be treated as a fraction and normalized. Thus, a normalized integer can be assigned use mode "unnormalized."
An integer greater than 248 is stored as a normalized number.

Binary Signed VFL

Formats: (Fn)DD(B, FL, BS), ±xx---x.x---xE±yySn
DD(B, FL, BS), (Fn)±xx---x.x---xE±yySn
(R)DD(B, FL, BS), ±xx---xx Sn
DD(B, FL, BS), (R) ±xx xx Sn

A data definition of binary signed data may have either (Fn) or (R) entry modes, but not both at the same time. (Fn) implies that the number following it is written in a decimal radix, whereas (R) implies that the number following it is an integer. An integer subject to a radix entry mode must be written without the aid of E because E is not defined for a radix other than 10. A decimal fraction must have a controlling (Fn) entry mode. In the data description, either the field length or byte size, or both, may be omitted. The implied field length in this case is 64; the implied byte size is 1. The sign byte need not be specified unless the programmer desires to have flag or zone bits different from zero. Note that the sign bit position changes for a byte size less than 4. To make a number negative, specify the sign byte as:

BS = 1,	S1
BS = 2,	S2
BS = 3,	S4
BS = 4,	S10

If a number has no entry mode at all, it must be a decimal integer, but may in this case be written with the aid of the E notation.

Examples:

1. (F7)DD(B,,4), .005E3S13, -17; 143.2S11, (8) 77760, 777
   Implied field length is 64. Octal specification in the fourth D field overrules (F7) written before DD, but (F7) still applies to 777.

2. (2)DD(B, 16 8), 110101S377, (10) -972, 11101110S201
   Binary entry, overruled in only the second D field.

3. (F12)DD(B, 24), 1.324E3, -72.1E-4, 3.4E-4S1
   Implied byte size is 1.

4. DD(B), 1489, -1272, 1491, (F13) -972.16, 13948S1, 12E5
   Decimal integers, except where a field entry mode is written.

Binary Unsigned VFL

Formats: (Fn)DD(BU, FL, BS), xx---x.x---xE±yy
DD(BU, FL, BS), (Fn) xx---x.x---xE±yy
(R)DD(BU, FL, BS), xx---xx
DD(BU, FL, BS), (R) xx---xx
(Az)DD(BU, FL, BS), alphabetic information to "z"
(IQSz)DD(BU, FL, BS), alphabetic information to "z"

Numerical entry is exactly the same as in binary signed data except that no sign byte is formed, and if the byte size is left out of the dds, it is set to 8. Any sign or sign byte (with S) written with mode BU is ignored. The two alphabetic modes are permitted here. Note that the alphabetic entry mode must precede the DD, that there can be only one D field per statement, and that if the field length is omitted, it is set equal to the byte size.

Examples:

1. (F13)DD(BU, 30), 17.2, 183, (8) 70707

2. (A*)DD(BU, 48, 6), GLORIOUS FRIDAY, THE 13TH.*
   The mode and field length have no effect on the conversion and storage; they are used in compiling instructions that refer to the name of this statement. Field length 48 indicates that the programmer wants to process these characters in groups of 8.

3. (IQSS)DD(BU, 32, 8), DOG EAT DOG S

Decimal Signed VFL

Formats: (R)DD(D, FL, BS), ±xx---xxx Sn
DD(D, FL, BS), ±(R) xx---xx Sn
DD(D, FL, BS), ±xx---xxEyy Sn
(Fn) has no meaning for mode = D or DU

The two decimal modes in DD and DDI statements represent the only cases in which STRAP I converts numbers to an internal decimal radix. This conversion is limited to being available only from integers to integers. The radix entry mode indicates the radix in which the numbers are written on the card. Thus, it is possible to write an integer in binary or octal and have it converted to decimal for machine use. If no entry mode is given, decimal-to-decimal is implied and the E notation can be used to multiply an integer by positive powers of 10. If either the field length or byte size is omitted, the implied values are FL - 64, and BS = 4.

Examples:

1. DD(D), -9534812, +173E5, 18E10S13
Field length = 64; byte size = 4. A 4-bit sign byte is formed. Decimal-to-decimal conversion.

2. (2)DD(D, 20), 111010001101S7
Byte size = 4. Binary-to-decimal conversion.

3. DD(D, , 8), 432E3
Field length = 64. Decimal-to-decimal conversion. Four binary zeros are inserted in the zone positions of each byte.

Decimal Unsigned VFL

Formats: (R)DD(DU, FL, BS), xx---xx
DD(DU), FL, BS), (R) xx---xx
DD(DU, FL, BS), xx---xxxEyyy
(Az)DD(DU, FL, BS), alphabetic information to "z"
(IQSz)DD(DU, FL, BS), alphabetic information to "z"

The numerical conversion is just as in decimal signed mode except for the omission of the sign byte. Alphabetic conversion is exactly as in the binary unsigned mode, except that instructions referring to these data may be compiled as decimal operations. For alphabetic entry, implied field length is equal to byte size.

Examples:

1. DD(DU), 8430051, (8) 77241, 82E10
Field length = 64; byte size = 4.
An octal-to-decimal conversion is inserted between two decimal-to-decimal conversions.

2. (IQS3)DD(DU, , 8), PUSH PANIC BUTTON 3
Field length = 8.

Summary of Rules for DD Statements

Entry Mode	Appropriate Use Modes
Fn	U, B, BU
R	B, BU, D, DU
A	BU, DU
IQS	BU, DU

NOTE: Use mode N should have no entry mode.

Special Field Entry	Appropriate Use Modes
S	N, U, B, D
X	N, U

The floating decimal notation, using E to designate multiplication by powers of 10, is appropriate to all modes although it is always restricted to a decimal radix, and in the decimal use modes, is restricted to increasing the magnitude of decimal integers. If the field length is omitted from the dds, it will be assigned a value of 64, except in the case of alphabetic entry where it is set equal to the byte size. The maximum permissible field length for a DD statement is 64.

8.3.4 Pseudo-Operations That Define Symbols

Almost all pseudo-operations define symbols in the same manner. That is, any symbol appearing in the name field is assigned the current value of the location counter. The pseudo-operations DDI and SYN operate somewhat differently. A symbol appearing in the name field of a DDI or SYN operation is assigned the value that appears in the statement field of that entry. Thus, if SAM is the symbolic name of location 5000, and this location contains a normal DD operation, an instruction written as LVI, SAM is converted by the compiler to LVI, 5000. If the value contained at location 5000 is 7, and 5000 is a DDI or SYN operation, the same instruction (LVI, SAM) is compiled as LVI, 7. Thus, SAM is defined by its own statement field rather than by the contents of the location counter. Note that the DDI and SYN pseudo-operations do not create storage elements in the compiled program, but that the compiled DDF and SYN entries are placed in a table for use by the compiler.

DDI Data Definition Immediate

The DDI operation is intended primarily as a means of entering data to be used in immediate addressing instructions. First consider the compiler's normal treatment of immediate addressing instructions. In previous sections, immediate type instructions have been written with a decimal point in the address part. Remember that the decimal point absolutely locates the 18th bit of the address. If no decimal point is included in the address part of an immediate addressing VFL instruction, the instruction is compiled in a very different manner. In such cases, the field length contained in the (dds) of the instruction defines the position to be occupied by the low-order bit of the address. For example, the instruction LI (BU, 12), 15 is compiled such that the low-order bit of the number 15 is located in position 11 of the instruction. However, if the same instruction is written LI (BU, 12)15. 0, the decimal point overrules the normal manner of compiling and the low-order bit of the number 15 appears in position 17 of the instruction address.

It may appear that the DDI operation is not necessary because the absolute location of a converted address may be determined by the specified field length or by inserting a decimal point in the address field. Note, however, that no means is provided, in this method, for the compiling of immediate VFL decimal operands. All alternate methods of compiling immediate operands result in a conversion to pure binary. Thus, the main use of the DDI operation is the compiling of immediate decimal operands for VFL operations.

The pseudo-operation DDI is identical to DD except for the following points:

1. Only one major field may follow the operation field of the statement.
2. If no field length is specified, a field length of 24 is implied.
3. If the converted field is alphabetic and exceeds the field length, the low-order characters are dropped and an error indication is given on the listing.
4. Neither of the floating point modes may be used in a DDI statement.

If a DDI has a field length of less than 24, the number that it defines will appear in the leftmost portion of the address of the operation when it is compiled in an immediate operation. Unused bits in the right end of the address field will be zero. If the address field of an immediate operation contains arithmetic among symbols or symbols and integers, the arithmetic will be done in binary, regardless of how the symbols are defined or what the mode of operation is. All numeric entries in such an address field are handled exactly as other addresses and are converted to binary, never to decimal. Therefore, the only way to get a decimal number into the address field of an immediate operation without writing it in the 7030 BCD code explicitly is to symbolize it and use DDI.

If the instruction referring to a DDI statement contains no (dds), the (dds) of the DDI is invoked for use by the instruction. If both the DDI and the instruction contain a (dds), the data are compiled and placed in a table as per the (dds) of the DDI statement and used in the instruction as per the (dds) of that instruction. For example:

Name	Statement
JOE	DDI (DU), 9478
SAM	DDI (DU, 12), 342
BOB	DDI (B), -142

Statements JOE and SAM are compiled as decimal operands. JOE has an implied field length of 24 while SAM has a specified field length of 12. BOB is compiled as binary with an implied field length of 24 bits of which the low-order bit is the sign (implied byte size for signed binary is 1).

When compiling addresses for immediate operations, STRAP I assumes that a symbol defined by DDI has a sign byte if one is needed. It assumes that a symbol defined in any other way does not have a sign byte and compiles one having flag and zone bits equal to zero and byte size as specified in the dds.

It must be remembered that any arithmetic performed between symbols or symbols and integers of an immediate type instruction will be in binary even though the DDI which defines the symbol may specify decimal. Thus, in the two examples shown below, different results are obtained.

Name	Statement
SAM	DDI (DU, 12), 392
BILL	DDI (DU, 12), 12
HERE	LI, SAM + BILL

In this example, logical results are not obtained because the data are compiled as unsigned decimal but the arithmetic performed to produce the immediate operand at location HERE is binary.

Name	Statement
SAM	DDI (DU, 12), 392
BILL	DDI (DU, 12), 12
HERE	LI, SAM
THEN	+I, BILL

In this example, the data compiled as decimal are used as decimal and logical results are obtained.

SYN, Synonym

The operation SYN is stated in the format shown below.

Name	Statement
A	SYN (dds), Y

When any instruction is written with a symbolic address which is the name of a SYN operation, the effect is the same as if Y of the SYN had been written in the instruction. The meaning of SYN is always one of exact substitution. Instructions which contain SYN defined symbols are permitted to have their own data description field. If, however, no data description is given, the data properties of the final symbol not in parentheses are transferred to the name. Consider the following example of the use of SYN and the data properties of the final symbol:

Name	Statement
SAM	SYN(N), 1000.0
FLAG	SYN(BU, 3, 8),.61
	(intervening code)
	L, SAM + FLAG

LOAD loads only the flag from the floating point word SAM preparatory to some VFL arithmetic or tests on the flag.

A common use of the SYN operation is the defining of symbols which represent I-O channels. Thus, a program may contain the symbolic instruction: W, CNSL, CW. In such a case a SYN card may be used to define CNSL as follows:

Name	Statement
CNSL	SYN, (8) 23.40

In this example a decimal point is included in the Y field and, in octal, one-half the channel address of the console is expressed. A channel address of 39 results.

8.3.5 Pseudo-Operations That Give Directions to the Compiler

Set Location Counter

Mnemonic	Format
SLC	A SLC, Y

This pseudo-operation resets the location counter to the value of the address Y. The next instruction will be compiled at this address, subject to rounding upward conventions. For example, a floating point instruction will be located at the nearest full-word address. If Y is not a full-word address, the location counter will be rounded up to the nearest one.

Y must contain a bit address expression whose value is positive. An integer appearing in this field is interpreted as representing bits.

END

Mnemonic	Format
END	A END, Y

A card with the operation code END signals the end of an assembly, and is usually included as the last card of each symbolic program deck. A branch card is then punched with the binary output deck with an address Y. If a standard load program is used to load in the assembled program, the branch card will be recognized and the loading will be terminated. In such cases, the first instruction of the new program to be executed is specified in the Y field of the END statement.

Any symbol A in the name field is ignored.

Conditional No Operation

Mnemonic	Format
CNOP	A CNOP,

The pseudo-operation CNOP is used to insure that the instruction or data immediately following the CNOP will be assigned a full-word address by the compiler. When a CNOP is encountered, the location counter is immediately rounded up to the next half-word address. Then the compiler examines the location counter. If it already stands at a full-word address, the CNOP is ignored. If, however, the location counter is set to a half-word address, the machine instruction NOP is compiled. This has the effect of advancing the location counter 32 bits or one half-word to the next full-word address.

Any symbol A appearing in the name field is assigned a full-word address when the CNOP is ignored, or a half-word address when a NOP is compiled.

Terminate Loading and Branch

Mnemonic	Format
TLB	TLB, Y

As is the case with END, this operation is intended for use with a standard loader.

The pseudo-operation "Terminate Loading and Branch" is similar to an END statement with one major distinction. TLB does not stop the assembly process. Therefore, TLB may be assembled at any point in a symbolic deck where a transition card is desired. The branch card thus produced will interrupt the loader when encountered in a binary deck and transfer control to the instruction at location Y. The remainder of the program is loaded under program control.

Extract

Mnemonic	Format
EXT	A EXT, (I, J) STATEMENT

The extract pseudo-operation has the following meaning: First, STATEMENT is compiled as in any legal machine instruction. Then the field beginning at I and ending at J is extracted and compiled in the position in the program where the EXT occurs. The symbol A appearing in the name field is assigned a data description (BU, J-1 + 1, 8) and is attached to the quantity compiled. If EXT is used to specify the extraction of anything beyond the range of the single statement that follows it, up to 64 zeros will be added.

Example: EXT(18, 47) + (B, 18, 7), 73.16

First the full-word instruction + (B, 18, 7), 73.16 is formed. Then bits 18 through 47 (the first bit in the instruction is numbered 0 according to 7030 custom) are extracted and placed in the program being compiled. The dds (BU, 30, 8) is formed. The location counter is advanced 30 bits.

NOTE: STATEMENT must be a legal machine instruction, not a pseudo-operation.

Data Reservation

Mnemonic	Format
DR	A DR(dds), N

A DR reserves space for data. The operation causes N fields of the kind described in the data description to be reserved; that is, the instruction location counter is skipped forward a quantity in bits equal to the product of N and the field length specified in the dds. Any symbol A appearing in the name field is attached to the first field reserved, as is the dds. Therefore, whenever A appears as the principal address in an instruction, this dds is invoked in the same manner as with a DD or DDI statement. Thus:

```
JOE                    DR(BU, 8, 8), (10)
```

reserves ten 8-bit fields (skips the location counter forward 80 bits). The dds (BU, 8, 8) is attached to JOE. JOE is attached to the first 8-bit field reserved.

Data Reservation and Set to Zero

Mnemonic	Format
DRZ	A DRZ (dds), N

This pseudo-operation causes the same action as DR, except that the reserved fields are all set to zero.

Print Single-Spaced

Mnemonic	Format
PRNS	PRNS

This pseudo-operation causes the assembly listing to be printed with single spacing. Double spacing is the normal mode unless PRNS is written.

Print Double-Spaced

Mnemonic	Format
PRND	PRND

This psudo-operation restores printing to the normal double-spacing condition after the use of a PRNS.

Punch Full Cards

Mnemonic	Format
PUNFUL	PUNFUL

If the binary deck is to be punched with data in all 80 columns, this psudo-operation must be included in the symbolic program. If PUNFUL is not used, the assembled binary deck will be punched in a format intended for use with a standard loader. The standard load programs are not yet complete. Note that the no-ECC mode is used for PUNFUL.

Punch Normally

Mnemonic	Format
PUNNOR	PUNNOR

This pseudo-operation restores standard punching after the use of PUNFUL. Cards punched in the standard format will include a check sum, first word address, identification, and so on.

Skip Paper

Mnemonic	Format
SKIP	SKIP, i

If the i = 0, the assembly listing will restore the paper immediately. If i ≠ 0, one half-page will be skipped.

Punch ID

Mnemonic	Format
PUNID	PUNID, XXXXXXXX

The first eight characters following the comma are punched in columns 73-80 of each card in the binary deck produced by the assembly program. Thus, a PUNID card should be used to identify each assembly. The X's may be any legal card code characters. Of course, this operation is not intended for use on PUNFUL assemblies.

Print ID

Mnemonic	Format
PRNID	PRNID, COMMENT

When PRNID is encountered, the entire contents of this card are immediately printed on-line and on the output tape as well. PRNID provides a means of heading the assembly listing with such information as the problem name, programmer, and so on.

Suppress Error Marks

Mnemonic	Format
SEM	SEM, A, B, C

The operation code SEM, followed by a blank address field, causes all error marks detected in the statements that follow the SEM statement to be suppressed in the listing. Any particular error flag or group of flags may be suppressed by writing the letters or characters identifying the flags in the address field, separated by commas. Thus, SEM, Q, T suppresses error flags Q and T only. Certain error marks will not be suppressed regardless of SEM.

Resume Error Marks

Mnemonic	Format
REM	REM, A, B, C

A REM restores normal error marking on the listing after a SEM has been used. The ability to specify individual error flags is also available with REM. Thus, following the statement SEM, Q, T the pseudo-operation REM, Q restores error flagging involving flag Q only, while flag T remains suppressed.

TAIL

Mnemonic	Format
TAIL	TAIL, CHARACTER

The statement TAIL, X causes a block of statements to have the symbol X appended as a suffix to each symbol appearing in each statement of the "tailed" block. The block is ended by the next tail statement or by an untail statement. (Untail is equivalent to a tail statement with a null field.)

In STRAP I the symbol X may be any single alphabetic or numeric character. In STRAP II the tail symbol may be any legal symbol.

If a basic symbol is defined two or more times in different blocks, then "within block" references may be made in the normal manner. References from one block to another must use the method:

```
OP, JOE$X
```

where the desired JOE is defined in the block tailed by X, and is defined there only once. Also permitted is:

```
OP, JOE$
```

which references a JOE defined in an untailed block. An untailed block behaves exactly like a block tailed by a blank.

Note that the tail character is included in the name field, and in STRAP I the name, including the tail character, may not exceed six characters.

9 ERROR INJECTION

THE 7030 performs a large amount of automatic error checking. It is desirable to have a method for the deliberate introduction of errors into the computer to test the operation of the check circuits. A method of introducing such errors is provided by the error injection system. Basically, the error injection system provides a program-controlled method of altering the output of the I checker. The operation of the I checker itself is not altered by error injection, except that error indications may be suppressed. As in normal operation, any information (instruction or data) which is passed through the I checker has all possible check bits generated for it.

For example, a data word which is to be stored in main storage is passed through the checker where I unit parity, execution or lookahead parity, and ECC bits are generated for the data word. The appropriate check bits, as determined by the destination of the information, are then transmitted along with the data word. In this example, because the data word is to be routed to storage, the ECC bits are selected as the proper check bits to be routed along with the data word. The error injection system does not alter the data bits at the output of the I checker. Rather, certain of the check bits are inverted under program control. Whether or not error injection is active and, if so, which bits are to be inverted, is program controlled by setting one bits in certain positions of address 3, the upper-lower-bounds register. The following conditions must be met in order to operate the error injection system.

1. The computer must be placed in "maintenance mode" by the toggle switch provided on the maintenance console.

2. The special error injection switch located on the maintenance console must be turned on.

3. Bit 58 of location 3 (the error injection trigger) must be set to one. The error injection trigger may be set under program control while in the maintenance mode. In maintenance mode the trigger may also be turned on and off by pushbuttons on the console.

When the error injection trigger is off, normal programming is permitted. When the trigger is on (only possible in maintenance mode), control bits located in the upper-lower-bounds register designate any errors to be injected at the output of the I checker. The specific functions designated by the control bits are:

Bit Position	If 0	If 1
0	no action	Suppress ECC and I parity error indications from I checker
1	"	Invert: P3(12-15), P11 (0-17) and C0
2	"	Invert: P4(16-23), P12 (18-23) and C1
3	"	Invert: P5 (24-31), P13(24-27), and C2
4	"	Invert: P6 (32-39), P14(28-31), and C4
5	"	Invert: P7 (40-47), P15(32-49), P16(46-49) and C8

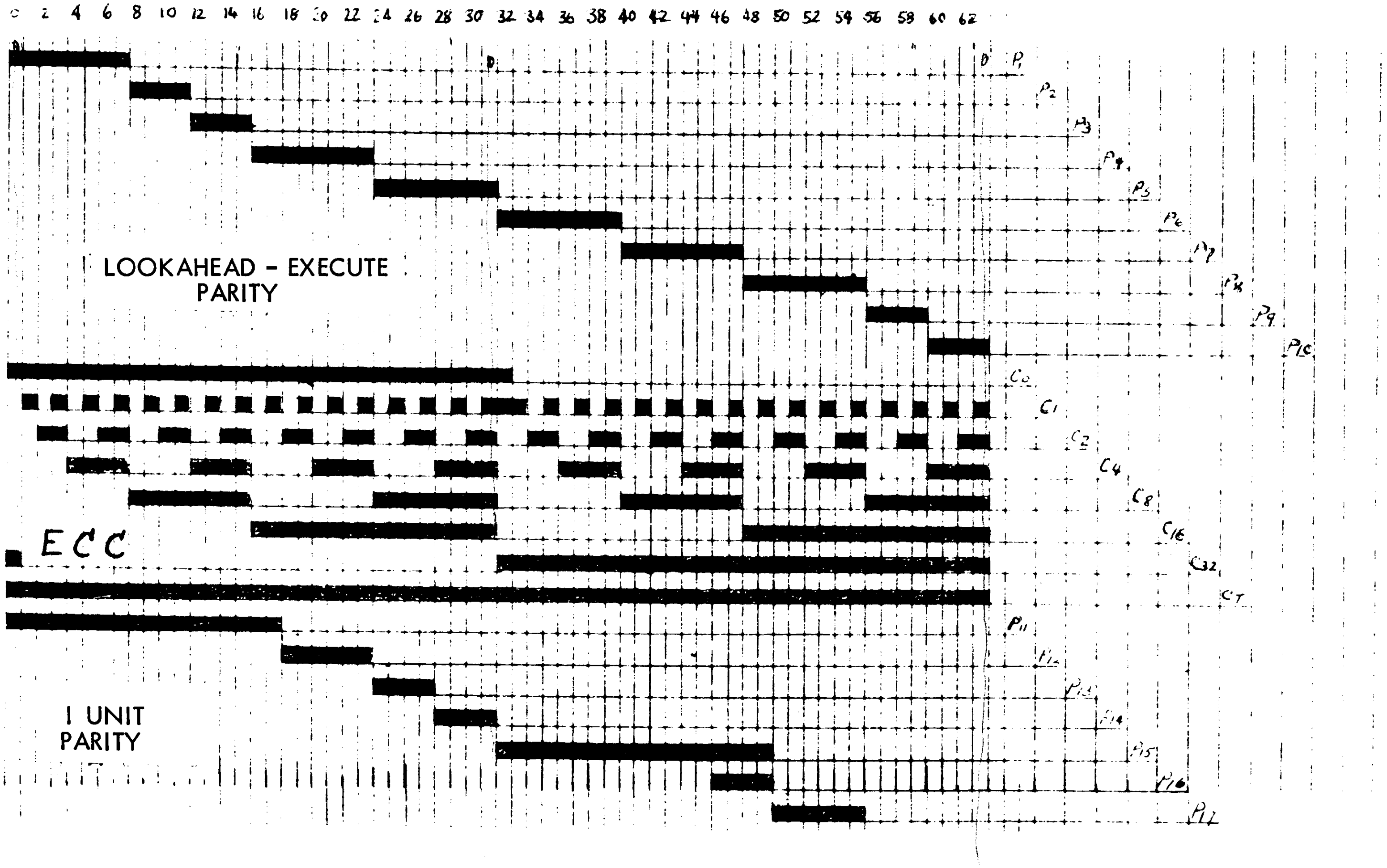
Bit Position	If 0	If 1
6	no action	Invert: P8 (48-55), P17 (50-55), and C16
7	"	Invert: P9 (56-59) and C32
8	"	Invert: P10 (60-63) and CT
58	no error injection	Permits bits 0-8 to become effective to control check bit inversion, thereby providing error injection.

The data bit coverage provided by the various parity and ECC bits is presented in Figure 9-1. The names given to the check bits, such as P13, are not necessarily so labeled in the machine. The bits are labeled here merely for the convenience of reference. Note that no method is provided to invert one type of check bit only. For example, if 3.01 is set to one, P3 (execute parity on positions 12-15), P11 (I unit parity on positions 0-17), and ECC bit C0 are all inverted. The type of error injected into the system depends upon the type of information passing through the I checker.

Note from the above paragraph that a great deal of care must be used when writing a routine for error injection. For example, if a program is to be written to introduce ECC errors into the system, some measure must be taken to insure that the program itself is not injected with errors. Once the control bits and the error injection trigger have been set, instructions and data receive injected errors. A full-word instruction carrying a parity inconsistency is always NOP'ed. A full-word containing two half-word instructions has both instructions NOP'ed only if a parity discrepancy exists on both. If a parity discrepancy exists only on one of the half-word instructions, then only that half-word instruction is NOP'ed; the other is executed properly. Error injection can be turned off under program control by providing a half-word instruction (FP, STORE, etc.) that contains no parity discrepancy. Because instructions having parity errors are NOP'ed , the program to be used to control error injection must be written with extreme care. In the absence of such care, it is possible for the program to lose control of the computer. A point may be reached where all instructions are NOP'ed and error injection must be terminated by use of the switches on the maintenance console. If the program is to maintain control, it is necessary that the instruction which is to terminate error injection be positioned in the program in such a manner that it does not receive a parity error and, therefore, not be NOP'ed.

A second factor that must be considered is the relative position of the program at the time error injection is turned on. A program having an instruction sequence 1, 2, 3, 4, 5, ----n, in the computer usually finds that when 1 is being executed by the arithmetic unit, lookahead may contain 2 and 3, and 4, 5 may be in the I unit undergoing preparation. If instruction 1 is designated to turn on error injection it is apparent that instructions in lookahead are immune to error injection, as may be a portion of the instructions in the I unit.

Error injection is used for two major purposes. One is the generation of data words with incorrect ECC bits for testing the ECC circuits. When used in this manner, data words only are to be affected by error injection. This may be accomplished by insuring that the entire working program is positioned in lookahead and that portion of I unit immune to error injection at the time error injection is turned on. One method of accomplishing this is to follow the instruction which turns on error injection with a TRANSMIT BACKWARD. The TRANSMIT results in the storing of data words with incorrect ECC bits. The last transmission then sets bit 58 of location 3 to zero and thus terminates error injection.



P1 (0-7)
P2 (8-11)
P3 (12-15)
P4 (16-23)
P5 (24-31)
P6 (32-39)
P7 (40-47)

P8 (48-55)
P9 (56-59)
P10 (60-63)
P11 (0-17)
P12 (18-23)
P13 (24-27)

P14 (28-31)
P15 (32-49)
P16 (46-49)
P17 (50-55)

FIGURE 9-1 AREAS OF CHECK BIT COVERAGE

Once a word is stored with the appropriate ECC bits inverted, future use of this word results in single error correction. The data bits may then be interrogated to assure that the checker has inverted the proper data bit. This same procedure may be used when testing core storage. When used for this purpose, the inverted ECC bits represent a single-bit error and any new error introduced in storage results in a double-error indication, rather than being automatically corrected.

A second major use of error injection is the testing of the circuits which check the parity of instructions. When used for this purpose, the error injection system is used to inject deliberate parity errors into some of the programs instructions. The programmer then positions the instructions within the program in such a manner that predetermined instructions are NOP'ed. If any of these instructions avoid detection and are not NOP'ed, the result is affected accordingly. Usually, each instruction which is to be NOP'ed contributes a different amount to the result. Therefore, if the result is other than that expected, the programmer knows not only that an error injected instruction avoided detection, but also which instruction avoided detection. When using error injection in this manner, it is usually desired that error injection be active during the entire execution of the subject program. To assure that all instructions are subject to error injection, the instruction that turns on the error injection trigger may be followed to ten NOP instructions, followed by the program designed to be subject to error injection. A similar approach may be used when turning error injection off. The instruction that turns error injection off may be followed by ten NOP's to assure that the first significant instruction is not NOP'ed.

Extreme care should be exercised in programming the use of error injection. Full consideration should be given to the desired objective, check bits to be inverted, logical approach, and program provided. Error injection is the only mode of operation in which complete control of the system can be lost. Because of the large number of factors that must be considered, it is highly recommended that the interrupt system be disabled before operating error injection.

APPENDIX A

Location		Name	Length	Bit Address	
0		Zero	64	0-63	M
1	P,a	Interval timer	19	0-18	E
1	P,b	Time clock	36	28-63	I
2	P	Interruption address	18	0-17	M
3	P	Upper boundary	18	0-17	T
3	P	Lower boundary	18	32-49	T
3	P	Boundary control bit	1	57	T
4		Maintenance bits	64	0-63	K
5	b	Channel address	7	12-18	T
6		Other CPU	19	0-18	T
7		Left zeros count	7	17-23	T
7		All ones count	7	44-50	T
8		Left half of accumulator	64	0-63	T
9		Right half of accumulator	64	0-63	T
10		Accumulator sign	8	0-7	T
11	c	Indicators	64	0-63	T
12	d	Mask	64	0-63	T
13		Remainder	64	0-63	M
14		Factor	64	0-63	M
15		Transit	64	0-63	M
16-31		Index registers X0 through X15	64	0-63	I

P Permanently protected area of storage.

a Read-only except for STORE VALUE, STORE COUNT, STORE REFILL, and STORE ADDRESS

b Read-only.

c Bit positions 0-19 are read-only.

d Bit positions 0-19 are always ones, and bit positions 48-63 are always zeros.

M External storage location.

I Index core storage.

T Transistor register

K Maintenance keys.

APPENDIX B. PARTIAL LIST OF EXTENDED CHARACTER SET (ECS)

Character	Code		
	Binary Byte	Octal	Hexadecimal
&	0010 0000	40	20
+	0010 0001	41	21
$	0010 0010	42	22
=	0010 0011	43	23
*	0010 0100	44	24
(	0010 0101	45	25
/	0010 0110	46	26
)	0010 0111	47	27
,	0010 1000	50	28
;	0010 1001	51	29
'	0010 1010	52	2A
"	0010 1011	53	2B
a	0010 1100	54	2C
A	0010 1101	55	2D
b	0010 1110	56	2E
B	0010 1111	57	2F
c	0011 0000	60	30
C	0011 0001	61	31
d	0011 0010	62	32
D	0011 0011	63	33
e	0011 0100	64	34
E	0011 0101	65	35
f	0011 0110	66	36
F	0011 0111	67	37
g	0011 1000	70	38
G	0011 1001	71	39
h	0011 1010	72	3A
H	0011 1011	73	3B
i	0011 1100	74	3C
I	0011 1101	75	3D
j	0011 1110	76	3E
J	0011 1111	77	3F
k	0100 0000	100	40
K	0100 0001	101	41
l	0100 0010	102	42
L	0100 0011	103	43
m	0100 0100	104	44
M	0100 0101	105	45
n	0100 0110	106	46

Character	Code		
	Binary Byte	Octal	Hexadecimal
N	0100 0111	107	47
o	0100 1000	110	48
O	0100 1001	111	49
p	0100 1010	112	4A
P	0100 1011	113	4B
q	0100 1100	114	4C
Q	0100 1101	115	4D
r	0100 1110	116	4E
R	0100 1111	117	4F
s	0101 0000	120	50
S	0101 0001	121	51
t	0101 0010	122	52
T	0101 0011	123	53
u	0101 0100	124	54
U	0101 0101	125	55
v	0101 0110	126	56
V	0101 0111	127	57
w	0101 1000	130	58
W	0101 1001	131	59
x	0101 1010	132	5A
X	0101 1011	133	5B
y	0101 1100	134	5C
Y	0101 1101	135	5D
z	0101 1110	136	5E
Z	0101 1111	137	5F
0	0110 0000	140	60
1	0110 0010	142	62
2	0110 0100	144	64
3	0110 0110	146	66
4	0110 1000	150	68
5	0110 1010	152	6A
6	0110 1100	154	6C
7	0110 1110	156	6E
8	0111 0000	160	70
9	0111 0010	162	72
.	0111 0100	164	74
:	0111 0101	165	75
-	0111 0110	166	76
?	0111 0111	167	77

APPENDIX C

SYMBOLIC DESCRIPTIONS AND MNEMONICS FOR IBM 7030

The following list of mnemonics may be used with Strap-1 and Strap-2. A symbolic description of the mnemonic is given to assist the programmer. The operations symbols used are defined at the start of each section. Note that the same letter ("a" and "m" for example) has a different definition for floating point and for VFL. Carefully read the definition for each set. A more detailed description of the operation is in the IBM 7030 Reference Manual, Form A22-6530.

A specific title for each mnemonic is not given in cases where the mnemonic is derived from the basic operation by changing the sign and absolute modifiers.

In the case of VFL operations, the unsigned modifier must be implied by the data referred to or be explicitly stated in a dds.

FLOATING POINT OPERATIONS

Notation for Symbolizing the Floating Point Operations OP(dds), $A_{18}(I)$

Accumulator Operands

a = bits (0-59) of the accumulator, and the accumulator sign, bit 4 of the sign byte register.
b = bits (60-107) of the accumulator, and the accumulator sign.
ab = bits (0-107) of the accumulator, and the accumulator sign.
e(a) = bits (0-11) of a.
f(a) = bits (12-59) of a, and s(a).
s(a) = bit 4 of the sign byte register.
SB(a) = bits 4-7 of the sign byte register.
Fl (a) = bits 5-7 of the sign byte register.

Storage Operands

m = bits (0-59) of the storage word, and its sign, bit 60.
M = L(m) = the effective address.
e(m) = bits (0-11) of m.
f(m) = bits (12-59) of m, and s(m).
s(m) = bit 60 of the storage word.
SB(m) = bits (60-63) of the storage word.
Fl (m) = bits (61-63) of the storage word.

$FT = Factor operand; SB($FT) = bits (60-63) of $FT.
$RM = Remainder operand.

Add

+	$a+m \longrightarrow a$
-	$a-m \longrightarrow a$
+A	$a+\lvert m\rvert \longrightarrow a$
-A	$a-\lvert m\rvert \longrightarrow a$

1. b is unchanged.
2. Fl (a) is unchanged.

Add to Memory

M+	$m+a \longrightarrow m$
M-	$m-a \longrightarrow m$
M+ A	$\lvert m\rvert +a \longrightarrow m$
M- A	$\lvert m\rvert -a \longrightarrow m$

1. Fl (m) remain unchanged.
2. The entire accumulator and SB(a) remain unchanged.

Add to Fraction

F+	$f(ab)+f(m) \longrightarrow f(ab)$
F-	$f(ab)-f(m) \longrightarrow f(ab)$
F+ A	$f(ab)+\lvert f(m)\rvert \rightarrow f(ab)$
F -A	$f(ab)-\lvert f(m)\rvert \rightarrow f(ab)$

1. e(m) is ignored; the add is performed with e(a) on both operands.
2. The normalized mode operates in the same way as in D+.

Add to Exponent

E+	$e(ab)+e(m) \rightarrow \lvert e(ab)$
E-	$e(ab)-e(m) \rightarrow e(ab)$
E+ A	$e(ab)+\lvert e(m)\rvert \rightarrow e(ab)$
E-A	$e(ab)-\lvert e(m)\rvert \rightarrow e(ab)$

1. f(m) is ignored.
2. Strap-1 will assemble as unnormalized unless the normalized mode is requested by referring to normalized data or by using the dds = (N).

Add Immediate to Exponent

E+I	$e(ab)+e(M) \rightarrow e(ab)$
E -I	$e(ab)-e(M) \rightarrow e(ab)$
E+ AI	$e(ab)+\lvert e(M)\rvert \rightarrow e(ab)$
E -AI	$e(ab)-\lvert e(M)\rvert \rightarrow e(ab)$

1. The unnormalized mode is given unless overruled by dds = (N).

Shift Fraction

SHF	$f(ab)\cdot 2^{M} \longrightarrow f(ab)$
SHFN	$f(ab)\cdot 2^{-M} \longrightarrow f(ab)$
SHFA	$f(ab)\cdot 2^{\lvert M\rvert} \rightarrow f(ab)$
SHFNA	$f(ab)\cdot 2^{-\lvert M\rvert} \rightarrow f(ab)$
SHFL	$f(ab)\cdot 2^{\lvert M\rvert} \rightarrow f(ab)$
SHFR	$f(ab)\cdot 2^{-\lvert M\rvert} \rightarrow f(ab)$

1. Left shift if bit 11 of M = 0.
2. Right shift if bit 11 of M = 1.
3. The operation is not affected by the normalized modifier.
4. The exponent is not adjusted for the shift. e(a) is unchanged.
5. On a right shift, zeroes are introduced in bit 12.

Double Add

D+	$ab+m \longrightarrow ab$
D-	$ab-m \longrightarrow ab$
D+ A	$ab+\lvert m\rvert \longrightarrow ab$
D- A	$ab-\lvert m\rvert \longrightarrow ab$

1. PSH indicator goes on if the exponent difference exceeds 48.

Add to Magnitude

+MG	$R=\lvert a\rvert+m$
-MG	$R=\lvert a\rvert-m$
+MGA	$R=\lvert a\rvert+\lvert m\rvert$
-MGA	$R=\lvert a\rvert-\lvert m\rvert$

1. $R \rightarrow a$ if $R \geq 0$.
2. $0 \rightarrow f(a)$ if $R < 0$ and e(a) is unchanged.
3. s(a) is unchanged in either case.

Double Add to Magnitude

D+MG	$R=\lvert ab\rvert+m$
D-MG	$R=\lvert ab\rvert-m$
D+MGA	$R=\lvert ab\rvert+\lvert m\rvert$
D-MGA	$R=\lvert ab\rvert-\lvert m\rvert$

1. $R \rightarrow ab$ if $R \geq 0$.
2. $0 \rightarrow f(ab)$ if $R < 0$ and e(a) is unchanged.
3. s(a) is unchanged in either case.

Add Magnitude to Memory

M+MG	$R=m+\lvert a\rvert$
M-MG	$R=m-\lvert a\rvert$
M+MGA	$R=\lvert m\rvert+\lvert a\rvert$
M-MGA	$R=\lvert m\rvert-\lvert a\rvert$

1. $R \rightarrow m$ if $s(R)=s(m)$.
2. $0 \rightarrow f(m)$ if $s(R) \neq s(m)$.
3. s(m) is unchanged in either case.

Multiply

*	$a\cdot m \longrightarrow a$
*N	$a\cdot -m \longrightarrow a$
*A	$a\cdot \lvert m\rvert \longrightarrow a$
*NA	$a\cdot -\lvert m\rvert \longrightarrow a$

1. b in unchanged.

Double Multiply

D*	$a\cdot m \longrightarrow ab$
D*N	$a\cdot -m \longrightarrow ab$
D*A	$a\cdot \lvert m\rvert \longrightarrow ab$
D*NA	$a\cdot -\lvert m\rvert \longrightarrow ab$

1. (108-127) of accumulator are unchanged.

Multiply Factor and Add

*+	m· ($FT)+ ab → ab
*N+	-m· ($FT)+ ab → ab
*A+	|m| · ($FT)+ ab → ab
*NA+	- |m| · ($FT)+ ab → ab

1. The contents of $FT remain unchanged.

Divide

/	a/m → a
/N	a/-m → a
/A	a/ |m| → a
/NA	a/- |m| → a

1. No remainder is generated.
2. Quotient is 48 bits.
3. Pre-normalization of the operands is independent of the normalization modifier.
4. b is unchanged.

Reciprocal Divide

R/	m/a → a
R/N	-m/a → a
R/A	|m| /a → a
R/NA	- |m| /a → a

1. Performed similarly to divide.
2. 0 → (60-63)b; the rest of b is unchanged.

Double Divide

D/	ab/m → ab
D/N	ab/-m → ab
D/A	ab/ |m| → ab
D/NA	ab/- |m| → ab

1. Remainder in $RM
2. 0 b except bit 60, which contains a continuation of f (a).
3. No rounding.
4. SB(a) SB($RM).
5. Result capable of being rounded in a subsequent instruction.

Store Root

SRT	$\sqrt{a}$ → m
SNRT	$-\sqrt{a}$ → m
SRTA	$\sqrt{\lvert a \rvert}$ → m
SNRTA	$-\sqrt{\lvert a \rvert}$ → m

1. ab and SB(a) are unchanged.

Load

L	m → a
LN	-m → a
LA	|m| → a
LNA	- |m| → a

1. 0 → Fl (a).
2. b is unchanged.

Double Load

DL	m → a
DLN	-m → a
DLA	|m| → a
DLNA	- |m| → a

1. 0 → b.
2. 0 → Fl (a).

Load with Flag Bits

LWF	m → a
LWFN	-m → a
LWFA	|m| → a
LWFNA	- |m| → a

1. Fl (m) → Fl (a).

Double Load with Flag Bits

DLWF	m → a
DLWFN	-m → a
DLWFA	|m| → a
DLWFNA	- |m| → a

1. 0 → b.
2. Fl (m) → Fl (a).

Load Factor

LFT	m → $FT
LFTN	-m → $FT
LFTA	|m| → $FT
LFTNA	- |m| → $FT

1. ab and SB(a) are not changed.
2. s(m) → (60)$FT.
3. 0 → (61-63)$FT.

Store

ST	a → m
STN	-a → m
STA	|a| → m
STNA	- |a| → m

1. Fl (a) → Fl (m).
2. a is unchanged.

Store Rounded

SRD	a → m
SRDN	-a → m
SRDA	|a| → m
SRDNA	- |a| → m

1. A one is added in bit (60)b prior to the store: a and (60)b are unchanged.
2. Fl (a) → Fl (m).

Store Low Order

SLO	b → f(m)
SLON	-b → f(m)
SLOA	|b| → f(m)
SLONA	- |b| → f(m)

1. e(a) - 48 → e(m).
2. Fl (a) → Fl (m).
3. e(a) is unchanged.

Compare

K	a:m
KN	a:-m
KA	a:|m|
KNA	a:- |m|

1. Indicators AL, AE, and AH are set as follows:
   AL is set to one if a < m
   AE is set to one if a = m
   AH is set to one if a > m
2. Zero exponents of different sign are considered equal.
3. If the exponent difference is 48 the larger of the numbers is per sign and exponents regardless of fractions.

Compare for Range

KR	a:m
KRN	a:-m
KRA	a:|m|
KRNA	a:- |m|

1. If AH is off prior to this op, no indicators will be changed.
2. If AH is on:
   AL is unchanged.
   AE is set to one if a < m.
   AH is set to one if a ≥ m.

Compare Magnitude

KMG	a :m
KMGN	a :-m
KMGA	a :|m
KMGNA	a :- |m|

1. Same as Compare, except for accumulator comparand.

Compare Magnitude for Range

KMGR	a :m
KMGRN	a :-m
KMGRA	a : |m|
KMGRNA	a :- |m|

1. Same as Compare for Range, except for accumulator comparand.

VARIABLE FIELD LENGTH OPERATIONS

Notation for Symbolizing the Variable Field Length Operations OP(dds), A_{24}(I), OF_7(I')

Accumulator Operands

a = the accumulator operand whose:

1. Low order bit is defined by the offset;
2. Byte size is four for decimal arithmetic, eight for binary arithmetic;
3. Length includes all bits in the accumulator to the left of the offset;

4. Sign is indicated by bit four of the sign byte register.

$\bar{a}$ = the accumulator operand, a, but without sign.

a_{20} = the accumulator operand, a, with offset = 20.

Storage Operands

m = the storage operand whose:

1. High-order bit is defined by the bit address;
2. Byte size may be any number from one to eight, but is assumed to be four in the instruction lists below;
3. Length is defined by the field length in the dds;
4. Sign is bit s in the sign byte.

$\bar{m}$ = the storage operand in which all bytes are processed as data; a positive sign is assumed.

The unsigned storage operand is designated by the dds.

Bits 7.17 and 7.18 are the leftmost two bits of $LZC.

$FT = Factor Operand; s($FT) = bit 60; FL($FT) = bits (61-63).

$TR = 64-bit Transit Register.

Integer Operations

Operations which can have an immediate operand are followed by (I), except for *+.

Add

+	$a+m \longrightarrow a$	(I) 1. If the sign changes, bits to the right of the offset are complemented.
-	$a-m \longrightarrow a$	

Add To Memory

M+	$m+a \longrightarrow m$
M-	$m-a \longrightarrow m$

Add to Magnitude

+MG	$R=\bar{a}+m$	(I) 1. $R \rightarrow \bar{a}$ if $R \geq 0$. 2. 0→entire accumulator if $R < 0$. 3. s(a) is not changed by these operations.
-MG	$R=\bar{a}-m$	

Add Magnitude To Memory

M+MG	$R=m+\bar{a}$	1. R→m if s(R) = s(m). 2. 0→m if s(R) ≠ s(m). 3. s(m) is not changed.
M-MG	$R=m-\bar{a}$	

Multiply

*	$a \cdot m \longrightarrow a_{20}$	(I) 1. Multiplication takes place only if mode = B or BU. 2. The decimal mode gives LTRS and 00_2 to bits 7.17 and 7.18. 3. The length of a or m must be ≤ 48 bits in binary multiply. 4. The portion of the accumulator not containing the product is set to zero.
*N	$a \cdot -m \longrightarrow a_{20}$	

Multiply Factor and Add

*+	$m \cdot (\$FT)+a \rightarrow a$	(I) 1. Write: *I+ and *NI+ for an immediate operand. 2. Multiplication takes place only if mode = B or BU. 3. Decimal mode gives LTRS and 10_2 to bits 7.17 and 7.18.
*N	$-m \cdot (\$FT)+a \rightarrow a$	

Divide

/	$a/m \longrightarrow a$	(I) 1. Divide takes place only in the binary mode. 2. Decimal divide gives LTRS and 01_2 in bits 7.17 and 7.18. 3. The remainder is placed in $RM. The remainder sign, (60) $RM, is the same as the original s(a). Fl ($RM) = 0. 4. Bits to the right of the offset are cleared.
/N	$a/-m \longrightarrow a$	

Load

L	$m \longrightarrow a$	(I) 1. 0→Fl(a). 2. The entire accumulator is cleared before the load.
LN	$-m \longrightarrow a$	

Load with Flag Bits

LWF	$m \longrightarrow a$	(I) 1. Fl(m)→Fl(a).
LWFN	$-m \longrightarrow a$	

Load Factor

LFT	$m \longrightarrow \$FT$	(I) 1. 0→(61 - 63) $FT. 2. The offset field is ignored.
LFTN	$-m \longrightarrow \$FT$	

Load Transit and Set

LTRS	$m \longrightarrow \$TR$	(I) 1. Offset→$AOC. 2. 11_2→bits 7.17 and 7.18. 3. Indicator $BTR = 1 and $DTR = 0 if mode is B or BU. Indicator $DTR = 1 and $BTR = 0 if mode is D or DU.
LTRSN	$-m \longrightarrow \$TR$	

Store

ST	$a \longrightarrow m$	1. SB(a)→SB(m). 2. If the byte size is greater than four: Binary: zone bits of the sign byte register are stored in SB(m). Decimal: zone bits of the sign byte register are stored in each byte of m.
STN	$-a \longrightarrow m$	

Store Rounded

SRD

SRDN

These operations are the same as the corresponding Store operations, except for:

a. Binary: a one is added one bit to the right of the offset, prior to the store.

b. Decimal: 0101 is added one byte to the right of the offset, prior to the store.

c. The accumulator is unchanged, even if rounding occurs.

Add One to Memory

M+1	$m+1 \longrightarrow m$	1. The one is added to the low order byte. 2. The offset field is ignored.
M-1	$m-1 \longrightarrow m$	

Compare

K a:m
KN a:-m

(I) 1. The Compare operations set the AL, AE, and AH indicators.
AL is set to one if: $a < m$
AE is set to one if: $a = m$
AH is set to one if: $a > m$
2. All bits to the left of the offset in the accumulator participate in the compare.

Compare for Range

KR a:m
KRN a:-m

(I) 1. If the AH indicator is off prior to the operation, it is executed as a NOP.
2. If AH is on:
AL is unchanged.
AE is set to one if $a < m$
AH is set to one if $a \geq m$

Compare If Equal

KE a:m
KEN a:-m

(I) 1. If the AE indicator is off, no changes will occur.
2. If the AE indicator is on, the indicators are set as in Compare, K.

Compare Field

KF $\bar{a}$:m
KFN $\bar{a}$:-m

(I) 1. The indicators are set as in Compare.
2. The length of the accumulator comparand is the same as the length of the storage comparand.
3. The matching bits of both operands are compared.

Compare Field for Range

KFR $\bar{a}$:m
KFRN $\bar{a}$:-m

(I) 1. The accumulator comparand is the same as in Compare Field, KF.
2. The indicators are set as in Compare Range, KR.

Compare Field If Equal

KFE $\bar{a}$:m
KFEN $\bar{a}$:-m

(I) 1. The accumulator comparand is the same as in Compare Field, KF.
2. The indicators are set as in Compare If Equal, KE.

Logical Connectives OP(dds), A_{24} (I), OF_7 (I')

Note: If the operand from storage has a byte size (BS) less than eight, then eight minus BS (8 - BS) leading zeros are added to each byte from storage before the connect takes place. However, the storage operand is not changed in Cxxxx or CTxxxx.

Connect to Accumulator

$Cx_1x_2x_3x_4$ Result → a

Connect to Memory

$CMx_1x_2x_3x_4$ Result → m

Connect for Test

$CTx_1x_2x_3x_4$ Result is not stored.

$x_1x_2x_3x_4$ is a four-bit binary configuration to describe the type of connective; it is summarized:

Let: m = a bit from storage (may be an inserted leading zero if the byte size is less than 8).
a = a bit from the accumulator corresponding to m. The accumulator byte size always = 8.
x_1 = desired result if m = 0 and a = 0
x_2 = desired result if m = 0 and a = 1
x_3 = desired result if m = 1 and a = 0
x_4 = desired result if m = 1 and a = 1
Example: C1010 (BU, 64, 4), 0 will complement the entire 128-bit accumulator.

Pseudo-Connectives

LF	(Load Field)	LF = C0011
SF	(Store Field)	SF = CM0101

Immediate Connects

To indicate immediate addressing, write: $CIx_1x_2x_3x_4$, $CTIx_1x_2x_3x_4$, and LFI.

$AOC = All ones count register.
$LZC = Left zeros count register.

After a connective operation the two registers, $AOC and $LZC contain the indicated counts of the result. Because the result may not occupy the entire accumulator, $AOC and $LZC may not give the total count of ones and left zeros of the accumulator. However, these counts always give the correct count in CM or SF.

Convert Instructions

Definitions:
a_D = accumulator in decimal, four-bit bytes with specified offset.
a_B = accumulator in binary with specified offset.
a_{B20} = accumulator in binary with offset = 20.
a_{B68} = accumulator in binary with offset = 68.
m_B = storage operand in binary with specified byte size and field length.
m_D = storage operand in decimal with specified byte size and field length.
$TR = 64-bit transit register with a sign byte in the rightmost four bits.

Note: The conversion goes: from decimal to binary if the mode given is decimal; from binary to decimal if the given mode is binary.

Convert

CV $a_D \longrightarrow a_{B68}$ if mode = D or DU
or $a_{B68} \longrightarrow a_D$ if mode = B or BU
CVN $-a_D \longrightarrow a_{B68}$
or $-a_{B68} \longrightarrow a_D$

1. In binary a field of 48 bits is used.
2. The entire accumulator to the left of the offset is used.

Double Convert

DCV $a_D \longrightarrow a_{B20}$
or $a_{B20} \longrightarrow a_D$
DCVN $-a_D \longrightarrow a_{B20}$
or $-a_{B20} \longrightarrow a_D$

1. In binary, a field of 96 bits is used.
2. The entire accumulator to the left of the offset is used.

Load Converted

LCV	m_D	a_B	(I)	1. s(m) → s(a) 2. 0 → Fl(a) 3. The entire accumulator is cleared before the load.
or	m_B	a_D		
LCVN	$-m_D$	a_B	(I)	
or	$-m_B$	a_D		

Load Transit Converted

LTRCV	m_D	$\$TR_B$ (I)	1. The accumulator and offset are ignored. 2. 0 → Fl($TR) 3. s(m) → s($TR) 4. The entire $TR is cleared before the load.
or	m_B	$\$TR_D$	
LTRCVN	$-m_D$	$\$TR_B$ (I)	
or	$-m_B$	$\$TR_D$	

Progressive Indexing

Any VFL or Connective operation (when not immediate) may have a second operation enclosed in parentheses. The second operation may be V ± I, V ± IC, or V ± ICR.

Format: OP(OP_2)(dds), A_{24} (J), OF_7 (I')

Notes: 1. The original value field of J is the effective address of operation.

2. A_{24} is the immediate operand specified by J in V ± I, and so on, and the value field of J is incremented by ± A_{24} according to ± I. The incrementing takes place subsequent to note 1.

3. J may be $XO.

INDEXING OPERATIONS

Notation for symbolizing the Indexing Operations

Index Word Operands

J = bits (0 - 63) of the index word.
V = bits (0 - 24) of J.
C = bits (28 - 45) of J.
R = bits (46 - 63) of J.

Storage Word Operands

m = bits (0 - 63) of a storage word.
V(m) = bits (0 - 24) of m if the second operand is V.
(sign of V is in bit 24)
V(m) = bits (0 - 17) of m if the second operand is C or R.

Immediate Operands

m = bits (0 - 18) of the effective address if the second operand is V.
m = bits (0 - 17) of the effective address if the second operand is C or R.

Notes: 1. For clarity, the titles to the indexing and the branch operations have been omitted.

2. The indicators XF, XCZ, XVLZ, XVZ, and XVGZ are set by all of the direct and immediate index operations except KV, KC, KVI, KVNI, and KCI. These indicators are set before the refill (if any) takes place.
KV, KC, . . . ,KCI set the index compare indicators XL, XE, and XH.

Direct Index Arithmetic OP, J, A_{19} (I)

LX	m → J	A_{18}
LV	V(m) → V	1. M = A_{19} (I) 2. m = (M) 3. C_2 = The count field of J after modification
LC	V(m) → C	
LR	V(m) → R	

SX	J → m	1. A_{18}
SV	V → V(m)	
SC	C → V(m)	1. 0 → (18 - 24) of m.
SR	R → V(m)	1. 0 → (18 - 24) of m.
V+	V+V(m) → V	1. There is no V - etc.
V+C	V+V(m) → V C-1 → C_2	
V+CR	V+V(m) → V C-1 → C_2 (R) → (J) if C_2 = 0	
SVA	V → V(m)	1. V is truncated to 18, 19, or 24 bits, as is appropriate for the instruction containing V(m).
LVE	$(M)^n$ → V	1. (M) means contents of M $(M)^1$ " " " (M) . . . $(M)^n$ " " " $(M)^{n-1}$
KV → V:V(m) KC → C:V(m)		1. Indicators: XL, XE, XH are set by KV and KC. This setting is the only output of KV and KC.
RNX	J → (R($XO)) M → R($XO) m → J	1. Used for saving and restoring index registers.
LVS	(special format):	LVS, J, A^1, A^2, . . . ,A^n

$$\sum_{i=i}^{n} V(A^i) \longrightarrow V(J)$$

1. The sum may include any subset of the index words, each one appearin no more than once.
2. No indexing of the address field is allowed.

Immediate Index Arithmetic OP, J, A_{19}

Notes: 1. None of the immediate index instructions allow for indexing of the address. A_{19} is the effective address and is represented by A below.

2. The output of KVI, KVNI, and KCI is the setting of indicators XL, XE, and XH.

LVNI	-A → V	1. (19 - 24) of V are set to 0.	A is appended by 5 zero bits for the operation.
LVI	A → V	1. (19 - 24) of V are set to 0.	
LCI	A → C		
LRI	A → R		
V+I	V+A → V	1.	
V-I	V-A → V	1.	
V+IC	V+A → V C-1 → C	1.	
V-IC	V-A → V C-1 → C	1.	
V+ICR	V+A → V C-1 → C_2 (R) → (J) if C_2 = 0	1.	
V-ICR	V-A → V C-1 → C_2 (R) → (J) if C_2 = 0	1.	
C+I	C+A → C_2		
C-I	C-A → C_2		
KVI	(0 - 18) of V:A	1. (19 - 24) of V are compared with zeros.	
KVNI	(0 - 18) of V:A	1. (19 - 23) of V are compared with zeros and (24) of V is compared with 1 (minus).	
KCI	C:A		

Count and Branch Operations OP, J, B_{19} (K)

CB	$C_1 - 1 \longrightarrow C_2$ $IC_1 + 0.32 \longrightarrow IC$ if $C_2 = 0$ $M \longrightarrow IC$ if $C_2 \neq 0$
CBR	$C_1 - 1 \longrightarrow C_2$ $IC_1 + 0.32 \longrightarrow IC$ and $(R) \rightarrow (J)$ if $C_2 = 0$ $M \longrightarrow IC$ if $C_2 \neq 0$
CBZ	$C_1 - 1 \longrightarrow C_2$ $IC_1 + 0.32 \longrightarrow IC$ if $C_2 \neq 0$ $M \longrightarrow IC$ if $C_2 = 0$
CBRZ or CBZR	$C_1 - 1 \longrightarrow C_2$ $IC_1 + 0.32 \longrightarrow IC$ if $C_2 \neq 0$ $M \longrightarrow IC$ and $(R) \rightarrow (J)$ if $C_2 = 0$

1. K may be only 0 or 1.
2. M=the effective address of B_{19} (K).
3. IC_1 is the value of the instruction counter where the CB instruction is located.
4. C_1 and C_2 are the count field of J before and after the count portion of the instruction, respectively.

Note: In addition to the stated functions, the value field of J may be modified by placing + , - , or H after the above mnemonics. The modification of V takes place regardless of C_2 and before the refill (if any).

Example: In addition to the given functions of CB, we have:

CB	leave V alone
CB+	$V + 1.0 \rightarrow V$
CB-	$V - 1.0 \rightarrow V$
CBH	$V + 0.32 \rightarrow V$

Unconditional Branch Operations: OP, A_{19} (I)

B	M	$\longrightarrow IC$
BR	$M + IC_1 + 0.32$	$\longrightarrow IC$
BE	Enable M	$\longrightarrow IC$
BD	Disable M	$\longrightarrow IC$
BEW	Enable M Wait	$\longrightarrow IC$
NOP	$IC_1 + 0.32$	$\longrightarrow IC$

1. The unconditional branch instructions are the only branch instructions which allow a 4 bit index field, I. The conditional branch instructions may have only a 1-bit index field, K.
2. IC_1 is the value of the instruction counter where the instruction is located (i.e., the leftmost bit of the instruction).

Branch on Bit Operations: OP, A_{24} (I), B_{19} (K)

BB	$IC_1 + 0.32 \rightarrow IC$ if $m_1 = 0$ $M_2 \longrightarrow IC$ if $m_1 = 1$
BZB	$IC_1 + 1.0 \rightarrow IC$ if $m_1 = 1$ $M_2 \longrightarrow IC$ if $m_1 = 0$

1. $m_1 = (A_{24}(I))$, the bit being tested.
2. $M_2 = B_{19}(K)$, the branch address.
3. K=0 or 1; I=1 -15.

Note: The BB and BZB may have a suffix, Z, 1, or N, which, respectively, will set m_1 to zero or to one, or negate it. This function is independent of the success of the branch. For example, the following branch on bit instructions are permissible and perform the stated functions as well as:

BB	BZB	leave m_1 alone
BBZ	BZBZ	$0 \rightarrow m_1$
BB1	BZB1	$1 \rightarrow m_1$
BBN	BZBN	$-m_1 \rightarrow m_1$

Branch on Indicator Operations BIND, B_{19} (K)

BIND	$IC_1 + 0.32 \rightarrow IC$ if ind. = 0 $M \longrightarrow IC$ if ind. = 1
BZIND	$IC_1 + 0.32 \rightarrow IC$ if ind. = 1 $M \longrightarrow IC$ if ind. = 0

1. The indicators may not be set to 1 or negated with a BIND operation.

Notes: 1. The letters "IND" in BIND are replaced by the appropriate indicator mnemonics as shown in note 2 below.

2. The above operations can have a suffix, Z, which will cause the indicator being tested to be set to zero independently of the success of the branch. For example, BZXPOZ will set indicator XPO to zero arbitrarily. We may have: BXPO; BZXPO; BXPOZ; and BZXPOZ. The following list includes all of the indicator mnemonics which may be used in BIND, B_{19}(K), and their bit addresses.

Mnemonic	Name	Bit Address
	EQUIPMENT CHECK	
MK	Machine Check	11.0
IK	Instruction Check	11.1
IJ	Instruction Reject	11.2
EK	Exchange Control Check	11.3
	ATTENTION REQUEST	
TS	Time Signal	11.4
CPUS	CPU Signal	11.5
	INPUT-OUTPUT REJECTS	
EKJ	Exchange Check Reject	11.6
UNRJ	Unit Not Ready Reject	11.7
CBJ	Channel Busy Reject	11.8
	INPUT-OUTPUT STATUS	
EPGK	Exchange Program Check	11.9
UK	Unit Check	11.10
EE	End Exception	11.11
EOP	End of Operation	11.12
CS	Channel Signal	11.13
	(not available)	11.14
	INSTRUCTION EXCEPTION	
OP	Operation Invalid	11.15
AD	Address Invalid	11.16
USA	Unended Sequence of Addresses	11.17
EXE	Execute Exception	11.18
DS	Data Store	11.19
DF	Data Fetch	11.20
IF	Instruction Fetch	11.21
	RESULT EXCEPTION	
LC	Lost Carry	11.22
PF	Partial Field	11.23
ZD	Zero Divisor	11.24
	RESULT EXCEPTION-FLOATING POINT	
IR	Imaginary Root	11.25
LS	Lost Significance	11.26
PSH	Preparatory Shift Greater than 48	11.27
XPFP	Exponent Flag Positive	11.28
XPO	Exponent Overflow	11.29
XPH	Exponent High	11.30
XPL	Exponent Range Low	11.31
XPU	Exponent Underflow	11.32
ZM	Zero Multiply	11.33
RU	Remainder Underflow	11.34

	FLAGGING	
TF	T Flag	11.35
UF	U Flag	11.36
VF	V Flag	11.37
XF	Index Flag	11.38
	TRANSIT OPERATIONS	
BTR	Binary Transit	11.39
DTR	Decimal Transit	11.40
	PROGRAMMER INDICATORS	
PGO or PG		11.41
PG1		11.42
PG2		11.43
PG3		11.44
PG4		11.45
PG5		11.46
PG6		11.47
	INDEX RESULT	
XCZ	Index Count Zero	11.48
XVLZ	Index Value Less than Zero	11.49
XVZ	Index Value Zero	11.50
XVGZ	Index Value Greater Than Zero	11.51
XL	Index Low	11.52
XE	Index Equal	11.53
XH	Index High	11.54
	ARITHMETIC RESULT	
MOP	To-Memory Operation	11.55
RLZ	Result Less than Zero	11.56
RZ	Result Zero	11.57
RGZ	Result Greater than Zero	11.58
RN	Result Negative	11.59
AL	Accumulator Low	11.60
AE	Accumulator Equal	11.61
AH	Accumulator High	11.62
	MODE	
NM	Noisy Mode	11.63

TRANSMIT OPERATIONS: OP, J, $A_{18}(I)$, $A'_{18}(I')$

Notes:
1. Full words are transmitted in all Transmit and Swap instructions.
2. In the immediate operations, J is the count of the number of full words transmitted. J must be ≤ 16. If J = 0, 16 words are transmitted.
3. In the others (the direct transmission) the count field of J has the number of full words to be transmitted.

Transmit Forward

T	$(M_1) \longrightarrow (M_2)$ $(M_1+1) \longrightarrow (M_2+1)$ etc.	1. M_1 is the effective address of A_{18} (I) 2. M_2 is the effective address of $A'_{18}(I')$

Transmit Forward Immediate

TI	$(M_1) \longrightarrow (M_2)$ $(M_1+1) \longrightarrow (M_2+1)$ etc.	

Transmit Backward

TB	$(M_1) \longrightarrow (M_2)$ $(M_1-1) \longrightarrow (M_2-1)$ etc.	1. Both blocks are referred to in a backward direction.

Transmit Backward Immediate

TBI	$(M_1 \longrightarrow (M_2)$ $(M_1-1) \longrightarrow (M_2-1)$ etc.	

Swap Forward

SWAP	$(M_1) \longleftrightarrow (M_2)$ $(M_1+1) \longleftrightarrow (M_2+1)$ etc.

Swap Forward Immediate

SWAPI	$(M_1) \longleftrightarrow (M_2)$ $(M_1+1) \longleftrightarrow (M_2+1)$ etc.

Swap Backward

SWAPB	$(M_1) \longleftrightarrow (M_2)$ $(M_1-1) \longleftrightarrow (M_2-1)$ etc.

Swap Backward Immediate

SWAPBI	$(M_1) \longleftrightarrow (M_2)$ $(M_1-1) \longleftrightarrow (M_2-1)$ etc.

MISCELLANEOUS OPERATIONS: OP, $A_{19}(I)$

Store Instruction Counter If

SIC	$IC_1+1.0 \longrightarrow (0-18)$ of $A_{19}(I)$ if the following half word branch instruction is executed.	1. SIC; NOP will not store the IC.

Refill

R	$(R_M) \longrightarrow (M)$	1. R_M=refill field of word M

Refill If Count Is Zero

RCZ	$(R_M) \longrightarrow (M)$ if C field of M = 0	

Execute

EX	Execute $\longrightarrow$ (M)	1. The instruction located at M is executed. 2. Control then goes to the instruction following EX.

Execute Indirect and Count

EXIC	Execute $(M)^1$ $(M) + 1 \longrightarrow (M)$	1. The instruction whose address is located in M is executed.

Store Zero

Z	$0 \longrightarrow (M)$	1. Full word of zeros.

INPUT-OUTPUT INSTRUCTIONS: OP, $A_7(I)$, $A_{18}(I')$

Locate

LOC

Select Unit

SU

A_7 (I) represents a channel address; $A_{18}(I')$ represents:
1. The address of one of several units attached to channel $A_7(I)$; in this case LOC or SU must be given before a RD or W addressing this channel;
2. An address on the disk specified by $A_7(I)$. LOC=SU.

Read

RD — $A_7(I)$ represents a channel address; a reading operation in initiated for this channel (or for a unit attached to this channel if more than one unit is available and has been readied by a LOC instruction). $A_{18}(I')$ is the address of a control word.

Write

W — Initiates a writing operation. Analogous to RD except that the skip flag of the control word is ignored.

Release

REL — Immediately terminates any operation in progress at the unit specified in $A_7(I)$, the channel address, or in the last unit at $A_7(I)$ selected by a LOC instruction, if $A_7(I)$ consists of more than one unit.

Copy Control Word

CCW — The current control word corresponding to the addressed channel $A_7(I)$ is sent to $A_{18}(I')$.

LOC(SEOP)
RD(SEOP)
W(SEOP)
REL(SEOP)
CTL(SEOP)
SU(SEOP) — Same as LOC, SU, RD, W, REL, CTL except the SEOP bit in control word is set to 1; thus, program interruption on completion of an operation is suppressed, provided no exception conditions, such as unit check and end exception, are encountered.

Control

CTL — Initiates performance of certain functions at the channel indicated by $A_7(I)$, or at the last unit selected by a LOC instruction. The functions are indicated:

General I-O Unit (Standard for $A_{18}(I')$)

$A_{18}(I')$ =	0	Reserved light off
	1	Reserved light on
	2	Read-write check light on
	4	ECC mode
	5	No ECC mode

Card Reader and Card Punch
Standard, except $A_{18}(I')$ = 2 also causes a card to be offset in the stacker.

Tape Units
Standard, but in addition:

$A_{18}(I')$ =	4	ECC mode, odd parity
	5	No ECC mode, odd parity
	6	No ECC mode, even parity
	7	Rewind tape
	8	Space block (record)
	9	Backspace block (record)
	10	Space file
	11	Backspace file
	12	Write tape mark (E. O. F. mark)
	13	Erase long gap
	14	High-density mode (556 bits/inch)
	15	Low-density mode (200 bits/inch)

Inquiry Station, Printer, Console
Standard, except codes 4 and 5 are missing. On console, $A_{18}(I')$ = 3 causes the gong to sound.

APPENDIX D. CE INSTRUCTION MANUAL ABBREVIATIONS FOR 7030

A,B,C,D	Basic Registers
A-A-A-A	Controlled A Pulses
A-B-A-B	Controlled AB Pulses
AAAA	Free Running A Pulses
ABA	Adder Bus A
ABAB	Free Running AB Pulses
ABB	Adder Bus B
ABC	Arithmetic Bus Counter
AB REG	Accumulator Register
ABS	Absolute
ACC	Accept, Accumulator
ACC T	Access Time
ACCY	Accuracy
ACIB	Arithmetic Checker In Bus
ACOB	Arithmetic Checker Out Bus
ACPT	Accept
AD	Address Invalid
ADDR	Adder
ADR	Address
ADV	Advance
ADV EN SEQ	Advance Enables Sequence
AE	Accumulator Equal
AES	Advance Enables Sequence
AH	Accumulator High
AIC	Adder Input Checker
AL	Accumulator Low
ALD	Automated Logic Diagram
AM0	Group A Storage Zero
AM1	Group A Storage One
A TIMER	Arithmetic Bus Timer
AOB	Adder Out Bus
AR	Address Register
ARITH	Arithmetic
ANTIC	Anticipation
ATCB	Allow Time Clock Break In
AVAIL	Available
B-B-B-B	Controlled B Pulses
BA	Bit Address
BAAB	Bit Address-AB Register
BACD	Bit Address-CD Register
BBBB	Free Running B Pulses
BC	Bus Control
BCD	Binary Coded Decimal
BC1Y	Block Check 1 Y
BC2Y	Block Check 2 Y
BEW	Branch Enable and Wait
BFR	Buffer
BI	Branch Indicator Instruction
BIT	Bit
BM2	Group B Storage Two
BNDS	Boundaries, Bounds
BP	Break Point
BR	Branch
BRCY	Branch Recovery Operation
BROK	Branch Successful Trigger
BS	Byte Size
BT	Byte
BTR	Byte Register
BUS	Buses
BX	IBM 7619 Exchange
CAR	Carry
CB	Circuit Breaker
CBJ	Channel Busy Reject
CCW	Copy Control Word
CD2Y	Condition 2Y
CD YR	Condition Y Right
CDZ DEC	Condition Z Decode
CF	Chain Flag
CH	Channel
CH ADR	Channel Address
CHAR REG	Character Register
CHK	Check; Checker Cycle Control Sequencer
CHK LT	Check Light
CHKR	Checker
CK	Check
CKT	Circuit
CL	Clear
CMD	Cumulative Multiplicand
CNIDC	Condition Nonidentifiable Machine Check Indicator
CNSL	Console
CNT 1	Count Sequences
CNTR	Counter
COMP	Complement
COND	Condition
CP	Clock Pulse
CPU	IBM 7101 Central Processing Uni
CPU S00T019	CPU Other Signal Triggers (Signal 00 to 19)
CS	Channel Signal
CSA	Carry Save Adder
CTR	Counter
CTRL	Control
CTRL CKT	Control Circuits
CW	Control Word
CW MOD 1 CY	Control Word Modification 1.2 Cycle
CWA	Control Word Address
CWTC	Control Word Type Cycle
CY	Cycle
CZRF	Count Zero Refill Trigger
DBL	Double
DC	Direct Current
DCL	Die Card Lever

APPENDIX D. (Cont'd)

DCR	Decoder
DCU	Disk Synchronizer; Deskewing Unit
DEC	Decode
DEC 4	Decode 4 Sequencer
DEL	Delay
DF	Data Fetch
DISC	Disconnect
DISK SNC	Disk Synchronizer
DISPL	Display
DLY	Delay
DLYD	Delayed
DRVR	Driver
DS	Data Store
DVD	Dividend
DVR	Divisor
DW	Data Word
DWA	Data Word Address
DWAR	Data Word Address Register
DZT	Digit Zero Trigger
EAC	End Around Carry
ECC	Error Correcting Code
EE	End Exception
EHOM	End of High Order Mark
EI MODE	Enter Instruction Manual Mode Trigger
EI OP	Enter Instruction Manual Operation Trigger
EK	Exchange Control Check
EKJ	Exchange Check Reject
EMA	Exchange Memory Address
EMAD	Exchange Memory Address Decoder
EMAR	Exchange Memory Address Register
EMF	External Memory Fetch Sequencer
EN	Enable
EN DIS	Interrupt Enabled, Disabled Trigger
END EXE	End Execute, Exception
EOA	End of A Register
EOC	End of C Register, End of Card
EOF	End of File
EOS	End Operation Suppress Same as SEOP
EPGK	Exchange Program Check
EPK	Exchange Program Check
EQ	Equal
ERR	Error
EVEN BR	Even Branch
EX	Exchange
EX CHK	Exchange Check
EX CHK REJ	Exchange Check Reject
EX WAIT	Execute Wait Trigger
EXC	Exception, Execute
EXCH	Exchange
EXE	Execute, Execution Exception
EXIC, EXID	Execute Indirect and Count
EXID, OP	Execute Indirect Operation
EXMD	Execute Mode Trigger
EXOP	Execute Operation
EXP	Exponent
EXT	External
FL	Field Length
FLL 1	Fetch Level Lookahead Load Sequences
FLP	Floating Point
F/N	Branch If On/Branch If Off
FODX	Fetch Outstanding Trigger
FP INCMPL	Floating Point Incomplete trigger
FPZ ALT	Floating Point Z Alternator
FR	Free Running, From, Frame
FST	First
FULL WD S	Full Word Straight
FULL WD S	Full Word, Not Straight
FW TGR	Full Word Trigger
FWD	Forward, Forwarding
GEN	Generate
GLFT	Geometric Load First Time Trigg
GLR	Geometric Load Register
GT	Gate
H	High
HALT SYNC	Program Halt Sync Trigger
HK	Housekeeping
HLT REQ	Program Halt Required
HO	Hold-Over
HOM	High Order Mark
HOR	Horizontal
HSB	Handle Sign Byte Trigger
HSCLN	Houseclean
HX	Disk Synchronizer
I-O	Input or Output
I-UNIT	Instruction Unit
IAOB	Index Adder Output Bus
IAOB	Instruction Adder Out Bus
IAR	Interrupt Address Register
IAU	Index Adder Unit
IAUC	Indexing Arithmetic Unit Counter
IBL	Initial Buffer Loading
IC	Instruction Counter

IC BUFF	Instruction Counter Buffer
ICA	Instruction Counter Adder
ICADV	Instruction Counter Advance
ICB	Instruction Counter Buffer
ICBC, 1Y,2Y	Instruction Counter Block Check 1Y, 2Y
ICFO1Y	Instruction Fetch Outstanding 1Y
ICF O2Y	Instruction Fetch Outstanding 2Y
ICIB	Instruction Checker in Bus
ICL1	LA Load Instruction Counter, Sequencer
ICOB	Instruction Checker Out Bus
ICOR	Instruction Word Correct
ICR	Instruction Counter Register
ICRN1Y	Instruction Counter Return to 1Y
ICRN2Y	Instruction Counter Return to 2Y
ICW	Initial Control Word
ID	Indicator, Indicator Driver
IDC	Identifiable Error Trigger
IE	Instruction Error
IEI	I-Unit Exception Indicators
IEXP	Instruction Execute In Progress Trigger
IF	Instruction Fetch
IF 1Y(2Y)	Instruction Fetch to 1Y(2Y)
IHW	Instruction Half-Word
IHW EX ZR	Instruction Half-Word Execute Z Right
IHW RDY ZR	Instruction Half-Word Ready ZR
IJ	Instruction Reject
IJ IDC	Instruction Reject Identifiable Check
IK	Instruction Check
IMM	Immediate
IMOB	Instruction-Memory-Out-Bus
IND	Indicator
INFO	Information
INH	Inhibit
INHBT	Inhibit
INHIB	Inhibit
INIT	Initial
INST	Instruction
INSTR	Instruction
INT	Internal
INT	Internal Operand Bit Interrupt
INT	Internal Operand Bit
INTR, INTRPT	Interrupt
INST, LEV	Instruction Level
IPL GO	Initial Program Load Go Trigger
IPL MODE	Initial Program Load Mode Trigger
IPL OP	Initial Program Load Operation
IPL PB TGR	Initial Program Load Push-button Trigger
IPL REQ	Initial Program Load Required Trigger
IR	Imaginary Root
IRCY	I-Recovery
IR GAP	Inter-Record Gap
IRF	Internal Register Fetch Sequencer
IRPT OP	Interrupt Recovery Operation
IRPT-BIN	Interrupt Branch or Indicator
IUIR	Instruction Unit Updated Indicator Register
IWC 1	Instruction Word Check 1Y
IWC 2	Instruction Word Check 2Y
IWCA	Instruction Word Check Alternator
K	Compare
LA	Lookahead
LA RCY MD	Lookahead Recovery Mode
LAAR	Lookahead Address Register
LAEA	Lookahead Effective Address
LAIC	Lookahead Instruction Counter
LAICIB	Lookahead Instruction Checker In Bus
LAL	Lookahead Load
LAM 1	Left Address Modify 1 Sequences
LAMIB	Lookahead Storage In Bus
LAOP	Lookahead Operation Code
LARCY	Lookahead Recovery
LA EN	Lookahead Enable
LA LEV	Lookahead Level
LB	Lower Bounds
LC	Level Check
LC	Lost Carry
LCV	Load Converted
LD	Load
LDE	Late Decode Enable
LD PLS MEM	Load Pulse Memory
LEA	Left Effective Address
LEV	Level
LF	Level Filled
LFE	Level Fill E
LFT	Load Factor
LGA	Load Geometric Address
LHFW	Left Half of Full-Word
LID	Load Indirect
LLP	Last Load Pulse
LMO	Leftmost Ones
LO	Low
LOBND	Lower Boundary
LOC	Locate
LOD	Left Ones Detect
LOP	Load Operation Code Into LA Sequencer
LP	Load Point

APPENDIX D. (Cont'd)

LRCR	Longitudinal Parity Check Register
LS	Latch Scan, Large-Small, Load-Store
LS	Lost Significance
LST	Load Store Into LA Sequencer
LTC	Latch
LTH	Latch
LTRS	Load Transit and Set
LWF	Load With Flag
LZ	Leave or Set to Zero
LZC	Left Zeros Counter
MAB	Memory Address Bus
MAC	Memory Address Check
MAN	Manual
MANOP GO	Manual Operation Go Trigger
MANOP SYNC	Manual Operation Sync Trigger
MAR	Modify Addressable Registers
MAR	Memory Address Register
MB	Memory Bus
MBC	Memory Bus Clock
MCD	Multiplicand
MCND	Multiplicand
MDBI	Memory Data Bus In
MDBO	Memory Data Bus Out
MDR	Memory Data Register
MECH RDY	Mechanical Ready
MEM	Memory
MEM ADD REG	Memory Address Register
MEM ADR SEL REG	Memory Address Select Register
MEM ADR SEL	Memory Address Selector
MEM BFR REG	Memory Buffer Register
MEM END TR	Memory End Trigger
MES	Message
MF	Multiple Flag
MIB	Memory In Bus
MK	Machine Check
MK	Mark
MOB	Memory Out Bus
MOD	Modification
MOD	Module
MOD	Modulus
MOD ZL	Y to Z Transfer Modify ZL Inputs
MOD ZR	Y to Z Transfer Modify ZR Inputs
MPY	Multiply
MPYC	Multiply and Add, Multiply Cumulative
MPYR	Multiplier
M/C	Marginal Check
MSEC	Millisecond
MTC	Master Tests Complete
MULT	Multiple

NSEC	Nanosecond
NEG	Negative
NIDC	I-Unit Nonidentifiable Check
NIDC	Non Identifiable Error Trigger
NM	Noisy Mode
NO OP	No Operation
NORM	Normal
N/C	Normally Closed
N/O	Normally Open
OCC	Operand Check Counter
ODD BR	Odd Branch
OFF	Offset
OP	Operation
OP	Operation Code Invalid
OPCD	Operand CD
OP CHK	Operand Check
OP CORR	Operand Correct
OSC	Oscillator
OUT	Outgoing
OUT REG	Output Register
P	Parity
PAR	Parity
PAR VER REG	Parity Verifier Register
PAU	Parallel Arithmetic Unit
PBCL	Punch Brush Card Lever
PDF	Power Distribution Frame
PF	Partial Field
PIRCY	Prepare for I-Unit Recovery
PK	Program Check
PL	Pipeline
PLF	Pipeline Off
PNL	Panel
POS	Position
POS	Positive
PP	Partial Product
PREP	Prepare
PREPN	Preparation
PRES	Present
PRG	Program
PRG HLT	Program Halt
PRG HLT RQ	Program Halt Request
PRG STRT R	Program Start Required
PRG STRT S	Program Start Sync
PRI	Primary
PRI	Priority
PROC	Processing
PROD	Product
PSH	Prepare Shift Greater Than 48
PSH	Pre-Shift
PSS	Print Sub Scan

PTY	Parity
PU	Punch Unit
PUSA	Prepare for Unending Sequence of Addresses
PUL	Pulse
PWR	Power
PX	Progressive Indexing
PX OP	Progressive Indexing Operation
PX REG	Progressive Index Register
PX REQ	Progressive Indexing Required
R-A	Relative-Absolute
R-W	Read-Write
RA	Return Address
RA	Residue Adder
RAM 1	Right Address Modify 1 Sequencer
RAR	Refill Address Register, Return Address Register
RAR P	Return Address Register Parity
RASR	Read Assembly Register
RBL	Residual Byte Length
RD	Read
RDD	Read Delay Disconnect
RD ADR HI	Read Address High
RD ADR LO	Read Address Low
RDY	Ready
REA	Right Effective Address
REC	Receive
REC	Record
REC	Recovery
REC STOR MK	Record Storage Mark
RECOMP	Recomplement
RECOMP CONTROL	Recomplement Control
REG	Register
REJ	Reject
REL	Release
REQ	Request
RES	Residue
RESP	Response
RET ADR	Return Address
REV	Reverse
RFL	Residual Field Length
RIC	Read In Clock
RIS	Read In Step
RGZ	Result Greater than Zero
RHFW	Right Half of Full-Word
RLL	Recovery Level Lookahead Load Sequence
RLZ	Result Less than Zero
RM	Residue Multiplier
RMDR	Remainder
RMI	Roll Mode Interrogation
RN	Result Negative
RN 1Y	Return to 1Y
RN2Y	Return to 2Y
RND	Round
RO	Read-Out
ROC	Read Out Clock
ROS	Read Out Step
RO INTR	Read-Out Interrupt
RPT	Repeat
RSR	Read Storage Register
RST	Reset, Restore
RST 1	I-Unit Recovery Reset
RST 2	Lookahead Recovery Reset
RSU	Round Set Up
RTNY	Return to Y Trigger
RU	Remainder Underflow
RWC	Read-Write Control
RWCR	Read Word Count Register
RWD	Rewind
RWE	Read-Write Error
RZ	Result Zero
S	Store
S B REG	Sign Byte Register
S R-W	Scatter Read-Write
SA	Sense Amplifier
SAB	Sample A and B Pulses, Sign of AB Register
SABC	Sample A and B Controlled Pulses
SABR	Sample A and B Running Pulses
SAC	Sample A Controlled Pulses
SAD	Store Address
SAR	Sample A Running Pulse
SAU	Serial Arithmetic Unit
SBC	Storage Bus Control
SBC	Sample B Controlled Pulses
SBC	Six Bit Counter
SBR	Sample B Running Pulse
SC	Shift Counter
SCC	Store Check Counter
SCD	Sign of CD Register
SCL	Stacker Card Lever
SCN	Scan
SCNG	Scanning
SCNR	Scanner
SCTL	Shift Control
SCTR	Sector
SDOP	Single Display Operation
SEL	Select
SEL REG	Select Register
SEOP	Suppress End Operation
SERV	Service
SF	Skip Flag
SGL	Single
SGL DISP	Single Display Manual Operation

APPENDIX D. (Cont'd)

SGL STOR	Single Store Manual Control Trigger
SGN	Sign Modifier
SH	Shift
SIG	Signal
SIM	Simulate
SING DISP	Single Display Manual Operation
SING OP R, S	Single Operation Control Trigger
SING STOR	Single Store Manual Control Trigger
SIR	Set Indicators and Reset Trigger
SLL 1	Store Level LA Load Sequencer
SLO	Store Low Order
SM	Sample Memory Pulse
SMP, SMPL	Sample
SMS	Standard Modular System
SNC	Synchronize
SPL REC TGR ON	Split Record Trigger On
SPND INTLK	Suspend Recovery Interlock for Y Register Address
SR	Storage Register
SRAB	Sign Register AB
SRCD	Sign Register CD
SRND	Store Rounded
SRT	Store Root
SS	Single Shot
SSOP	Single Store Operation
SSQ	Store Square Root
ST	Start Trigger
STC	Sample Test Complete
STCR	Stacker
STICA	Store Instruction Counter IF
STO, STOR	Store
STO CHK	Store Check
STORE WAIT	Special Store Wait Trigger
SU	Set Up
SUR	Set Up for Recomplementation
SVC	Service
SW	Switch
SWP	Swap
SYNC	Synchronize
SYS	System
SZ	Store Zero
T	Time, Transfer Bus
T-C	True-Complement
TBC	Transfer Bus Counter
TC ADV REQ	Time Clock Advance Request
TC SYNC	Time Clock Sync
TCL	Throat Card Lever
TCOP	Time Clock Operation
TCU	IBM 7613 Tape Control
TE	Trailing Edge
TF	Data Flag T, Turn Off
TGR	Trigger
TMR	Timer
TMT-SWP	Transmit-Swap
TN	Turn On
TOB	Transfer Out Bus
TRF	Transfer
TRIG	Trigger
TRUE/COMPL	True or Complement
TS	Time Signal
TSM	Transmit
TU	Tape Unit
U	Unit
UB	Upper Bounds
U BUSY	Unit Busy
U RDY	Unit Ready
UF	Data Flag U
UIR	Updated Indicator Register
UK	Unit Check
UK STOP	Unit Check Stop
ULD	Upper/Lower Boundaries
ULBR	Upper/Lower Boundary Register
UNCOR	Uncorrectable
UN NOT RDY	Unit Not Ready
UNRJ	Unit Not Ready Reject Indicator
USA	Unending Sequence of Addresses
USEC	Microsecond
UXIR	Updated Index Indicator Register
VAT	Variable Autotransformer
VER	Verifier
VF	Data Flag V
VFL	Serial Arithmetic Unit
VFL	Variable Field Length
VFL 1, 2, 3, 4, 5	Lookahead Loading Sequencer, VFL 1, 2, 3, 4, 5
VRC	Vertical Parity Check, Vertical Redundancy Check
WBC	Word Boundary Crossover
WBC and DEC Z	Y to Z Word Boundary Crossover or Decode Z Inputs
WBC TGR	Word Boundary Crossover Trigger
WC	Word Count
WCR	Word Count Register
WD	Word
WI	Write In
WNA	W Operand Address-Nonexistent
WR	W Register
WR	Write
WR CL ADV	Write Clock Advance
WR IR GAP	Write Inter-Record Gap
WRD	Word
WSA	W Operand Address Special
WXA	W Operand Address Index (16-31)

XAB	Index Address Bus
XAC	Index Address Check
XAU	Index Adder Unit
XCL	Index Storage Clear Sequencer
XCS	Index Core Storage
XCZ	Index Count Zero
XE	Index Equal
XF	Index Fetch
XF	Index Flag
XFER	Transfer
XF IND	Index Fetch Indicator
XF and DEC ZL	Y to Z Transfer Index Fetch or Decode ZL Inputs
XFN	Generated Flag Negative
XFP	Generated Flag Positive
XH	Index High
XL	Index Low
XL END	Index Low Indicator
XOF 1	Lookahead Loading Sequencer, Index Operand Fetch
XPFP	Propagated Flag Positive
XPH	Exponent High
XPL	Exponent Low
XPM	Exponent Medium
XPN	Exponent High Negative
XPO	Exponent Overflow
XPU	Exponent Underflow
XR	Index Register
XS	Index Storage
XSF	Index Storage Fetch Sequencer
XSRT	Index Storage Read Test
XST	Index Storage Store Sequencer
XSWT	Index Storage Write Test
XTC	Index Time Clock
XVGZ	Index Value Greater than Zero
XVLZ	Index Value Less than Zero
XVZ	Index Value Zero
YL ZL	Y to Z Transfer Y Left to Z Left Inputs
YL ZR	Y Left to Z Right Transfer
YR ZL	Y to Z Transfer YR ZL Inputs
YR ZR	Y to Z Transfer Y to Z Right Inputs
ZD	Zero Divisor
ZDF	Z Data Fetch
ZDS	Z Data Store
ZL	Z Register Left Half
ZL BLOK 1Y	Z Left Block Instruction Fetch to 1Y
ZL BLOK 2Y	Z Left Block Instruction Fetch to 2Y
ZL SPND 1Y	Z Left Suspend Instruction Fetch to 1Y
ZL SPND 2Y	Z Left Suspend Instruction Fetch to 2Y
ZLAD	Z Left Operand Address Invalid
ZLFL PT	Z Left Floating Point
ZLIDC	Z Left Identifiable Check
ZLIF	Z Left Instruction Fetch
ZLIHW	Z Left Instruction Half Word
ZLMODR	Z Left Modification Required
ZLMT	Z Left Empty Trigger
ZLMT COA	Z Left Empty Condition On Accept
ZLNA	Z Left Operand Address Nonexistent
ZLSA	Z Left Operand Address Special 0-1
ZLXA	Z Left Operand Address Index 16-31
ZOA	Z Operand Address Decoder
ZOAL	Z Operand Address Left
ZOAR	Z Operand Address Right
ZR	Z Register
ZR BLOK 1Y	Z Right Block Instruction Fetch to 1
ZR BLOK 2Y	Z Right Block Instruction Fetch to 2
ZR FL PT	Z Right Floating Point
ZR FP	Z Right Floating Point
ZR SPND 1Y	Z Right Suspend Instruction Fetch to 1Y
ZR SPND 2Y	
ZR SPND 2Y	Z Right Suspend Instruction Fetch to 2Y
ZRAD	Z Right Operand Address Invalid
ZRIDC	Z Right Identifiable Check
ZRIF	Z Right Instruction Fetch
ZRIHW	Z Right Instruction Half Word
ZRMT	Z Right Empty Trigger
ZRMT COA	Z Right Empty Condition On Accept
ZRNA	Z Right Operand Address-Nonexiste
ZRSA	Z Right Operand Address Special (0-15)
ZRXA	Z Right Operand Address Index (16-31)
1Y	1Y Register
1YAD	1Y Address Invalid
1YIDC	1Y Identifiable Check
1Y 1F	1Y Instruction Fetch
1Y MC	1Y Storage Address Error Check
1Y MT	1Y Empty Trigger
2Y	2Y Register
2YAD	2Y Instruction Address Invalid
2YIDC	2Y Identifiable Check
2Y 1F	2Y Instruction Fetch
2Y MC	2Y Storage Address Error Check
2YMT	2Y Empty Trigger
4 S RING	Four-Stage Ring
6 S RING	Six-Stage Ring
8 S RING	Eight-Stage Ring
⊻	Exclusive OR

IBM®

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

Printed in U.S.A. R23-9695