IBM

GA21-9353-0
S5280-01

IBM 5280 Distributed Data System

Functions Reference Manual

Preface

This reference manual is intended for people who need the following information about the 5280:

- An overview of system programming and system function
- A description of system and partition data areas
- A description of machine addressing and object code instructions
- How to use system diagnostic aids

In the appendixes are hexadecimal conversion and addition tables, EBCDIC and ASCII charts, and SCS control codes.

Related Publications

- *IBM 5280 Data Areas and Diagnostic Aids Handbook*, SY31-0595
- *IBM 5280 Assembler Language Reference Manual*, SC21-7790
- *IBM 5280 Communications Utilities Reference Manual*, SC34-0247

First Edition (February 1980)

Changes are periodically made to the information herein; these changes will be reported in technical newsletters or in new editions of this publication.

Use this publication only for the purposes stated in the *Preface.*

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Product Information Development, Department 997, 11400 Burnet Road, Austin, Texas 78758. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever. You may, of course, continue to use the information you supply.

Contents

Chapter 1. 5280 Functions and Storage Overview

Components of the 5280 system include the:

- 5281 Data Station
- 5282 Dual Data Station
- 5285 Programmable Data Station
- 5286 Dual Programmable Data Station
- 5288 Programmable Control Unit
- 5256 Printer
- 5225 Printer

All data stations and the control unit may contain diskette drives. The 5285 and 5288 can have an optional printer attachment. The 5285 and 5288 can contain an optional communications attachment. A system controller contained in the 5285, 5286, and 5288 handles all system functions. The 5281 and 5282 data stations do *not* have a controller and, therefore, must be attached to a data station or control unit.

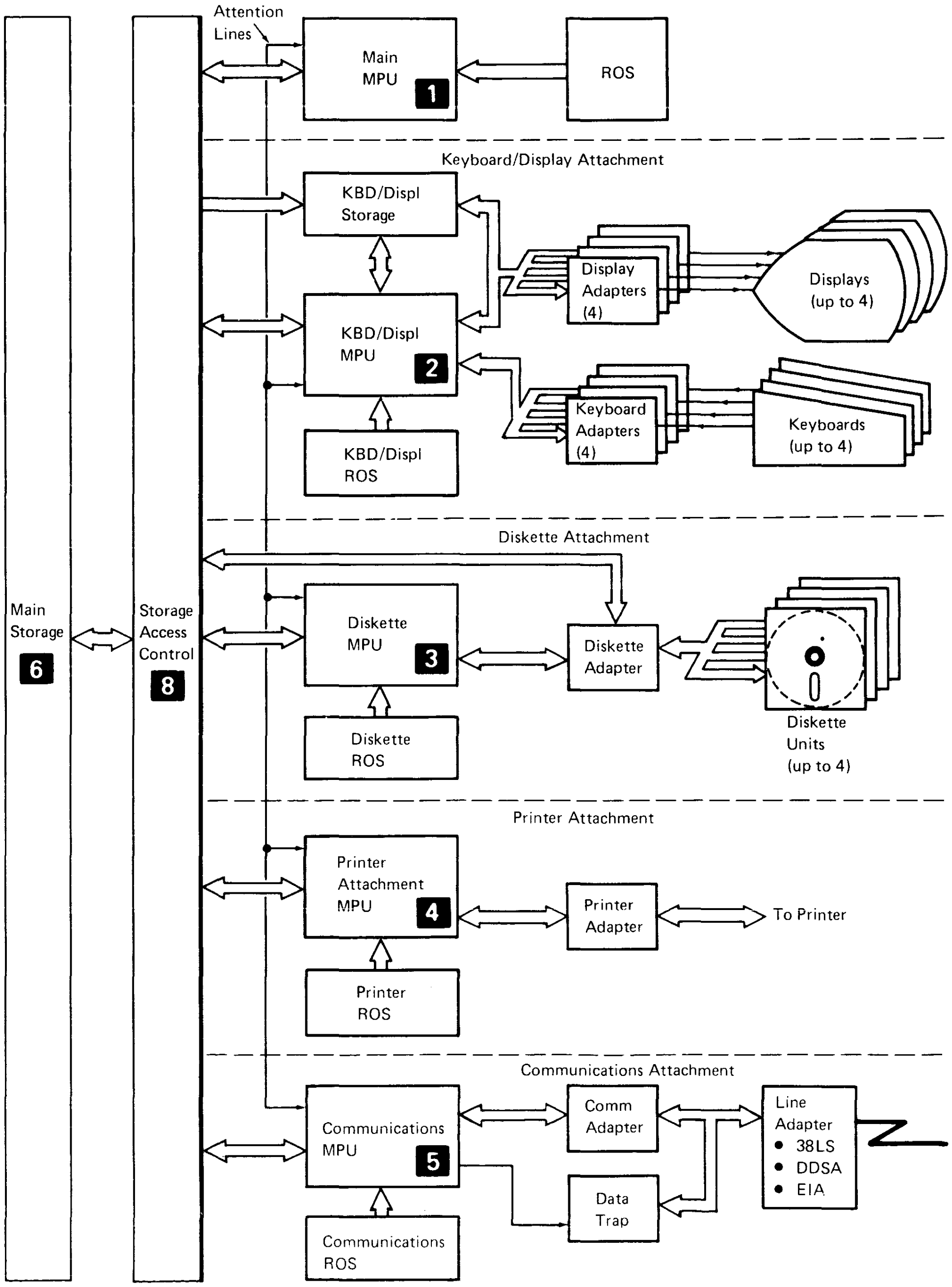
SYSTEM CONTROLLER

The 5280 system controller contains a main microprocessor, the partitioned main storage, and the device attachments. The device attachments contain the device microprocessors. The main microprocessor and the device microprocessors work independently of each other but share the same main storage.

The following illustration shows the main components of the system controller.

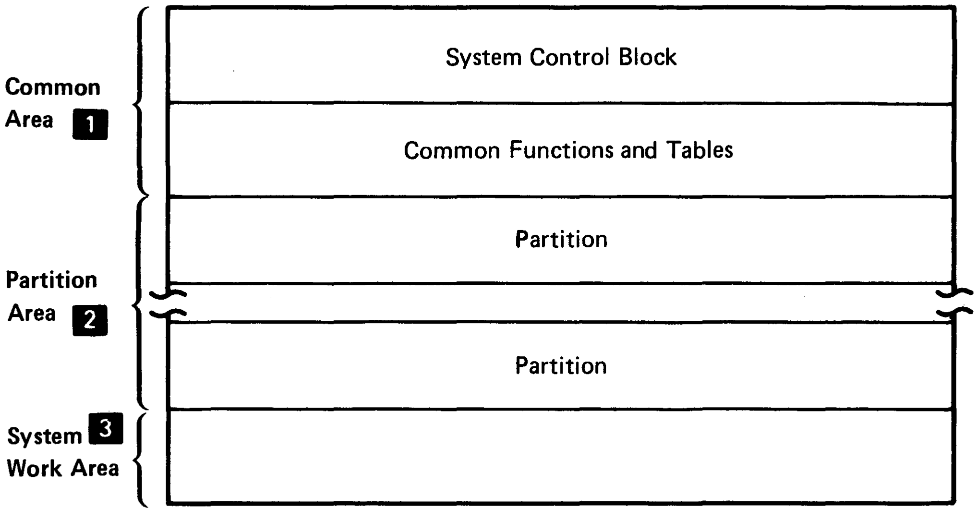
The main microprocessor 1 performs all of the non-I/O (input/output) operations, such as mathematical computations and data movement. The main microprocessor also controls the device microprocessors 2 through 5

The device microprocessors control all the operations for the attached devices. The main microprocessor communicates with the device microprocessors via IOBs (input/output control blocks) in main storage 6 and hardware attention lines 7 When the main microprocessor determines that work is required of a device microprocessor, it puts information into the appropriate IOB and activates an attention line to the device microprocessor. When the device microprocessor detects the attention from the main microprocessor, it reads the IOB and performs the requested work. The storage access control 8 directs access to main storage for all the microprocessors.



MAIN STORAGE

Main storage is divided into the areas illustrated in the following figure:



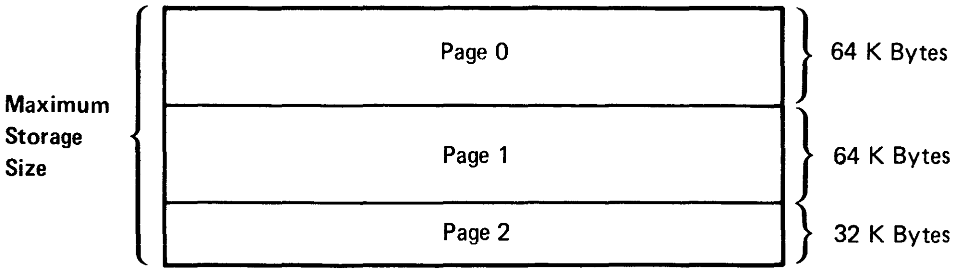
The common area 1 is always located at the beginning of main storage. It consists of the system control block and the common functions and tables.

The partition area 2 contains up to eight partitions. Except for the first and last partition, each partition can be up to 64 K bytes in length. The first partition can be up to 64 K minus 256 bytes, and the last partition can be up to 64 K minus 768 bytes in length. Total main storage size can be up to 160 K. A program can be loaded into each partition. After a program is loaded into a partition, the partition contains the IOBs, registers, indicators, formats, I/O buffers, tables, data areas, work area, and object code instructions required for the execution of the program.

The last 256 bytes of main storage are used by the controller as a system work area 3

Main Storage Addressing

Main storage is divided into 64 K byte sections referred to as pages. There can be a maximum of 160 K bytes of main storage.



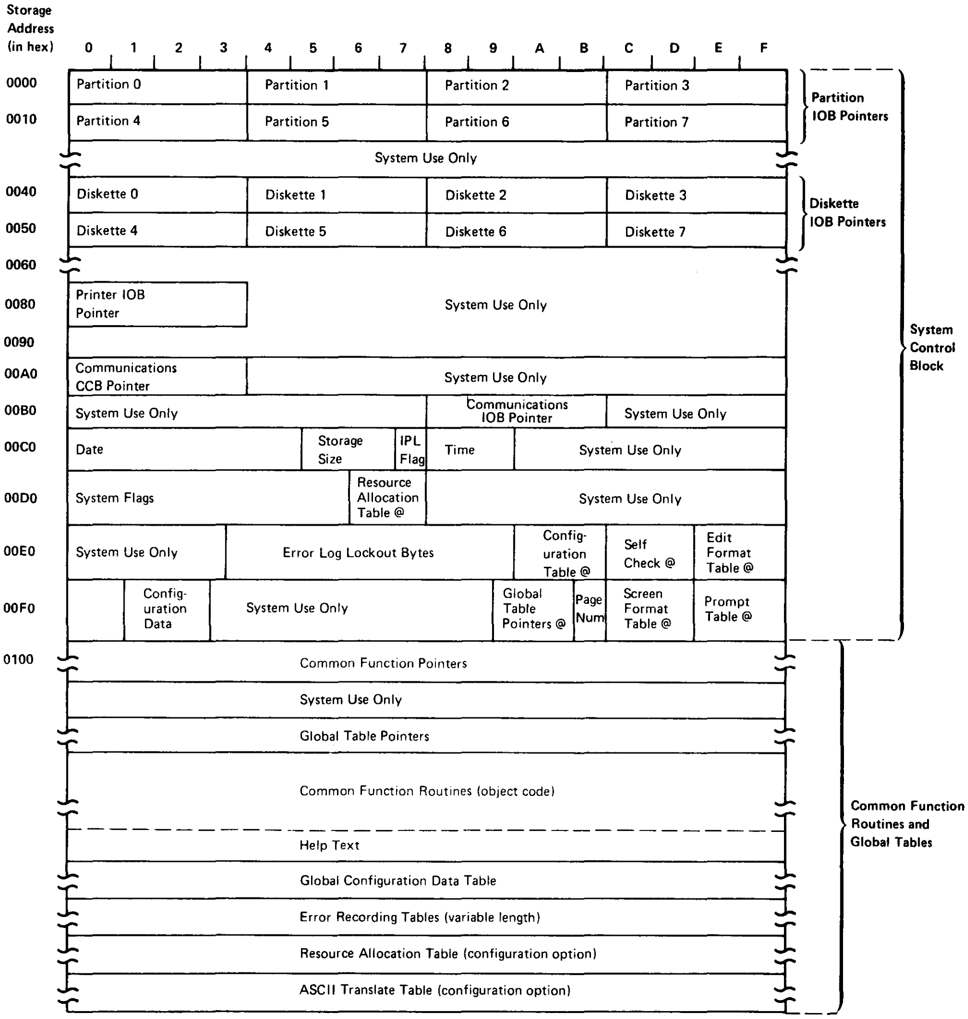
A partition cannot cross a page boundary, and therefore cannot be greater than 64 K bytes in length. Each byte within a partition can be uniquely addressed with 16 bits, from hex 0000 to FFFF.

Although the common area is always located on page 0, a partition may be on any main storage page.

When an application program addresses an area outside the partition, a 4-bit page number precedes the 16-bit storage address. This 20-bit address is used when a partition addresses an area within the common area.

COMMON AREA

The following is a general illustration of the system control block and common functions and tables located in the common area of main storage.



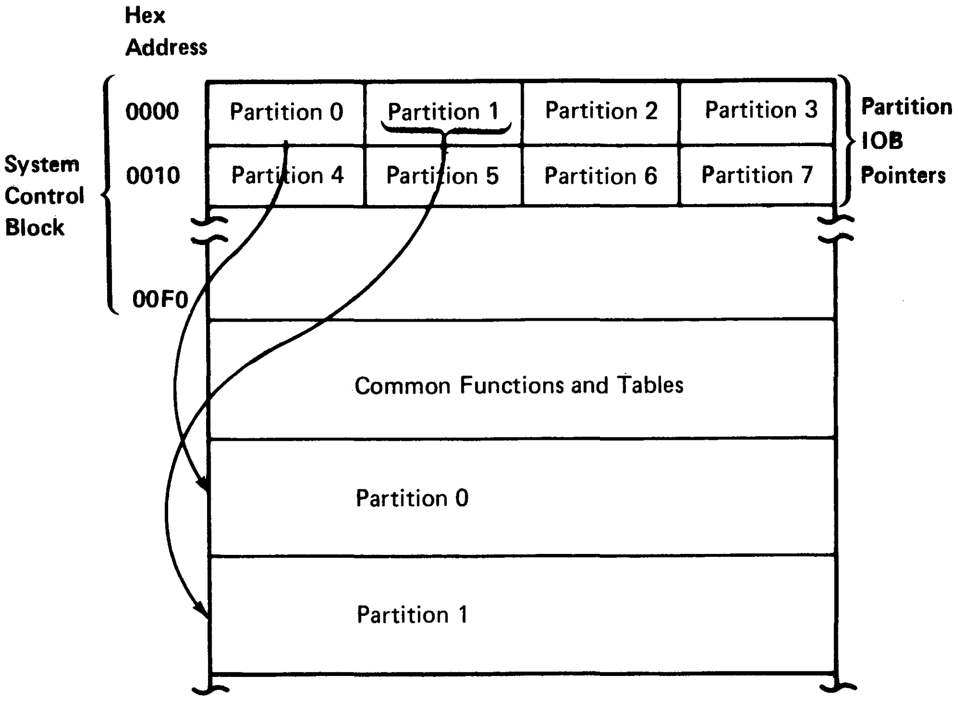
Storage Address (in hex)							
0000	Partition 0	Partition 1	Partition 2	Partition 3			
0010	Partition 4	Partition 5	Partition 6	Partition 7			
	System Use Only						
0040	Diskette 0	Diskette 1	Diskette 2	Diskette 3			
0050	Diskette 4	Diskette 5	Diskette 6	Diskette 7			
0060							
0080	Printer IOB Pointer	System Use Only					
0090							
00A0	Communications CCB Pointer	System Use Only					
00B0	System Use Only	Communications IOB Pointer	System Use Only				
00C0	Date	Storage Size	IPL Flag	Time	System Use Only		
00D0	System Flags	Resource Allocation Table @	System Use Only				
00E0	System Use Only	Error Log Lockout Bytes	Config-uration Table @	Self Check @	Edit Format Table @		
00F0		Config-uration Data	System Use Only	Global Table Pointers @	Page Num	Screen Format Table @	Prompt Table @
0100	Common Function Pointers						
	System Use Only						
	Global Table Pointers						
	Common Function Routines (object code)						
	Help Text						
	Global Configuration Data Table						
	Error Recording Tables (variable length)						
	Resource Allocation Table (configuration option)						
	ASCII Translate Table (configuration option)						

System Control Block

The system control block is located in the first 256 bytes of the common area. The fields of the system control block are assigned to fixed locations; the fields contain pointers to the partitions, device IOBs, and global system tables that are not assigned to fixed locations. Other system control block fields contain date, timer, and configuration information.

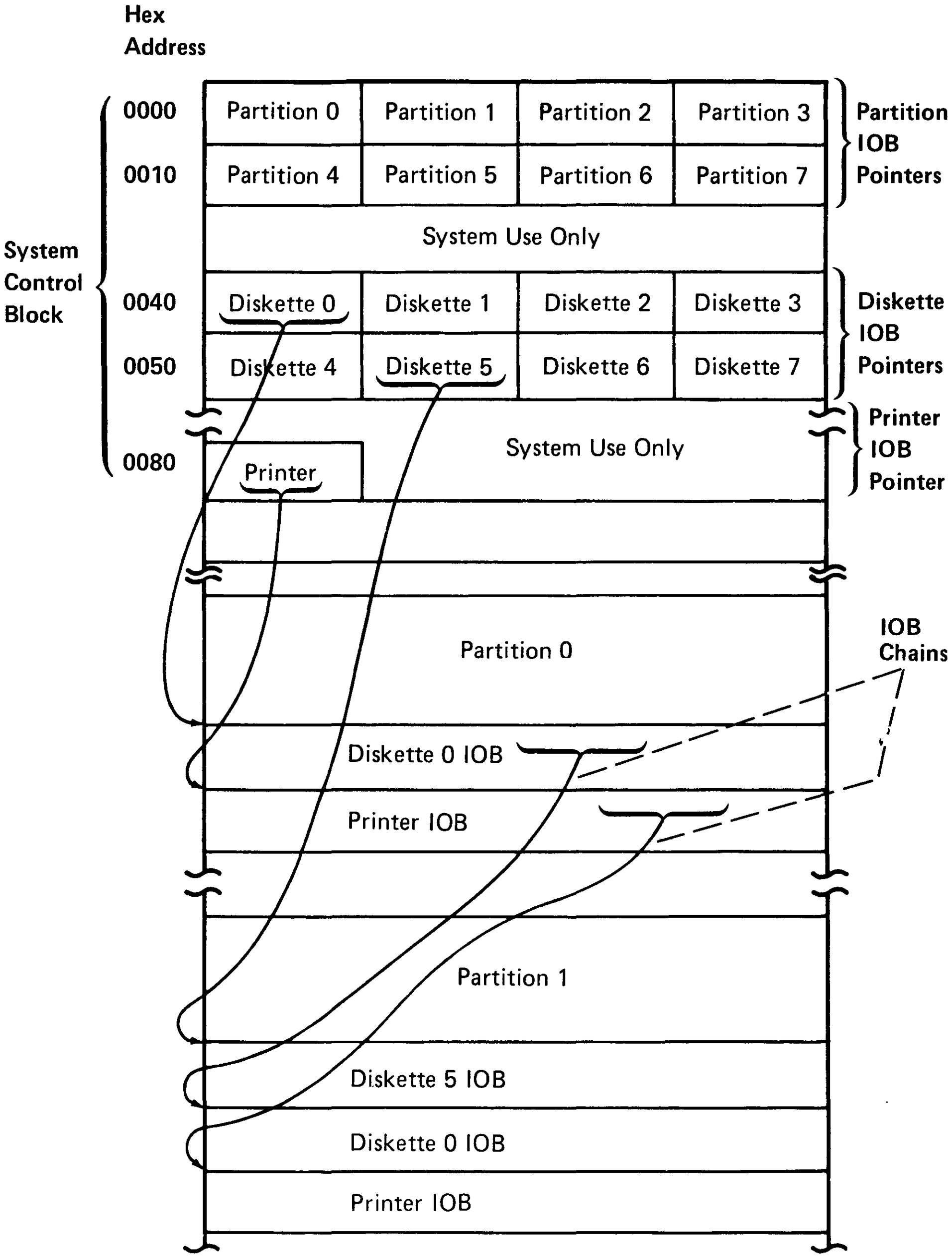
Partition Pointers

Each partition pointer is a 4-byte block of information about a partition. This information includes whether a program has been loaded into the partition, whether the partition is a foreground or background partition, and the address of the beginning of the partition. The partition IOB is always stored in the first 256 bytes of a partition, so this address is also the absolute address of the partition IOB. The partition IOB contains information about the partition and the program loaded into the partition. The main microprocessor uses the information in the partition pointer to find the partition; it uses the information in the partition IOB to execute the object code instructions stored within the partition.



Device IOB Pointers

Each device IOB pointer is a 4-byte block of information about a diskette drive, a printer, or the communications attachment. The information indicates whether the device is attached and includes the address of the first device IOB assigned to that device. If more than one IOB is assigned to one I/O device, an IOB chain is used; each device IOB contains the address of the next assigned device IOB. The device IOBs are stored within the main storage partitions and describe the I/O to be performed by each I/O device. The device microprocessor uses the information in the device IOB pointers to find the first device IOB. They use the information in the device IOB to perform the required I/O and to find the next device IOB. The last IOB on the chain points back to the first IOB.



Pointers to Global System Tables

Each global system table pointer contains the address of a global system table. System tables contain the addresses of prompts, formats, tables, and other data areas. System tables are used within each partition to contain the addresses of the data areas within that partition. The global system tables contain addresses of global data areas that are stored within the common area rather than within a partition. Data areas stored within a partition can be used only by that partition; however, global data areas can be used by any partition. Global data areas include a printer configuration table, screen formats, prompts for keyboard/display I/O, edit formats for diskette, printer, or communications I/O, data tables for table operations, and self-check data for self-check operations.

Common Functions and Help Text

Following the system control block is an area of variable length that contains common function routines. These routines can be called from any partition; return is made to the calling partition.

The routines stored in the common functions area depend upon the individual system. A table of help text messages may be included in the common area. These messages can be called from the keyboard in response to the Help key.

Configuration Table

A configuration table is included in the common area if one or more printers are attached to the system. The address of the configuration table is stored in the system control block.

The configuration table has one entry for each printer. Each entry has such information as the device subaddress and the number of entries the printer has in the soft error count table.

Error Recording Tables

Two error tables are stored in the common area as global tables 0 and 1: (1) the system hardware error log, and (2) the soft error count table. The system error log is of variable length and is used by the microprocessors to record system hardware-related errors. Each table entry has information to identify the device, IOB and program associated with the error. The soft error count table is used by the printer microprocessor to record the number of soft printer errors that occur during program executions. These error tables provide a history of system hardware-related errors and I/O errors that can be written to a diskette with a special error log dump program. See the *Data Areas and Diagnostics Aids Handbook* for information about communications error tables.

Resource Allocation Table

The optional resource allocation table specifies the logical devices that can be accessed by each partition. Each table entry contains a logical device ID and the physical address of the device. The logical device ID, a 2-character ID assigned to the device during system configuration time, can be used to address the device. The main microprocessor uses the logical ID to find the physical address of the device in the resource allocation table.

ASCII Translate Table

Data is stored in main storage in EBCDIC notation. However, data in another notation can be translated to EBCDIC as it is read into an I/O buffer. Or data can be translated from EBCDIC to another notation as it is read from an I/O buffer. The optional ASCII translate table can be used by any partition to translate data to or from ASCII notation. The ASCII table is another global table.

PARTITIONS

There may be up to 8 partitions numbered sequentially from zero. There must be at least one partition for each keyboard. A partition is of variable length, but it cannot cross a 64 K byte boundary. The number, size, and location of the partitions is defined at system configuration time. The first 256 bytes of each partition contains control information at fixed displacements from the beginning of the partition. The next 3840 bytes may be used as needed for indicators, decimal registers, or binary registers. This area is followed by a variable length storage area. The last 256 bytes of each partition is used for a work area. Each byte of a partition is addressable relative to the first byte of the partition. The following illustrates the areas of a main storage partition.

Relative Hex Address	
0000	Partition IOB
0040	Logical I/O Table
0080	Keyboard/Display IOB
0100	Indicators and Registers
1000	Storage for Object Program, Buffers, Device IOBs, and Other Data Areas
	Partition Work Area (256 bytes)

Partition IOB

The partition IOB describes the partition and the program loaded into the partition. The main microprocessor loads this information into the fields of the IOB, using information from the common area and from the application program. During program execution, the main microprocessor uses the information to determine the partition status, the program status, the address of the next executable instruction, and how long to execute instructions within the partition before going to the next partition.

The absolute address of the beginning of the partition is stored in the IOB. The main microprocessor adds this address to the relative addresses stored in the partition to generate absolute addresses for the program instructions.

A timer is set when the main microprocessor enters a partition. The IOB specifies how long the main microprocessor executes instructions within the partition. This time is determined by the application program. The main microprocessor exits the partition when the time limit is reached or when it encounters a nonoverlapped I/O instruction that is to be handled by a device microprocessor.

Logical I/O Table

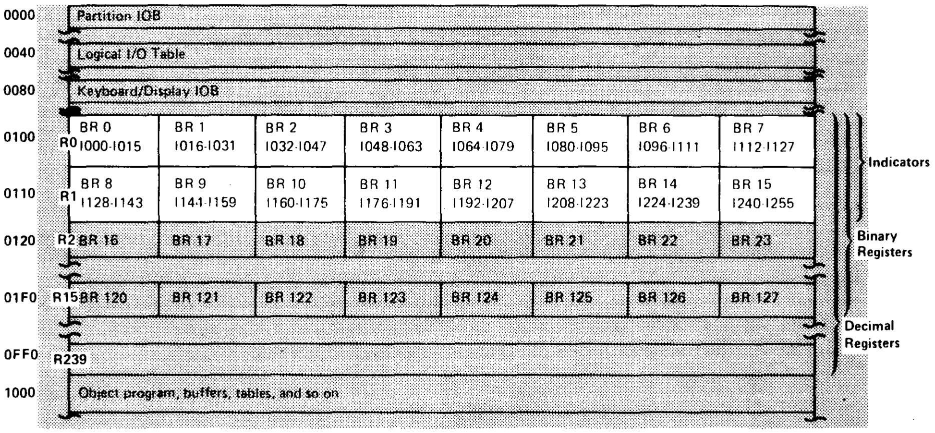
The logical I/O table consists of one 4-byte entry for each IOB that is used in the program. Each entry contains the address of the IOB, flags, and other information describing the IOB. The entries are numbered sequentially from hex 00 to 15, corresponding to the numbers assigned to the IOBs. The keyboard/display is always entry zero. When the main microprocessor encounters an I/O instruction during program execution, the instruction specifies the number assigned to the IOB that describes the work. The main microprocessor uses this number as an index into the logical I/O table; the entry at this index contains the address of the IOB and specifies the I/O device that is to perform the work.

Keyboard/Display IOB

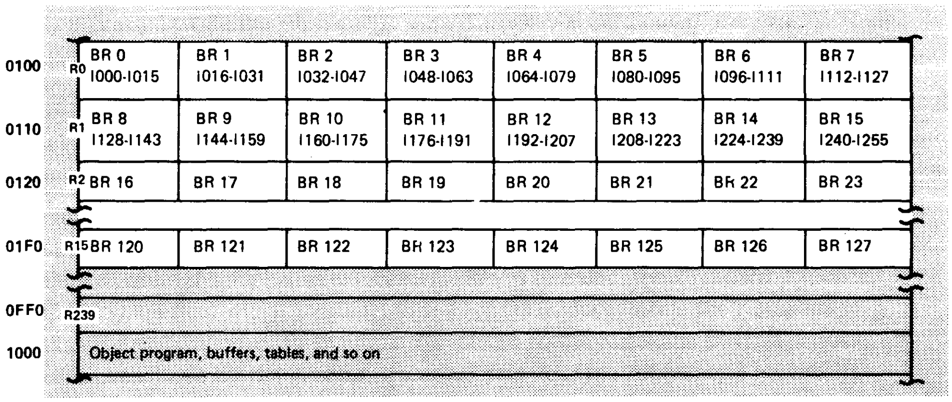
Every application program must have a properly initialized keyboard/display IOB. The keyboard/display IOB contains information to control all I/O via the keyboard/display to which the partition is assigned. This information includes the address of the I/O buffer, the address of the object code that controls the format of the records on the screen and in the I/O buffer, and the address of control tables located in keyboard/display storage. Keyboard display storage is not part of main storage; it is located within the keyboard/display attachment. The keyboard/display storage contains translate tables and other control information used by the keyboard/display microprocessor to process keystrokes and to display characters on the screen.

Registers and Indicators

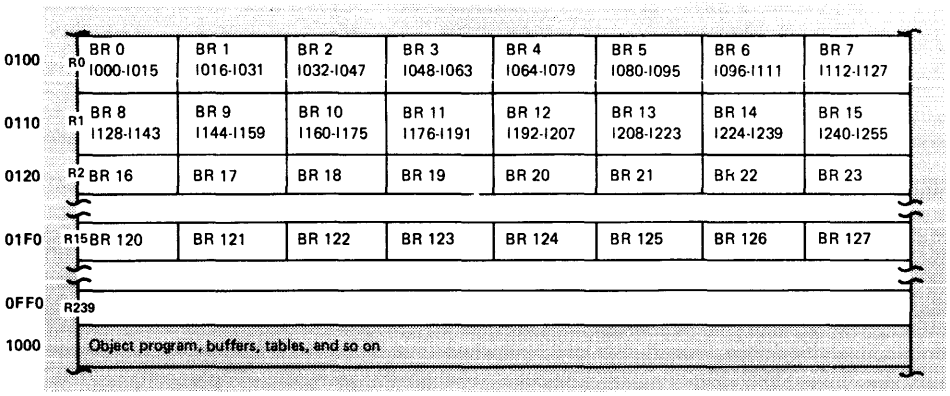
Immediately following the partition control area are bytes that can be used for indicators, binary registers, and decimal registers. The first 32 bytes contain 255 indicators. The indicators are numbered sequentially from zero. The first 100 indicators are user indicators, and the remaining indicators are used by the system. The indicators are located in the bytes that also can be used for the first 16 binary registers or the first two decimal registers.



The first 256 bytes of this area, including the bytes where the indicators are located, can be used for 128 two-byte binary registers. The first 16 binary registers are used for the indicators, and the next 16 binary registers are used by the system. The remaining binary registers may be used for binary arithmetic or logical operations by the application program. The binary registers are located in the bytes that also can be used for the first 16 decimal registers.



The remaining bytes of this area, up to relative address hex 0FFF, can be used for 16-byte decimal registers. Counting the first 16 decimal registers, which can be used for the binary registers, there are 240 decimal registers. Decimal registers R16 through R239 can be used for decimal arithmetic or logical operations by the application program. Decimal registers store data in EBCDIC notation and can support sign control.



0100	R0	BR 0 1000-1015	BR 1 1016-1031	BR 2 1032-1047	BR 3 1048-1063	BR 4 1064-1079	BR 5 1080-1095	BR 6 1096-1111	BR 7 1112-1127
0110	R1	BR 8 1128-1143	BR 9 1144-1159	BR 10 1160-1175	BR 11 1176-1191	BR 12 1192-1207	BR 13 1208-1223	BR 14 1224-1239	BR 15 1240-1255
0120	R2	BR 16	BR 17	BR 18	BR 19	BR 20	BR 21	BR 22	BR 23
01F0	R15	BR 120	BR 121	BR 122	BR 123	BR 124	BR 125	BR 126	BR 127
0FF0	R239								
1000		Object program, buffers, tables, and so on							

Any of the bytes up to relative address hex 0FFF that are not used for registers are used for data storage. The bytes following hex 0FFF can be used only for data storage.

Partition Work Buffer

The last 256 bytes of a partition are used as a partition work buffer. This work buffer is used during load operations, trace operations, decimal arithmetic operations, self-check, and formatting. The application program does not access this area.

System Work Buffer

The last 256 bytes of main storage are used as a system work buffer. This system work buffer is not associated with any partition, and it is not accessed by an application program.

Foreground and Background Partitions

One main storage partition is permanently assigned to each keyboard/display. A partition that is permanently assigned to a keyboard is a foreground partition. Any partition that is not permanently assigned to a keyboard is a background partition.

When a program executing in a background partition needs to use a keyboard, it can cause an edge indicator to be displayed on the keyboard/display screen. This indicator notifies the operator that a background partition needs the keyboard. The operator can interrupt the program that is using the keyboard and attach the background partition. When the background partition no longer needs the keyboard, the partition must be detached to give control of the keyboard back to the interrupted program. Only one partition can be attached to a keyboard/display at any given time.

INPUT AND OUTPUT BUFFERS

There must be at least one physical buffer in main storage for each IOB in a program that has I/O instructions. The physical buffer length must be a multiple of 128 bytes. Double buffering can be used for minimal delays in interactive programs; a second physical buffer is set up so the 5280 can process data in one while an input or output operation is being performed with the other. Double buffers are also required to duplicate fields of a previous record into the same field of a current record. The 5280 keeps track of the buffers and the records that are in the buffers.

Data sets can be blocked for better utilization of diskette space; a logical buffer is set up and the blocking and deblocking functions are performed automatically by the 5280. Or the logical buffer can be omitted and logical records can be blocked and deblocked directly to and from the physical buffer.

EXTERNAL STATUS PROCESSING

While an I/O device is processing I/O, it may encounter a condition that the device microprocessor cannot handle, such as an error condition or a condition that requires operator intervention or execution of object code instructions. When this occurs, the device microprocessor stops processing the I/O, places a condition code into the device IOB, sets an external status flag in the device IOB, and sets an attention line to the main microprocessor. The device microprocessor continues to service the other IOBs.

When the main microprocessor determines that an IOB has the external status flag set, it enters the partition and executes appropriate object code instructions to resolve the conditions. The instructions are determined by the application program. When the application program has resolved the condition, the main microprocessor resets the external status flag and goes to the next partition. The device microprocessor returns to the IOB only when it again receives an I/O command. The I/O command may be a reissue of the last I/O command.

LOADING A PARTITION

At IPL, a program can be loaded into any main storage partition. At any time after IPL, a partition can be loaded by a program instruction or by the standard load processor in the common function area. The standard load processor prompts for load parameters to be entered from the keyboard. A program instruction can prompt for load parameters to be entered from the keyboard, or can obtain the load parameters from a storage area. The load parameters include the partition number, the device ID or physical address, and the name of the data set to load. The load operation can load a data set into another partition or can reload the same partition with the same or a different data set. After the main microprocessor obtains the load parameters, it attempts to load the data set from diskette into the partition.

Unless the diskette sector size is greater than 256 bytes, the first read will cause 256 bytes to be read into the partition. If the sector size is greater than 256 bytes, the first read will cause one sector to be read into the partition. In either case, the rest of the object data set will be read into the partition in 4 K byte blocks.

The data set is read from the diskette from the BOE (beginning of extent) to the EOD (end of data). There must be no gaps of unused diskette space between BOE and EOD. The first block that is read into the partition contains the partition IOB. The main microprocessor checks the length specified in the partition IOB and then checks the length of the partition being loaded. If the size of the partition being loaded is sufficient for the data set, the load proceeds until all data in the data set is read from the diskette. If the size of the partition being loaded is not sufficient, a load error results.

Partial Overlay

A partial overlay can spot load a section of object code or data into a partition without destroying the program object code already in the partition. A partial overlay is initiated by a program instruction. The load parameters must include the address where the partial overlay begins. When the partial overlay is completed, control returns to the instruction following the load instruction that initiated the partial overlay operation.

Error Recovery

There are two methods of error recovery that may be used when an error occurs during a load operation. One method allows the main microprocessor to handle error recovery. The other method uses error recovery procedures written by the user. The load instruction indicates which method of error recovery is used.

User Defined Error Recovery

When a program instruction loads a data set into another partition, or if the load takes place through a common function, the load instruction can indicate that user defined error recovery procedures will handle error recovery. If the load operation is successful, control returns to the *second* instruction following the load instruction. If an error occurs during the load operation, the main microprocessor places the error code into a system binary register (BR16) and returns control to the *first* instruction following the load instruction. This instruction usually branches to the error recovery procedures.

Main Microprocessor Error Recovery

There are four types of error recovery procedures, depending on the type of load taking place when the error occurred. When any type of error occurs, the main microprocessor sends an error message to the screen and waits for the operator to press the Reset key. After the reset, error recovery is as follows for the different types of loads:

Global load, prompts for the load parameters to be entered from the keyboard. After reset, the load prompt is redisplayed with the original information that was entered. The operator can then enter the correct information.

Program instruction reloading the same partition, with the standard load prompt in the common functions area available. After reset, the load instruction is replaced with the standard load prompt, which prompts for the load parameters to be loaded from the keyboard.

Program instruction reloading the same partition, with no standard load prompt available. There is no way to retry this type of load. The main microprocessor issues an exit instruction and goes to the next partition. The partition that was being loaded is available to be loaded by another partition.

Program instruction loading another partition. After reset, the load instruction is not retried. The partition that was being loaded is made available to be loaded by another load instruction or by the standard load processor. Control returns to the instruction following the load instruction.

SUBROUTINES

The 5280 supports a variable-length address stack for use during subroutine calls and returns. The assembler places the address (relative to the start of the partition) of the address stack into BR18. During program execution when a subroutine call is executed, the main microprocessor places the 2-byte absolute address of the next sequential instruction into the address stack pointed to by BR18. Then the content of BR18 is incremented by 2 so that it points to the next available 2-byte entry in the address stack. When a return is executed, the content of BR18 is decremented by 2, and the address stored in the address stack at the location pointed to by BR18 is taken as the return address.

ADDRESS VALIDITY CHECKING

Addresses in assembler language instructions are specified in the following two ways: (1) directly by a 2-byte address in the object code that was generated by a reference to a label in the source code, and (2) indirectly by an address in a binary register (this address is usually calculated), to which a displacement may be added to provide an offset into the base address. No validity checking is made for direct addresses; because the 2-byte address in the object code is generated by a reference to a label in the source code, the referenced label must be valid and within the partition for the code to assemble correctly. For indirect addresses (except addresses that access areas within the common area), the 5280 checks the address to which access is being made to verify that the address is within the partition. If a displacement is included in the instruction, it is added to the base address and the resulting address is checked to verify that it is within the partition. No validity checking is made on addresses that access areas within the common area (20-bit addresses). No additional checking is made to an address within an instruction that is modified by the INXEQ instruction; if the INXEQ instruction modifies an address within an instruction and the resulting address points to an area of storage outside the partition, unpredictable results will occur.

OPENING A DATA SET IOB

The main microprocessor uses the OPEN instruction to prepare for I/O processing. When the main microprocessor executes an OPEN, it places the IOB on the IOB chain, initializes (or updates) information in the IOB, and verifies data set sharing capabilities.

To process an OPEN, the main microprocessor:

1. Obtains the IOB pointer address from the logical I/O table entry for this data set.

2. If the IOB pointer address specified in the logical I/O table is not between hex 40 and BC inclusive, or is not on a 4-byte boundary, the main microprocessor uses the device ID (bytes hex 60 and 61 of the IOB) as a search argument and searches the resource allocation table. If a match is made on the partition number and device ID, the main microprocessor takes the physical address given in the resource allocation table and uses it to open the data set IOB. If the system does not have a resource allocation table, an external status (0736) occurs. If no match is made on the device ID, an external status (0725) occurs. If the physical address that is found in the resource allocation table is invalid, an external status (0726) occurs. There are two IOB pointers in this range (address hex A0 and A4) that are used exclusively by the communications access method to access the communications microprocessor; these IOB pointers are not to be used by the application program. No checking is made to ensure that the application program does not use these IOB pointers, and unpredictable results may occur if they are used.

3. Determines the proper attention line to use, based on the IOB pointer address, and checks to determine if the device to open is installed. The main microprocessor does this by checking the third byte of the IOB pointer for a nonzero value. After checkout, each device microprocessor places hex FF in this byte to signal that the device is installed. During an open, the main microprocessor detects this nonzero value and continues doing the open. When the main microprocessor places the address of an IOB in the IOB chain, it leaves bit 1 of the third byte on so that there will always be a nonzero value there for later opens. If the device is not installed, this byte is left at hex 00 after checkout; the main microprocessor interprets this zero value to indicate that the device is not installed and will force an external status (0731) on any attempt to open the device.

4. Checks to determine if the data set IOB is already open. If it is already open, skip to step 8.

5. Checks to determine if there are any other IOBs on the chain. If the new IOB is label update, and if there are other IOBs on the chain, an external status (0733) occurs. If there are no other IOBs on chain, the address of the label update IOB is placed on the chain and bit 3 of byte 0 of the IOB pointer for this IOB is set; this marks the IOB chain as nonshare. If the new IOB is not label update, and if bit 3 of byte 0 of the IOB pointer is set, external status (0733) occurs. If there are no other IOBs on the chain, the main microprocessor places the address of the IOB on the chain and goes to step 8.

6. If there are other IOBs on the chain, the main microprocessor checks the share specifications.

7. If the share/access specifications are valid, the main microprocessor places the IOB on the chain. If they are not valid, external status (0727) occurs.

8. Saves the commands and operands in the IOB, turns on bits 0 and 1 in byte 0 of the IOB, and raises the attention line to the appropriate device microprocessor.

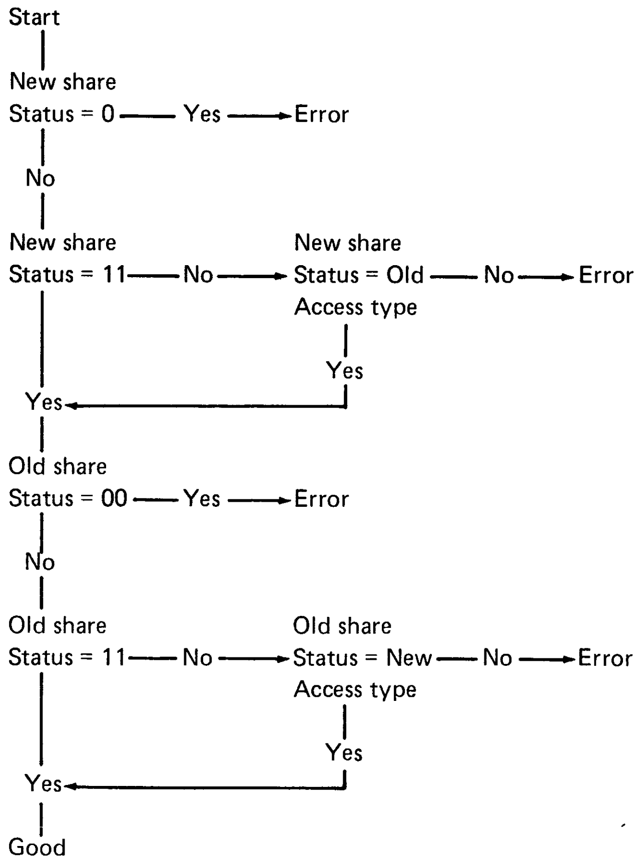
Formatting is not supported during an open (or allocate). If the HDR1 label should be formatted, a formatted read from the physical buffer should be executed after the open.

Share Data Set Opens

When a request is made to open, data set sharing is verified. A test is made to determine if the device subaddress of the new IOB matches that of the first IOB in the chain. If they do not match, the test is made on the next IOB in the chain. If there is a match, a test is made to determine if the IOB pointer address for the new data set is between hex 40 and 7C, inclusive. If it is not within this range, a match has been found and the share/access checking continues. If the IOB pointer address is within this range, an additional check based on data set names is made. If the data set name of the new IOB matches the data set name of the old IOB, the share/access checking continues. If the new data set name does not match the data set name of the old IOB, a mismatch has occurred and the next IOB in the chain is checked. For each match found, options based on read/write, share/don't share must compare. Four bits are assigned to contain the following access and share information:

Bit	Meaning
0	Read
1	Write
2	Read share allowed
3	Write share allowed

The following diagram shows how the main microprocessor compares the access type to the share options:



If the compare of options does not match according to the following diagram, an external status (0727) occurs. The error code is saved in the IOB, the appropriate external status code and external status bit are set, and a branch is taken to the external status subroutine.

R=Read
W=Write
S=Share

Old New	R/ RS	R/R &WS	R/ WS	R/ NS	W/ RS	W/R &WS	W/ WS	W/ NS	R&W/ RS	R&W/ R&WS	R&W/ WS	R&W /NS
R/RS	OK	OK										
R/R &WS	OK	OK			OK	OK			OK	OK		
R/WS					OK	OK						
R/NS												
W/RS		OK	OK									
W/R &WS		OK	OK			OK	OK			OK	OK	
W/WS						OK	OK					
W/NS												
R&W /RS		OK										
R&W/ R&WS		OK				OK				OK		
R&W /WS						OK						
R&W /NS												

KEYBOARD/DISPLAY I/O CONTROL

The keyboard/display attachment consists of a keyboard adapter, a display adapter, keyboard/display storage, and the keyboard/display microprocessor. Optional magnetic stripe readers and an optional elapsed time counter may also be included.

The keyboard/display microprocessor handles all data entry via the keyboard. It can handle up to four keyboards. For each keyboard it processes keyboard functions and data entry, and detects keystroke errors. It processes keystrokes and handles the character display according to the keyboard/display storage information. It uses a screen format control string, which is generated from the application program, to control the format of the input record as it is displayed on the screen and entered into the I/O buffer.

Keyboard/Display Storage

Each display has an assigned keyboard/display storage area. Within this area is a refresh buffer for the screen, and translate tables and other control information used by the keyboard/display microprocessor to interpret keystrokes and to display characters. The translate tables include the: (1) scan code translate table, which translates each keystroke scan code to a corresponding EBCDIC value that can be placed into the main storage I/O buffer; (2) display translate table, which translates each EBCDIC value to a display code before it is displayed on the screen; (3) validity table, which defines such things as the EBCDIC codes that are valid for each character set; and (4) diacritic table, which defines diacritic character combinations. Other control information in the keyboard/display storage area defines configuration of the lines on the screen and the symbols displayed on the status line for particular field definitions. The keyboard/display IOB specifies the address in keyboard/display storage of the storage area assigned to the keyboard.

Screen Format Control String

A source statement in the application program generates a string of object code, referred to as a screen format control string, that describes the format of each input record. This screen format control string specifies the length and valid characters for each input field, and describes prompts, display attributes, duplication fields and constant insert fields. It indicates the position on the screen where each field and prompt is to be displayed, and the position in the I/O buffer where each field is to be placed. The application program specifies the screen format control string and the I/O buffer to use, and the addresses of the string and buffer are stored in the keyboard/display IOB.

As the keyboard/display microprocessor processes each field of the screen format control string, it places the input data into the I/O buffer and displays it on the screen. However, the keyboard/display microprocessor cannot move the data from the I/O buffer to other main storage locations, or to another I/O device. When a screen format control string is completed, the keyboard/display microprocessor places a record advance condition code into the IOB and reports external status to the main microprocessor. The main microprocessor must process the contents of the I/O buffer according to the application program instructions.

Functions and Modes

When a function key is pressed, the keyboard scan code is translated by the keyboard/display storage translate tables to an EBCDIC code. This EBCDIC code initiates the appropriate function. The function may be processed by the keyboard/display microprocessor, by the application program, or by both.

The data entry mode may affect the way the function is processed. The keyboard/display microprocessor supports several modes of entry. (See the keyboard flags at hex displacement 3E in the keyboard/display IOB for a list of the modes.) The modes are selected by the application program, which must set the assigned mode flags in the keyboard/display IOB. The keyboard/display microprocessor controls the keyboard/display I/O and functions in the mode specified by the mode flags.

Magnetic Stripe Reader

The optional magnetic stripe reader reads a character string that is stored on a badge. When the badge is inserted into the reader, the character string is read into a buffer within the reader. The keyboard/display microprocessor reports an external status condition to the main microprocessor. The main microprocessor then executes the application program subroutine that reads the character string into main storage and processes it.

Elapsed Time Counter

The optional elapsed time counter records elapsed real time. The keyboard/display microprocessor maintains a timer that increments a 2-byte field in the system control block every 1.6 seconds. A program instruction can read this 2-byte field and the 1-byte timer value into a main storage area to measure the time elapsed during a job or during a portion of a job.

Errors Detected by the Keyboard/Display Microprocessor

The keyboard/display microprocessor detects keystroke errors and keyboard/display hardware errors. Most keystroke errors are handled by the keyboard/display microprocessor, which displays an error code on the screen and waits for the operator to press the Reset key. All hardware errors are entered into the error recording table in the common area. In addition, certain conditions cause the application program to be notified via external status.

DISKETTE I/O CONTROL

The diskette attachment consists of a diskette adapter and the diskette microprocessor. Each diskette microprocessor can handle up to 4 diskette drives.

The diskette microprocessor handles all data I/O functions for the diskette drives. These functions include reading and writing data set records, blocking and deblocking records, searching data set records, and managing shared data sets. The diskette microprocessor also handles allocating data sets, opening data sets, and closing data sets. It can also change data set labels on a diskette and insert or delete records.

Although all data is stored within main storage in EBCDIC notation, the diskette microprocessor can read data set records in another notation and translate them to EBCDIC, or it can translate EBCDIC records to another notation and then write the translated records to a diskette. The translation requires translate tables, which may be within a main storage partition or within the common area.

Error Recovery and External Status

Initial attempts to recover from errors are tried by the diskette microprocessor. When an error occurs during an I/O operation, the operation may be retried a certain number of times; the number depends on the operation and the type of error. If the error is not resolved by the diskette microprocessor, the diskette microprocessor places a 4-digit condition code in the diskette IOB and reports external status. When the main microprocessor determines that an external status condition is pending in the diskette IOB, it uses the condition code to find the appropriate subroutine in the application program to resolve the condition.

PRINTER CONTROL

The printer attachment consists of a printer adapter and a printer microprocessor. The printer microprocessor can handle up to four 5256 printers and one 5225 printer. The 5256 printer is a tabletop matrix character printer. The 5225 is a floor standing line printer.

The format of the printed output may be modified by SCS (standard character string) control characters. The SCS control characters may be placed in the printer output data stream by the application program unless the program is using an SCS conversion data set or unless no modification is desired. Each data set is described with a control statement in the source program. If the data set description specifies the data set type as an SCS conversion data set, the main microprocessor places the SCS control characters in the printer output data stream.

The printer microprocessor handles blocking and deblocking of output records. It also handles data sets that specify share attributes. For the printer, share attributes indicate that more than one data set can use the same printer.

Error Recovery and External Status

Initial attempts to recover from certain errors are tried by the printer microprocessor or by the printer. If the error or external status condition is not resolved by the printer or printer microprocessor, the printer microprocessor places a 4-digit condition code in the printer IOB and reports external status. When the main microprocessor determines that an external status condition is pending in the printer IOB, it uses the condition code to find the appropriate subroutine in the application program to resolve the condition.

The printer microprocessor records errors in the error tables, which are located in the common area.

COMMUNICATIONS CONTROL

The communications attachment consists of communications adapter, a communications data trap, and the communications microprocessor. The communications microprocessor supports one communications line. The adapter can provide data link support for BSC or SDLC protocol. The data trap is used by the communications microprocessor to store diagnostic information. See the *Data Areas and Diagnostic Aids Handbook* and the *Communications Utilities Reference Manual* for information about communications.

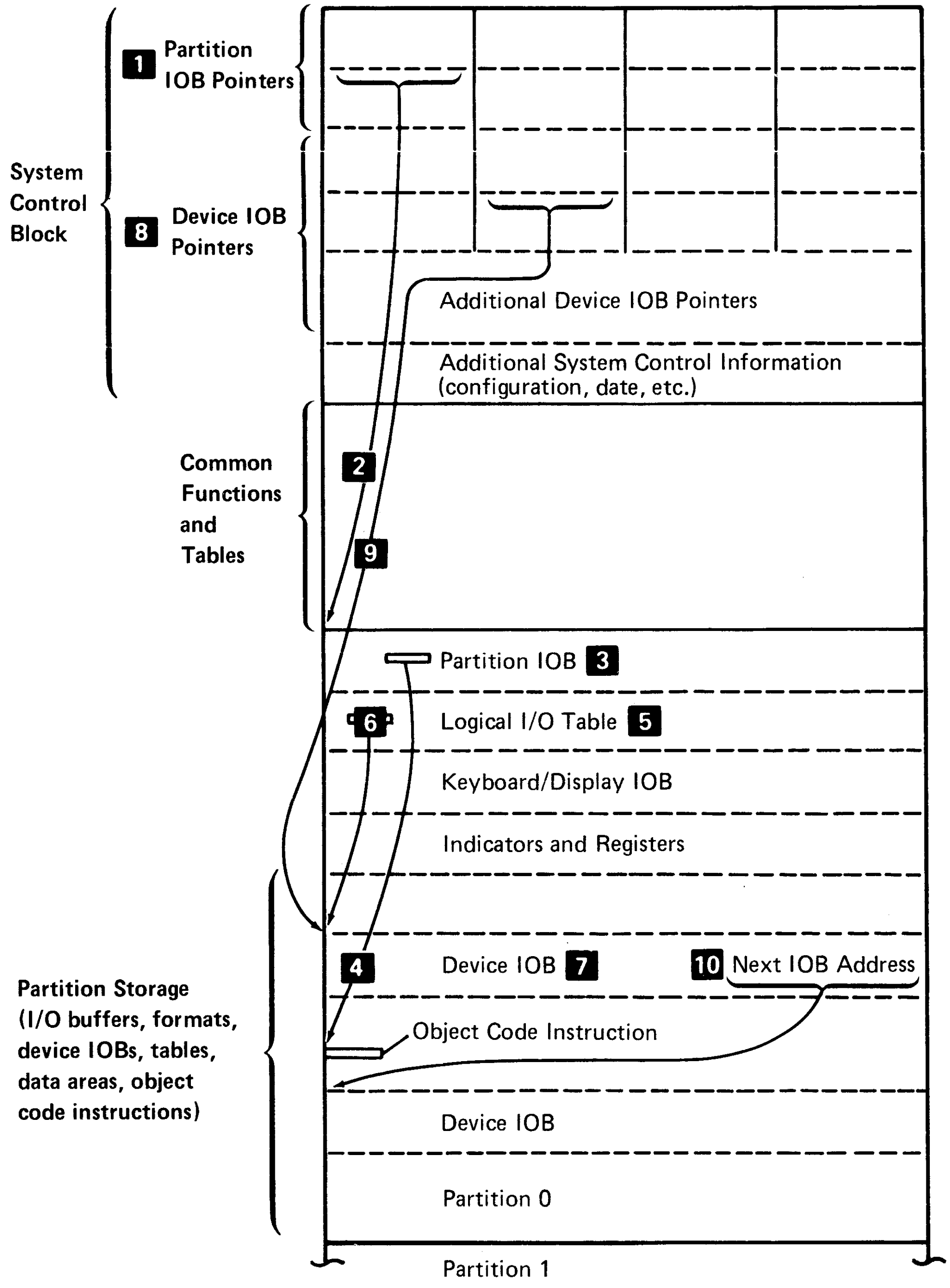
The communications microprocessor handles communications I/O, including sending status information and data to the host, receiving data and status information from the host, and blocking and deblocking records. The communications microprocessor uses a communications access method, which may be an IBM program product or a program written by the user, to control communications operations. The communications microprocessor interfaces with the communications access method through the communications control block. The communications access method, in turn, interfaces with the application program through the communications IOB. The communications IOB is described by a control statement in the source application program. The communications access method and communications control block must be loaded into a main storage partition. The application program and communications IOB are loaded into another main storage partition.

Error Recovery and External Status

The communications microprocessor attempts to recover from certain I/O errors and records errors in an error recording table located within the communications access method partition. If the error or condition is not resolved by the communications microprocessor, the communications access method places a 4-digit condition code in the communications IOB and reports external status.

TYPICAL OPERATION

This illustration is used with the typical operation description on the following pages.



The main microprocessor checks the partition IOB pointer 1 until it finds a pointer that indicates that a program is loaded in the partition. If there are no active attention lines pending, the main microprocessor goes to the address indicated in the partition IOB pointer 2 . The first 256 bytes of the partition holds the partition IOB 3 .

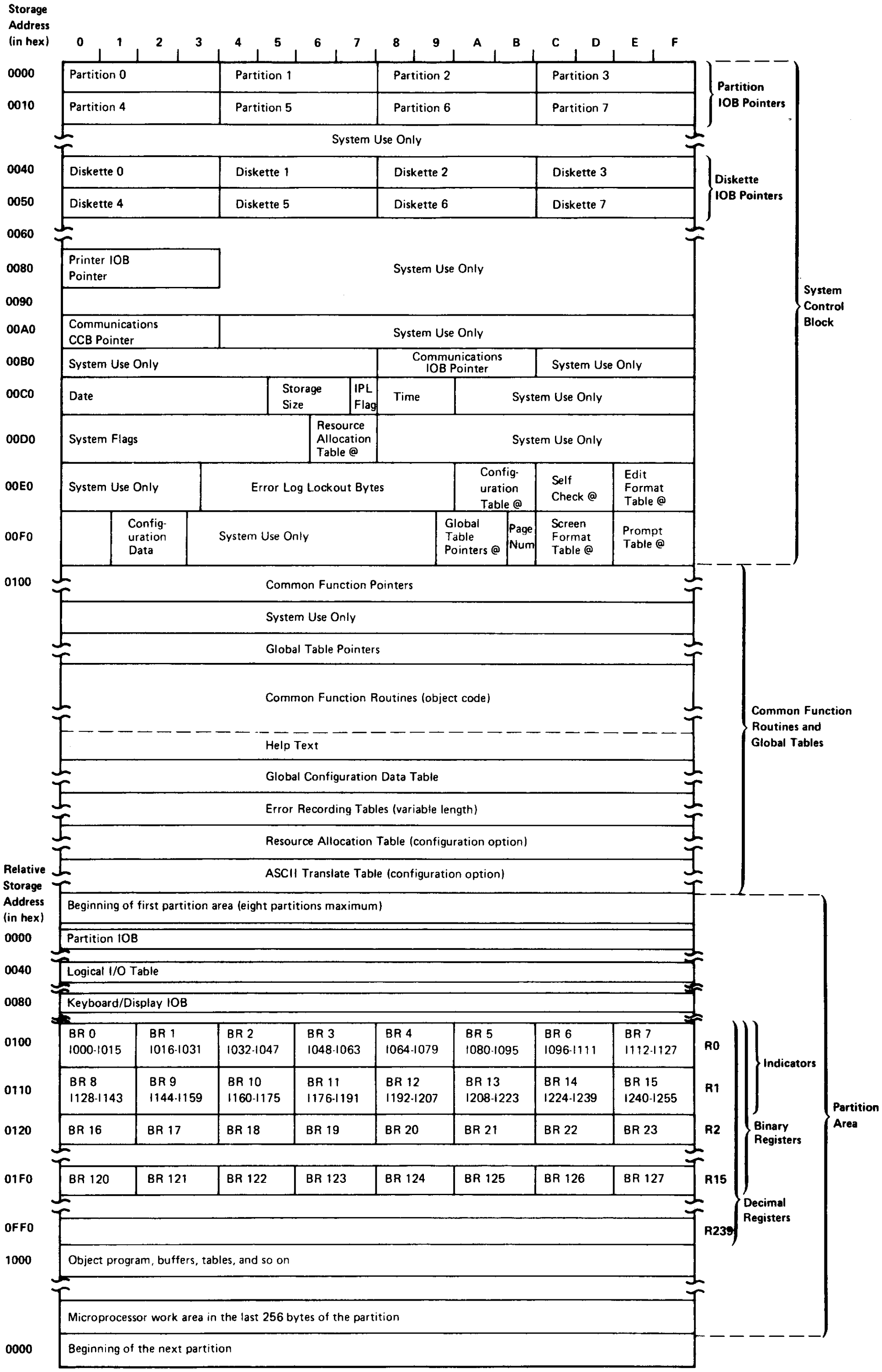
The partition IOB contains such information as the partition size and the address of the object code instruction to execute next. When the main microprocessor enters the partition, it sets a timer. This timer controls how long the main microprocessor is to remain within the partition. The main microprocessor then goes to the object code instruction address in the partition storage area 4 . It executes instructions in the storage area until the time limit is up or until it encounters a nonoverlapped I/O instruction. If the timer times out, the main microprocessor completes the execution of the instruction it is currently working on, returns to the partition IOB and stores the address of the next instruction to execute when it returns to this partition, and goes back to the system control block. If no active attention lines are pending, it continues checking the partition IOB pointers; when it finds a partition IOB pointer that indicates that a program is loaded in the partition, it goes to that partition and performs the same steps as described above.

If the main microprocessor encounters an I/O instruction before the timer times out, it uses the data set number specified in the instruction as an index into the logical I/O table 5 . It goes to the appropriate entry in the logical I/O table to find the address 6 of the device IOB that describes the I/O operation. The main microprocessor then goes to the device IOB 7 , loads the instruction into the IOB, and activates the device attention line to the appropriate I/O device. If the I/O instruction specified overlapped I/O, the main microprocessor continues executing instructions within the partition while the I/O device is performing the I/O. If the instruction specified nonoverlapped I/O, the main microprocessor exits the partition. The instruction following the I/O instruction is not executed until the I/O instruction is completed by the device.

When a device microprocessor senses an active device attention line, it checks the device IOB pointers 8 in the system control block until it finds a pointer that contains an IOB address. It then goes to the address 9 and performs the work described in the IOB. The IOB contains the instruction op code and parameters, the address of the I/O buffer or buffers, and other information such as format addresses and data set type. When the device microprocessor encounters a condition that it cannot handle, it clears the first two bits of the status byte and sets the external status bit in the status byte of the device IOB, and activates an attention line to the main microprocessor. If the device microprocessor finishes the I/O work in a normal way, it clears the first 2 bits of the status byte in the device IOB. The device microprocessor then checks the device IOB to determine the address of the next IOB on the IOB chain for this device 10 . It processes the IOBs on the chain until it encounters an IOB that is marked as the first on the chain. Except for the printer microprocessor, which has only one IOB pointer, the device microprocessor then returns to the system control block and checks the next device IOB pointer. If it finds another device IOB pointer that contains an IOB address, it goes to the IOB and uses the I/O device associated with the IOB pointer to process the IOB chain as described above.

This page intentionally left blank

This section describes the 5280 main storage data areas. The following figure shows the main storage organization for the 5280.


Storage Address (in hex)
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000 Partition 0 | Partition 1 | Partition 2 | Partition 3
0010 Partition 4 | Partition 5 | Partition 6 | Partition 7
Partition IOB Pointers
System Use Only
0040 Diskette 0 | Diskette 1 | Diskette 2 | Diskette 3
0050 Diskette 4 | Diskette 5 | Diskette 6 | Diskette 7
Diskette IOB Pointers
0060
0080 Printer IOB Pointer | System Use Only
0090
00A0 Communications CCB Pointer | System Use Only
00B0 System Use Only | Communications IOB Pointer | System Use Only
00C0 Date | Storage Size | IPL Flag | Time | System Use Only
00D0 System Flags | Resource Allocation Table @ | System Use Only
00E0 System Use Only | Error Log Lockout Bytes | Configuration Table @ | Self Check @ | Edit Format Table @
00F0 Configuration Data | System Use Only | Global Table Pointers @ | Page Num | Screen Format Table @ | Prompt Table @
System Control Block
0100 Common Function Pointers
System Use Only
Global Table Pointers
Common Function Routines (object code)
Help Text
Global Configuration Data Table
Error Recording Tables (variable length)
Resource Allocation Table (configuration option)
ASCII Translate Table (configuration option)
Common Function Routines and Global Tables
Relative Storage Address (in hex)
Beginning of first partition area (eight partitions maximum)
0000 Partition IOB
0040 Logical I/O Table
0080 Keyboard/Display IOB
0100 BR 0 1000-1015 | BR 1 1016-1031 | BR 2 1032-1047 | BR 3 1048-1063 | BR 4 1064-1079 | BR 5 1080-1095 | BR 6 1096-1111 | BR 7 1112-1127 | R0
0110 BR 8 1128-1143 | BR 9 1144-1159 | BR 10 1160-1175 | BR 11 1176-1191 | BR 12 1192-1207 | BR 13 1208-1223 | BR 14 1224-1239 | BR 15 1240-1255 | R1
0120 BR 16 | BR 17 | BR 18 | BR 19 | BR 20 | BR 21 | BR 22 | BR 23 | R2
01F0 BR 120 | BR 121 | BR 122 | BR 123 | BR 124 | BR 125 | BR 126 | BR 127 | R15
Indicators
Binary Registers
Decimal Registers
0FF0 R239
1000 Object program, buffers, tables, and so on
Microprocessor work area in the last 256 bytes of the partition
0000 Beginning of the next partition
Partition Area

SYSTEM CONTROL BLOCK

The system control block occupies the first 256 bytes of main storage and contains partition pointers, device IOB pointers, and pointers to system tables.

Hex Displacement	Length in Bytes (in Hex)	Description
0000	20	Partition Pointers: (one 4-byte block for each possible partition). Each 4-byte block has the following meaning:

Byte 0

Bit(s)		*Meaning*
0	1 =	A program is being loaded into the partition.
1	1 =	The partition is being attached to the keyboard.
2		System use only.
3	1 =	A keyboard attention occurred during a nonoverlapped, nonkeyboard-I/O operation.
4-7		System use only.

Byte 1

Bit(s)		*Meaning*
0	0 =	Background partition.
	1 =	Foreground partition.
1	1 =	There is no program in this partition; therefore, a program can be loaded.
2	1 =	An attention from the main microprocessor to the keyboard/display microprocessor is pending.
3	1 =	An attention from the keyboard/display microprocessor to the main microprocessor is pending.
4-7		When not 0000, the main microprocessor is accessing the partition.

Byte 2	High-order address of the beginning of the partition. Hex FF indicates this partition is not defined.
Byte 3	Page number in storage where this partition is located.

Hex Displacement	Length in Bytes (in Hex)	Description
0020	20	System use only.
0040	20	Diskette IOB Pointers (eight 4-byte blocks). Each 4-byte block has the following meaning:
		Byte 0 Flag Byte
		Bit(s) *Meaning*
		0 1 = The diskette microprocessor has locked the IOB pointer; the main microprocessor cannot use the IOB pointer while this bit is on.
		1-2 System use only.
		3 1 = A label update data set is open. The main microprocessor cannot put another IOB on this chain.
		4-7 When not 0000, the main microprocessor is using the IOB chain.
		Byte 1 The high-order address of the first IOB on the chain.
		Byte 2
		Bit(s) *Meaning*
		0 The low-order bit of the IOB address.
		1 1 = The diskette drive is installed for this IOB pointer.
		2-3 System use only.
		4-7 The page number in storage where the IOB is located.
		Byte 3 Diskette microprocessor save area.
0060	20	System use only.
0080	4	Printer IOB pointer (same meaning as a diskette IOB pointer, displacement 0040).
0084	1C	System use only.

Hex Displacement	Length in Bytes (in Hex)	Description
00A0	4	Communications CCB pointer (one 4-byte block). The 4-byte block has the following meaning:

Byte 0

Bit(s)	*Meaning When 1*
0	The CCB pointer is valid for use by the communications feature.
1	The CCB pointer is available for CAM use.
2	The CAM load parameter list is located at the address specified in bytes 1 and 2.
3	The CAM is loaded and ready to accept commands from the application program.
4-7	The partition number of the partition that initiated the loading of CAM.

Byte 1 When bits 0 or 1 of byte 0 = 1, this byte contains the high-order byte of the CCB address (relative to the beginning of the page). When bit 2 of byte 0 = 1, this byte contains the page number of the load parameter list.

Byte 2

Bit(s)	*Meaning*
0-3	System use only.
4-7	When bit 1 of byte 0 = 1, this byte contains the page number of the CCB storage location. When bit 2 of byte 0 = 1, this byte contains the high-order byte of the address of the load parameter list.

Byte 3 Hexadecimal FF if the communications microprocessor is installed.

00A8	10	System use only.

Hex Displacement	Length in Bytes (in Hex)	Description
00B8	4	Communications IOB Pointer: Byte 0 *Bit(s)* — *Meaning* 0 — The CAM has locked the IOB pointer; the main microprocessor cannot use the IOB if this bit is 1. 1-3 — System use only. 4-7 — If not 0000, the main microprocessor is accessing the chain. Byte 1 — The high-order address of the first IOB on the chain. Byte 2 *Bit(s)* — *Meaning* 0 — The low-order IOB address. 1-2 — System use only. 3 — 1=CAM is optional. 4-7 — The page number of the IOB location. Byte 3 — Hex FF if the CAM is operational.
00BC	4	System use only.
00C0	5	Date information as follows: Byte 0 — Year minus 1900. Bytes 1-2 — Day of the year. Byte 3 — Month (date is invalid if this byte = 00). Byte 4 — Day of the month.
00C5	2	Storage size as follows: Byte 0 — Number of 64 K-byte pages of storage. Byte 1 — Number of 256-byte blocks of storage on the last page.
00C7	1	IPL Flag.

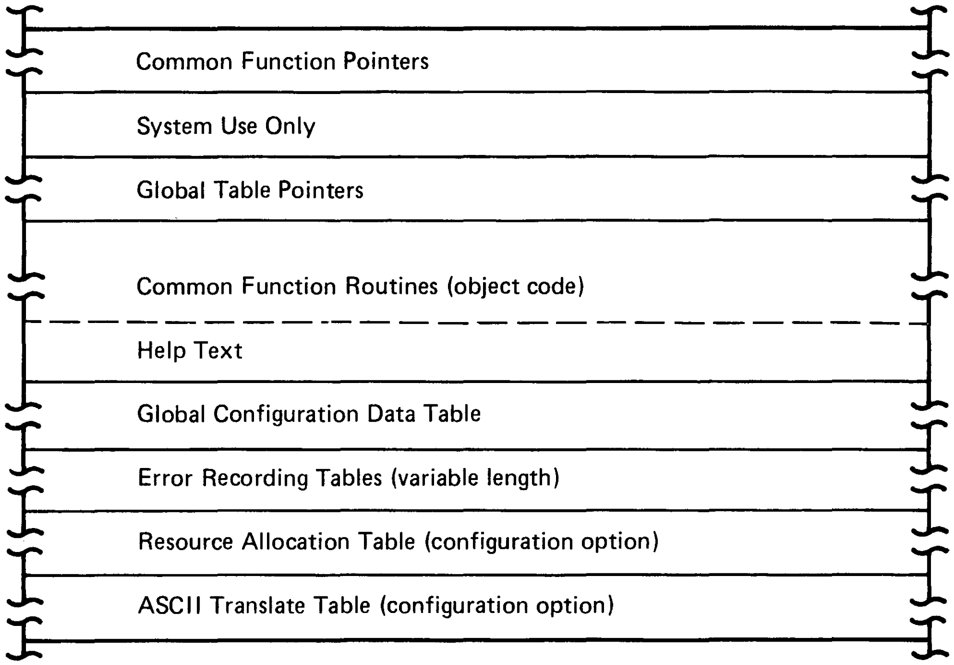
Hex Displacement	Length in Bytes (in Hex)	Description
00C8	2	High-order 2 bytes of time (1.6 seconds per count) since the system was powered on (if the elapsed time counter is installed). Updated by the keyboard/display microprocessor.
00CA	6	System use only.
00D0	8	System flags as follows: Byte 0 *Bit(s)* *Meaning* 0-3 System use only. 4 1 = The resource allocation table is in storage. 5-7 System use only. Byte 1 During IPL, hexadecimal FF indicates the main microprocessor is ready for IPL data to be loaded. Not hexadecimal FF indicates that a diskette microprocessor is loading IPL data. Byte 2 The device address of the IPL diskette. Byte 3 The IPL device subaddress. Bytes 4-5 System use only.
00D6	2	The address of the resource allocation table, relative to the beginning of page 0.
00D8	B	System use only.
00E3	7	Error log lockout bytes: Each device IOB pointer is assigned a bit in the first-level lockout bytes. Up to 8 device IOB pointers share a bit in the second-level lockout byte. For entry to the error log, all bits in the first-level byte containing the device IOB pointer lockout bit must be 0, and all bits in the second-level lockout byte must be 0. To use the table, the microprocessor must: (1) Set the bit in the first-level lockout byte corresponding to the IOB pointer of the device that has the error. All other bits in that byte must be 0. (2) Set the bit in the second-level lockout byte. All other bits in that byte must be 0. (3) Set bits 4-7 of byte 1 of the pointer to system table 0 to hex FF. This half-byte must have been zero.

Hex Displacement	Length in Bytes (in Hex)	Description
00E3 (cont.)		First-level error lockout bytes: Byte E3 0 Partition 0 1 Partition 1 2 Partition 2 3 Partition 3 4 Partition 4 5 Partition 5 6 Partition 6 7 Partition 7 Byte E4 System use only (must be zero). Byte E5 0 Diskette 0 1 Diskette 1 2 Diskette 2 3 Diskette 3 4 Diskette 4 5 Diskette 5 6 Diskette 6 7 Diskette 7 Byte E6 Printer Byte E7 System use only (must be zero). Byte E8 System use only (must be zero).
00E9		Second-level error lockout byte: *Bit(s)* *Use* 0 Partitions 0-7. 1 System use (must be 0). 2 Diskettes 0-7. 3 Printer and system use only (must be zero). 4-7 System use only (must be zero).
00EA	2	Address of the global configuration table, relative to the beginning of page 0. The address is set by the configuration utility.
00EC	2	Address of global self check control block, relative to the beginning of page 0.
00EE	2	Address of the global edit format table, relative to the beginning of page 0.

Hex Displacement	Length in Bytes (in Hex)	Description
00F0	1	System use only.
00F1	1	Main microprocessor configuration data; initialized by the configuration program as follows: *Bit(s)* *Meaning* 0-3 The number of partitions to scan. 4-7 The number of the partition at which to start scanning.
00F2	7	System use only.
00F9	2	Address of the system table pointers, relative to the beginning of page 0.
FB	1	The page number of the global screen format table and the global prompt table.
FC	2	The address of the global screen format table, relative to the beginning of the page specified in displacement FB.
FE	2	The address of the global prompt table, relative to the beginning of the page specified in displacement FB.

COMMON FUNCTIONS AND GLOBAL TABLES

The common functions and global tables begin at address hexadecimal 0100, and may include different areas depending on the system and whether the user selected IBM options. The following diagram is a general description of the common functions and global tables as they are if the common area SYSDPRT2 (the default area) is selected. If the user writes this code instead of using the IBM code, the addresses and areas are not required to be the same as described in the diagram. Following the general description is a complete description of the global tables.



Common Function Pointers

System Use Only

Global Table Pointers

Common Function Routines (object code)

Help Text

Global Configuration Data Table

Error Recording Tables (variable length)

Resource Allocation Table (configuration option)

ASCII Translate Table (configuration option)

Global Configuration Table

The address of the global configuration table is at hexadecimal EA, EB in main storage. This table contains information about the printer for the printer microprocessor. There are two header bytes, followed by an 8-byte entry for each printer configured. Hexadecimal FFFF indicates the end of the table. The following is the format of the 2 header bytes, and of the 8-byte entry.

Hex Displacement	Length in Bytes (in Hex)	Description
	2	Header Bytes:

Byte 1	The address of the printer IOB pointer (hexadecimal 0080).
Byte 2	

Bits	*Meaning*
0-3	The number of entries in the configuration table minus 1.
4-7	The length of a table entry minus 1 (hex 7).

Hex Displacement	Length in Bytes (in Hex)	Description
	8	Printer Entry:

Byte 1	Device subaddress.

Bits	*Meaning*
0-2	System use only.
3-4	Printer port number.
5-7	Printer station address.

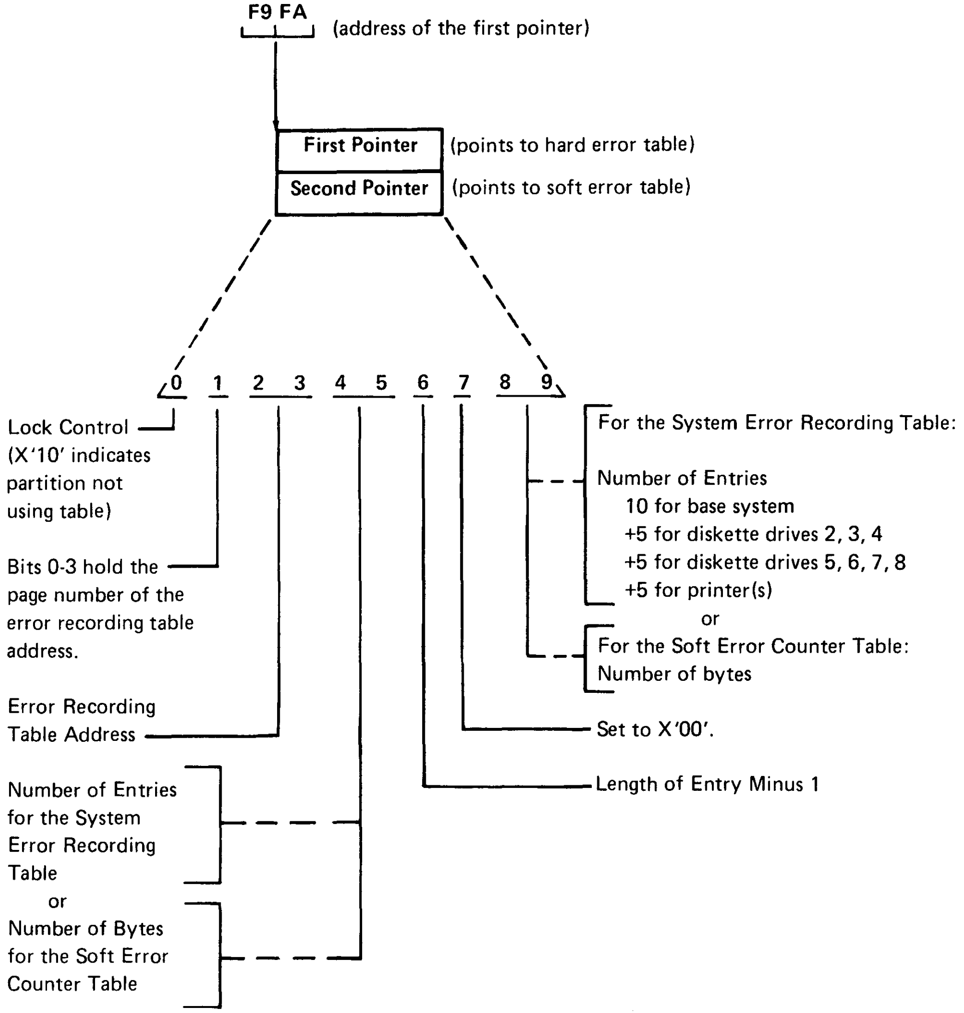
Bytes 2-3	Displacement into the table to this entry (must be nonzero).
Byte 4	Table length (hexadecimal 14).
Byte 5	Printer error encoding type: A0 = Bit encoding 20 = Byte encoding
Byte 6	Must be zero.
Byte 7	Number of 128-byte blocks in printer buffer (hex 02).
Byte 8	Must be zero.

Error Recording Tables

A system hard-error table and a soft-error table are stored in the common area. The system hard-error table is used by the microprocessors to record system hardware-related errors. The soft-error table is used by the printer microprocessor to record the number of I/O errors that occur during program execution. These error tables provide a history of system hardware-related errors and I/O errors that can be written to a diskette with a special error table dump program.

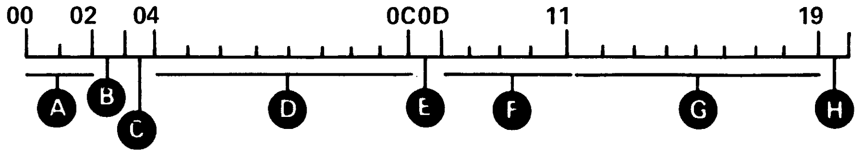
How to Find the Error Recording Tables

The error recording table pointers are stored in the common area, in the format of a system table. The 2-byte address of the error recording table pointers is located in the system control block, at displacement hex F9. The first pointer always contains the address of the hard error table, and the second pointer always contains the address of the soft error table. The pointers are 10 bytes in length, in the following format:


F9 FA
(address of the first pointer)
First Pointer
(points to hard error table)
Second Pointer
(points to soft error table)
0 1 2 3 4 5 6 7 8 9
Lock Control
(X'10' indicates partition not using table)
Bits 0-3 hold the page number of the error recording table address.
Error Recording Table Address
Number of Entries for the System Error Recording Table
or
Number of Bytes for the Soft Error Counter Table
Length of Entry Minus 1
Set to X'00'.
For the System Error Recording Table:
Number of Entries
10 for base system
+5 for diskette drives 2, 3, 4
+5 for diskette drives 5, 6, 7, 8
+5 for printer(s)
or
For the Soft Error Counter Table:
Number of bytes

Hard Error Table Format

The hard-error table can contain up to 25 entries. Each entry contains up to 26 bytes of error information in the following format:



A Error Code: See *Error Code Format.*

B Address of IOB Pointer[2]

C Device Subaddress (printer only)[1]

D Program Name

E IOB Identification:
Bits 0-3 = The logical unit number from the logical I/O table.[1]

Bits 4-7 = The partition number in which the IOB is stored.

F Device Status: See *Device Status* for a description.

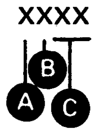
G Data Set Name[1]

H Number of Duplicate Errors: Maximum is X'FF'.

[1] This field contains all 0s for the keyboard/display MPU.

[2] For the keyboard/display MPU, this field points to the foreground partition associated with this keyboard. If the partition is a background partition, this field points to the foreground partition with which this background is associated.

Error Code Format



A Device Identification as Follows:

0 = Main microprocessor
1 = Keyboard/display microprocessor
2 = Printer microprocessor
3 = Diskette microprocessor
4 = SNA
5 = BSC
6-8 Not used
9 = Application program
C = Previous data set (SNA)
D = Previous data set (BSC)

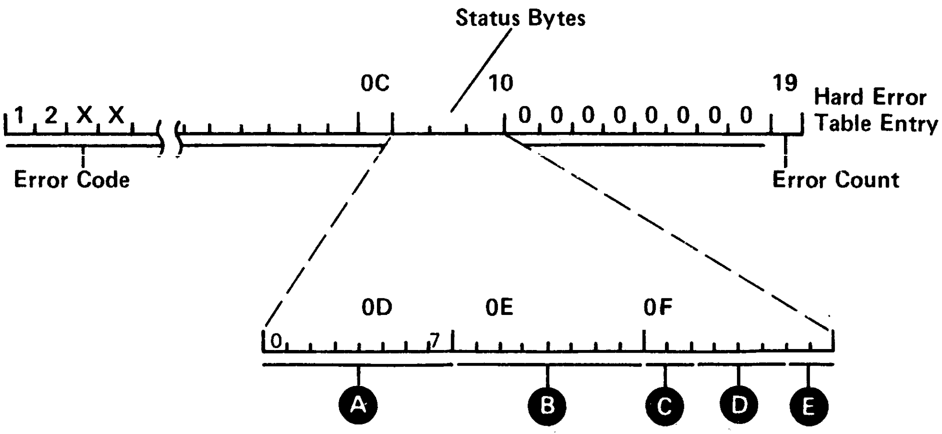
B Error Category as Follows:

1 = Intervention required
2 = Hard error (operation is not retried)
3 = Retriable error (retried x times)
4 = IOB error (user error)
5 = Soft error (retried successfully)
6 = Exception status (such as the Cancel key pressed on the 5256 printer)
7 = Warning error (user can continue)
8 Not used
9 = User program terminated

C Specific Error Condition: See the *IBM 5280 Message Manual,* GA21-9354, for a description of specific conditions.

Device Status

For the Keyboard/display MPU: The device status bytes have the following meaning for error codes 1200, 1201, and 1202. For 1204 all status bytes are undefined.



A For error codes 1200 and 1201, status 0D has the following meaning:

Bit(s)	*Meaning*
0	1 = A keyboard/display feature card is installed.
1	1 = Keyboard/display storage selected is on a feature storage card. (see Note)
2	1 = Keyboard/display storage selected is on the keyboard/display MPU card. (see Note)
3-7	Keyboard/display storage access status as follows: 01111 = Accessed by the keyboard/display MPU or by a translate cycle. 10111 = Storage accessed by display 1 hardware. 11011 = Storage accessed by display 2 hardware. 11101 = Storage accessed by display 3 hardware. 11110 = Storage accessed by display 4 hardware.

For error code 1202, byte 0D contains the invalid scan code.

B For error codes 1200 and 1201, byte 0E has the following meaning:

Bit(s) *Meaning*

0-2 High-order bits of the keyboard/display storage address when the error occurred.

3 0 = The error occurred when translating and writing to the display refresh buffer.

4 0 = The last storage access was for a read operation.

1 = The last storage access was for a write operation (diagnostic use only).

5 Parity is even (should be even for a read; can be either for a write).

6-7 Indicates model as follows:

00 = 5288
01 = Not Used
10 = 5286
11 = 5285

For error code 1202, byte 0E contains the EBCDIC translation for the invalid scan code.

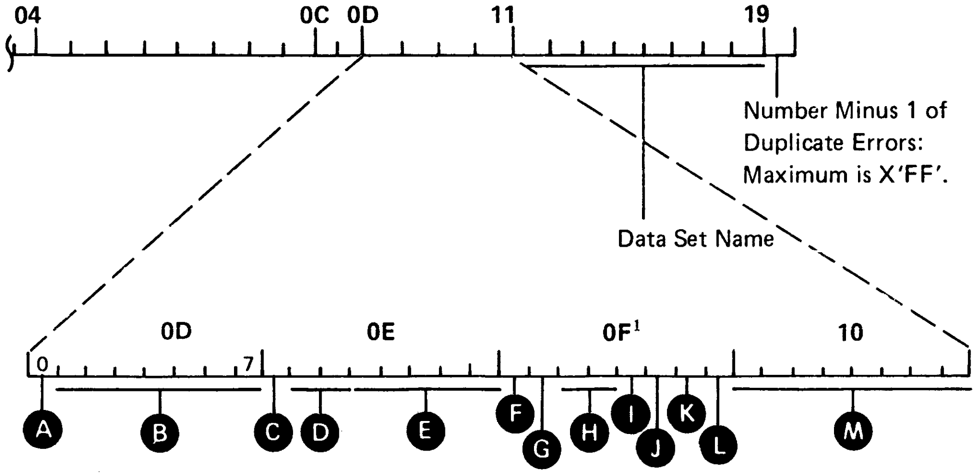
C Not Used

D 0000

E Number of Keyboards Detected (as attached) By the Keyboard/Display MPU

Note: For error codes 1200 and 1201, if bit 1 and bit 2 of status 0D are both 0, an invalid address was accessed.

For the Diskette MPU: The device status bytes have the following meaning.



A Failing Head Number:

0 = Head 0
1 = Head 1

B Failing Track Number

C 0 = FM
1 = MFM

D Sector Size:

00 = 128
01 = 256
10 = 512
11 = 1024

E Failing Sector Number—valid for the following codes:

3301
3302
3303
3306
3501
3502
3503
3506

F CRC error occurred

G ID found during search

H 01 = Control address mark (AM) was detected
10 = Missing address mark
11 = Bad track accessed

I Storage overrun: The diskette MPU was unable to obtain the required storage cycles to transfer data.

J Error during a verify read operation.

K Command not complete: The diskette MPU has not completed the operation requested.

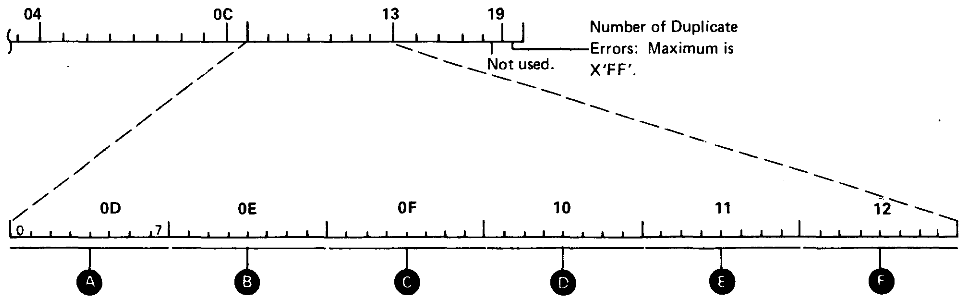
L When 1, the write or erase gate was active during a read operation; or the write or the erase gate was not active during a write operation.

M Command sent to the diskette adapter by the diskette MPU.[2]

[1] If byte 0F is equal to FF, the track contains no IDs.

[2] If the command is hex Ax or 2x, bytes 0D and 0E may not be valid.

For the Printer Attachment MPU: The status bytes have the following meaning.



A Not used.

B First Poll Response Byte

Bit(s) *Meaning*

0 The printer MPU is busy.

1 The printer received bad data—parity error.

2 The printer is not ready.

3 The printer has outstanding status, which must be read by the printer attachment MPU.

B 4-6 *Exception status from the printer:*

000 = No exception status.

010 = The printer received an invalid activate command. A read command must be followed by a read activate command and a write command must be followed by a write activate command.

011 = Undefined exception status.[1]

100 = The printer received an invalid command.

101 = Printer storage overrun: The printer received too much data or too many commands.

110 = Undefined exception status.[1]

111 = The printer was powered off and then powered on.

7 Not used by the 5280.

C Second Poll Response Byte

Bit *Meaning When 1*

0 The printer received an invalid SCS character (usually a programming error).

1 The printer received an invalid SCS parameter (usually a programming error).

2 The printer receive buffer is full.

3 The printer operation is complete.

4 The Cancel Request key was pressed on the printer.

5 The printer mechanism is not ready (usually a voltage missing at the printer).

6 End of forms

7 The printer received an unprintable character (this bit should only be on if the SGEA command is set to stop).

D Outstanding Status from Printer

Bit	*Meaning When 1*
0	Print wire check
1	Emitter slow speed check
2	Emitter fast speed check
3	Emitter sequence check
4	No emitter pulses
5	Emitter overrun: Printer MPU cannot keep up with the emitter pulses.
6	Forms stopped
7	Forms position check

E Encoding Type:

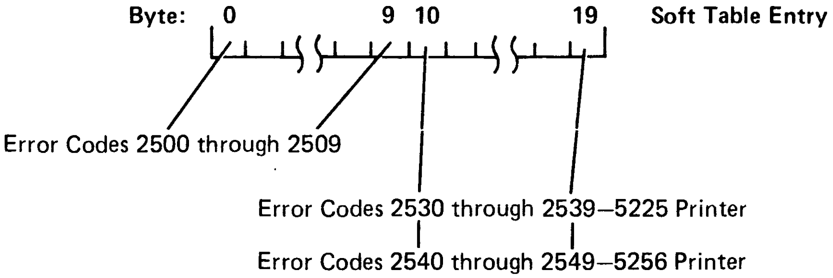
A0 = bit encoding
20 = byte encoding

F Must be 00.

[1] This exception status should not be received from the printer. If it is, it usually indicates a line hit.

Soft Error Table Format

The printer soft error recording table contains a count for each printer soft error that occurred. The table contains one entry for each printer on the system. Each entry is 20 bytes long, and each byte is assigned to a specific error code as follows:



Byte: 0
9 10
19
Soft Table Entry
Error Codes 2500 through 2509
Error Codes 2530 through 2539—5225 Printer
Error Codes 2540 through 2549—5256 Printer

If a count byte reaches 255 (hexadecimal FF), the error is stored in the hard-error table and is no longer counted.

Resource Allocation Table

The resource allocation table defines the physical address of each logical device that can be used by each partition. The table is created and initialized as a user option during the system configuration portion of the SCP. If a resource allocation table has been created and placed into the common area, the system flag at address hex D0, bit 4 is 1; the address of the table is at hex D6-D7 in the system control block.

The resource allocation table consists of a 4-byte partition header for each partition, followed by a 4-byte device entry for each device that can be used by that partition. When the main microprocessor attempts to open an IOB that specifies a logical device ID instead of a physical address, it uses the resource allocation table to find the physical address. The main microprocessor searches the table until it finds the first entry for that partition or the first global partition entry. It then searches the device entries for a matching device ID. If no match is found, it continues searching the partition headers for another entry for the partition or another global partition entry. The search continues until the matching device ID is found or until the table is exhausted. If no match is found, an error is reported. If a match is found, the device at the physical address specified in the table is used to open the data set.

The format of the partition header records and the device entries are as follows.

Partition Header

Byte:	0	1	2	3
	Partition ID	Number of Entries		

Bytes **Meaning**

0 Partition ID Number:

FO = Global entry (each device entry applies to all partitions)
FF = (see the description for bytes 2 and 3)

Bit(s) *Meaning*

3 1 = The partition number in bits 4 through 7 is not valid.
0 = A valid partition number is in bits 4 through 7.
4-7 The number (0 through 7) of the partition to which the entries apply.

1 The number (0 through 255) of device entries for this partition.

2-3 If bytes 0 and 1 contain hexadecimal FFFF (indicating the end of the table), bytes 2 and 3 indicate the number of bytes still available in the table.

If byte 0 is hex FF and byte 1 is not, byte 1 contains the page number and bytes 2 and 3 contain the remainder of the address of the next section of the resource allocation table.

Device Entries

Byte:	0	1	2	3
	Device ID		Device Physical Address	

Bytes	**Meaning**
0-1	*Device ID:* The EBCDIC code for the logical device ID. This ID is compared to the ID specified in the IOB during an open, and when they are equal, the device with that physical address is opened.
2-3	*Physical address:* Byte 2 is the address of the IOB pointer for this device. During an open, the main microprocessor moves byte 2 into the logical I/O table, and byte 3 to the IOB.

ASCII Translate Table

The ASCII translate table contains two 256-bytes sections. The first section is used for input, to translate EBCDIC notation to ASCII. The second 256-byte section is used for output, to translate ASCII notation to EBCDIC. The hex value of each character is used as an offset into the appropriate translate table, and the original hex value is replaced with the hex value at that offset.

PARTITION AREA

The partition area contains the program executed by the main microprocessor and the information required to execute these programs.

All addresses shown in the following description are relative to the beginning of the partition area.

Hex Displacement	Length in Bytes (in Hex)	Description
0000	40	Partition IOB (see *Partition IOB*).
0040	40	Logical I/O table (see *Logical I/O Table*).
0080	80	Keyboard/display IOB (see *Keyboard/Display IOB*).
0100	80	Indicators I000 through I254, binary registers BR0 through BR15, and/or decimal registers R0 and R1. (See *System Indicators Within a Partition* for a list of indicators that are used by system microprocessors.)
0120	Variable (224 if all binary registers are used)	Binary register BR16 through BR127 and/or decimal register R2 through R15. (See *System Registers Within a Partition* for a list of registers that are used by system microprocessors.)
Variable (0200 if all binary registers are assigned)	Variable (3584 if all decimal registers are used)	Decimal registers R16 through R239.
Variable (pointed to in the partition IOB)	Variable	Object code, buffers, tables, diskette and printer IOBs (see *Diskette IOB* and *Printer IOB* later in this chapter), work areas, and other user program areas.
Variable	256	The last 256 bytes of a partition area are used as a microprocessor work area.

Partition IOB

The following is a general description of the partition IOB. Following this general description is a complete description of each field of the IOB. All addresses shown are hexadecimal displacements from the beginning of the partition. No validity checking is made on any of the values in the bytes of the following IOB. If any of these bytes are modified by the application program, unpredictable results may occur.

00	**Program Name** 4							
08	Partition Length 5	Control Flags 3,6		Main Microprocessor Error Code 2,3,6	System Use Only 6	Program Start Address, High 3,4	Absolute Address of Program 3,6	System Use Only 6
10	Instruction Address Pointer 3,4		Common Area Page Number 1,3,5	Page and Current Instruction Flags 3,5	Partition Page Number 2,3,6	Program Length 2,3,4	Address of Program Check Routine 2,3,4	
18	Address of System Table for Data Tables 1,3,4		Main Microprocessor Save Area 3,6		Program Execution Timer 2,3,4	Number of Last Edit Format 2,3,6	Currency Edit Characters 3,4	
20	Decimal Edit Character 3,4	Comma Edit Character 3,4	Edit Count 3,4	Partition Number 2,3,6	Address of System Table for Edit Formats 1,3,4		Address of Self Check Control Block 1,3,4	
28	System Use Only 6		Load Flags 3,6		Save Area For Subroutine Return Address 3,6		Save Area for Trace 3,6	Remaining Number of Bytes to Load 3,6
30	Page of Partition Being Loaded 3,6	IOB Address of Partition Being Loaded 3,6	Address of Partition Work Buffer 3,6	Part. Number of Partition Being Loaded 3,6	IOB Pointer Address for Diskette 3,6	I/O Flags 3,6	Current Instruction Address 3,6	
38	Page Number of Data to Dump 3,6	Address of Data to Dump, High 3,6	Number of Bytes to Dump 3,6	Trace Flags 3,6	Address-Stop Instruction Address 3,6		Configuration Information 3,6	System Use Only 6

1. Application program can read or write this field.
2. Application program can only read this field.
3. Used by the main microprocessor.
4. This field must be initialized by the object module.
5. This field must be initialized at IPL time.
6. This field must be zero in the object module.

Hex Displacement	Length in Bytes (in Hex)	Description
00	8	Program Name (eight EBCDIC characters that identify the program in this partition).
08	1	Partition length in number of 256-byte blocks minus 1.
09	1	Control Flags: *Bit(s)* *Meaning* 0 1 = IOB is initialized (a program is loaded). 1 1 = Keyboard is attached to this partition. 2-7 System use only.
0A	1	Control Flags: *Bit(s)* *Meaning* 0 1 = Tracing through a Call or Return instruction, to or from the common function area. 1 System use only. 2 1 = Processing a newly invoked Cmd, C key sequence. (Waiting for keystroke after the Cmd and C keys have been pressed.) 3-6 System use only. 7 1 = Waiting for a response (ENTR) for the global load prompt.
0B	1	Main microprocessor error code.
0C	1	System use only.
0D	1	High-order part of the program start address.
0E	1	Absolute program start address. Used by the microprocessor when returning from a common area subroutine.
0F	1	System use only.
10	2	Points to the next instruction to be executed when the microprocessor begins executing code in the partition. When the microprocessor begins executing code in this partition for the first time, it adds the value in byte 0D of this IOB to this address to make it an absolute address. The microprocessor then updates this address before leaving the partition.

<table>
<tr><th>Hex Displacement</th><th>Length in Bytes (in Hex)</th><th>Description</th></tr>
<tr><td>12</td><td>1</td><td>Page number of the common functions as follows:
<table>
<tr><th><em>Bit(s)</em></th><th><em>Meaning</em></th></tr>
<tr><td>0-3</td><td>Page number of common function area 1.</td></tr>
<tr><td>4-7</td><td>System use only.</td></tr>
</table></td></tr>
<tr><td>13</td><td>1</td><td>Page and current instruction flags as follows:
<table>
<tr><th><em>Bit(s)</em></th><th></th><th><em>Meaning</em></th></tr>
<tr><td>0-1</td><td>01 =</td><td>Instructions being executed are in the common function area.</td></tr>
<tr><td></td><td>10 =</td><td>Instructions being executed are in the partition.</td></tr>
<tr><td></td><td>11 =</td><td>System use only.</td></tr>
<tr><td>2-3</td><td></td><td>System use only.</td></tr>
<tr><td>4-7</td><td></td><td>Page number for the instruction currently being executed.</td></tr>
</table></td></tr>
<tr><td>14</td><td>1</td><td>Number of the page where this partition is located (range 0-2).</td></tr>
<tr><td>15</td><td>1</td><td>Length of the program in this partition (in 256-byte blocks).</td></tr>
<tr><td>16</td><td>2</td><td>Address of program-check subroutine.</td></tr>
<tr><td>18</td><td>2</td><td>Address of the system table that contains 8-byte entries, each of which defines a table in this partition. (See <em>System Table for Data Tables</em> under <em>System Tables</em> for the format of the table entries.)</td></tr>
<tr><td>1A</td><td>2</td><td>Microprocessor save area</td></tr>
<tr><td>1C</td><td>1</td><td>Program execution time (time slice timer) as follows:
<table>
<tr><th><em>Bit(s)</em></th><th></th><th><em>Meaning</em></th></tr>
<tr><td>0</td><td>1 =</td><td>Ignore attentions.</td></tr>
<tr><td>1</td><td>1 =</td><td>Keyboard external status occurred after a RESUME or CNENTR operation.</td></tr>
<tr><td>2</td><td>1 =</td><td>Keyboard external status occurred during a nonoverlapped, nonkeyboard operation.</td></tr>
<tr><td>3</td><td></td><td>System use only.</td></tr>
<tr><td>4-7</td><td></td><td>Length of time divided by 4 (in milliseconds) to execute instructions in the partition (initialized by the assembler).</td></tr>
</table></td></tr>
<tr><td>1D</td><td>1</td><td>Binary number assigned to the last edit format index used.</td></tr>
</table>

Hex Displacement	Length in Bytes (in Hex)	Description
1E	2	Characters used as the edit currency characters during formatted read or write operations.
20	1	Character used as the edit decimal character during formatted read or write operations.
21	1	Character used as the edit comma character during formatted read or write operations.
22	1	Number of characters between edit commas during formatted read or write operations.
23	1	The partition number assigned to this partition (00-07; set by the microprocessor at load time).
24	2	Address of the edit format system table (table that contains 2-byte entries that point to the local edit format strings stored within the partition).
26	2	Address of the self check control block.
28	2	System use only.
2A	2	Load Flags as follows:

Byte 2A

Bit		*Meaning*
0	1 =	Foreground partition is loading the program.
	0 =	Background partition is loading the program.
1	1 =	Program is being loaded in a foreground partition.
	0 =	Program is being loaded in a background partition.
2	0 =	Program is loading a program into this same partition.
	1 =	Program is loading a program into another partition.
3	1 =	The loader issued an ENTR command.
4	1 =	Partial load.
	0 =	Regular load.
5	1 =	Attach a background partition if possible.
6		System use only.
7	1 =	User error recovery specified.

Hex Displacement	Length in Bytes (in Hex)	Description
2A (cont.)		Byte 2B

Bit	*Meaning When 1*
0	User external status routines are not available.
1	One automatic retry was attempted.
2	Close was issued by the loader.
3	Closing open data sets.
4	Error message requested by the loader.
5	Global screen format used.
6	EXIT in foreground partition.
7	Detach the background partition at the end of the load operation.

Hex Displacement	Length in Bytes (in Hex)	Description
2C	2	Microprocessor save area.
2E	1	Trace save area.
2F	1	Number of 256-byte blocks left to load.
30	1	Storage page number in which partition is being loaded.
31	1	Address of partition IOB for partition being loaded (used only while loading a program).
32	1	Address of partition work buffer for partition being loaded (used only during load operation).
33	1	Partition pointer address of the partition being loaded.
34	1	Address of diskette IOB Pointer for diskette doing the load operation (used only while loading a program).
35	1	I/O Flags:

Bit	*Meaning When 1*
0	Nonoverlapped I/O pending.
1	Nonoverlapped ENTR pending.
2	Formatted read requested.
3	System use only.
4	Keyboard operation pending.
5	During SCS conversion, the logical buffer has data remaining to be converted.
6	Console function request pending.
7	ENTR issued by loader.

Hex Displacement	Length in Bytes (in Hex)	Description
36	2	Address of current instruction (used during trace).
38	1	Page number of data to dump.
39	1	High-order address of data to dump.
3A	1	Number of 256-byte blocks of data to dump.
3B	1	Trace Flags: *Bit* *Meaning When 1* 0 Address-stop address reached. 1 Address-stop request pending. 2 First time through trace or address-stop. 3 Dump flag. 4 Request trace of binary instructions. 5 Request trace of miscellaneous instructions. 6 Request trace of decimal instructions. 7 Request trace of branching instructions.
3C	2	Instruction address after which execution stops during step-stop mode.
3E	1	Configuration information: *Bit(s)* *Meaning* 0-3 Number of partition pointers remaining to scan. 4-7 Number of the partition pointer at which the main microprocessor begins scanning.
3F	1	System use only.

Logical I/O Table

The logical I/O table is located at relative displacement hex 40 through hex 7F of each partition. Each data set IOB is represented with a 4-byte entry in the logical I/O table. The table entries are stored within the table in the order of the data set numbers.

The logical I/O table entries have two formats: one format is used for the keyboard/display IOB and the other format is used for diskette IOBs and printer IOBs.

The following is a general description of the logical I/O table. Following the general description is a complete description of the two formats that are used for the logical I/O table entries. The address of each table entry is the data set number times 4 plus hex 40.

Keyboard/Display Format

Byte **Meaning**

0 Not used (initialized to hexadecimal 00).

Flags:

Bit(s)		*Meaning*
0	1 =	Special processing is required. This bit is set on when an external status condition is detected during a RESUME, CANCEL, or ENTR operation and indicates that external status processing is required after these operations have completed.
1	1 =	An ENTR command is pending.
2-3		Not used, set to zero.
4	1 =	A keyboard/display operation is pending.
5-7		Not used, set to zero.

2-3 Address of the keyboard/display IOB (always 0080).

Diskette and Printer Format

Byte	**Meaning**
0	IOB pointer address.
1	I/O Class:

Bit(s)		*Meaning*
0-1		I/O type:
	01 =	Output only
	10 =	Input only
	11 =	Input and output
2	1 =	The main microprocessor is waiting to remove an IOB from an IOB chain.
3	1 =	The main microprocessor is waiting to add an IOB to an IOB chain.
4	1 =	The main microprocessor is waiting to lock the first IOB on a chain during an open.
5		Not used
6	1 =	The IOB is on an IOB chain.
7	1 =	An I/O request is pending for this device.

2	The high-order address of the IOB.
3	Flags:

Bit(s)		*Meaning When 1*
0		Low-order address of the IOB.
		Device type as follows:
1	1 =	Diskette
2		Not used
3	1 =	Printer
4	1 =	Not used
5	1 =	Communications
6-7		Not used

Note: Only one of bits 1-7 may be on at any one time.

Keyboard/Display IOB

The keyboard/display IOB is located at relative displacement hex 80 through hex FF of each partition.

The following is a general description of the keyboard/display IOB. Following this general description is a complete description of each field of the IOB. All addresses shown are hexadecimal displacements from the beginning of the IOB. To find the address relative to the beginning of the partition, add hex 80 to the displacement. No validity checking is made on any of the values in the bytes in the following IOB. If any of these bytes are modified by the application program, unpredictable results may occur.

00	IOB System Status 2	Foreground Background Flags 2,7,8,9	IOB Pointer Address 2,8,9	IOB Lockout 2	Error Code 2		Next Instruction Address 2	
08	Command Op Code 2	Current Screen Format Control String Byte Address 2		Current Prompt Table Address 2		Partition Prompt Table Address 3,6,7		External Status 2
10	External Status Routine Address 2,7		Keyboard Flags 2	Keyboard Flags 2	System Use Only	Last Diacritic Character 2	Save Area for Address of Current Screen Format Control String Byte 2	
18	Keyboard Bit Map 1,7						Return Key Code 2	Operation Code 2
20	Operation Parameter 1 2		Operation Parameter 2 2		Operation Parameter 3 2		Last Key Scan Code 2	Last Key EBCDIC 2
28	Current Field Address in Main Storage 2		Current Field Address in Keyboard/Display Storage 2		Current Record Buffer Address 3,4,7		Previous Record Buffer Address 3,4,7	
30	Displacement to Current Character in the Buffer 2		Address of the Current Cursor Position 2		Alphabetic Right-Adjust Character 6,7	Numeric Right-Adjust Character 6,7	Address of Storage Duplication Table 3,4,7	
38	Keyboard Flags 5		Keyboard Flags 2		Keyboard Flags 5	Keyboard Flags 2	Keyboard Flags 5	
40	Picture Check Displacement 2	Information from Screen Format Control String 2						
48	Information (Continued)	Picture Check Subfield Counter 2	Fixed Prompt Line 3,4,7	Keystroke Counter 6,8			Verify Correction Keystroke Counter 6,8	
50	Address of Storage Area in Keyboard/Display Storage 2,8		Address of Diacritic Translate Table 2,8		Address of Status Line Refresh Buffer 2,8		Address of Katakana Trans. Tbl. 2,8	Address of Scan Code Trans. Tbl. 2,8
58	Address of Screen Refresh Buffer 2,8		Number of Lines on Screen 2,8	Number of Characters per Line 2,8	Keyboard Configuration Information 2,8		Address of Validity Table 2,8	Display Line Map 2,8
60	Display Line Map (continued) 2,8			Language Group 2,8	Address of Cursor Address Register 2,8	Address of Control Area 2,8	Address of Display Control Register 2,8	System Use Only
68	Current Character Position within the Field 2		Current Field Number 2		Address of the Fixed Prompt Line 2		Current Position Counter 2	

<table>
<tr><td>70</td><td colspan="2">Positions Remaining in the Field
2</td><td colspan="2">Current Record Buffer Position
2</td><td>Normal Display Attribute
3,4,7</td><td>High Intensity Display Attribute
3,4,7</td><td colspan="2">Microprocessor Save Area
2</td></tr>
<tr><td>78</td><td>EBCDIC for Blank Check
3,4,7</td><td colspan="2">Address of Partition Screen Format Control String Table
3,4,7</td><td>EBCDIC for Verify Mismatch
2</td><td>EBCDIC for Duplication Mismatch
2</td><td>Microprocessor Save Area
2</td><td>Microprocessor Work Area
2</td><td>Main Microprocessor Lockout 2</td></tr>
</table>

1. An application program can change this field at any time.
2. An application program should not change this field.
3. Normally, an application program will not change this field.
4. An application program can change this field, but only when an ENTR command is not being processed.
5. See field descriptions for restrictions.
6. An application program can change this field, but only when an ENTR command is not being processed or when EXTR processing is suspended, such as during external status processing (before a resume is issued).
7. Initialized by the assembler.
8. Initialized during IPL.
9. Initialized by the program loader.

Hex Displacement	Length in Bytes (in Hex)	Description
00	1	IOB System Status:

Bit(s)		*Meaning*
0-1	11 =	Main microprocessor sent a command to the keyboard/display microprocessor: It cannot send another until the keyboard/display microprocessor sets the bits to 00.
	01 =	Keyboard/display microprocessor accepted the command; however, processing is not yet complete.
	00 =	No command pending.
2	1 =	Keyboard/display microprocessor is awaiting an operator response to a keyboard operation. The keyboard/display microprocessor sets this bit on if bit 4 of this byte is on and the operation code is hexadecimal 09 (place key-entered data in main storage).
3	1 =	External status sensed; the keyboard/display microprocessor sets this bit to 1 when it senses an external status condition. The main microprocessor clears the bit when it begins processing the external status. See also bit 7 in this byte.
4	1 =	Main microprocessor has requested a keyboard/display operation. The keyboard/display microprocessor clears the bit after it completes the operation.

Hex Displacement	Length in Bytes (in Hex)	Description			
00 (cont.)		*Bits*			*Meaning*
		5			System use only.
		6	1	=	Current command processing has been temporarily interrupted. The keyboard/display microprocessor sets this bit on when a command is interrupted (time slice elapsed) and clears the bit when it resumes processing the command.
		7	1	=	An external status condition is pending or is being processed. The keyboard/display microprocessor sets the bit when it senses an external status. The main microprocessor clears the bit after it has processed the external status condition, or the keyboard/display microprocessor clears the bit when it receives a RESUME operation that requests enable external status.
01	1	Foreground and Background Flags:			
		Bits			*Meaning*
		0	0	=	Keyboard data is for the foreground partition.
			1	=	Keyboard data is for the background partition.
		Bit 0 is meaningless in a background partition.			
		1	1	=	Keyboard operation in progress that requires input from the keyboard. When the operation is complete, the keyboard/display microprocessor sets the bit to 0.
		Bit 1 is meaningless in a background partition.			
		2	1	=	Background partition is attached; meaningless in a background partition.
		3	1	=	Bits 4-7 do not contain a valid partition number.
		4-7			In a foreground partition, bits 4-7 contain the number of an attached background partition (bit 2 = 1 and bit 3 = 0) or the number of the background partition that has a keyboard operation in progress (bit 2 = 1 or 0 and bit 3 = 0). In a background partition, bits 4-7 contain the foreground partition number with which this background is associated.

Hex Displace-ment	Length in Bytes (in Hex)	Description
02	1	The low-order byte of the address (in the system control area) of the 4-byte IOB pointer for this partition.
03	1	IOB Lockout: *Bit(s)* *Meaning* 0 1 = Keyboard/display microprocessor is using this IOB. The main microprocessor cannot use this IOB while this bit is on. 1-3 System use only. 4-7 When bits 4-7 are not 0, the main microprocessor is using the IOB.
04	2	The keyboard/display microprocessor uses these 2 bytes to store keyboard error codes. (Error codes are stored in 4-digit, zone-stripped format.)
06	2	The absolute address of the next sequential instruction following the command or operation issued to the keyboard/display microprocessor.
08	1	The op code of the command currently being processed by the keyboard/display microprocessor. When this byte = 00, no ENTR command is being processed.
09	2	During the processing of the ENTR command, bytes 09 and 0A contain the address (relative to the beginning of the partition or to the beginning of the page that contains the global table) of the first byte in the screen format byte group that is currently being processed. Before the keyboard/display microprocessor begins processing the ENTR command, bytes 09 and 0A have the following meaning: Byte 9 *Bit* *Meaning* 1 0 = Screen format control string is in the partition. 1 = Screen format control string is in the global area. Byte 0A The table entry number of the screen format control string to be used for this ENTR command.

Hex Displacement	Length in Bytes (in Hex)	Description
0B	2	While an ENTR command is being processed, these bytes contain the address of the first byte of the current prompt table. If the prompt table is in the partition, the address is relative to the beginning of the partition. If the prompt table is in the global area, the address is relative to the beginning of the storage page that contains the global area.
0D	2	Address of the partition prompt table relative to the start of the partition.
0F	1	External Status Information: *Bit(s)* *Meaning* 0-1 System use only. 2 1 = Indicates that the keyboard/display and main microprocessors are operating in diagnostic mode, such as dump or trace. 3-7 External status condition number (see the *Assembler Language Reference Manual* for a description of external status conditions).
10	2	Address, relative to the beginning of the partition, of the table or subroutine that is used when processing external status conditions.
12	1	Keyboard Flags: *Bit(s)* *Meaning* 0 0 = There is more than one external status subroutine. They are accessed via a subroutine table. 1 = External status is handled by one subroutine. 1-4 System use only. 5 0 = If byte 13, bit 5 is 1, Katakana keyboard lock is alphameric lowercase. 1 = If byte 13, bit 5 is 1, Katakana keyboard lock is Katakana lowercase. 6-7 00 = Katakana keyboard shift default is alphameric lowercase. 01 = Katakana keyboard shift default is alphameric uppercase. 10 = Katakana keyboard shift default is Katakana lowercase. 11 = Invalid value.

Hex Displacement	Length in Bytes (in Hex)	Description
13	1	Keyboard Flags:

Bit(s)		*Meaning*
0	0 =	Katakana Shift Lock key is up.
	1 =	Katakana Shift Lock key is held down.
1	0 =	Katakana Uppershift key is up.
	1 =	Katakana Uppershift key is held down.
2	0 =	Katakana Lowershift key is up.
	1 =	Katakana Lowershift key is held down.
3	0 =	For Katakana, alphameric uppershift key is up. For non-Katakana, numeric shift key is up.
	1 =	For Katakana, alphameric uppershift key is held down. For non-Katakana, numeric shift key is held down.
4	0 =	For Katakana, alphameric lowershift key is up. For non-Katakana, alpha shift key is up.
	1 =	For Katakana, alphameric lowershift key is held down. For non-Katakana, alpha shift key is held down.
5	0 =	For Katakana, keyboard shift is not locked. For non-Katakana, shift default is lowercase.
	1 =	For Katakana, keyboard shift is locked. The type of lock is specified in byte 12, bit 5. For non-Katakana, shift default is uppercase.
6-7		Current shift for all keyboards:
	00 =	Alphameric lower
	01 =	Alphameric upper
	10 =	Katakana lower
	11 =	Katakana upper

Hex Displacement	Length in Bytes (in Hex)	Description
14	1	System use only.
15	1	EBCDIC of the last diacritic character entered.
16	2	Save area for the address of the current format control string (bytes 09 and 0A) while a secondary format control string is being processed.
18	6	Keyboard bit map: contains a bit for each function key that can be processed either partially or totally by the keyboard/display microprocessor or by an object code program. If the bit is 1, the corresponding function key is processed by the object code program. See Appendix C, *Keyboard Functions: EBCDIC Codes and Bit Numbers* for a description of the bit map and corresponding functions.
1E	1	EBCDIC of the key that caused a return from screen format processing to the object code program.
1F	1	Op code of the keyboard/display operation issued by the main microprocessor (see Chapter 4).
20	2	Parameter 1 for the op code in byte 1F.
22	2	Parameter 2 for the op code in byte 1F.
24	2	Parameter 3 for the op code in byte 1F.
26	2	Contains the scan code (byte 26) and EBCDIC value (byte 27) of the last key pressed as follows: ● The function key during external status condition 1. ● The first character of a hexadecimal pair. ● The keystroke following the Command key during external status condition 8. ● The keystroke that caused the error during external status condition 8. ● The function key when the microprocessor processes a function key other than the Command, Reset, or Shift key.
28	2	Address in main storage (relative to the beginning of the partition) of the first byte of the field currently being entered or processed.

Hex Displacement	Length in Bytes (in Hex)	Description
2A	2	Address in the refresh buffer (keyboard/display storage) of the first byte of the field currently being entered or processed.
2C	2	Address (relative to the beginning of the partition) in main storage of the first byte of the current record buffer for data being entered.
2E	2	Address (relative to the beginning of the partition) in main storage of the previous record buffer that normally contains the previous record entered (used for record duplication).
30	2	Displacement from the beginning of the current record buffer to the current character position.
32	2	The absolute address of the current cursor position or screen position pointer within the refresh buffer in keyboard/display storage. (The cursor is displayed within data entry fields only. The screen position pointer is maintained between fields.)
34	1	The EBCDIC character that is inserted into the nondata positions of an alphabetic, right-adjust field when the right-adjust is performed.
35	1	The EBCDIC character that is inserted into the nondata positions of a numeric, right-adjust field when right-adjust is performed.
36	2	Address in main storage, relative to the beginning of the partition, of the storage duplication table.

Hex Displacement	Length in Bytes (in Hex)	Description
38	1	Keyboard Flags:

Bit(s)		*Meaning*
0	1 =	Operation is continued; the keyboard/display microprocessor sets this bit on when a keyboard operation (such as a data movement) is executed by separate, successive operations. The application program should not change this bit.
1		Set by the keyboard/display microprocessor to indicate that the key code is from an object code program instead of the keyboard. Indicates that the microprocessor must perform functions that are not performed when the code is from the keyboard. The application program should not change this bit.
2	1 =	At least one field of the record has been processed. Used to determine if a record advance operation should be performed. The application program should not change this bit.
3-5		Used by the keyboard/display microprocessor to keep track of format control string processing when check indicator for bypass specifications are encountered in the string. The application program should not change these bits.
6-7		Meaning of the current position displayed on the status line. Normally, no change is made in these 2 bits by the application program. If the application program does change these bits, it should not change them while an ENTR command is being processed.
	00 =	Position of the next character to be entered relative to the first character of the record.
	01 =	Position in the current record buffer in main storage (relative to the beginning of the buffer) in which the next character will be stored.
	10 =	Relative position of the cursor on the screen (from the first position).
	11 =	Position in the current field (relative to the beginning of the field) in which the next character will be stored. For format level 0, the position of the current 1-byte field relative to the beginning of the format level 0 field.

Hex Displacement	Length in Bytes (in Hex)	Description
39	1	Keyboard Flags:

Bit		*Meaning*
0		Set to 1 by the keyboard/display microprocessor when the object code program issues the request for error operation (KERRST). Reset to 0 when the object code program issues the request error reset operation (KERRCL). The application program should not change this bit.
1	1 =	The keyboard/display microprocessor detected a keying error. The bit is turned off when the reset key is pressed. The application program should not change this bit.
2	1 =	Keyboard is open for data entry. Set when an ENTR command or RESUME instruction is executed. Cleared when a CNENTR command is executed or when an external status condition is detected. The application program should not change this bit.
3	1 =	The cursor is within a field.
4	1 =	If a keying error occurs, the keyboard/display microprocessor checks the status line. If the status line is not displayed, it displays it. When the error is reset, it replaces the status line with the extra line from the refresh buffer.
	0 =	Do not check the status line. Normally, no change is made in this bit by the application program. If the application program does change this bit, it should not change it while an ENTR command is being processed.
5	1 =	Status line is displayed. Indicates that the extra line must be displayed when the error is reset. The application program should not change this bit.
6	1 =	A mismatch error occurred while verifying a constant insert field. The application program should not change this bit.
7		Set by the keyboard/display microprocessor when the perform keyboard operation (hex 11) is issued by the object program. It indicates that the EBCDIC code came from the object code program; therefore, the microprocessor does not check the keyboard bit map. The application program should not change this bit.

Hex Displacement	Length in Bytes (in Hex)	Description		
3A	1	Keyboard Flags:		
		Bit		*Meaning When 1*
		0		System use only.
		1		The command key was the last key pressed.
		2		The last two keys pressed formed a command key sequence.
		3		Data being entered is in hexadecimal format following a hexadecimal command key sequence.
		4		One hexadecimal digit has been entered following a hexadecimal command key sequence.
		5		System use only.
		6		One hexadecimal digit has been entered into the current position of a hexadecimal field.
		7		The last key entered was diacritic.
3B	1	Keyboard Flags:		
		Bit(s)		*Meaning*
		0		Set by the keyboard/display microprocessor when it sets external status condition 11 (magnetic stripe reader data in buffer). Reset when the object code program reads the data or resets the reader.
		1-3		Indicates the screen size for which the program was written:
			1	1 = 1920
			2	1 = 960
			3	1 = 480
		4	1 =	Special verify mode.
		5-7		Displacement minus 2 from the address of the first byte of the picture check screen format group to the address of the first picture check subfield byte.

Hex Displacement	Length in Bytes (in Hex)	Description
3C	1	Keyboard Flags:

Bit(s)		*Meaning When 1*
0	1 =	Activate the clicker for function keys. Normally, no change is made to this bit by the application program. If the application program does change this bit, it should not change it while an ENTR command is being processed.
1	1 =	The home (Record Backspace) key was pressed with the cursor in the first position of the record. The application program should not change this bit.
2	1 =	Secondary screen format being processed. The application program should not change this bit.
3	1 =	Screen format control string is outside the partition. The application program should not change this bit.
4-7		When bit 3 = 1, bits 4 through 7 contain the page number of storage in which the screen format control string is stored. The application program should not change this bit.

Hex Displacement	Length in Bytes (in Hex)	Description
3D	1	Keyboard Flags:

Bit	*Meaning When 1*
0	During a verify operation, the digit entered in the last position of a field exit required field in which the Field– key is allowed does not match the digit currently in the record.
1	*Dup key enable flag:* The Dup key is not allowed. (See note.)
2	*Monocase enabled flag:* Characters that may be monocase, as defined in the validity table, are displayed and stored in the buffer in uppercase. See note.
3	*Field exit minus key enable flag:* The Field– key is not allowed in a numeric field. (See note.)
4	*Special verify enable flag:* If the 5280 is in verify mode, data entry is allowed without verify checking against the current record contents. When the field is exited, normal verify mode is restored. (See note.)

Note: This flag is set to 0 when the keyboard/display microprocessor begins executing an ENTR command but can be changed by the change-keyboard-control-flag control group in the screen format control string.

Hex Displacement	Length in Bytes (in Hex)	Description
3D (cont.)		

Bit		*Meaning*
5		When screen format processing is interrupted by a return to the object program, this bit indicates the direction the format should be processed when format processing resumes, as follows:
	0 =	Forward processing when interrupt occurred.
	1 =	Backward processing when interrupt occurred.

Meaningless if byte 1E is 0.

Bit		*Meaning*
6		During a verify operation:
	0 =	The sign of the last position of a field exit required field has *not* been verified.
	1 =	The entire last position of a field exit required field has been verified.
7	1 =	Indicates to the microprocessor not to perform character edit checks or perform checks for data required, mandatory enter, mandatory fill, and blank check fields. The bit is turned on when keyboard operation hexadecimal 06 is executed. Resets to 0 when each field is advanced into or backspaced into.

Hex Displacement	Length in Bytes (in Hex)	Description
3E	1	Keyboard Flags:

Bit	*Meaning When 1*
0	Keyboard is in enter mode. (See note.)
1	Keyboard is in update mode. (See note.)
2	Keyboard is in rerun mode. (See note.)
3	Keyboard is in verify mode. (See note.)
4	Keyboard is in insert mode. The application program should not change this bit.
5	Keyboard is in field correct mode. The application program should not change this bit.
6	Keyboard is in display mode. (See note.)
7	Fixed prompts are not displayed. Normally, no change is made to this bit by the application program. If the application program does change this bit, it should not change it while an ENTR command is being processed.

Note: An application program can change bits 0, 1, 2, 3, and 6, but only when an ENTR command is not being processed. After program load, these bits are maintained by the application program to determine the current mode for formatted data entry. When an ENTR is outstanding, one and only one of bits 0, 1, 2, 3, and 6 must be set.

<table>
<tr><th>Hex Displacement</th><th>Length in Bytes (in Hex)</th><th>Description</th></tr>
<tr><td>3F</td><td>1</td><td>Keyboard Flags:
<table>
<tr><th><em>Bit</em></th><th></th><th><em>Meaning</em></th></tr>
<tr><td>0</td><td></td><td>Set to 0 when the cursor enters a field. Set to 1 when data is entered into the field. When the cursor leaves the field, the microprocessor ORs this bit with the modified data indicator that is assigned to this field: See <em>System Indicators within a Partition.</em> (See Note 1.)</td></tr>
<tr><td>1</td><td>1 =</td><td>The last position of a field exit required field has been entered. A nondata key is required to exit the field. The application program should not change this bit.</td></tr>
<tr><td>2</td><td>1 =</td><td>Awaiting a record advance key. The application program should not change this bit.</td></tr>
<tr><td>3</td><td>1 =</td><td>Auto dup/skip is enabled. (See Note 1.)</td></tr>
<tr><td>4</td><td>1 =</td><td>Auto enter is enabled. (See Note 1.)</td></tr>
<tr><td>5</td><td>1 =</td><td>Alternate record advance is enabled. (See Note 1.)</td></tr>
<tr><td>6</td><td>1 =</td><td>Data is displayed when in rerun mode. (See Note 2.)</td></tr>
<tr><td>7</td><td>1 =</td><td>A verify mismatch error is pending. The application program should not change this bit.</td></tr>
</table></td></tr>
<tr><td>40</td><td>1</td><td>Displacement from the first byte of the current screen format picture check group to the current subfield format byte.</td></tr>
</table>

Notes:

1. The application program can change this bit, but only while an ENTR command is not being processed or when ENTR processing is suspended, as during external processing before the RESUME is issued.
2. The application program can change this bit, but only while an ENTR command is not being processed.

Hex Displacement	Length in Bytes (in Hex)	Description
41	8	Contains information about the current field as shown in the following bytes (from the format control string) while the field is being processed: (See *Screen Format Control Strings* for more information.) *Byte* — *Meaning* 41 — First byte of the field group. 42-43 — Field length minus 1. 44 — Field attribute byte. 45 — Field attribute extended byte. 46-47 — Storage duplicate table displacement. 48 — Screen format picture check (PIC) byte.
49	1	The number of bytes accumulated in a picture-check subfield; picture check processing ends when bits 1 through 3 of this byte equal the subfield length in bits 1 through 3 of the picture check byte.
4A	1	Screen line on which fixed prompts are displayed.
4B	3	Nonverify-correction keystroke counter.
4E	2	Verify-correction keystroke counter.
50	2	Address, in keyboard/display storage, of the storage area that contains keyboard control information (see Chapter 3).
52	2	Address, in keyboard/display storage, of the diacritic translate tables.
54	2	Address, in keyboard/display storage, of the status line refresh buffer.
56	1	High-order byte of the address, in keyboard/display storage, of the Katakana translate table.
57	1	High-order byte of the address, in keyboard/display storage, of the scan code translate table.
58	2	Address, in keyboard/display storage, of the main refresh buffer.
5A	1	Number of lines on the screen.
5B	1	Number of characters per screen line.

Hex Displacement	Length in Bytes (in Hex)	Description
5C	1	Keyboard Configuration Information: *Bit(s)* *Meaning* 0 0 = Single screen 1 = Dual screen 1 0 = Single screen or station 0 of dual screen. 1 = Dual screen, station 1. 2-3 System use only. 4 1 = Katakana keyboard. 5 1 = Proof keyboard. 6 1 = Typewriter keyboard. 7 1 = Data entry keyboard.
5D	1	Keyboard Configuration Information: *Bit(s)* *Meaning When 1* 0 Not used. 1 Maximum screen size is 1920. 2 Maximum screen size is 960. 3 Maximum screen size is 480. 4-7 Not used.
5E	1	High-order byte of the address in keyboard/display storage of the validity table.
5F	4	Display line map: Bits 0-25 of the 4-byte group indicate which screen lines are displayed (lines 0-25 respectively). Bits 26-31 are 0.
63	1	Language group; the number selected from the language/keyboards-type table during configuration.
64	1	Low-order byte of the address in keyboard/display storage of the cursor address register.
65	1	High-order byte of the address in keyboard/display storage of the control area.
66	1	Low-order byte of the address in keyboard/display storage of the display control register.
67	1	System use only.
68	2	Displacement into the current field (0 to field length minus 1) to the current character position.

Hex Displacement	Length in Bytes (in Hex)	Description
6A	2	The relative field number (0 to maximum number of fields minus 1) of the field within the screen format control string currently being processed. A format level 0 specification equals one field.
6C	2	Address, in keyboard/display storage, of the fixed prompt line.
6E	2	Value of the current position counter (4-digit, zone-stripped format) displayed on the status line during keyboard entry.
70	2	Number of positions (4-digit, zone-stripped format) remaining in the field. The low-order two digits are displayed on the status line.
72	2	The relative position (in binary format) minus 1 in the current record buffer where the next character entered, if valid, will be stored.
74	1	Normal display attribute from .KBCRT (NMIN) statement.
75	1	Highlight display attribute from .KBCRT (HLIN) statement.
76	2	Microprocessor save area.
78	1	EBCDIC value used to check blank-check fields (usually hexadecimal 40).
79	2	Address, relative to the start of the partition, of the screen format control string table that is used to locate the format control strings with the partition.
7B	1	The EBCDIC character for the key that caused the last verify mismatch error.
7C	1	The EBCDIC character for the dup data that caused a verify mismatch.
7D	1	Microprocessor save area.
7E	1	Microprocessor work area.
7F	1	Main microprocessor lockout byte. When nonzero, the main microprocessor is using the IOB.

System Indicators Within a Partition

The first one hundred indicators within a partition may be used as the user wishes. The other indicators, however, are assigned a specific purpose for use during program execution. The indicator assignments are as follows:

Indicator	Condition	Meaning If Set to 1
I100		System use only
I101	Table search	Result is higher
	TRT	Byte not found
	CLC	String 1 greater than string 2
I102	Table search	Result is lower
	TRT	Byte is found
	CLC	String 1 is less than string 2
I103	Table search	Result is equal
	TRT	Byte found in last position (EOF)
	CLC	String 1 is equal to string 2
I108	External status	Restricted external status processing
I109	Program check	Program check error
I110		Background partition
I115	SCS	LSTLN overflow
I116		System use only
I117	Self check	Self check error
I118	SRAT	Resource allocation table search error
I119	HEXBIN	Attempt to convert invalid hex EBCDIC to binary
I120	Decimal divide	Divide error (denominator=0)
I121	Edit format	Invalid edit format conversion request
I122	Arithmetic	Decimal to binary conversion error
I123	Decimal multiply	Multiply overflow (+, *, /)
I124	Decimal arithmetic	Decimal arithmetic overflow
I125	Table search	Entry not found

Indicator	Condition	Meaning If Set to 1
I126	Table write	Attempt to extend table beyond its limit
I127	Table instruction	Table instruction error
I128-159		Reserved
I160-191	ENTR	Field modification indicators. Each indicator represents a field in the screen format, up to 32 fields. If there are more than 32 fields in the screen format, each indicator represents every 32nd field. I160 represents field 0, field 32 and so on. A format level zero specification is represented with one indicator for the entire group of 1-byte fields. All field modification indicators are set to 0 when an ENTR is encountered. While the ENTR is being processed, each time the cursor is advanced or backspaced into a field, bit 0 of byte hex BF of the keyboard/display IOB in the partition is set to 0. If data is entered into the field, bit 0 of the byte at hex BF is set to 1. When the cursor exits the field, bit 0 of the byte at hex BF is ORed with the field modification indicator that represents the field.
I192-254		Used by SYSKEU, DE/RPG, and other programs to communicate with common function routines.

System Registers Within a Partition

Several binary registers are used by the system during program execution. These registers are listed below, with the conditions or instructions that affect each register.

Register	Condition	Register Contents
BR16	LOAD	Relative record number for relative record read; also contains error code after a load error.
BR16	TRT	Address of the last position that translated to a non-zero character.
BR17	TRT	Function byte.
BR18	Subroutine	Address of next available entry position in the partition subroutine stack.
BR19	Keyboard external status	Current field starting address, relative to the beginning of the partition, of the field within the current record buffer.
BR20	Keyboard external status	Current field starting address within the screen refresh buffer in keyboard/display control storage.
BR21	Keyboard external status	Field definition and field length minus 1 of the current field.
BR22	External status	Relative address of the last data set IOB to report external status. Not used for keyboard/display external status.
BR23	External status	External status condition code, to be used as the index into the external status error table of subroutine addresses.
BR24		Used by SCP for PTF log.
BR25	LOAD	Physical device address of the device performing the load.
BR26-31		System use only.

Diskette IOB

Following is a general description of the diskette IOB. Following this general description is a complete description of each field of the IOB. Addresses shown are hexadecimal displacements from the beginning of the IOB. No validity checking is made on any of the values in the bytes of the following IOB. If any of these bytes are modified by the application program, unpredictable results may occur.

<table>
<tr><td>00</td><td colspan="1">IOB System Status
1</td><td colspan="2">IOB Chaining Information
1</td><td colspan="1">Page Data and Flags</td><td colspan="2">Error Code
1</td><td colspan="2">Next Instruction Address
1</td></tr>
<tr><td>08</td><td colspan="1">Command Op Code
1</td><td colspan="3">Command Operands
1</td><td colspan="2">Logical Buffer Address</td><td colspan="1">Translate Table Number
1,2</td><td colspan="1">External Status</td></tr>
<tr><td>10</td><td colspan="2">Address of External Status Subroutine or Subroutine Table</td><td colspan="1">Main Microprocessor Flags
1,2,3</td><td colspan="1">Data Set Flags
1,2,3</td><td colspan="2">Address of Data Set Name
1,2</td><td colspan="1">System Use Only</td><td colspan="1">Partition Address, High
1</td></tr>
<tr><td>18</td><td colspan="2">Physical I/O Buffer 1 (PB1) Address and Length
1,2</td><td colspan="1">PBI Track
1</td><td colspan="1">PBI Sector
1</td><td colspan="2">Logical Record Length
1,2,3</td><td colspan="2">Block Length
1,2,3</td></tr>
<tr><td>20</td><td colspan="2">Physical I/O Buffer 2 (PB2) Address and Length
1,2</td><td colspan="1">PB2 Track
1</td><td colspan="1">PB2 Sector
1</td><td colspan="2">Defective Sector Count
1,3</td><td colspan="2">Microprocessor Save Area
1,3</td></tr>
<tr><td>28</td><td colspan="2">Displacement to Next Record Space
1,3</td><td colspan="6">Microprocessor Save Area
1,3</td></tr>
<tr><td>30</td><td colspan="2">Pointer to HDR1 Label Address
1,3</td><td colspan="1">Sector Length
1,3</td><td colspan="1">Number of Additional Index Cylinders
1,3</td><td colspan="1">Number of Sectors per Block
1,3</td><td colspan="1">Number of Sectors per Track
1,3</td><td colspan="2">Track and Sector Number of Beginning of Extent (BOE)
1,3</td></tr>
<tr><td>38</td><td colspan="3">Relative Record Number of End of Data (EOD)
1,3</td><td colspan="3">Relative Record Number of End of Extent (EOE)
1,3</td><td colspan="2">Track and Sector Number of End of Data (EOD)
1,3</td></tr>
<tr><td>40</td><td colspan="1">System Use Only
4</td><td colspan="1">Table Number of Key Index File
1,2,4</td><td colspan="2">Number of Records Between Keys
1,2,3,4</td><td colspan="2">Key Position
1,2,4</td><td colspan="1">Key Length Minus 1
1,2,4</td><td colspan="1">System Use Only
4</td></tr>
<tr><td>48</td><td colspan="4">Microprocessor Save Area
1,3</td><td colspan="2">Data Set Type
1,2</td><td colspan="1">Adapter Error Status
1</td><td colspan="1">Microprocessor Save Area
1</td></tr>
<tr><td>50</td><td colspan="2">Number of Bytes to Read or Write (PB1)
1</td><td colspan="2">Microprocessor Save Area
1,3</td><td colspan="1">Seek Count
1</td><td colspan="3">Microprocessor Save Area
1,3</td></tr>
</table>

<table>
<tr><td>58</td><td colspan="2">Number of Bytes to Read or Write (PB2)
1</td><td colspan="2">Number of Nulls Between Blocks</td><td colspan="4">Microprocessor Save Area
1,3</td></tr>
<tr><td>60</td><td colspan="2">Device Identification</td><td>Diskette IOB Identifier</td><td>Deleted Record Character
1,3</td><td colspan="3">Microprocessor Save Area
1,3</td><td>Current Record Pointer
1,3</td></tr>
<tr><td>68</td><td colspan="2">Current Record Pointer (continued)
1,3</td><td colspan="6">Microprocessor Save Area
1,3</td></tr>
<tr><td>70</td><td colspan="8">Microprocessor Save Area
1,3</td></tr>
<tr><td>78</td><td colspan="6">Microprocessor Save Area
1,3</td><td colspan="2">System Use Only
1,3</td></tr>
</table>

1. An application program must not alter this field while the IOB is active.
2. Initialized by the assembler.
3. Initialized by the device at open time.

   If both 2 and 3 are specified for a field, it indicates that the field can be initialized by either the assembler or the device, except for bytes 12 and 13, which are initialized by both the assembler and the device.

4. These values apply only to keyed data sets. For SCS conversion data sets, these bytes have a different meaning. See the complete description of the fields for the SCS values.

Hex Displacement	Length in Bytes (in Hex)	Description
00	1	IOB System Status:

Bit(s)		*Meaning*
0-1	11 =	The main microprocessor sent a command to the diskette MPU. It cannot send another command until the diskette microprocessor sets the bits to 00.
	10 =	System use only.
	01 =	Diskette is executing the command; buffers are now in use.
	00 =	No command pending.
2	1 =	The diskette microprocessor has work to do.
3	1 =	The diskette microprocessor sets this bit on when it senses an error or external status. The main microprocessor clears the bit after the external status or the error condition has been processed.
4	1 =	The diskette microprocessor is performing a physical operation for this data set.
5		System use only.
6	1 =	The IOB is first in chain.
7		System use only.

Hex Displacement	Length in Bytes (in Hex)	Description
01	1	IOB Chaining Information: *Bit(s)* *Meaning* 0 1 = Diskette microprocessor is processing the chain pointer. The main microprocessor cannot use the chaining information when this bit is on. 1-3 System use only. 4-7 When nonzero, the main microprocessor is accessing the chain pointer flags.
02	1	High-order byte of the address of the next IOB in the chain.
03	1	Page Data and Flags: *Bit(s)* *Meaning* 0 Low-order address of the next IOB in the chain. 1-3 System use only. 4-7 Number of the page in main storage where the next IOB on the chain is located.
04	2	External status error code in 4-byte packed decimal format (not reset by the system).
06	2	The absolute address of the next sequential instruction following the operation issued to the diskette MPU.
08	1	Op code, see Chapter 4.
09	3	Instruction operand. These bytes contain the rightmost 3 bytes of the object code instruction. See Chapter 4 for the meanings of these bytes.
0C	2	Address of the logical I/O buffer relative to the beginning of the partition.
0E	1	The number of the table used to translate EBCDIC characters to ASCII, ASCII characters to EBCDIC, or other character set translations. Hex FF indicates no translation requested.
0F	1	External status category.
10	2	Address of the external status subroutine table.

<table>
<tr><th>Hex Displace-ment</th><th>Length in Bytes (in Hex)</th><th>Description</th></tr>
<tr><td>12</td><td>1</td><td>Main Microprocessor Flags:
<table>
<tr><th><em>Bit(s)</em></th><th></th><th><em>Meaning</em></th></tr>
<tr><td>0</td><td>0</td><td>= There is more than one external status subroutine. They are accessed via a subroutine table.</td></tr>
<tr><td></td><td>1</td><td>= External status conditions are handled by one subroutine.</td></tr>
<tr><td>1</td><td>1</td><td>= An error occurred when opening a data set.</td></tr>
<tr><td>2</td><td>1</td><td>= SCS conversion data set; logical buffer is empty.</td></tr>
<tr><td>3</td><td>1</td><td>= SCS conversion is in progress for this IOB.</td></tr>
<tr><td>4</td><td>1</td><td>= SCS last line status flag.</td></tr>
<tr><td>5</td><td>1</td><td>= An error detected by the main microprocessor is outstanding.</td></tr>
<tr><td>6</td><td>1</td><td>= CLOZ operation logically complete.</td></tr>
<tr><td>7</td><td>1</td><td>= SCS purge in progress, set during CLOZ operation.</td></tr>
</table></td></tr>
<tr><td>13</td><td>1</td><td>Data Set Flags:
<table>
<tr><th><em>Bit</em></th><th><em>Meaning When 1</em></th></tr>
<tr><td>0</td><td>The IOB is open.</td></tr>
<tr><td>1</td><td>Logical buffer is within the physical buffer.</td></tr>
<tr><td>2</td><td>Diskette is using double physical buffers.</td></tr>
<tr><td>3</td><td>Diskette microprocessor is waiting for a shared data set conflict to be resolved. The shared data set is being used by another IOB.</td></tr>
<tr><td>4</td><td>On open, the logical record and block size are set to equal the sector size.</td></tr>
<tr><td>5</td><td>I/O MPU requires repeat of last command. The main MPU decrements the external status table return address for repeat when a RETURN instruction is used.</td></tr>
<tr><td>6</td><td>Not used.</td></tr>
<tr><td>7</td><td>SCS continuation of transparent data across physical buffers, or data set keys are in ascending order.</td></tr>
</table></td></tr>
<tr><td>14</td><td>2</td><td>Address of data set name.</td></tr>
<tr><td>16</td><td>1</td><td>System use only.</td></tr>
<tr><td>17</td><td>1</td><td>Partition address, high-order byte: The value in byte 17 is added to each address in the IOB to convert it to an absolute storage address. This byte also points to the beginning of the partition IOB and is used to find table addresses.</td></tr>
</table>

Hex Displacement	Length in Bytes (in Hex)	Description
18	4	Physical I/O Buffer 1: Byte 18 and bit 0 of byte 19 contain the address, relative to the beginning of the partition, of the beginning of physical I/O buffer 1. Byte 19, bits 1 through 7 contain the number of 128-byte blocks allocated to the buffer in main storage. Byte 1A contains the head and track number where physical I/O buffer 1 starts on diskette (bit 0 = head number). Byte 1B contains the sector number where physical I/O buffer 1 starts on diskette (set to hexadecimal 00 anytime the buffer is invalid, such as: quick release, early write, or if an error occurs).
1C	2	The logical record length of the records in the data set. Not used by diskette MPU in SCS conversion processing.
1E	2	Block length for blocking logical records on diskette.
20	4	Physical I/O Buffer 2: Byte 20 and bit 0 of byte 21 contain the address, relative to the beginning of the partition, of the beginning of physical I/O buffer 2. Byte 21, bits 1 through 7 contain the number of 128-byte blocks allocated to the buffer in storage. Byte 22 contains the head and track number where physical I/O buffer 2 starts on diskette (bit 0 = head number). Byte 23 contains the sector number where physical I/O buffer 2 starts on diskette (set to hexadecimal 00 anytime the buffer is invalid, such as: quick release, early write, or if an error occurs).

<table>
<tr><th>Hex Displacement</th><th>Length in Bytes (in Hex)</th><th>Description</th></tr>
<tr><td>24</td><td>2</td><td>The number of defective sectors encountered.</td></tr>
<tr><td>26</td><td>2</td><td>Microprocessor save area.</td></tr>
<tr><td>28</td><td>2</td><td>Negative displacement to the next available record space from the end of the last block.</td></tr>
<tr><td>2A</td><td>6</td><td>Microprocessor save area.</td></tr>
<tr><td>30</td><td>2</td><td>Relative record number of the HDR1 address label for this data set.</td></tr>
<tr><td>32</td><td>1</td><td>The diskette sector length as follows:
<table>
<tr><th><i>Hex</i></th><th><i>Meaning</i></th></tr>
<tr><td>01</td><td>128-byte sector length for diskette 1 or 2.</td></tr>
<tr><td>02</td><td>256-byte sector length for diskette 1, 2, or 2D.</td></tr>
<tr><td>04</td><td>512-byte sector length for diskette 1, 2, or 2D.</td></tr>
<tr><td>08</td><td>1024-byte sector length for diskette 2D.</td></tr>
</table></td></tr>
<tr><td>33</td><td>1</td><td>The number of additional index cylinders on the diskette. For example, if this number is 4, there are five index cylinders on this diskette.</td></tr>
<tr><td>34</td><td>1</td><td>The number of sectors per block, which is block length divided by sector length plus 1 if there is a remainder.</td></tr>
<tr><td>35</td><td>1</td><td>The value 26, 15, or 8 to indicate the number of sectors per track.</td></tr>
<tr><td>36</td><td>2</td><td>The BOE track and sector number: For diskette 1, byte 36 = cylinder number and byte 37 = sector number. For diskette 2 and 2D; byte 36, bits 0-6 = cylinder number, and bit 7 = head number; byte 37 = sector number.</td></tr>
<tr><td>38</td><td>3</td><td>The relative record number of the last logical record in the data set.</td></tr>
<tr><td>3B</td><td>3</td><td>The relative record number of the last logical record space available in the data set.</td></tr>
<tr><td>3E</td><td>2</td><td>The EOD track and sector number: For diskette 1, byte 3E = cylinder number and byte 3F = sector number. For diskette 2 and 2D; byte 3E, bits 0-6 = cylinder number, and bit 7 = head number; byte 3F = sector number.</td></tr>
</table>

Hex Displacement	Length in Bytes (in Hex)	Description
40	1	Used only for SCS conversion data sets; a pointer into the physical I/O buffer where the data is formatted.
41	1	For keyed data sets, the table number of the keyed index file in main storage. Hex FF cannot be used. See *Addressing Through a System Table*, in Chapter 4, for information about finding tables in storage. For SCS conversion data sets, the line number of the current line.
42	2	For keyed data sets, the number of logical records between key entries on indexed files. For SCS conversion data sets, the line that generated external status (42) and the page size (43).
44	2	For keyed data sets, the location of the index key within the logical record. For SCS conversion data sets, the address of the format table entry being processed on open, which contains the SGEA (set graphic error action) parameters. After open, byte 44 has the number of blanks processed, and byte 45 has the number of bytes processed in the logical buffer.
46	1	For keyed data sets, the length minus one of the index key. For SCS conversion data sets, the number of characters processed in the line.
47	1	Used only for SCS conversion data sets; the number of characters per line.
48	4	Microprocessor save area.

Hex Displace-ment	Length in Bytes (in Hex)	Description
4C	2	Data set types as follows: Byte 4C: *Bit* — *Meaning When 1* 0 Read allowed. 1 Write allowed. 2 Read shared. 3 Write shared. 4 Label update data set. 5 Diskette microprocessor builds an index table when opening keyed data sets. 6 Keyed data set. 7 Set EOD equal to BOE when opening. Byte 4D: *Bit(s)* — *Meaning When 1* 0 Early write. 1 Quick release. 2 Translation of HDR1 labels required. 3 Diskette MPU does not check for overlapped extents or duplicate data set names. 4 Standard character string conversion is requested. 5 Pointer mode data set. 6-7 System use only.
4E	1	Used to store temporary status information.
4F	1	Microprocessor save area.
50	2	Number of bytes to read or write; used in conjunction with physical buffer 1.
52	2	Microprocessor save area.
54	1	The number of tracks to seek as follows: *Bit(s)* — *Meaning* 0 1 = Seek high. 0 = Seek low. 1-7 The number of remaining tracks to seek for this data set. (Seek operations can be overlapped with either a read or a write operation.)

Hex Displacement	Length in Bytes (in Hex)	Description
55	3	Microprocessor save area.
58	2	Number of bytes to read or write; used in conjunction with physical buffer 2.
5A	2	Number of nulls between blocks.
5C	4	Microprocessor save area.
60	2	Logical device identification from the resource allocation table.
62	1	Diskette IOB Identifier: *Bits* *Meaning* 0-3 The logical I/O table number. 4-7 The partition number for the partition in which this IOB is located.
63	1	The user specified character that indicates a logically deleted record for I or E exchange data sets.
64	3	Microprocessor save area.
67	3	A record number used as a pointer to keep track of positions within the data set. It is not necessarily the same as the record number of the record in the logical buffer.
6A	14	Microprocessor save area.
7E	2	System use only.

Printer IOB

Following is a general description of the printer IOB. Following this general description is a complete description of each field of the IOB. The addresses shown are hexadecimal displacements from the beginning of the IOB. No validity checking is made on any of the values in the bytes of the following IOB. If any of these bytes are modified by the application program, unpredictable results may occur.

00	IOB System Status 1,2,3	IOB Chaining Information 1,2		Page Data and Flags	Error Code 1,3		Next Instruction Address 1	
08	Command Op Code 1,2	Command Operands 1,2,3			Logical Buffer Address 1,2,3		Translate Table Number • 2	External Status 1,3
10	Address of External Status Subroutine Table 1		Main Microprocessor Flags 1	Data Set Flags 1,2,3	Address of Data Set Name 1		Printer Subaddress 1,2,3	Partition Address, High 1,2
18	Physical I/O Buffer 1 Address and Length 1,2,3		Buffer Remainder 1,2,3,4		Logical Record Length 1,2		Block Length• 1,2,3,4	
20	Physical I/O Buffer 2 Address and Length 2,3		Number of Printer Buffers Remaining to Transmit 2,3,4		Number of Bytes Sent in Last Transmission 2,3,4		Number of Bytes of Buffer Being Used 2,3,4	
28	Information From the Global Configuration Table					Device Physical Buffer Size 2,3	System Use Only	
30	System Use Only		Physical Record Length 1,2,3,4	Printer Line Length 2,3,4	Busy Timer 2,3,4		Close Timer 2,3,4	
38	Microprocessor Save Area	System Use Only 2,3,4	Number of Records Remaining to be Sent to Physical Buffer 2,3,4		Number of Printer Buffers to Transmit 2,3,4		Number of Logical Records to Transfer to Buffer 2,3,4	
40	SCS Parameters 1							
48	Microprocessor Save Area				Data Set Type 1,2		Last Poll Response Before an Error 2,3,4	
50	Status from Printer 2,3,4	Response to Last Poll 2,3,4		Status from Last Read 2,3,4	Microprocessor Save Area 2,3,4			Command Flag

58	System Use Only				Number of Bytes of Physical Buffer Being Used in Pointer Mode	
60	Printer Identification 1	Printer IOB Identifier 1	Microprocessor Save Area 2,3,4	Error Code Build Area 2,3,4	System Use Only	Current Record Number 1,2,3,4
68	Current Record Number (continued)	Microprocessor Save Area 2,3,4		Number of Logical Records Remaining in Pointer Mode 2,3,4	System Use Only	
70	System Use Only					
78	System Use Only					

1. Accessed by the main microprocessor.
2. Read by the printer microprocessor.
3. Written by the printer microprocessor.
4. Initialized by the printer microprocessor.

Hex Displacement	Length in Bytes (in Hex)	Description
00	1	IOB System Status:

Bit(s)		*Meaning*
0-1	11 =	The main microprocessor sent a command to the printer attachment microprocessor. The main microprocessor cannot send another command until the printer microprocessor sets the bits to 00.
	10 =	The printer attachment microprocessor has completed logical work for the command but is still doing physical work.
	01 =	System use only.
	00 =	No command pending. (Printer may still be busy.)
2	1 =	The printer attachment microprocessor has physical work to do.
3	1 =	The printer attachment microprocessor sets this bit on when it detects an error or external status. The main microprocessor clears the bit and processes the external status with the subroutine indicated.
4-5		System use only.
6	1 =	This is the first IOB on the chain.
7		System use only.

<table>
<tr><th>Hex Displace-ment</th><th>Length in Bytes (in Hex)</th><th>Description</th></tr>
<tr><td>01</td><td>1</td><td>IOB Chaining Information:
<table>
<tr><th><i>Bit(s)</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>The printer attachment microprocessor is processing the chain pointer. The main microprocessor cannot use the chaining information when this bit is 1.</td></tr>
<tr><td>1-3</td><td>System use only.</td></tr>
<tr><td>4-7</td><td>When nonzero, the main microprocessor is accessing the chain pointer.</td></tr>
</table></td></tr>
<tr><td>02</td><td>1</td><td>High-order byte of the address of the next IOB in the chain.</td></tr>
<tr><td>03</td><td>1</td><td>Page Data and Flags:
<table>
<tr><th><i>Bit(s)</i></th><th><i>Meaning</i></th></tr>
<tr><td>0</td><td>The low-order address bit of the next IOB in the chain.</td></tr>
<tr><td>1-3</td><td>System use only.</td></tr>
<tr><td>4-7</td><td>Page number where the next IOB in the chain is located.</td></tr>
</table></td></tr>
<tr><td>04</td><td>2</td><td>External status error code in 4-byte packed decimal format (only valid if byte 0, bit 3 is 1).</td></tr>
<tr><td>06</td><td>2</td><td>The absolute address of the next sequential instruction following the operation issued to the printer attachment MPU.</td></tr>
<tr><td>08</td><td>1</td><td>Command op code. See Chapter 4.</td></tr>
<tr><td>09</td><td>3</td><td>Command operand. These bytes contain the rightmost 3 bytes of the object code instructions. See Chapter 4 for the meanings.</td></tr>
<tr><td>0C</td><td>2</td><td>Address of the logical buffer, relative to the beginning of the partition.</td></tr>
<tr><td>0E</td><td>1</td><td>Number of the table used to translate EBCDIC characters to ASCII, ASCII characters to EBCDIC, or other character set translation. Hex FF indicates no translation required.</td></tr>
<tr><td>0F</td><td>1</td><td>External status category.</td></tr>
<tr><td>10</td><td>2</td><td>Address of the external status subroutine table.</td></tr>
</table>

Hex Displacement	Length in Bytes (in Hex)	Description
12	1	Main Microprocessor Flags: *Bit* — *Meaning When 1* 0 — All external status conditions handled by one subroutine. 1 — An error occurred while opening the data set. 2 — SCS conversion data set; logical buffer is empty. 3 — SCS conversion is in progress for this IOB. 4 — SCS last line status flag. 5 — An error detected by the main microprocessor is outstanding. 6 — CLOZ operation is logically complete. 7 — SCS purge in progress, set during CLOZ operation. Bits 2-7 are set and maintained by the main microprocessor.
13	1	Data Set Flags: *Bit* — *Meaning When 1* 0 — IOB is open.[1] 1 — Logical buffer is within physical buffer. 2 — Double physical buffers are used. 3-4 — Not used. 5 — I/O MPU requires repeat of last command. Main MPU decrements the external status table return address to cause the repeat when a RETURN instruction is used. 6-7 — System use only.
14	2	Address of the storage area that contains the data set name.
16	1	Device Subaddress: *Bit(s)* — *Meaning* 0-2 — Not used. 3-4 — Port address. 5-7 — Station address.

[1] If you issue a second open to a data set without closing the data set, this bit is no longer a valid indicator that the data set is open.

Hex Displacement	Length in Bytes (in Hex)	Description
17	1	High-order byte of the address of the beginning of the partition. The printer microprocessor adds this address to all relative addresses to form the absolute address.
18	2	Byte 18 and byte 19, bit 0 contain the address of the beginning of the physical I/O buffer 1 relative to the beginning of the partition. Byte 19, bits 1-7 contain the number of 128-byte blocks allocated to the buffer in main storage.
1A	2	Number of bytes of physical buffer not being used.
1C	2	Logical record length of records in the data set.
1E	2	Block length; can be either 128 or 256. If not specified in program, the block length is set to physical I/O buffer 1 size (maximum length is 256).
20	2	Address of the start of the physical I/O buffer relative to the beginning of the partition, and buffer length; same format as bytes 18-19.
22	2	Number of printer buffers remaining to transmit.
24	2	Number of bytes to be sent to the printer in the last transmission.
26	2	Number of bytes of the physical buffer being used.
28	6	Information for the printer attachment MPU from the global configuration table: Byte 28 Displacement from the beginning of the soft error log to the first entry for this printer. Byte 2A Number of entries allocated to the soft error log for this printer. Byte 2B Error encoding type as follows: A0 = Bit encoding 20 = Byte encoding Byte 2C Always 00. Byte 2D Number of 128-byte blocks in device physical buffer (2).

Hex Displace-ment	Length in Bytes (in Hex)	Description
2E	4	System use only.
32	1	Physical record length.
33	1	Printer line length; set to logical record length at open time. If the logical record length is longer than the maximum print line, zero record length is transmitted to cause the printer to use its default line length.
34	2	Busy timer (busy time-out results in 2291 error).
36	2	Close timer (close time-out results in a 2292 error).
38	1	Microprocessor save area.
39	1	System use only.
3A	2	Number of logical records remaining to be transferred to the physical buffer.
3C	2	Number of printer buffers that will be transmitted.
3E	2	Number of logical records that will be transferred to the physical buffer.
40	C	SCS conversion parameters, used only with SCS conversion data sets.

Byte(s)	*Meaning*
40	A pointer into the physical I/O buffer where the data is formatted.
41	The line number of the current line.
42	The line that generates external status.
43	The page size.
44-45	The address of the format table entry being processed on open, which contains the SGEA (set graphic error action) parameters. After open, byte 44 has the number of blanks processed, and byte 45 has the number of bytes processed in the logical buffer.
46	The number of characters processed in the line.
47	The number of characters per line.
48-4B	Microprocessor save area.

Hex Displacement	Length in Bytes (in Hex)	Description
4C	1	Data Set Type: *Bit(s)* *Meaning* 0 1 = Read allowed (causes error code 2402). 1 1 = Write allowed. 2 Not used. 3 1 = Write shared. (A printer may be used by more than one IOB.) 4-7 Not used.
4D	1	Data Set Type: *Bit(s)* *Meaning* 0 1 = Early write data set. (Transmit a logical record each time it is transferred to the physical buffer.) 1 Not used. 2 Always 0. 3 Not used. 4 1 = SCS conversion requested. 5 1 = Pointer mode data set. 6-7 Not used.
4E	2	Last poll response that occurred before an error was detected; also placed in the system hard error table.
50	1	Status from the printer; also placed in the system hard error table.
51	2	The response to the last poll command.
53	2	Status from the last read status command.
55	2	Microprocessor save area.
57	1	Command flag; indicates the last command issued.
58	6	System use only.
5E	2	Number of bytes of the physical buffer being used in pointer mode.
60	2	Printer ID.
62	1	Printer IOB identifier.

Hex Displacement	Length in Bytes (in Hex)	Description
63	1	Microprocessor save area.
64	2	Used to build the error code before it is transferred to bytes 04-05.
66	1	System use only.
67	3	Current record number: initialized to hexadecimal 00 at open time and used during pointer mode to indicate the number of records transferred to the buffer since open.
6A	2	Microprocessor save area.
6C	2	The number of logical records remaining to be transferred to the physical buffer in pointer mode.
6E	12	System use only.

SYSTEM TABLES

System tables contain the addresses of certain data areas. When an assembler source program allocates and labels one of these data areas, the system stores the address of the area in the appropriate system table. When a source program instruction refers to one of these data areas, the instruction specifies the label assigned to the area. Then when the source program is assembled, the assembler converts the label to the index into the system table where the address of that data area is stored. During program execution, when an object code instruction contains a system table index, the system finds the address of the area at that index into the appropriate system table.

System tables may be located within a main storage partition or within the common area. System tables within the partition contain addresses of data areas within the partition. System tables within the common area contain addresses of global data areas located in the common area. The partition or device IOB contain the addresses of the system tables within the partition. The system control block contains the addresses of the system tables in the common area.

The data areas that are addressed through a system table are the:

- Data tables
- Edit format control strings
- Screen format control strings
- Prompts and constant inserts
- Main storage duplicate areas (cannot be in the common area)

System Table for Data Tables

The system table for data tables is built by the assembler when it processes the .TABLE control statements; one system table entry is generated from each .TABLE control statement. The address of the system table for data tables that are located within the partition is in the partition IOB at relative address hex 18. The address of the system table for global data tables is in the system control block at absolute address hex F9.

The system table for data tables within the partition consists of one 8-byte entry for each data table. The format of the 8-byte entry is as follows:

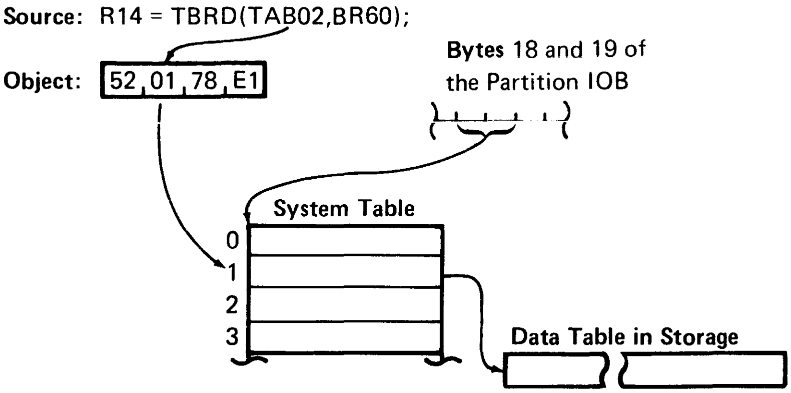
Bytes	Meaning
0-1	Table address: the relative address of the data table
2-3	Entry number: the number of the last table entry used
4	Entry length: the number of bytes minus 1 of a table entry
5	Bypass length: the length of the bypass portion of the table entry
6-7	Maximum entries: the maximum number of entries the table can have

The index for the system table for tables within the partition must be in the range 0 through 127. The index for the system table for global tables must be in the range 128 through 254. The first two global tables are reserved for system error tables; one global table may be an ASCII translate table.

The system table for global tables must always be located on storage page zero. The entries are 10 bytes long, in the following format:

Bytes	Bits	Meaning
0	0	Lock bit 0 = Table locked only for 1 table instruction. 1 = Table locked by TLCK instruction until TUNLCK instruction is issued.
	1-2	Not used
	3	0 = Valid partition number in bits 4-7. 1 = No valid partition number in bits 4-7.
	4-7	Partition number of partition using the table
1	0-3	Storage page number where the table is located
	4-7	Always 0001
2-9		As for bytes 0-7 of system table for data tables within the partition.

An object code table instruction contains the system table index for the table to access in the second byte of the 4-byte instruction. The following illustration shows how the system table index is used to access a data table within the partition. The data table labeled TAB02 was the second table set up with a .TABLE control statement.


Source: R14 = TBRD(TAB02,BR60);
Object: 52 01 78 E1
Bytes 18 and 19 of the Partition IOB
System Table
0
1
2
3
Data Table in Storage

System Table for Edit Format Control Strings

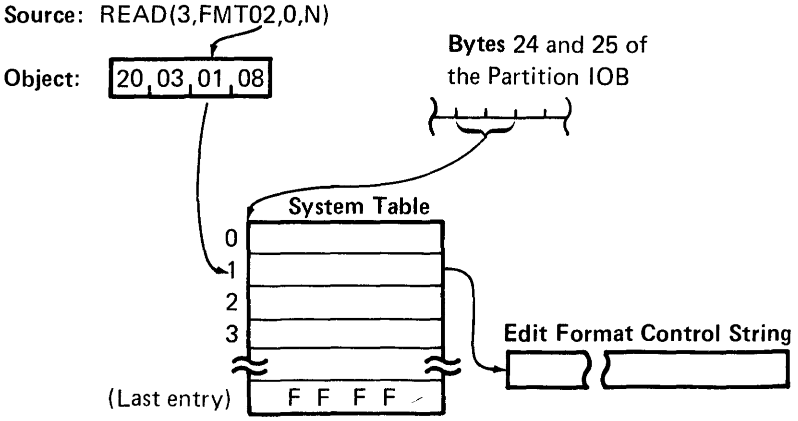
The system table for edit format control strings is built by the assembler when it processes the .FMT control statements; one system table entry is generated by each series of .FMT control statements. The address of the system table for edit format control strings that are located within the partition is stored in the partition IOB, at relative address 24. The address of the system table for global edit format control strings is stored in the system control block at absolute address hex EE. The system table for global edit format control strings must always be located on storage page 0.

The system table for edit format control strings located within a partition consists of one 2-byte entry for each control string. The 2-byte entry contains the address, relative to the beginning of the partition, where the control string is located. There may be up to 127 edit format control strings within a partition, represented by system table indexes 0 through 126. The last entry in the system table for edit format control strings always contains hex FFFF. If no edit formats are set up with the .FMT control statement series in a source program, a system table for edit format control strings is built; the only 2-byte entry in the table contains FFFF.

The system table for global edit format control strings consists of one 3-byte entry for each global edit format control string. The 3-byte entry contains the storage page number in the first byte, and the control string address (relative to the beginning of the storage page) in the second and third bytes. There may be up to 127 global edit format control strings (numbered 128 to 254), represented in the system table with indexes 0 through 126, where index 0 represents format 128. The last entry in the system table always contains hex FFFF.

When a source program instruction refers to an edit format, it includes the format label. The assembler converts the label to a format number from 0 to 127.

The following illustration shows how the system table is used to find an edit format control string that is located within the partition. In the illustration, FMT02 is the second edit format set up with a .FMT control statement series.


Source: READ(3,FMT02,0,N)
Object:
20 03 01 08
Bytes 24 and 25 of the Partition IOB
System Table
0
1
2
3
(Last entry)
F F F F
Edit Format Control String

System Table for Screen Format Control Strings

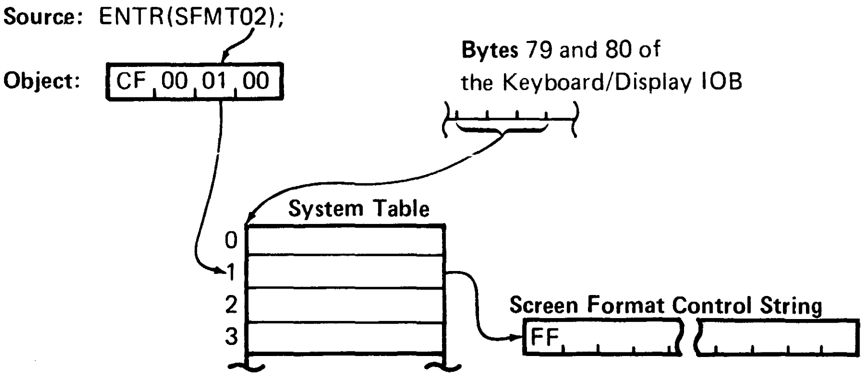
The system table for screen format control strings is built by the assembler when it processes the .SFMT control statements; one system table entry is generated from each series of .SFMT control statements. The address of the system table for screen format control strings that are located within the partition is stored in the keyboard/ display IOB at hex 79, relative to the start of the IOB. The address of the system table for global screen format control strings is stored in the system control block, with the storage page number at hex FB and the address at hex FC.

The system table for screen format control strings that are located within the partition consists of one 2-byte entry for each control string. The 2-byte entry contains the address, relative to the beginning of the partition, where the control string is located. There may be up to 256 control strings within a partition, represented by system table indexes 0 through 255.

The system table for global screen format control strings consists of one 2-byte entry for each global control string. The 2-byte entry contains the address, relative to the beginning of the storage page (in hex FB), where the control string is located. There may be up to 256 global control strings represented by system table indexes 0 through 255. The first global screen format control string is used by the system for the standard load prompt.

The ENTR command in the source program includes the label of the screen control format to use. The assembler converts the label to the system table index, and also determines whether the control string is within the partition or in the common area. If the control string is within the common area, bit 9 of the 4-byte object code instruction is set to 1. During program execution, if bit 9 equals 1 the address of the system table is taken from the system control block. If bit 9 equals 0 the address of the system table is taken from the keyboard/display IOB within the partition.

The following illustration shows how the system table is used to find a screen format control string that is located within the partition. In the illustration, the screen format labeled SFMT02 was the second screen control format set up with a series of .SFMT control statements.


Source: ENTR(SFMT02);
Object:
CF 00 01 00
Bytes 79 and 80 of the Keyboard/Display IOB
System Table
0
1
2
3
Screen Format Control String
FF

System Table for Prompts and Constant Inserts

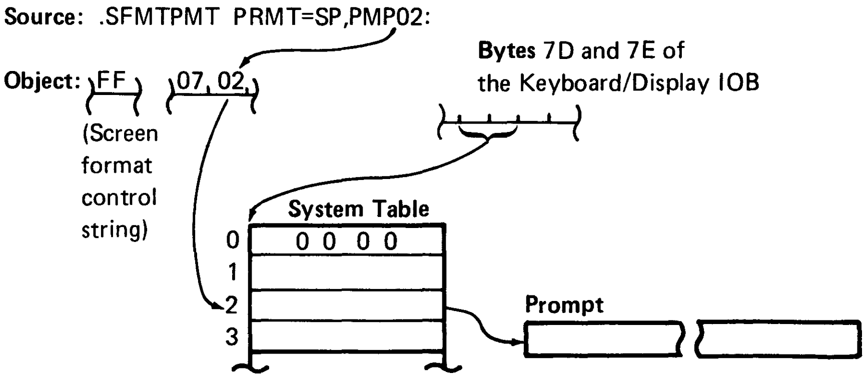
The system table for prompts and constant inserts is built by the assembler when it processes .DC control statements with the parameter TYPE=PRMT. The address of the system table for prompts and constant inserts that are located within the partition is stored in the keyboard/display IOB at hex 7D, relative to the start of the IOB. The address of the system table for global prompts and constant inserts is stored in the system control block at absolute address hex FE. The storage page number where the system table is located is stored in the system control block at hex FB. (It must be on the same storage page as the system table for global screen format control strings.)

The system table for prompts and constant inserts that are located within the partition consists of one 2-byte entry for each prompt or constant insert. The 2-byte entry contains the address, relative to the beginning of the partition, where the prompt or constant insert is located. The first entry in the system table always contains 2 bytes of zeros. The address of the first prompt or constant insert is at index 1 in the table.

The system table for global prompts and constant inserts consists of one 2-byte entry for each global prompt and constant insert. The 2-byte entry contains the address, relative to the beginning of the storage page, where the prompt or constant insert is located. The first entry contains 2 bytes of zeros. The first prompt or constant insert is at index 1 in the table. During program execution, if the screen format control string that referred to the prompt or constant insert is a global screen format control string (indicated by bit 9 of the object code ENTR command), the system table for global prompts and constant inserts is used.

In a source program, a prompt is referred to in a .SFMTPMT control statement; a constant insert is referred to in a .SFMTCNS control statement. The assembler converts the labels included in the control statements to system table indexes. During program execution, when a screen format control string refers to a prompt or constant insert system table index, the system finds the address of the prompt or constant insert in the system table at that index.

The following illustration shows how the system table is used to find a prompt that is located within the partition. The prompt labeled PMP02 was the second prompt set up by a .DC control statement with the TYPE=PRMT parameter.


Source: .SFMTPMT PRMT=SP,PMP02:
Object: FF 07, 02,
(Screen format control string)
Bytes 7D and 7E of the Keyboard/Display IOB
System Table
0
0 0 0 0
1
2
3
Prompt

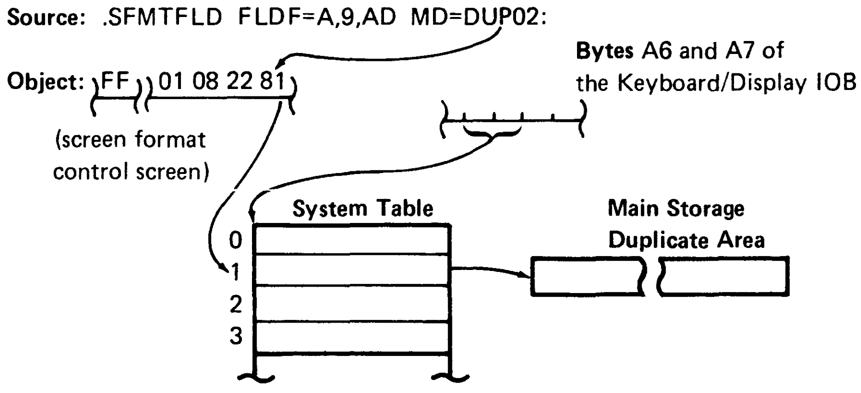
System Table for Main Storage Duplicate Areas

The system table for main storage duplicate areas is built when the assembler processes the .DC control statements that have the parameter TYPE=MDUP. The address of the system table for main storage duplicate areas that are located within the partition is stored in the keyboard/display IOB at hex displacement A6. Global main storage duplicate areas cannot be specified.

The system table for main storage duplicate areas consists of one 2-byte entry for each main storage duplicate area within the partition. The 2-byte entry contains the address, relative to the beginning of the partition, where the area is located. The address of the first main storage duplicate area is in the system table at index 0.

In a source program, a main storage duplicate area is referred to in an .SFMTFLD control statement with an MD=label (duplicate from the label) or an MS=label (store to the label) parameter. The assembler converts the labels to system indexes. During program execution, when a screen format control string refers to a system table index, the system finds the address of the main storage duplicate area in the system table at that index.

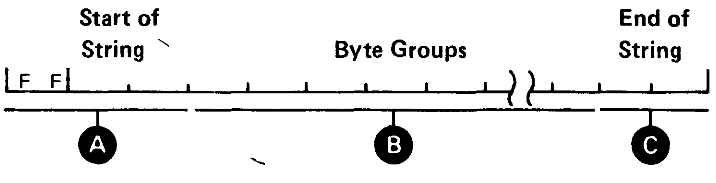
The following illustration shows how the system table is used to find a main storage duplicate area. In the illustration, the area labeled DUP02 is the second main storage duplicate area set up by a .DC control statement with a TYPE=MDUP parameter.


Source: .SFMTFLD FLDF=A,9,AD MD=DUP02:
Object: FF 01 08 22 81
(screen format control screen)
Bytes A6 and A7 of the Keyboard/Display IOB
System Table
0
1
2
3
Main Storage Duplicate Area

SCREEN FORMAT CONTROL STRINGS

The keyboard/display microprocessor uses screen format control strings to format and check data that is entered via the keyboard, displayed on the screen, and stored in the current record buffer in main storage. Screen format control strings are assembled as specified in the source program. For example, with the assembler language, screen formats are specified by the .SFMTST, .SFMTCNS, .SFMTPMT, .SFMCTL, .SFMTFLD, and .SFMTEND statements.

Control information, data fields, prompts, and display attributes are specified by a byte or a byte group in the control string. The order in which the control string is assembled is the order in which the string is processed. The following diagram is a generation description of the contents of the control string. Following this general description is a complete description of each type of specification that can be in the control string.



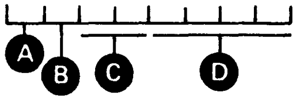
A Each screen format control string must begin with hex FF, followed by a byte group ID and control byte that indicates the start of the screen format control string. See *Start of Control String* under *Control Byte Group.*

B Each byte group contains an ID (see *Byte Group ID*) and other bytes to describe a control specification (see *Control Byte Group*), data field (see *Data Field Byte Groups*), prompt (see *Constant Insert Data and Prompts*), or display attribute (see *Display Attributes*).

C A byte group ID and control byte that indicates the end of the screen format control string. See *End of Control String* under *Control Byte Group.*

Byte Group ID

The type of format specification in each byte group in the control string is identified by the first byte of the group as follows:



A 1 = This is the last byte of the group.[4]

B 1 = Return control to the object code program.[3]

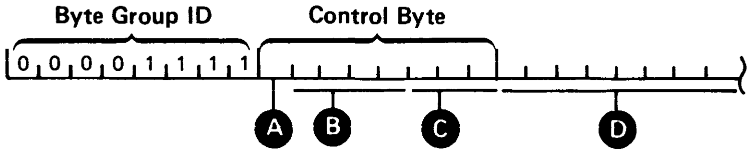
C 00 = Field is neither right-adjust nor field exit required.
01 = Field is right-adjust, alphabetic fill.[1]
10 = Field is field exit required.
11 = Field is right-adjust, numeric fill.[1]

D 0000 = Field is picture check field.[1]
0001 = Field is alphabetic.
0010 = Field is numeric.
0011 = Field is hex.
0100 = Field is Katakana.
0101 = Format level zero.
0110 = Fixed position prompt.
0111 = Standard prompt or constant insert data.
1000 = Invalid specification.[2]
1001 = Field is alphabetic only.
1010 = Field is numeric only.
1011 = Field is digits only.
1100 = Field is Katakana only.
1101 = Invalid specification.[2]
1110 = Display attribute.
1111 = Control specification (see *Control Byte Group*).

[1] If picture check is specified, the field cannot be right-adjust or processed right to left.
[2] Bit values 1000 and 1101 cause external status for invalid format control string to be posted.
[3] If bit 1 is on, the keyboard/display microprocessor returns control to the object code program. When the control string is processed forward, control returns after the format group is processed. When the control string is processed backward (a backspace key was pressed), control returns before the format group is processed.
[4] Bit 0 of each byte in the control string indicates whether this byte is the last byte of a group.

Control Byte Group

Control (such as start of control string and end of control string) is specified in the control string by one or more control bytes. The control byte(s) always follow a control string byte group ID.



Ⓐ 1 = Last byte of this control byte group.

Ⓑ Control Code Modifiers: See *Control Code Description.*

Ⓒ Control Code (see *Control Code Description*):

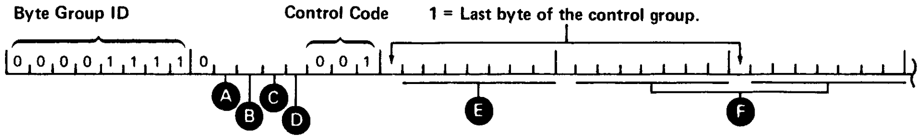
000 = Invalid code.
001 = Change pointer position.
010 = Start of control string.
011 = End of control string.
100 = Check indicator for bypass.
101 = Execute secondary format.
110 = Invalid code.
111 = Change keyboard flags.

Ⓓ Additional Control Bytes: Used for codes 001, 100, 101.

Control Code Description

000 Invalid Code: Control code 000 or 110 causes external status for invalid control string to be posted.

001 Change Pointer Position: Control code 001 causes the microprocessor to change the screen position pointer and/or the current record buffer pointer as follows:



A 1 = Change the screen position pointer.

B 1 = Change the current record buffer position pointer.

C 0 = Add the number of positions to the pointer if the string is processed forward; subtract the number of positions if the string is processed backward.

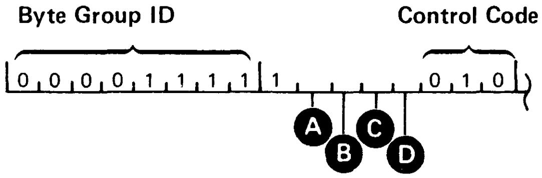
1 = Subtract the number of positions from the pointer if the string is processed forward; add the number of positions if the string is processed backward.

D Not used.

E Number minus 1 of positions to move the pointer if less than 128; otherwise this byte contains 7F and the next 2 bytes indicate the number minus 1 to move the pointer.

F Number minus 1 of positions to move the pointer if the previous byte = 7F.

010 Start of Control String: Control code 010 indicates the start of a screen format control string, as follows:



A 1 = Begin this screen format at the current screen position.

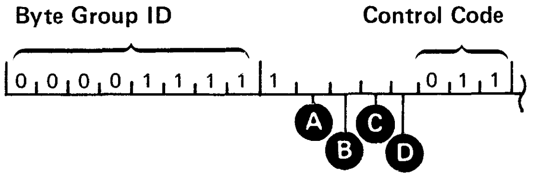
0 = Begin this screen format at row 2, column 1.

B Not used.

C 1 = In enter mode, move prompts and display attributes to the screen, and move data from the current record buffer to the screen. (In modes other than enter, this function is performed automatically.)

D 1 = Clear the screen (except the status line) before any function is performed.

011 End of Control String: Code 011 indicates the end of the format control string as follows:



A Not used.

B 1 = Sound the buzzer.

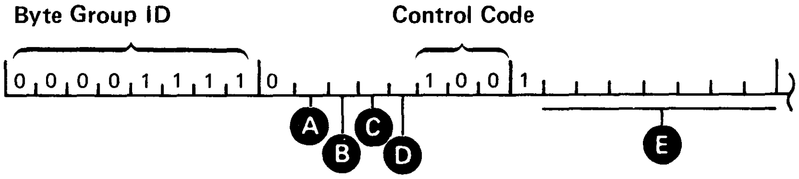
C 1 = Clear the screen except for the status line.

D Not used.

For a primary format, end of control string is processed at record advance time and during a cancel ENTR operation (CNENTR). The status line counters, field shift, and hex display are set to blanks and external status condition 6 is posted to the object program. For a secondary format, end of control string modifiers are ignored; end of control string indicates the end of the secondary format.

100 Check Indicator for Bypass: A check indicator for bypass control group is located at the beginning of and at the end of the end of the section of control string to be conditionally bypassed. If the indicator has the value specified for bypass, all field, display attribute, constant insert, and prompt specifications are bypassed. However, the cursor and current record buffer pointer are moved past the space on the screen and in the current record buffer where the bypassed fields, display attributes, constant inserts, and prompts would have appeared. If the bypass specifications are encountered in a forward direction, the current field counter is incremented by the number of fields bypassed. If the bypass specifications are encountered in a backward direction, the current field counter is decremented. If a return to program (RG), change buffer position pointer (BFPS), change screen position pointer (CSPS), or control specification to change status is encountered during bypass, it is processed as normal. If an execute secondary format (ES) specification is encountered, the fields and control specifications of the secondary format are processed as described above for a bypass.

The check indicator for bypass control group has the following format:



A 1 = Beginning of format string byte groups to bypass.

0 = End of format string byte groups to bypass.

B 1 = Bypass if the indicator is off.

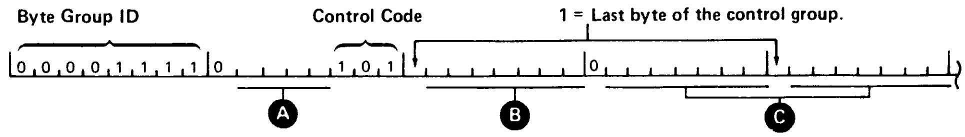
0 = Bypass if the indicator is on.

C Not used.

D Not used.

E Indicator Number (0-127)

101 Execute Secondary Format: The execute secondary format control group causes the keyboard/display microprocessor to interrupt processing this string, process a secondary control string, and return to this string. The format of the execute secondary format control group is as follows:

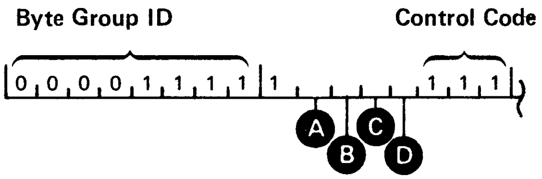


A Not used.

B Index into the system table for screen format control strings, where the address of the secondary format is stored, if the index is less than 128; otherwise this byte contains 7F and the index is specified by the following 2 bytes.

C If the previous byte is 7F, these bits specify the index into the system table, where the address of the secondary format is stored.

111 Change Keyboard Flags: The change keyboard flags control group causes the microprocessor to change the status of the keyboard flags. That is, if the flag is on, it is turned off; if it is off, it is turned on. The keyboard flags are turned off at the start of the processing of each ENTR command.



A 1 = Change the status of the Dup key enable/disable flag.

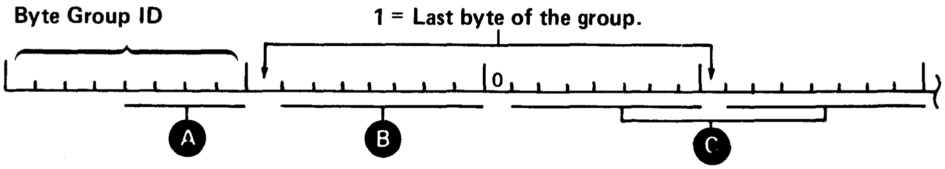
B 1 = Change the status of the monocase enable/disable flag.

C 1 = Change the status of the Field Exit key enable/disable flag.

D 1 = Change the status of the special verify mode enable/disable flag.

Data Field Byte Groups

A data field byte group specifies the format of a data field as it is entered via the keyboard, displayed on the screen, and stored in the current record buffer. The field starts at the current screen position and current record buffer pointer position. Data fields longer than 1 byte require a length specification in the data field format group as shown in the following diagram:



A Byte Group ID—Must specify one of the following:

0000 (picture check)	0101 (format level 0)
0001 (alphabetic)	1001 (alphabetic only)
0010 (numeric)	1010 (numeric only)
0011 (hexadecimal)	1011 (digits only)
0100 (Katakana)	1100 (Katakana only)

B Field length minus 1 if the field length is less than 128; otherwise, this byte contains 7F and the following 2 bytes specify the length minus 1.

C If the previous byte is 7F, these bits specify the length minus 1 of the field.

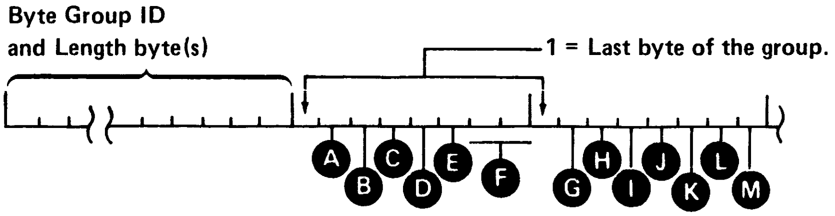
A data field with only the following attributes requires only the byte group ID and (if the field is longer than 1 byte) the field length byte(s) to describe the field in the screen format control string:

- Basic field
- Format level zero field
- Right adjust field
- Field exit required field
- Manual duplicate field from the previous buffer

A data field with additional attributes requires additional bytes to specify field attributes, storage duplication areas, or picture specifications.

Field Attributes and Storage Duplication Group

Field attributes may be specified with 1 or 2 bytes, as necessary. For store and duplicate fields, the attribute byte(s) must be followed by additional byte(s) that specify where to find the address of the duplicate or store area. The format is as follows:



A 1 = Another attribute byte follows this attribute byte.

B 1 = Main storage duplicate field.[1]

C 1 = Verify bypass field.

D 1 = Signed numeric field.

E 1 = Data required field.

F 00 = Field is not auto dup, auto skip, or bypass.
01 = Auto skip field.
10 = Auto dup field.
11 = Bypass field.

G Not used.

H 1 = Main storage store field.[1]

I 1 = Right to left field.

J 1 = Absolutely automatic field.

K 1 = Blank check field.

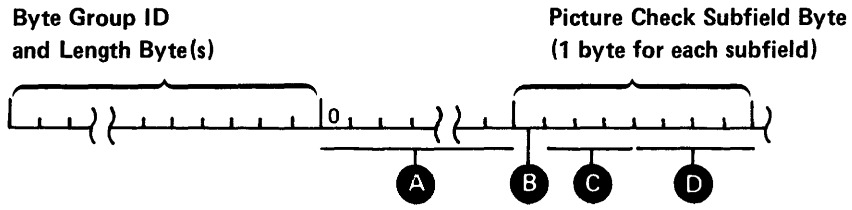
L 1 = Mandatory enter field.

M 1 = Mandatory fill field.

[1] For main storage duplicate and store fields, an index specification must follow the attribute byte(s). The index specification is 1 byte long if the index is less than 126; it is 3 bytes long if the index is 126 or greater (see *Execute Secondary Format*) under *Control Byte Group* for the format of the index specification. The index specified is the entry number into the system table for main storage duplicate areas, where the address of the duplicate area is located. The address of the system table is in bytes hex 46 and 47 of the keyboard/display IOB.

Picture Check Subfield Group

Following are the specifications for picture check subfields:



A Field Attribute Byte(s): See *Field Attributes and Storage Duplication Group,* the previous topic in this section.

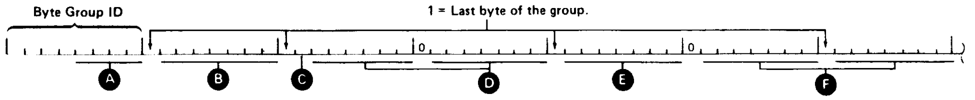
B 1 = Last byte in this group.

C ·Subfield Length Minus 1 (0-7).

D 0001 = Subfield is alphabetic.
0010 = Subfield is numeric.
0011 = Subfield is hex.
0100 = Subfield is Katakana.
1001 = Subfield is alphabetic only.
1010 = Subfield is numeric only.
1011 = Subfield is digits only.
1100 = Subfield is Katakana only.

Constant Insert Data and Prompts

Constant insert format groups specify the location and the length of constant insert data to be moved to the screen and inserted into the current record buffer. Prompt format groups specify the length and location of a prompt to be moved to the screen fixed prompt line or to current screen position. Following are the control string specifications for constant insert and prompts:



Ⓐ 0110 = Fixed position prompt.
0111 = Standard prompt or constant insert.

Ⓑ Index into the system table for prompts, where the address of the prompt is stored if the index is less than 126; otherwise this byte is 7F and the index is specified in the following 2 bytes. For constant inserts, this byte must be 7F.[2]

If this byte is xx000000, the fixed prompt line is cleared.[1]

Ⓒ 1 = Specification is for constant insert data.
0 = Specification is for prompt.

Ⓓ If the previous byte is 7F, these bits specify the index into the system table, where the address of the constant insert or prompt is stored.

Ⓔ Length minus 1 of the constant insert or prompt if the length is less than 128; otherwise, this byte is 7F and the length minus 1 is specified by the following 2 bytes.[1]

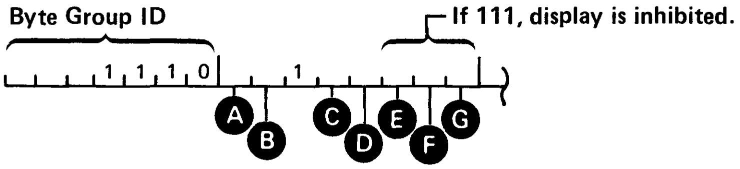
Ⓕ If the previous byte is 7F, these bits specify the length minus 1.[1]

[1] If clear the fixed prompt line is specified, the prompt line is cleared (the number of positions specified in the length bytes of the format group) beginning with the first position of the line. If the length is not specified in the format group, the full line is cleared.

[2] If the constant insert or prompt is stored within the partition, the address of the system table is in bytes hex 0D and 0E (address hex 8D and 8E relative to the start of the partition) of the keyboard/display IOB. If the constant insert or prompt is stored within the common area, the address of the system table is in bytes hex FE and FF of the system control block.

Display Attributes

A display attribute format specification consists of 2 bytes, the format identifier and a display attribute byte, as shown below. The display attribute is moved to the screen at the current screen position.



A 1 = Last byte in the group.

B Not used.

C 1 = Column separators displayed.

D 1 = Blink.

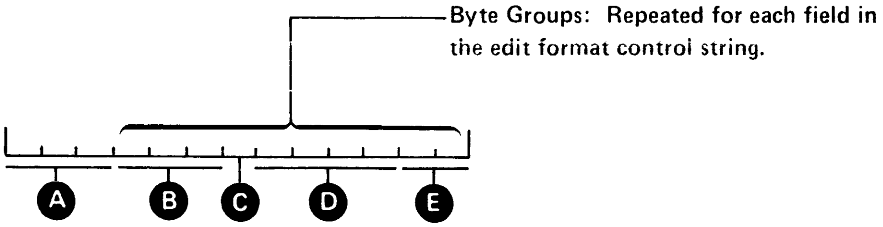
E 1 = Underline.

F 1 = High intensity.

G 1 = Reverse image.

EDIT FORMAT CONTROL STRINGS

Control information and field descriptions are specified by groups of bytes in an edit format control string. The order in which the control string is assembled is the order in which the string is processed. The following diagram is a general description of an edit format control string. Following this general description is a description of each type of specification that can be in the control string.


Byte Groups: Repeated for each field in the edit format control string.
A
B
C
D
E

A Header bytes: 3 header bytes always begin a format control string. If data directed formatting is used, these bytes specify the condition character information.

B End Flag and Displacement: 1 or 3 bytes that indicate the last control string in a series and specify the displacement of the field from the previous field.

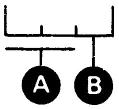
C Edit Flags: 1 byte that indicates data types and edit control information.

D Buffer and Storage Specifications: 4 bytes indicate buffer length and the length and address of the storage area to which, or from which, data is moved.

E Optional Bytes: See *Second Optional Edit Control Byte* and *Picture Specification* under *Byte Groups.*

Header Bytes

The first 3 bytes of a control string are header bytes. If a condition character is used for data directed formatting, the header bytes specify the condition character and the position in the record where the condition character is located.



A Condition Character Position: The displacement minus 1 from the left of the I/O buffer where the character is located. If no condition character is specified, these bytes contain hex FFFF.

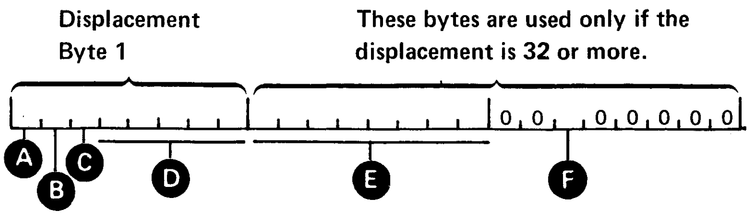
B Condition Character: If no condition character is specified, this byte contains a blank (hex 40).

The header bytes are followed by a series of field description and edit control bytes. Each field in the record is represented by one group of bytes, which begin with the end flag and displacement bytes.

Byte Groups

End Flag and Displacement

One or three bytes specify the displacement from the rightmost position of the previous field to the rightmost position of the current field. The displacement byte also contains a flag that indicates the end of the format control string series. If the displacement is less than 32, 1 byte contains the displacement and the end flag. If the displacement is greater than or equal to 32, 3 bytes are used: the first 2 bytes specify the displacement, and the third byte contains the end flag.


Displacement
Byte 1
These bytes are used only if the displacement is 32 or more.
0 0 0 0 0 0 0
A
B
C
D
E
F

Ⓐ Displacement Length:

0 = Displacement is less than 32; displacement value is specified by bits 3 to 7 of this byte.
1 = Displacement is 32 or greater; displacement value is specified by bits 3 to 7 of this byte and bits 0 to 7 of the next byte. A third byte is used for the end flag.

Ⓑ Displacement direction:

0 = Forward displacement.
1 = Backward displacement.

Ⓒ End flag: 1 = last in this series.

Ⓓ Displacement: If bit 0 of this byte is 0, this is the displacement. If bit 0 of this byte is 1, this is the high-order 5 bits of the displacement.

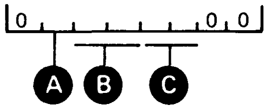
Ⓔ Displacement Byte 2: The low-order 8 bits of a displacement of 32 or more.

Ⓕ End Flag:

0 = Not last in series.
1 = Last in this series.

Edit Flags

The edit flags specify the data type of the data that is moved to or from the I/O buffer, and the type of the I/O buffer. They also indicate whether the optional bytes are used to specify edit or picture descriptions.



A Data Type:

0 = Binary.
1 = Decimal.

B I/O Buffer Type:

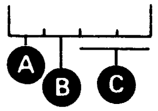
00 = Binary buffer.
10 = Decimal buffer.
11 = Hexadecimal buffer.
01 is not valid.

C Optional Bytes specification:

00 = No optional edit bytes or picture specifications are used.
01 = One optional edit byte is used.
10 = Two optional edit bytes are used.
11 = Picture specification is used.

Buffer and Storage Specifications

Four bytes specify the number of positions in the I/O buffer and in the storage area the field uses, and the address of the storage area.



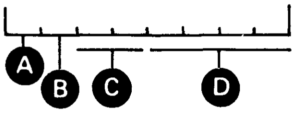
A I/O buffer positions: The number minus 1 of positions the field uses in the I/O buffer.

B Storage positions: The number minus 1 of positions the field uses in the storage area.

C Storage Address: The address of the beginning of the storage area to which, or from which, data is moved.

First Optional Edit Control Byte

This edit control byte may be used only when a decimal buffer is used. If a picture specification is used for the field, this edit control byte is not used, the picture specification follows the storage area address in the edit format control string. The decimal control character, comma control character, and currency control character are found in the partition IOB.



A Comma Control:

0 = No comma control.
1 = Insert the comma control character to separate groups of digits.

B Decimal Control:

0 = Insert blank (hex 40) between the decimal and fraction portions of a number.
1 = Insert decimal control character between the decimal and fraction portions of a number.

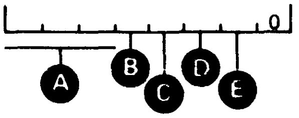
C Fill Character:

00 = Blank fill.
01 = Zero fill.
10 = Asterisk (*) fill.
11 = Floating currency character.

D Displacement from the right of the field to the position where the decimal control character is to be inserted, or 0000 if decimal control is not being used.

Second Optional Edit Control Byte

This edit control byte may be used only when a decimal buffer is used. If a picture specification is used for the field, this edit control byte is not used; the picture specification follows the storage area address in the format control string. The decimal control character, comma control character, and currency control character are found in the partition IOB.



Ⓐ Sign control:

000 = Do not change sign zone in the buffer.
001 = Change sign zone in the buffer to positive (hex F).
100 = Insert blank or minus sign in the field.
101 = Insert a minus or plus sign in the field.
110 = Insert two blanks or the characters CR in the field.
111 = Insert two blanks or the characters DB in the field.

Ⓑ Zero Suppress Control: Valid only with insert decimal.

0 = Force 0 to the left of the decimal control character if the field is 0.
1 = Blank fill if result is 0.

Ⓒ Date Edit Control:

0 = No date edit.
1 = Date edit (bit 5 may be 0 or 1, and all other bits must be 0).

Ⓓ Date Edit Control Character:

0 = Use a slash for data edit (mm/dd/yy).
1 = Use a period for date edit (mm.dd.yy).

Ⓔ Currency Control Character:

0 = No fixed currency character.
1 = Fixed currency character.

Picture Specification

Picture specifications are used only to write to a decimal buffer. If a picture specification is used, the optional edit control bytes are omitted; in the format control string, the first picture byte follows the storage area address.

A picture specification consists of a series of 1-byte hex codes. Each hex code pertains to the corresponding byte in the decimal buffer. Each series of hex codes, ending with an end of string byte, describes one subfield of the current field description. Picture specifications are of variable length; however, a picture specification for a global format is limited to 10 bytes, including the end of string byte.

Hex Code **Meaning**

00 Decimal digit. A decimal digit is accepted in the corresponding position of the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
12345	0000000000	12345

01 Suppress leading zeros. If the character in this subfield position is a leading zero, it is replaced with a blank in the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
00123	0101010000	123

02 Insert blank. A blank is inserted into this position in the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
12345	0000020000	12 345

03 Insert blank if zero. If the character in this subfield position is zero, it is replaced with a blank in the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
10203	0303030000	1 203

04 Insert asterisk. If this subfield position is a leading zero, it is replaced with an asterisk in the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
00123	0404040404	**123

Hex Code **Meaning**

05 Insert comma character. A comma character is inserted into the buffer at this position unless zero suppression has occurred. If zero suppression has occurred, a blank is inserted. Examples:

Subfield Input	*Hex Codes*	*Output to Buffer*
00123	010105000000	123
00123	000005000000	00,123

06 Insert slash. A slash is inserted into the buffer at this position unless zero suppress has occurred. If zero suppression has occurred, a blank is inserted. Examples:

Subfield Input	*Hex Codes*	*Output to Buffer*
000285	0101060101060101	2/85
000285	0000060000060000	00/02/85

07 Insert decimal character. A decimal character is inserted into the buffer at this position unless zero suppression has occurred. If zero suppression has occurred, a blank is inserted. Examples:

Subfield Input	*Hex Codes*	*Output to Buffer*
123456	0005000000070000	1,234.56
0001	0101070101	1

08 Stop zero suppression. Zero suppression is stopped at this position in the buffer. This code must be followed by a 05, 06, or 07 code. The 08 code does not insert a blank or any character into a buffer position. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
0001	010108070000	.01

09 Insert currency character. A fixed currency character is inserted into the buffer if only one 09 code is used. If an 09 code is placed into every leading digit position of the subfield, a floating currency character is placed into the buffer at the left of the most significant digit. The currency character requires two bytes of buffer space. Examples:

Subfield Input	*Hex Codes*	*Output to Buffer*
01234	09010101070101	$ 12.34
01234	090909070000	$12.34

Hex Code **Meaning**

0A Insert minus sign. If the field value is negative, a minus sign is inserted into this position of the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
00012	0A0101010101	- 12

0B Insert plus sign. If the field value is positive, a plus sign is inserted into this position of the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
12345	0B0000000000	+12345

0C Insert sign. The appropriate sign is inserted into this position of the buffer. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
12345	0C00000000	-12345

0D Insert CR. If the value of the subfield is negative, the characters CR are inserted into the buffer. If the value is positive, the two buffer positions are blank. Example:

Subfield Input	*Hex Codes*	*Output to Buffer*
00123	0101010101010D	123CR

0E Insert DB. If the value of the subfield is negative, the characters DB are inserted into the buffer. If the value is positive, the two buffer positions are blank.

0F End of string flag. The hex code string for each subfield must end with 0F.

This page intentionally left blank

Chapter 3. Keyboard/Display Storage

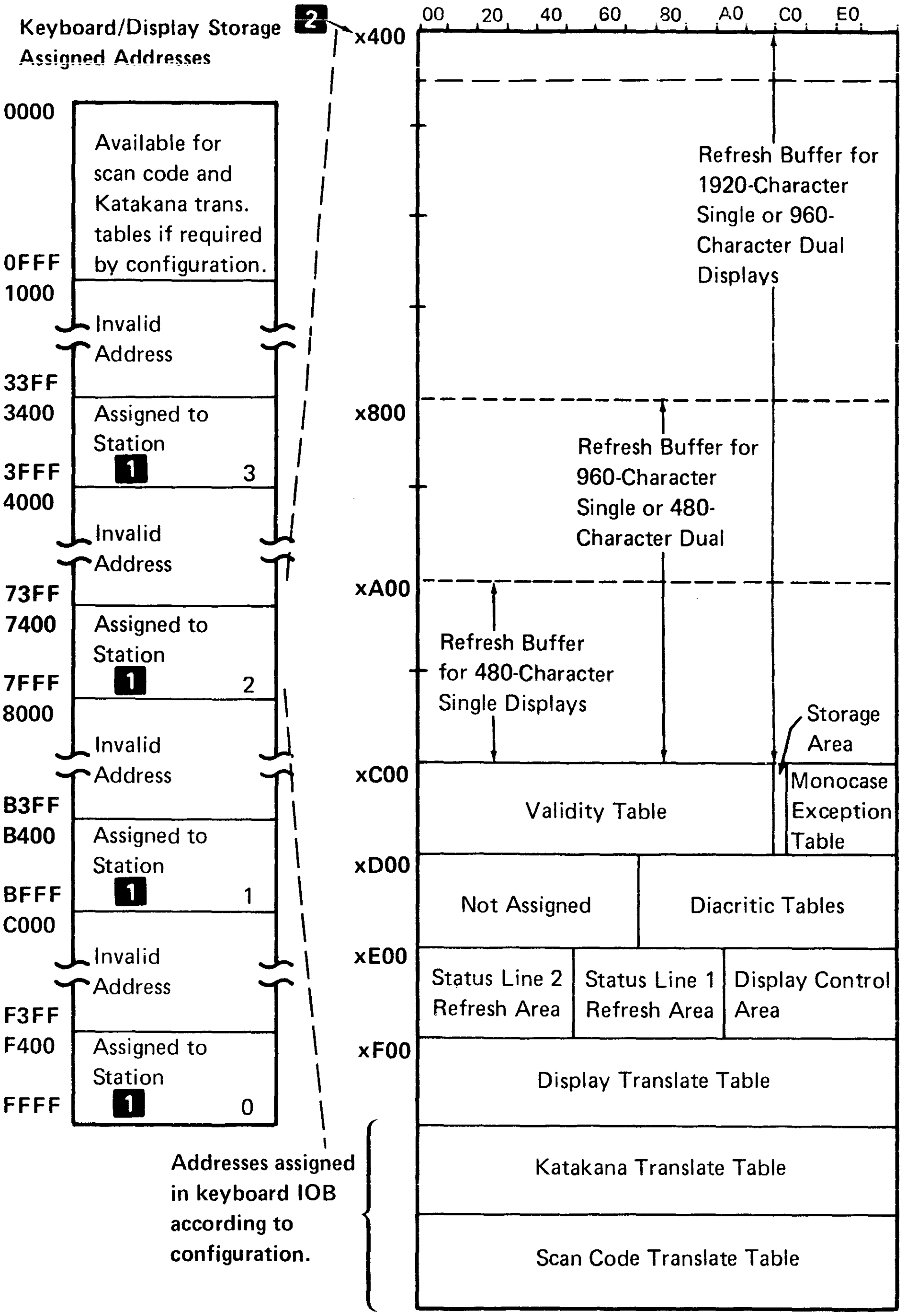
The keyboard/display storage provides control information and refresh buffers for processing keystrokes and for displaying characters on the display screen. Each keyboard/display unit uses a separate portion of keyboard/display storage. The total size of the portion of keyboard/display storage used by each keyboard/display unit depends on the size of the refresh buffer necessary for the keyboard/display unit's screen.

The keyboard/display storage is loaded during system IPL from the IPL diskette. The keyboard/display IOB in each partition contains the addresses of the keyboard/display storage areas used by that partition's keyboard/display unit.

The following is a general description of the data areas and refresh buffer areas within keyboard/display storage. The addresses for each keyboard/display unit begins with an x, which represents hex F, B, 7, and 3 for keyboard/display units 1 through 4 respectively. For example, if all keyboard/display units have a screen size of 1920 characters, the keyboard/display storage for unit 1 begins at hex F400, for unit 2 at hex B400, for unit 3 at hex 7400, and tor unit 4 at nex 3400. There is also additional storage in a fifth section, which starts at hex 0000 and which is shared by the four units.

On a dual unit, the two keyboards share the same keyboard/display storage section. The first keyboard (keyboard 0) uses the lower-numbered rows of the refresh buffer, control register 0, cursor address register 0, and status line refresh buffer 1. The second keyboard (keyboard 1) uses the higher-numbered rows of the refresh buffer, control register 1, cursor address register 1, and status line refresh buffer 2.

The following illustration shows the format of keyboard/display storage as it is generated for IPL bv the system configuration program (SYSCON).



Notes:

1 Station is either single or dual display. Keyboards for a dual share one block of storage.

2 x = F for station 0; B for station 1; 7 for station 2; 3 for station 3.

Because the keyboard/display IOB in each partition contains pointers into the keyboard/display storage, the validity table, storage area, diacritic table, scan code translate table, and the Katakana translate table (if required) can be located anywhere in keyboard/display storage as long as the tables that require alignment on a 256-byte boundary are properly aligned. However, the refresh buffer, status line refresh area, and display translate table for a particular keyboard must all be located in the same section of keyboard/display storage (section F, B, 7, or 3). The display translate table must always begin at address xF00, and the display control area must begin at address xEA0 of the appropriate section of keyboard/display storage.

REFRESH BUFFER AREA

The keyboard/display storage contains refresh buffers for each keyboard/display unit. These buffers act as refresh areas for display characters. The refresh area for the status line(s) is separated from the refresh area for the remainder of the screen. This separate area is in addition to the refresh area appropriate for a particular screen size.

When a keystroke is processed by the keyboard/display microprocessor, it is translated from the keystroke scan code to EBCDIC code. The EBCDIC code is placed into the current record buffer in main storage within the partition associated with the keyboard, and translated to display code. The display code is then placed into the refresh buffer in order to be displayed on the screen. The hexadecimal representations of screen attributes are also placed into the refresh area.

VALIDITY TABLE

The validity table defines:

- The EBCDIC values used in the alphabetic only, numeric only, and Katakana only character sets.
- The EBCDIC values of keys defined as diacritics.
- The EBCDIC values that have to be translated to uppercase when the monocase function is enabled.
- The scan codes of keys that are not typamatic.
- The scan codes of keys that can be shifted from lowercase alphameric only if a shift key (not including the Shift Lock key) is simultaneously pressed, such as the function keys to the left of the keyboard.

The validity table contains 1-byte entries that are in the following format:

Bit	Meaning When 1
0	Ignore the typamatic action in the scan code.
1	Shift only if the shift key is pressed.
2	System use only (initialized to 0).
3	Translate EBCDIC code to uppercase if monocase function is enabled.
4	EBCDIC code used for diacritic.
5	EBCDIC code belonging to Katakana-only character set.
6	EBCDIC code belonging to numeric-only character set.
7	EBCDIC code belonging to alphabetic only character set.

Bits 0 and 1 in the table are used when the table is accessed using a scan code. The 7-bit scan code is an index into the validity table to retrieve the corresponding 1-byte entry.

Bits 3 through 7 are used when the table is accessed using an EBCDIC. The value hex 40 is subtracted from the EBCDIC code to establish the offset into the table in order to retrieve the corresponding 1-byte entry.

STORAGE AREA

The storage area holds information needed for interpreting keystrokes and maintaining the status line, and a monocase exception table. Following is a description of the first 16 bytes of this storage area:

Byte	Description
0	Display code for the insert mode indicator.
1	Display code for the alphabetic shift symbol.
2	Display code for the numeric shift symbol.
3	Display code for the hexadecimal shift symbol.
4	Display code for the Katakana shift symbol.
5	Scan code for the Hex key function.
6	Scan code for the Power On Reset key function.
7	Scan code for the Console key function.
8	Not used.
9	Display code for the alphabetic-only shift symbol.
10	Display code for the numeric-only shift symbol.
11	Display code for the digits-only shift symbol.
12	Display code for the Katakana-only shift symbol.
13-15	Not used.

Display codes for the shift symbols are displayed on the status line to show the keyboard shift status of the current field.

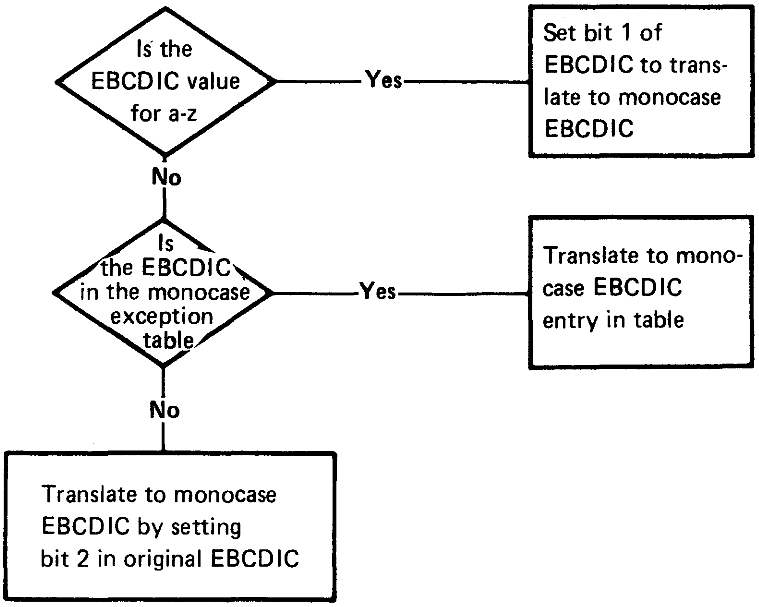
Scan codes for the command (function) keys in bytes 5, 6, and 7 are processed by the system. These functions are initiated by pressing the Cmd key first, then the command key.

Monocase Exception Table

Following the first 16 bytes is a monocase exception table. The monocase exception table contains character values that cannot be conveniently converted from lowercase to uppercase. (See the logic shown below.) The table begins at displacement hex 10 into the storage area. The table contains pairs of bytes (lowercase code/uppercase code) that provide translation from lowercase EBCDIC to uppercase EBCDIC. The byte pairs are in ascending order of the EBCDIC for the lowercase values. The length of the table is variable, depending on the number of entries required. The table always ends with hex FFFF; if the table contains no other entries, it contains only hex FFFF.

Exception EBCDICs, Lowercase	Exception EBCDICs, Uppercase
ae	AE
.	.
.	.
.	.
FF	FF

A bit in the validity table is used to specify that an EBCDIC can be monocase. If the monocase flag is set and an EBCDIC value is entered (by a keystroke or diacritic or hex key sequence during formatted data entry, or by keyboard operation hex 0A [pass scan code] or 0B [pass EBCDIC], or by the KACCPT instruction) that can be monocase, the system translates the lowercase EBCDIC to its corresponding uppercase EBCDIC. The following shows how the system translates the EBCDIC to monocase:


Is the EBCDIC value for a-z
Yes
Set bit 1 of EBCDIC to translate to monocase EBCDIC
No
Is the EBCDIC in the monocase exception table
Yes
Translate to monocase EBCDIC entry in table
No
Translate to monocase EBCDIC by setting bit 2 in original EBCDIC

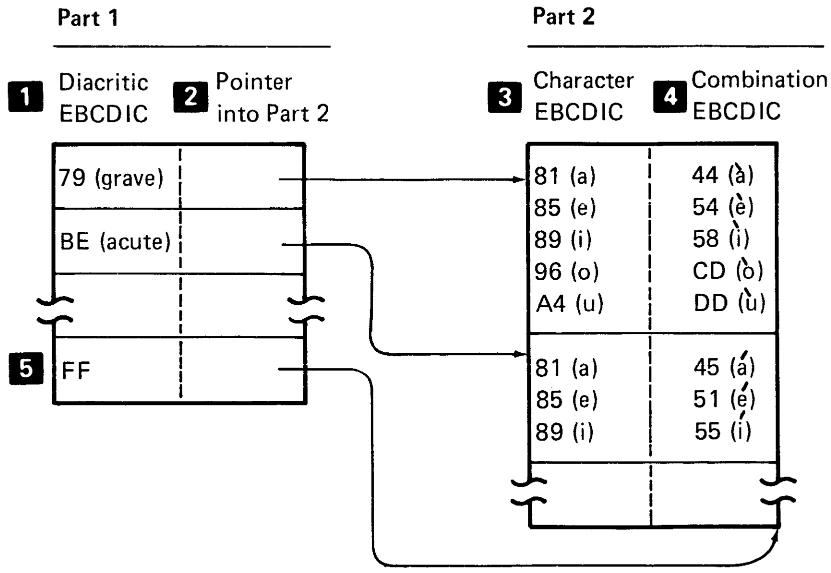
DIACRITIC TABLE

The diacritic table provides composite EBCDIC codes that represent the diacritic-character pairs for characters defined as diacritic in the validity table.

The diacritic table is in two parts. Part 1 contains 2-byte entries for each diacritic defined. Byte 1 is the EBCDIC code for each diacritic and byte 2 is a pointer into part 2 of the table.

Part 2 of the diacritic table contains the EBCDIC code for each character that can be used with a diacritic-character pair and also contains the composite EBCDIC code that represents the diacritic-character pairs.

The following shows how the diacritic table is used:



1 EBCDIC for valid diacritics, arranged in ascending order of diacritic EBCDIC value.

2 Displacement (from the beginning of part 1) into part 2 of the table, where the EBCDIC for the first of the characters that can be validly combined with the diacritic is stored.

3 EBCDIC values for characters that can be validly combined with the diacritic.

4 EBCDIC of the combined character and diacritic. For each diacritic, this section is arranged in ascending order of the character EBCDIC value.

5 The last entry in part 1 contains hex FF, in the first byte, and a pointer to the byte following the last combination EBCDIC in part 2 of the table.

REFRESH AREAS FOR THE STATUS LINES

There are two status line refresh buffers in keyboard/display storage for each unit. These buffers are referred to as status line 1 buffer and status line 2 buffer. The status line 1 buffer is used as a refresh area for the status line of a single data station or station 0 of a dual-display data station. The status line 2 buffer is used as a refresh area for the status line of station 1 of a dual data station. The status line is usually displayed on row one of the screen. However, if a screen format uses all of the rows on the screen, the status line can be removed from the screen so that row one of the format can be displayed on row one of the screen. The status line is maintained in the status line refresh buffer whether or not it is being displayed on the screen.

DISPLAY CONTROL AREA

The display control area contains:

- Display attributes for the beginning of each row on the display screen.
- The refresh buffer address of the first position of the row.
- Control registers that provide control for the upper and lower halves of the display screen.
- Cursor address registers that provide the current refresh buffer address of the cursor.

The display control area begins with strings of 3-byte entries; one entry for each row on the display screen.

The first byte of a 3-byte entry contains the display attributes for each row. The format of the first byte is:

Bits	Attribute Description
0-1	System indicator: 00 = None 01 = None 10 = Dash 11 = Solid rectangle
2	Valid row starting attribute. This bit must be 1 in order for bits 3 through 7 to be effective.
3	1 = Column separator.
4	1 = Blink.
5	1 = Underscore.
6	1 = High intensity.
7	1 = Reverse image.

Note: If bits 5, 6, and 7 are all on (111), no data is displayed.

The second and third byte contain the refresh buffer address of the first position of the row.

The first 3-byte entry in the display control area describes row 1, the second entry row 2, and so on through row 25. Row 0 is described by the twenty-sixth entry.

For dual display stations, rows 0 through 11 are assigned to display station 0; rows 14 through 25 are assigned to display station 1.

Control registers 0 and 1 follow the strings of 3-byte entries. Control register 0 is used for the display screen of a single display station, or for display station 0 of a dual display station. Control register 1 is used for display station 1 of a dual display station.

The format of control register 0 is:

Bits	**Meaning When 1**
0	Inhibit display of upper half of screen if single, station 0 if dual.
1	Not used (initialized to 0).
2	Blink cursor for display station 0.
3	Blink upper half of the display screen if single, station 0 if dual.
4	Reverse image of upper half of screen if single, station 0 if dual.
5-7	Not used (initialized to 00).

The format of control register 1 is:

Bits	**Meaning When 1**
0	Inhibit display of lower half of screen if single, station 1 if dual.
1	Not used (initialized to 0).
2	Blink cursor for display station 1.
3	Blink lower half of screen if single, station 1 if dual.
4	Reverse image of lower half of screen if single, station 1 if dual.
5-7	Not used (initialized to 000).

Following the control register bytes there are two 2-byte cursor address registers. These registers contain the current refresh buffer address of the cursor. Cursor address register 0 stores the cursor address for a single display station or for display station 0 of a dual display station. Cursor address register 1 stores the cursor address for display station 1 of a dual display station.

xEA0	Row starting attribute Row starting address high Row starting address low	Row 1
xEA3	Row starting attribute Row starting address high Row starting address low	Row 2
xEE8	Row starting attribute Row starting address high Row starting address low	Row 25
xEEB	Row starting attribute Row starting address high Row starting address low	Row 0
xEEE	Not used	
xEF2	Control register 0	
xEF3	Control register 1	
xEF4	Cursor address register 0, high	
xEF5	Cursor address register 0, low	
xEF6	Cursor address register 1, high	
xEF7	Cursor address register 1, low	

DISPLAY TRANSLATE TABLE

The display translate table converts EBCDIC code to display code so characters are displayable on the display screen. The display translate table must be located in the last 256 bytes of the keyboard/display storage area assigned to the unit.

KATAKANA TRANSLATE TABLE

The Katakana translate table is required for a display station with a Katakana keyboard. Some keytops on the Katakana keyboards have more than two characters. The right side of the keytop has Katakana characters; the left side has alphameric symbols.

The translate table converts scan codes to EBCDIC for the Katakana characters.

The table is divided into two 128-byte segments. The lowerhalf of the table (offset hex 00 to 7F) is used when the keyboard is in Katakana lowershift. The upper half of the table (offset hex 80 to FF) is used when the keyboard is in Katakana uppershift.

SCAN CODE TRANSLATE TABLE

The scan code translate table converts scan codes to EBCDIC for all keyboards. Katakana keyboards also use the Katakana translate table.

The scan code translate table is divided into two 128-byte segments. The lower half of the table (offset hex 00 to 7F) is used when the keyboard is in alphameric lowershift. The upper half of the table (offset hex 80 to FF) is used when the keyboard is in alphameric uppershift.

Bits 1 through 7 of the 8-bit scan code are used as an offset into the table, either into the lower half or into the upper half, depending on the keyboard shift status. For example, a scan code of hex 09 locates the EBCDIC in the tenth byte of the lower half of the table if the keyboard is in lowershift.

Chapter 4. Object Code Instruction Format

Each object code instruction is 4 bytes long. The first byte always contains the operation code. The other three bytes contain flags, addresses and other data.

ADDRESSING METHODS WITHIN A PARTITION

In a source program instruction, a storage area or another instruction is referred to by a label. A register is referred to by a label or by a number. When source instruction is converted to object code, these labels and numbers are converted to addresses. An address in an object code instruction is always relative to the beginning of the partition. When the object code instruction is executed, the relative address is added to the absolute address of the beginning of the partition. The absolute address of the beginning of the partition is stored in displacement hex 0D in the partition IOB.

Because instructions and registers must begin on specific boundaries, the 16-bit address can be compressed. The bits in the object code instruction that are not used for the address are used for other purposes, such as flags. A relative address in an object code instruction is in one of the following formats:

- *16-bit address* to locate any byte within a partition
- *14-bit address* to locate an instruction
- *8-bit address* to locate a decimal register
- *7-bit address* to locate a binary register

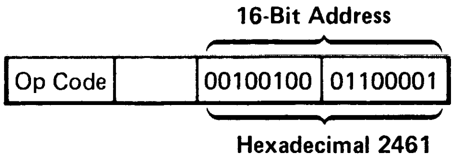
In addition to the relative addresses, an object code instruction may contain the following types of data:

- *8-bit instruction displacement*, used with certain branch instructions to locate an instruction.
- *8-bit indicator number* to locate an indicator.
- *8-bit index* into a system table to locate the address of a format, prompt, duplication area, or table.
- *Constant.*

Addressing a Byte Within the Partition

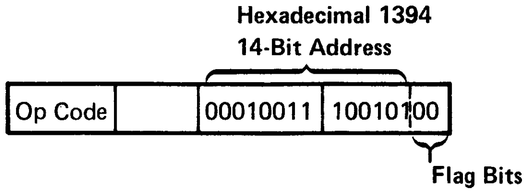
The size of the partition cannot be greater than 64 K bytes; therefore, any byte within the partition can be addressed with 16 bits (hex 0000 through FFFF). A 16-bit address is stored in the third and fourth byte of an object code instruction.

Example:



Addressing an Object Code Instruction

Because object code instructions begin on 4-byte boundaries, the last 2 bits of the 16-bit address are always zeros. These 2 bits can be used for flags; the high-order 14 bits are used to address the instruction. In an object code instruction, a 14-bit address is stored in the high-order 14 bits of the third and fourth bytes as follows:



Instruction Displacement

In certain branch instructions, the label in the source instruction is converted to a displacement rather than to an address. An instruction displacement is the number of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken. An instruction displacement is 8 bits long and is stored in the fourth byte of an object code instruction. A positive displacement can cause a forward jump of up to 128 object code instructions. A negative displacement is stored in the twos complement of the displacement value. A negative displacement causes a backward jump of up to 128 object code instructions from the instruction following the branch instruction.

Addressing a Decimal Register

Each decimal register begins on a 16-byte boundary from hexadecimal 0100 to 0FF0 (relative to the beginning of the partition).

In a source program, a decimal register is specified by a register number or a label, which is converted to a 16-bit address in the object code. All 16-bit addresses for decimal registers begin and end with zero, as the following chart shows:

R62 is stored at location hex 04E0.

Hex	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
01	0	1	2	3	4	5	6	7	8	9	10	11	12	13	~~14~~	15
02	16	17	18	19	20	21	22	23	24	25	26	27	28	29	~~30~~	31
03	32	33	34	35	36	37	38	39	40	41	42	43	44	45	~~46~~	47
04	~~48~~	~~49~~	~~50~~	~~51~~	~~52~~	~~53~~	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	~~59~~	~~60~~	~~61~~	62	63
05	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
06	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
07	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
08	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
09	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
0A	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
0B	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
0C	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
0D	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
0E	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
0F	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239

Following is an alternative method to convert a register number (using R62 as an example) to a 16-bit address:

1. Multiply the register number by 16: 16 x 62 = 992

2. Convert the product to hexadecimal: decimal 992 = hexadecimal 03E0

3. Add hexadecimal 0100: 03E0 + 0100 = 04E0

When the program is assembled, the 16-bit address is compressed to an 8-bit address:

16-Bit Address 04E0 → 4E → E4 **8-Bit Address**

1. Remove the zeros from the beginning and end of the 16-bit address.

2. Reverse the remaining two digits.

Addressing a Binary Register

Each binary register begins on a 2-byte boundary from hexadecimal 0100 to 01FE (relative to the beginning of the partition). In a source program, a binary register is specified by a register number or a label, which is converted to a 16-bit address in the object code. All 16-bit addresses for binary registers begin with 01, as the following chart shows:

BR62 is stored at location hex 017C.

Hex	0	2	4	6	8	A	C	E
010	0	1	2	3	4	5	6	7
011	8	9	10	11	12	13	14	15
012	16	17	18	19	20	21	22	23
013	24	25	26	27	28	29	30	31
014	32	33	34	35	36	37	38	39
015	40	41	42	43	44	45	46	47
016	48	49	50	51	52	53	54	55
017	56	57	58	59	60	61	62	63
018	64	65	66	67	68	69	70	71
019	72	73	74	75	76	77	78	79
01A	80	81	82	83	84	85	86	87
01B	88	89	90	91	92	93	94	95
01C	96	97	98	99	100	101	102	103
01D	104	105	106	107	108	109	110	111
01E	112	113	114	115	116	117	118	119
01F	120	121	122	123	124	125	126	127

Following is an alternative method to convert a binary register number (using BR62 as an example) to a 16-bit address:

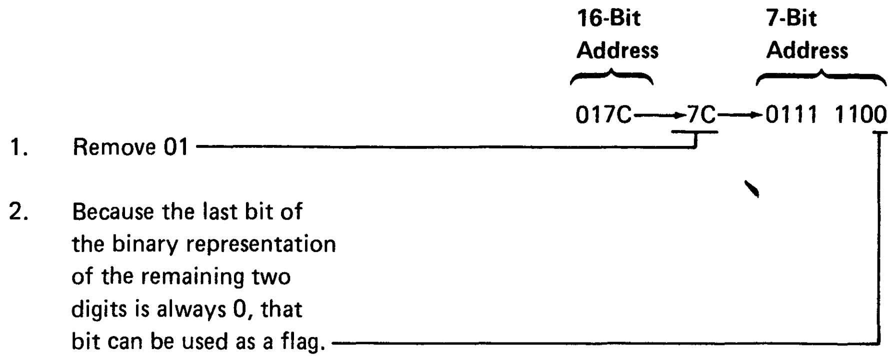
1. Multiply the register number by 2: 62 x 2 = 124

2. Convert the result to hexadecimal: decimal 124 = hexadecimal 007C

3. Add hexadecimal 0100: 007C + 0100 = 017C

If a binary double register is referred to in a source instruction, the address in the object code is the address of the rightmost register.

When the source program is assembled, the 16-bit address for a binary register is compressed to a 7-bit address:



16-Bit Address

7-Bit Address

017C → 7C → 0111 1100

1. Remove 01

2. Because the last bit of the binary representation of the remaining two digits is always 0, that bit can be used as a flag.

Indicator Addressing

An indicator is specified in a source instruction with a label or a decimal number (I0 to I254). This label or decimal number is converted to the hexadecimal representation of the indicator number in the object code.

For example:

Indicator Number	Hex Value	Binary Value Stored in the Instruction
I47	2F	00101111
I06	06	00000110
I126	7E	01111110

Addressing through a System Table

In a source program, a format, prompt, data table, or main storage duplication area is referred to with a label. In the object code, this label is converted to an index into a system table. This system table holds the addresses of the labeled data areas, and the index specifies the position in the system table where the appropriate address is stored. The index for a format or table is stored in one byte of the object code instruction; however, the index for a prompt or main storage duplication area is stored within the screen format control string. Except for prompts, the first address in a system table is at index 0. The following chart shows the valid range for the system table index for each type of data area:

	Valid
	Index
Type	**Values**
Screen format	0-255
Edit format	0-127
Data table	0-127
Prompt	1-(see note)
Duplication area	0-(see note)

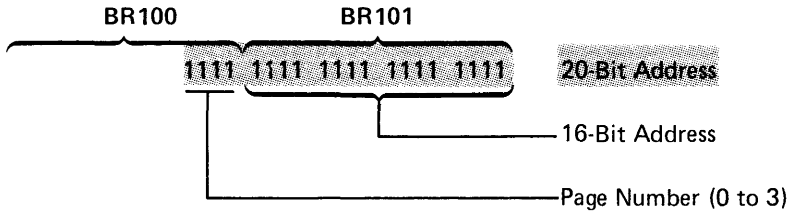
Note: The number of prompts and duplication areas is limited only by storage size and performance considerations.

The formats of the system tables are described in Chapter 2 under *System Tables.*

ADDRESSING METHODS OUTSIDE THE PARTITION

A 10-bit address must be used to address any location outside the partition. The format of a 20-bit address is a 16-bit address preceded by a 4-bit storage page number. Main storage is divided into storage pages; each storage page is 64 K bytes (K = 1024 bytes). A page with less than 64 K bytes is a partial page. The 16-bit address can address any byte within the page; therefore, the 20-bit address that includes the page number can address any byte within main storage.

An instruction never contains a 20-bit address. In a source program, the 20-bit address must be stored in a double binary register. When a source instruction refers to the 20-bit address, it specifies the label or number of the leftmost register of the double binary register that holds the address. The assembler converts the register specification to a 7-bit compressed address of the *rightmost* register of the double register that holds the address. For example, a source program has a 20-bit address stored in BR100 and BR101. The source instruction specifies BR100(4), where the 4 represents a length of 4 bytes. The assembler stores the 7-bit compressed address of BR101 in the object code instruction; the flag bit is set to 1 to indicate that the register is part of a double register that holds a 20-bit address. BR101 holds the 16-bit address and the low-order 4 bits of BR100 specify the page number.



Addressing through a System Table

Format and tables stored in the common area are available to any partition. When a source program specifies that the format or table is in the common area (with an .XTRN control statement), the format or table is assigned a system table index that is greater than a valid index for a table or format within the partition. The following chart shows the range for a system table index for data areas outside the partition:

Type	Valid Index Values
Screen format	256-512
Edit format	128-254
Data table	128-254

For edit formats and data tables, the index is stored in one byte of the object code instruction. For screen formats, a bit is set in the object code for the enter instruction (hex CF) to indicate that the screen format is in the common area.

CONSTANTS

In a source program, a constant can be specified (1) as a decimal, hexadecimal, or binary value, (2) as a character, or (3) with a label that is equated to a value. In the object code, any form of constant is stored in the object code as immediate data. The following list shows the kinds of constants that are used in a source program.

- *Data set number:* The number of the current data set. The number can be any number from hexadecimal 1 to F and requires 4 bits of object code.
- *Length:* The length of data being used by the instruction.
- *Displacement:* The displacement into a data area; usually an optional parameter in a data movement instruction.
- *Mask:* A pattern of bits used in skip operations. Each mask requires 8 bits.

INSTRUCTION FORMAT

Mnemonic to Op Code Conversion Chart

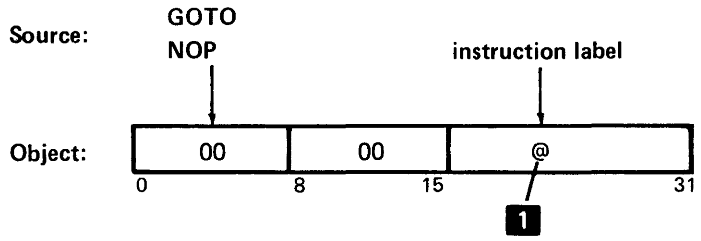
The object code instructions in this chapter are in op code (hexadecimal) order. If it is necessary to find an object code instruction by assembler language mnemonic, use the following chart to find the op code. The mnemonics in the chart are listed in alphabetic order.

	Op		Op
Mnemonic	Code	Mnemonic	Code
ALLOC	34	d(len,BRa) = BRb	A3
AND	42	d(len,BRn) = constant	B9
BINDEC	A6	d,Rn = constant	44
BINHEX	49	d(len,BRn) = Rn	7n
BRa = BRb	98	DECBIN	A7
BRa <=> BRb	45	DECR BRn	06
BRn = constant	99	DISPEX	C7
BRn(4) -=	96	DISPST	C7
BRn(4) + = nn	95	DUP	BD
BRn(4) - = nn	97	DVCTL	3D
BRn(4) +=	94	ENABLE	OC
BRn [(4)] /=	AB	ENTR	CF
BRn &=	9A	EXIT	2F
BRn &= d(len,BRn)	BA	GOTAB BRn	08
BRn &= nn	9B	GOTO	00
BRn V=	9C	GOTO BRn (indexed)	08
BRn V= d(len,BRn)	BC	GSCK	48
BRn V= nn	9D	HEXBIN	4A
BRn X=	9E	IF BRn EQ	6E
BRn X= d(len,BRn)	BE	IF BRn GE/LE	6F
BRn X= nn	9F	IF BRn GT/LT	6D
BRn +=	90	IF BRn NE	6C
BRn -=	92	IF BRn 0	03
BRn * =	AA	IF FMT	02
BRn + = d(len,BRn)	B0	IF Rn AN	0D
BRn - = d(len,BRn)	B2	IF Rn CK	0E
BRn = (indexed)	B8	IF Rn EQ	62
BRn - = n	93	IF Rn GE/LE	63
BRn + = n	91	IF Rn GT/LT	61
BRn(4) + = d(len,BRn)	B4	IF Rn NE	60
BRn(4) - = d(len,BRn)	B6	IF Rn SN	0F
BRn = Rn	A7	IF Rn 0	01
BUZZ	C7	IF Rn -	05
CALL	OB	IFB IS	BB
CALLTB	OB	IFB OFF	B5
CLC	AE	IFB ON	B7
CLICK	C7	IFC IS	4E
CLOZ	23	IFC NOT	4C
CNENTR	C7	IFD Rn EQ	66
CRTMM	CA		

Mnemonic	Op Code	Mnemonic	Op Code
IFD Rn GE/LE	67	Rn -	11
IFD Rn GT/LT	65	Rn +	10
IFD Rn NE	64	Rn *	18
IFDSI	25	Rn /	17
IFH BRn EQ	6A	Rn (32) *	15
IFH BRn GE/LE	6B	Rn (32) /	12
IFH BRn GT/LT	69	Rn = BRn	A6
IFHI	42	Rn = d(len,BRn)	7n
IFI In	07	Rn = label	8n
IFIR In	04	Rn = +nn	46
IFLO	42	Rn = -nn	47
INIT	33	RR	A1
INSBLK	32	RSTMG	C7
INXEQ	A5	RTIMER	C7
KACCPT	C7	RXORW	43
KDETCH	C5	SCRTC	C9
KERRCL	C7	SEARCH	24
KERRST	C7	SETOFF	B3
KEYOP	C7	SETON	B1
label = BRn	A2	SKIP WHILE	A0
label = constant	44	SL (binary)	A1
label = Rn	8n	SL (decimal)	1C
label = SL n	A1	SLS	1D
LCRTC	C8	SOFF	41
LOAD	2E	SON	40
MMCRT	CB	SR (binary)	A1
MOFF	1A	SR (decimal)	16
MVC	AC	SRAT	2B
MVC(BRn(4))	A4	SRR	1F
MVCR	AC	SRS	1E
MVCV	AC	SYSLCK	2C
MVER	19	SYSUNL	2D
NOP	00	TBBS	55
OPEN	22	TBDL	57
PAUSE	4F	TBFH	50
PDUMP	4F	TBFL	54
POSN	26	TBFX	53
READ	20	TBIN	56
READMG	C7	TBRD	52
REBF	21	TBRL	52
REPLFD	C3	TBWE	51
RESCAL	CD	TBWT	51
RESMXT	CD	TCLOZ	3F
RESUME	CD	TCTL	3F
RETEXT	0C	TINIT	22
RETURN	0C	TLCK	58
RL	A1	TOPEN	22
Rn =	14	TRANS	A8
Rn <=>	13	TREAD	2A

Mnemonic	Op Code	Mnemonic	Op Code
TROFF	4F	WAIT	36
TRON	4F	WFMCRT	3E
TRT	A9	WRBF	3C
TTERM	23	WRT	30
TUNLCK	59	WRTI	31
TWAIT	36	WRTS	35
TWRT	3A	ZONE	1B

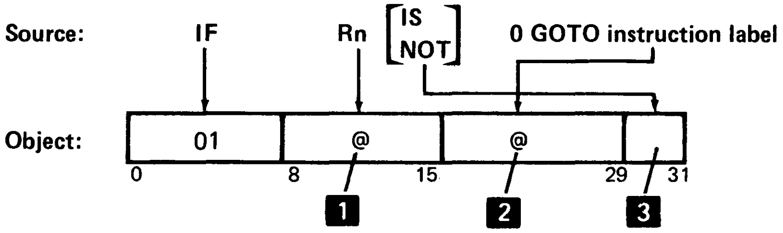
Unconditional Branch (GOTO/NOP)



1 Branch address: Branch to the instruction at this address. For NOP, this is the address of the next instruction.

The microprocessor branches to the branch address.

Test Decimal Register for 0 (Zero) or Blank (IF Rn 0)



1 Test register address: Test the register at this address.

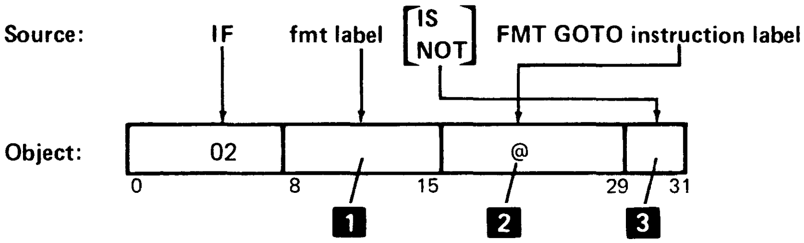
2 Branch address: Branch to the instruction at this address.

3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The test register contains zeros (hex F0s) or blanks (hex 40s) and IS is specified.
- Any byte of the test register contains a value other than blank or zero and NOT is specified.

Test Format Number (IF fmt)



1 Format: The number (hex 01-FE) of the format to use.

2 Branch address: Branch to the instruction at this address.

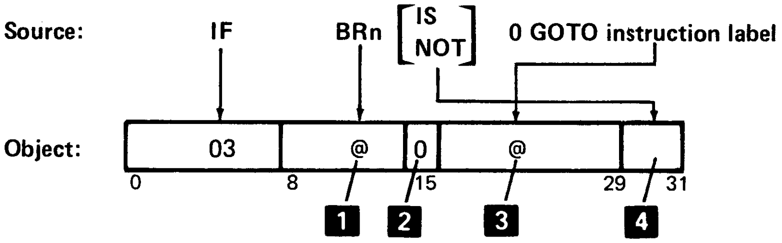
3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The format number is equal to the last format used and IS is specified
- The format number is not equal to the last format used and NOT is specified

The format number of the last format used is in the partition IOB at displacement hex 1D.

Test Binary Register for Zero (IF BRn 0)



1 Test register address: Test the register at this address.

2 Bit 15 is 0.

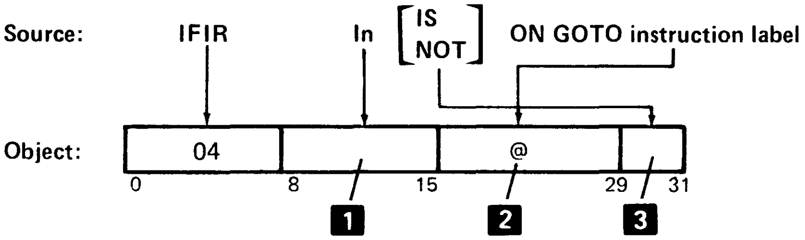
3 Branch address: Branch to the instruction at this address.

4 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The register contains zeros (hex 00s) and IS is specified
- The register contains a value other than zeros and NOT is specified

Test and Reset Indicator (IFIR In)



1 Indicator: The indicator number (hex 00-FE) of the indicator to test. The indicator number is mandatory.

2 Branch address: Branch to the instruction at this address.

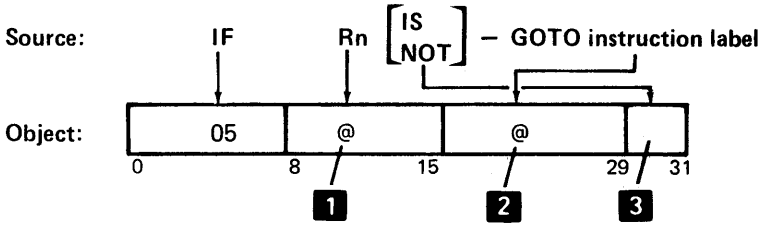
3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The indicator is on and IS is specified
- The indicator is off and NOT is specified

The microprocessor turns off (resets) the indicator whether it branches or not.

Test Decimal Register for Negative (IF Rn-)



1 Test register address: Test the register at this address.

2 Branch address: Branch to the instruction at this address.

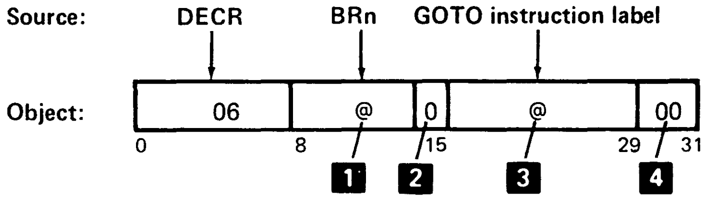
3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The zone portion of the rightmost byte in decimal register is hex D and IS is specified

- The zone portion of the rightmost byte in the register is not hex D and NOT is specified

Decrement Binary Register and Test for Zero (DECR BRn)



1 Test register address: Test the register at this address.

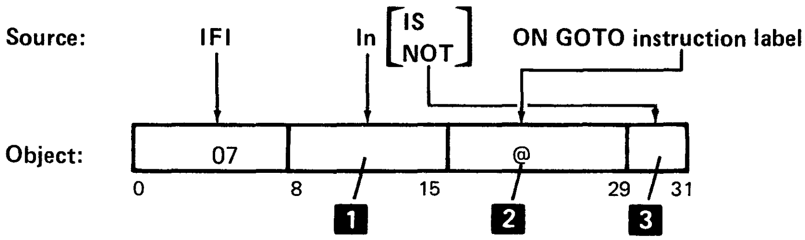
2 Bit 15 is 0.

3 Branch address: Branch to the instruction at this address.

4 Bits 30 and 31 are 00.

Each time this instruction is executed, the contents of the test register decrement by one and are then tested for zero. If the contents are not zero, the microprocessor branches to the branch address.

Test Indicator (IF In)



1 Indicator: The indicator number (hex 00-FE) of the indicator to test.

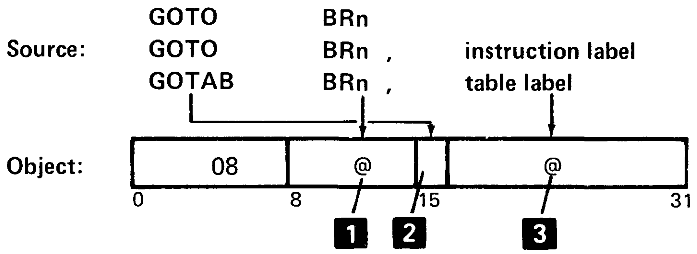
2 Branch address: Branch to the instruction at this address.

3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The indicator is on and IS is specified
- The indicator is off and NOT is specified

Indexed Branch (GOTO BRn/GOTAB BRn)



1 Index register address: The address of the register that contains the index. BR0 cannot be used as an index register.

2 Bit 15:
0 = GOTO
1 = GOTAB

Note: If bit 15 is 1, the microprocessor uses the table address and branches via that table.

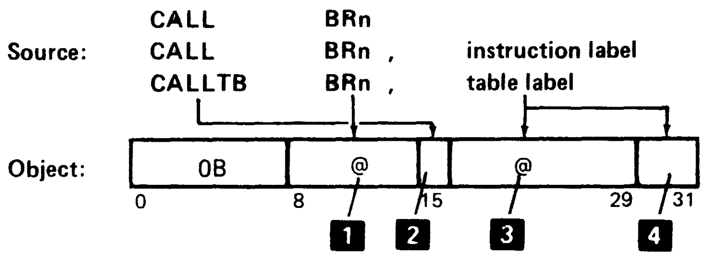
3 Branch address or table address: Branch to the instruction at this address, or use this table to find the branch address.

Note: This address is all zeros if a GOTO is specified with no instruction label operand.

If bit 15 is 0, the microprocessor adds the contents of the index register to the branch address and branches to the resulting address. If no label is specified, an indirect branch is made to the address in the index register **1**.

If bit 15 is 1, the microprocessor branches to the address in the table entry indicated by the index register, using the table indicated by **3**. If the index is 0, the first address in the table is used.

Subroutine Call (CALL/CALLTB)



1 Index register address: The address of the register that contains the index.

2 Bit 15:
0 = CALL
1 = CALLTB

3 Branch address for CALL: Branch to the instruction at this address.

Note: This address is all zeros if no instruction is specified.

Table address for CALLTB: The address of the table.

4 Bits 30 and 31 for CALL:
00 = Current area
01 = Common function area 1
10 = Base area
11 = Common function area 2

Bits 30 and 31 for CALLTB: The last 2 bits of the table address.

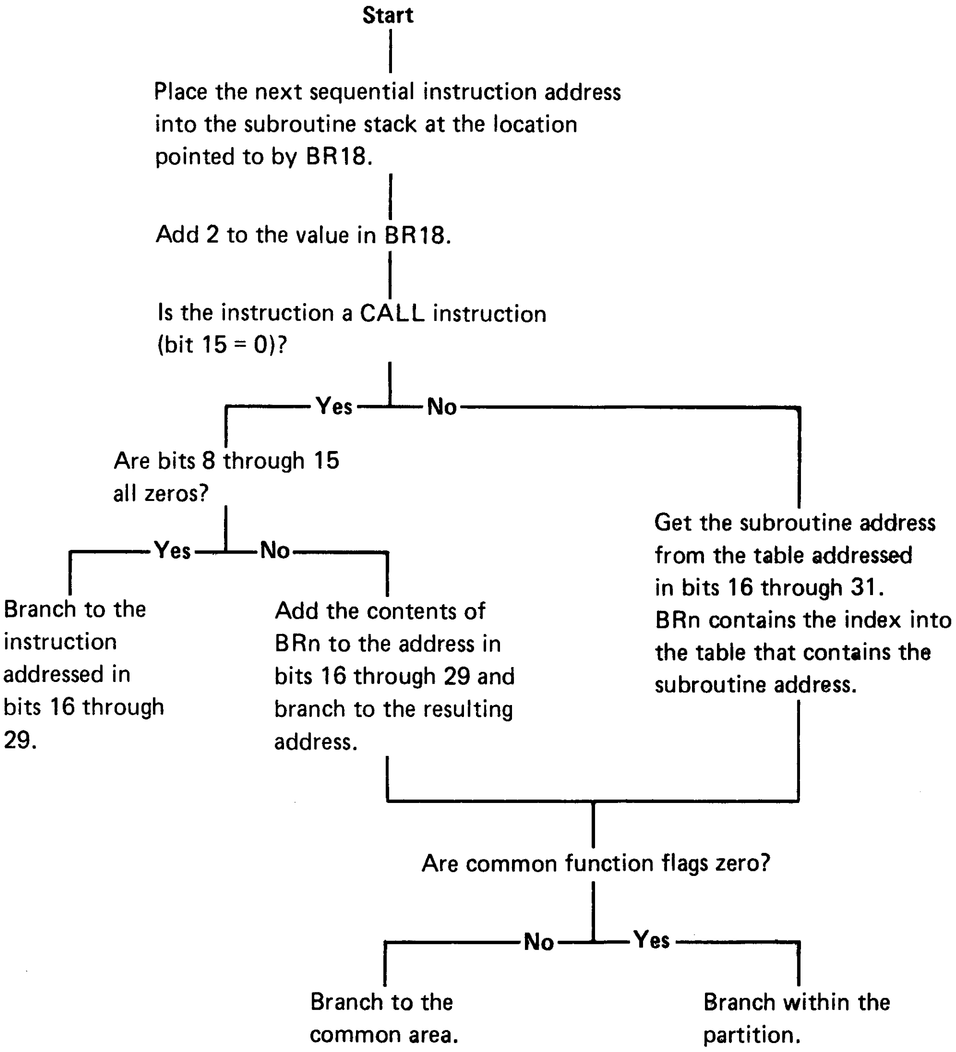
Note: Bits 0-15 of the table entry correspond to bits 16-31 of the CALL instruction. Bits 14 and 15 of the table entry may contain the common function flags described for bits 30 and 31 of the CALL instruction.

If bit 15 is 0, the microprocessor adds the contents of the index register to the branch address and branches to the resulting address. If bit 15 is 1, the microprocessor branches to the address in the table entry indicated by the index register, using the table indicated by **3** . If the index is 0, the first address in the table is used.

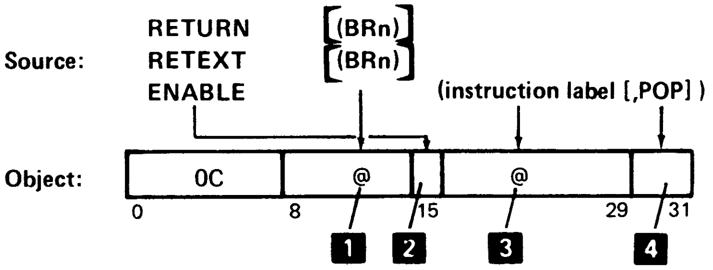
Execution Sequence

The CALL or CALLTB instruction causes the microprocessor to stop executing instructions in the main program and branch to a subroutine.

Following is the main microprocessor execution sequence for the CALL and CALLTB instructions:


Start
Place the next sequential instruction address into the subroutine stack at the location pointed to by BR18.
Add 2 to the value in BR18.
Is the instruction a CALL instruction (bit 15 = 0)?
Yes
No
Are bits 8 through 15 all zeros?
Yes
No
Branch to the instruction addressed in bits 16 through 29.
Add the contents of BRn to the address in bits 16 through 29 and branch to the resulting address.
Get the subroutine address from the table addressed in bits 16 through 31. BRn contains the index into the table that contains the subroutine address.
Are common function flags zero?
No
Yes
Branch to the common area.
Branch within the partition.

Subroutine Return or Enable External Status (RETURN/RETEXT/ENABLE)



1 Index register address for RETURN and RETEXT: The address of the register that contains the index.

Note: This address is all zeros if no index register is specified, or if ENABLE is specified.

2 Bit 15:
0 = RETURN
1 = RETEXT or ENABLE

3 Branch address for ENABLE: Branch to the instruction at this address.

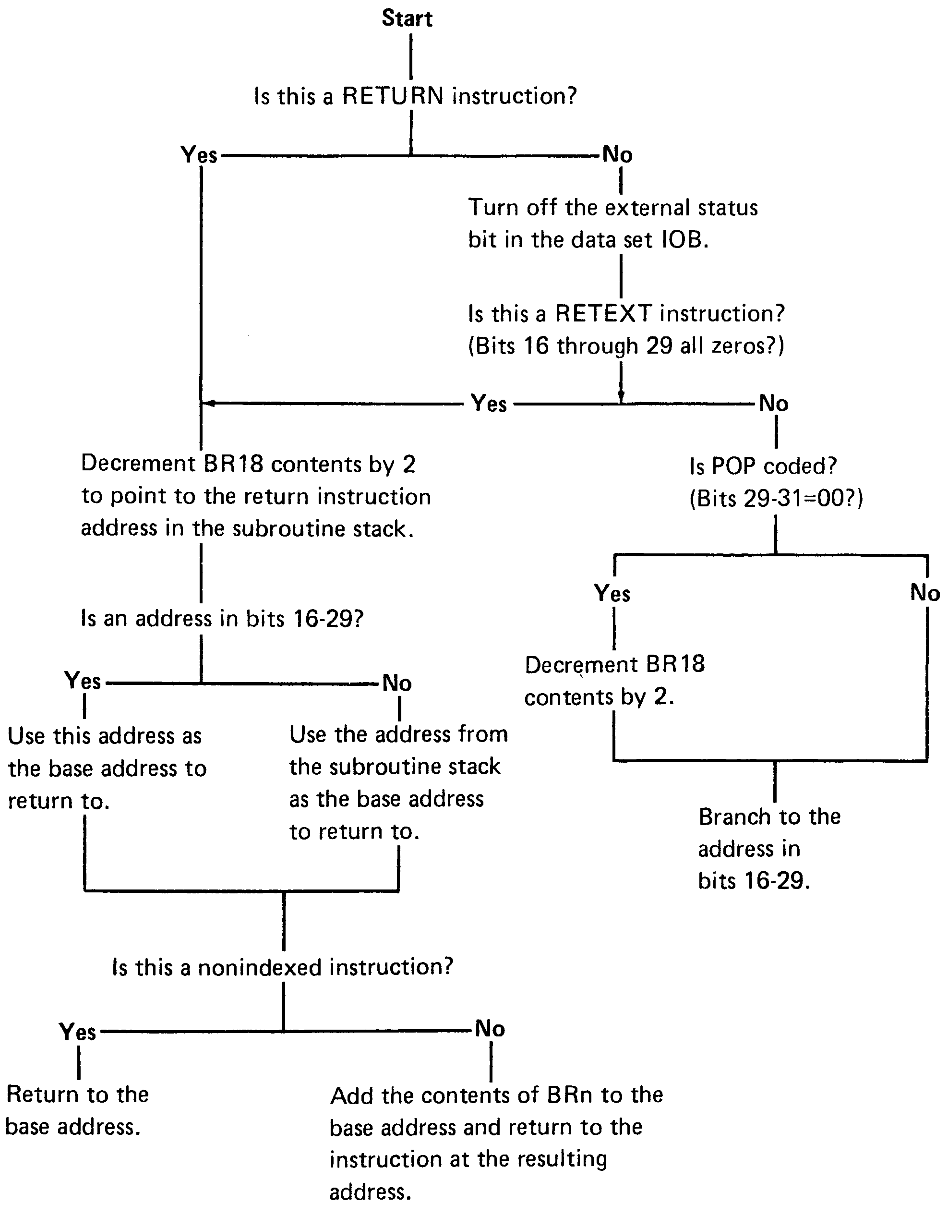
All zeros for RETURN and RETEXT.

4 Bits 30 and 31:
00 = RETURN and RETEXT
00 = ENABLE, if POP is specified
01 = ENABLE, if POP is not specified

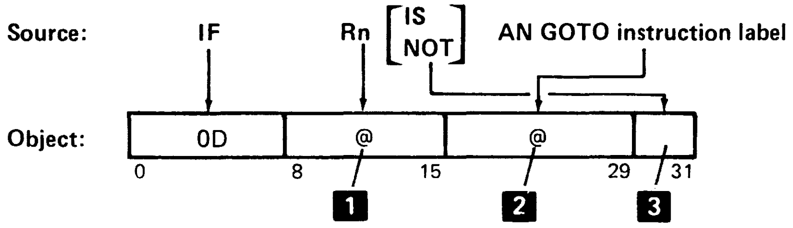
Execution Sequence

The RETURN instruction causes the microprocessor to stop executing the subroutine and return to the main program. RETEXT causes a return to the main program, and the main microprocessor turns off the external status outstanding bit in the status byte of the keyboard/display IOB. ENABLE causes the microprocessor to turn off the external status bit and, if POP is specified, to decrement the subroutine stack pointer, BR18.

Following is the main microprocessor execution sequence for the RETURN, RETEXT, and ENABLE instructions:


Start
Is this a RETURN instruction?
Yes
No
Turn off the external status bit in the data set IOB.
Is this a RETEXT instruction? (Bits 16 through 29 all zeros?)
Yes
No
Decrement BR18 contents by 2 to point to the return instruction address in the subroutine stack.
Is POP coded? (Bits 29-31=00?)
Yes
No
Is an address in bits 16-29?
Yes
No
Decrement BR18 contents by 2.
Use this address as the base address to return to.
Use the address from the subroutine stack as the base address to return to.
Branch to the address in bits 16-29.
Is this a nonindexed instruction?
Yes
No
Return to the base address.
Add the contents of BRn to the base address and return to the instruction at the resulting address.

Test Decimal Register for Absolute Number (IF Rn AN)



1 Test register address: Test the register at this address.

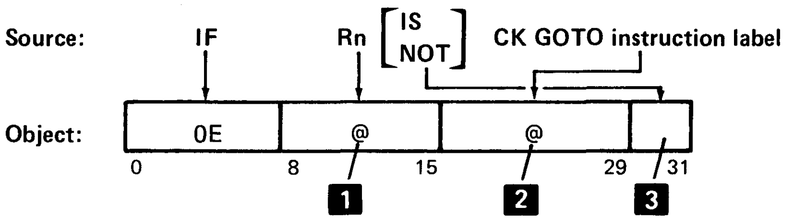
2 Branch address: Branch to the instruction at this address.

3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The test register contains a valid positive number or all blanks (hex 40), and IS is specified
- The test register does not contain a valid positive number, and NOT is specified

Test Decimal Register for Self-Check Digit (IF Rn CK)



1 Test register address: Test the register at this address.

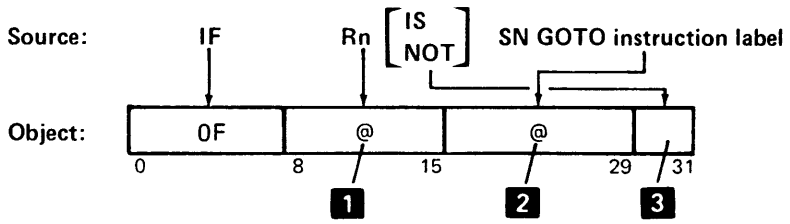
2 Branch address: Branch to the instruction at this address.

3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The self-check number in the test register is correct when it is checked by the self-check algorithm, and IS is specified
- The self-check number in the test register is not correct and NOT is specified

Test Decimal Register for Signed Number (IF Rn SN)



1 Test register address: Test the register at this address.

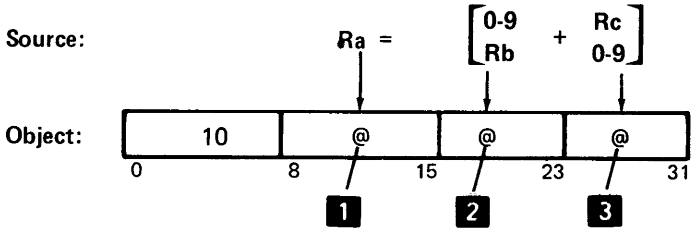
2 Branch address: Branch to the instruction at this address.

3 Bits:
00 = IS
01 = NOT

The microprocessor branches to the branch address if:

- The test register contains a valid signed numeric value and IS is specified
- The test register does not contain a valid signed numeric value and NOT is specified

Decimal Register Add (+)



1 Result decimal register address: The address of the decimal register that will contain the result of this operation.

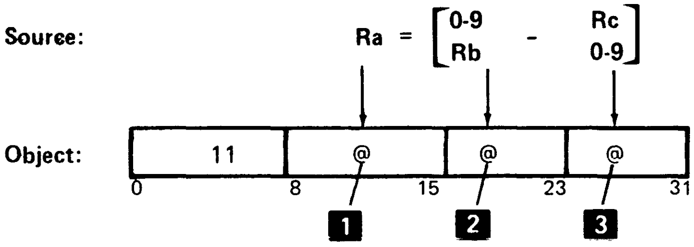
Note: If a carry results out of the high-order position in this register, the overflow indicator I124 is set on.

2 Factor 1 decimal register address: The address of the decimal register that contains factor 1, or a single-digit constant (hex 0-9) followed by hex 0.

3 Factor 2 decimal register address: The address of the decimal register that contains factor 2, or a single-digit constant (hex 0-9) followed by hex 0.

This instruction algebraically adds the factor 1 value to the factor 2 value and stores the sum in the result register.

Decimal Register Subtract (-)



1 Result decimal register address: The address of the decimal register that will contain the result of this operation.

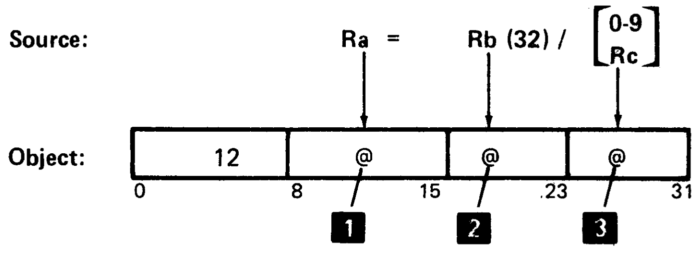
Note: If a carry results out of the high-order position in this register, the overflow indicator I124 is set on.

2 Factor 1 decimal register address: The address of the decimal register that contains factor 1, or a single-digit constant (hex 0-9) followed by hex 0.

3 Factor 2 decimal register address: The address of the decimal register that contains factor 2, or a single-digit constant (hex 0-9) followed by hex 0.

This instruction algebraically subtracts the factor 2 value from the factor 1 value and stores the result in the result register.

Decimal Double-Register Divide (/)



1 Result decimal register address: The address of the decimal register that will contain the result of this operation.

2 Factor 1 decimal register address: The address of the double decimal register that contains factor 1.

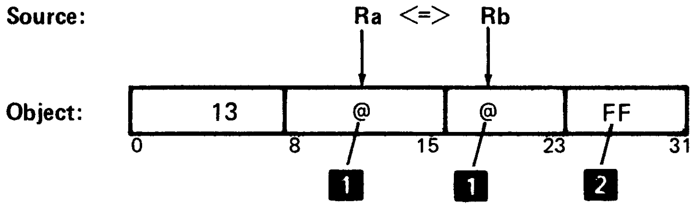
Note: Factor 1 is replaced with the remainder.

3 Factor 2 decimal register address: The address of the decimal register that contains factor 2, or a single-digit constant (hex 0-9) followed by hex 0.

This instruction divides the factor 1 value by the factor 2 value. The result is stored in the result register; the remainder is stored in the factor 1 register.

Note: If division by zero is attempted, the overflow indicator I124 and the divide error indicator I120 are set on.

Decimal Register Exchange (<=>)

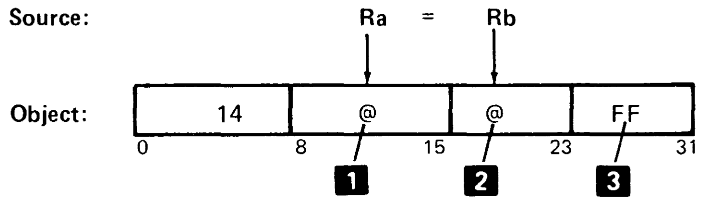


1 Decimal register addresses: The addresses of the decimal registers that exchange contents.

2 Set to hex FF.

This instruction swaps the contents of the specified decimal registers.

Decimal Register Copy (=)



1 Result decimal register address: The address of the copied-to decimal register.

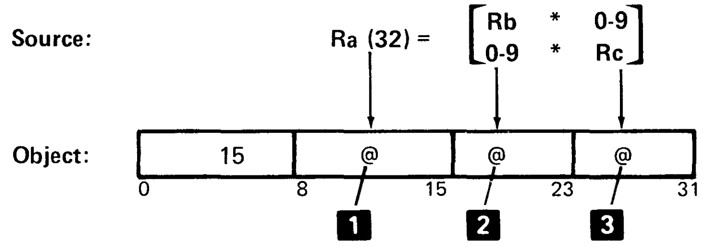
2 Factor 1 decimal register address: The address of the decimal register that contains data to copy or a constant 0-9.

Note: If a constant is used, it is placed in bits 16 through 19, and bits 20 through 23 are filled with zeros.

3 Set to hex FF.

The constant is copied into the result decimal register. The contsnt is placed into byte 15 of the decimal register, and bytes 0 through 14 are filled with blanks (hex 40s).

Decimal Double-Register Multiply (*)



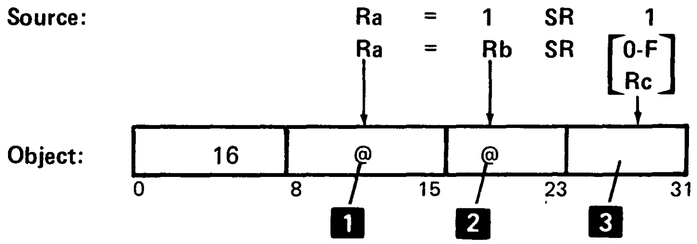
1 Result decimal register address: The address of the double decimal register that will contain the result of this operation.

2 Factor 1 decimal register address: The address of the decimal register that contains factor 1, or a single-digit constant (hex 0-9) followed by hex 0.

3 Factor 2 decimal register address: The address of the decimal register that contains factor 2, or a single-digit constant (hex 0-9) followed by hex 0.

This instruction multiplies the factor 1 value by the factor 2 value and stores the product in the result double decimal register.

Decimal Register Shift Right, Blank Pad (SR)



1 Result decimal register address: The address of the decimal register that will contain the shifted data upon completion of this operation.

2 Shift decimal register address: The address of the decimal register that contains the data to shift, or a constant 1 if you want to blank the result register.

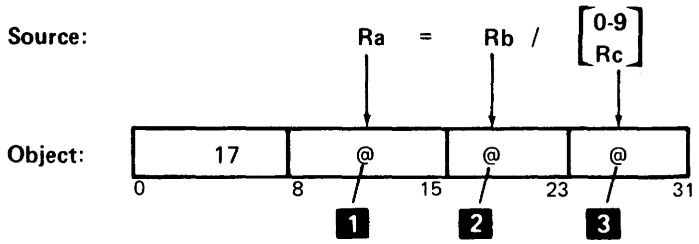
Note: If a constant 1 is used, 3 must also be 1. This is the quickest way to blank a decimal register.

3 Shift count: The number of bytes to shift the data. The shift count can be specified as a constant (hex 0-F) followed by hex 0, or as a decimal register that contains the shift count in the digits portion of the low-order byte of the register.

The bytes of the shift register are shifted right the number of bytes indicated by the shift count, and the shifted result is placed into the result register. The high-order bytes of the shifted result contain blanks (hex 40) for the number of positions shifted. If a negative number is shifted right, the D-zone is shifted out of the register and the register contents are no longer negative.

If a constant 1 is specified for the shift register, the bytes are shifted as though 2 were a decimal register with decimal 1 in the rightmost byte, and bytes 0-14 were blanks. The rightmost byte is shifted out of the register so the register contains only blanks. These blanks replace the contents of the result register.

Decimal Register Divide (/)



1 Result decimal register address: The address of the decimal register that will contain the result of this operation.

2 Factor 1 decimal register address: The address of the factor 1 decimal register.

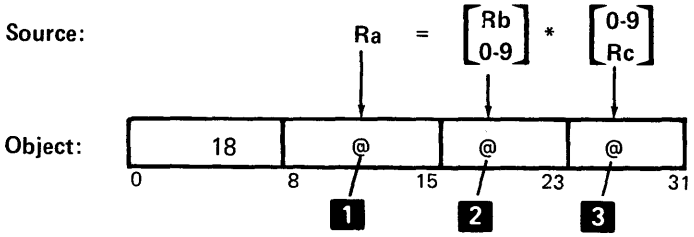
Note: Factor 1 is replaced by the remainder.

3 Factor 2 decimal register address: The address of the decimal register that contains factor 2, or a single-digit constant (hex 0-9) followed by hex 0.

This instruction divides the factor 1 value by the factor 2 value and stores the result in the result register.

Note: If division by zero is attempted, the overflow indicator 1124 and the divide error indicator 1120 are set on.

Decimal Register Multiply (*)



1 Result decimal register address: The address of the decimal register that will contain the result of this operation.

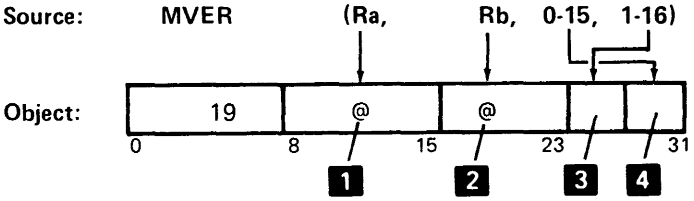
Note: If a carry occurs out of the high-order position in this register, the overflow indicator I124 and the multiply overflow indicator I123 are set on.

2 Factor 1 decimal register address: The address of the decimal register that contains factor 1, or a single-digit constant (hex 0-9) followed by hex 0.

3 Factor 2 decimal register address: The address of the decimal register that contains factor 2, or a single-digit constant (hex 0-9) followed by hex 0.

This instruction multiplies the factor 1 value by the factor 2 value and stores the product in the result register.

Decimal Registers, Move Partial Contents (MVER)



1 Result decimal register address: The address of the decimal register that will contain the moved data upon completion of this operation.

2 From decimal register address: The address of the decimal register from which data is moved. Data is moved left-to-right, starting with the byte that is specified in the MVER instruction. The contents of the from register remains unchanged.

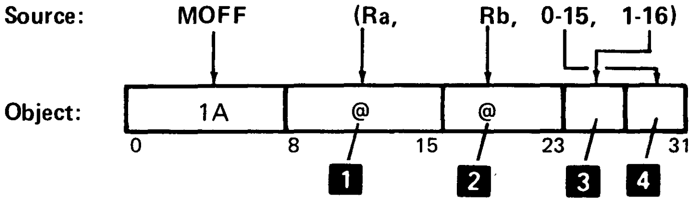
3 Byte count: The number minus 1 (hex 0-F) of bytes to be moved (that is, the length operand minus 1).

Note: If this number of bytes plus the displacement **4** is greater than 16, some of the data is moved into the register that follows the result register.

4 Displacement: The offset (hex 0-F) into both registers of the leftmost byte of data to move.

This instruction moves all or part of the contents of the from register into the result register. The movement is from the specified offset in the from register to the same offset in the result register.

Decimal Registers, Move Partial Contents with Offset (MOFF)



1 Result decimal register address: The address of the decimal register that will contain the moved data.

2 From decimal register address: The address of the decimal register from which data is moved. The contents of the from register remain unchanged.

3 Byte count: The number minus 1 (hex 0-F) of bytes to move. (That is, the length operand, minus 1.)

Note: If this number of bytes, plus the displacement 4 is greater than 16, some of the data is moved into the register that follows the result register.

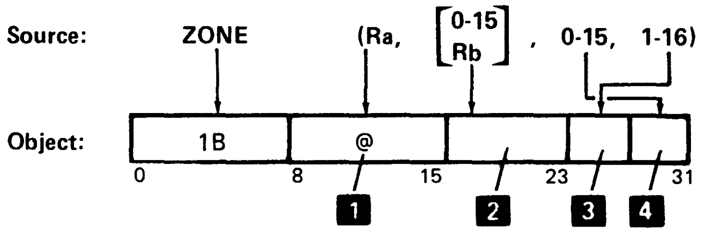
4 Displacement: The offset (hex 0-F) into the result register of the leftmost byte of moved data.

The rightmost number of bytes specified by 3 are moved from the 2 register to the result register. The data is moved from left to right and placed in the result register at the byte specified by offset.

The offset applies only to the result register (Ra), so the move is not limited to corresponding byte positions.

Note: If the sum of offset and length is greater than 16, bytes are moved into the register following the result register.

Decimal Register Zone Modification (ZONE)



1 Result decimal register address: The address of the decimal register that contains bytes to modify. The contents of this register are modified with either the zone modifying digit **2** or the zone portion of the rightmost character in the specified register.

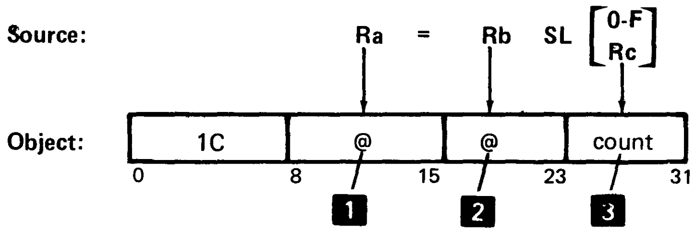
2 Zone modifying digit: The digit (hex 0-F) followed by hex 0, or the address of the decimal register that contains the modifying digit.

3 Length: The number minus 1 (hex 0-F), of bytes to modify.

4 Displacement: The offset (hex 0-F) into the result register of the leftmost byte to modify.

The bytes of the decimal result register (Ra) are modified, starting at the byte specified by offset and continuing to the right for the number of bytes specified by length. The hex character specified by the operand replaces the original zone of each byte specified. If the offset plus length exceeds 16 bytes, the bytes of the next register are also modified.

Decimal Register Shift Left, Blank Fill (SL)



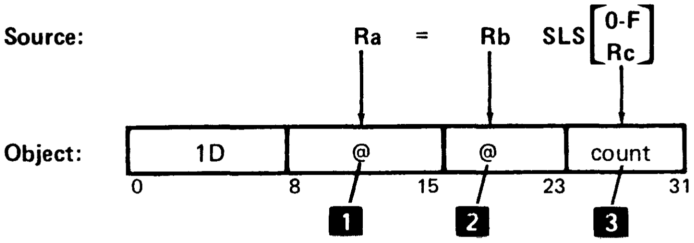
1 Result decimal register address: The address of the decimal register that will contain the shifted data upon completion of this operation. The low-order bytes of the result register are filled with blanks (hex 40). Data that is shifted out of the high end of the register is lost.

2 Shift decimal register address: The address of the decimal register that contains the data to shift. The contents of this register remain unchanged.

3 Shift count: The number of bytes to shift the data. The shift count can be specified as a constant (hex 0-F) followed by hex 0, or as a decimal register that contains the shift count in the digits portion of the low-order byte of the register.

The bytes of the shift register are shifted left the number of bytes indicated by the shift count, and the shifted result is placed into the result register. The low-order positions of the shifted result contain blanks (hex 40s) for the number of positions shifted. If a negative number is shifted left, the D-zone is shifted out of the units position, and the register contents are no longer negative.

Decimal Register Shift Left Signed (SLS)



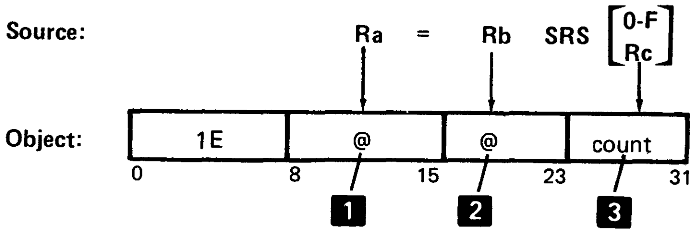
1 Result decimal register address: The address of the decimal register that will contain the shifted data upon completion of this operation. The low-order bytes of the result register are filled with zeros (hex F0). Data that is shifted out of the high end of the register is lost.

2 Shift decimal register address: The address of the decimal register that contains the data to shift. The contents of this register remain unchanged.

3 Shift count: The number of bytes to shift the data. The shift count can be specified as a constant (hex 0-F) followed by hex 0, or as a decimal register that contains the shift count in the digits portion of the low-order byte of the register.

The bytes of the shift register are shifted left the number of bytes indicated by the shift count, and the shifted result is placed into the result register. The low-order bytes of the shifted result contain zeros (hex F0s) for the number of positions shifted. If a negative number is shifted left, the units position of the result register retains the D-zone.

Decimal Register Shift Right Signed (SRS)



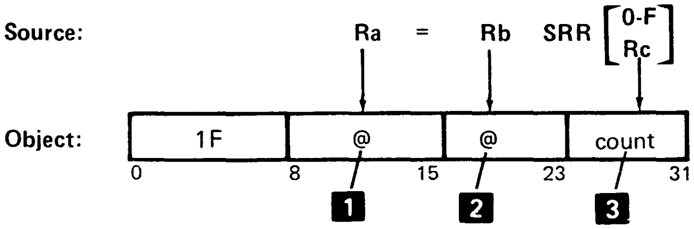
1 Result decimal register address: The address of the decimal register that will contain the shifted data upon completion of this operation. The high-order bytes of the result register are filled with zeros (hex F0). Data that is shifted out of the low end of the register is lost.

2 Shift decimal register address: The address of the decimal register that contains the data to shift. The contents of this register remain unchanged.

3 Shift count: The number of bytes to shift the data. The shift count can be specified as a constant (hex 0-F) followed by hex 0, or as a decimal register that contains the shift count in the digits portion of the low-order byte of the register.

The bytes of the shift register are shifted right the number of bytes indicated by the shift count, and the shifted result is placed into the result register. The high-order bytes of the result register contain zeros (hex F0s) for the number of positions shifted. Any blanks present are shifted without change. If the unshifted contents of the shift register contained a negative value, the result register contains a hex D in the zone portion of the rightmost byte. All other zones remain unchanged.

Decimal Register Shift Right and Round (SRR)



1 Result decimal register address: The address of the decimal register that will contain the shifted data upon completion of this operation. The high-order bytes of the result register are filled with zeros (hex F0). Data that is shifted out of the low end of the register is lost.

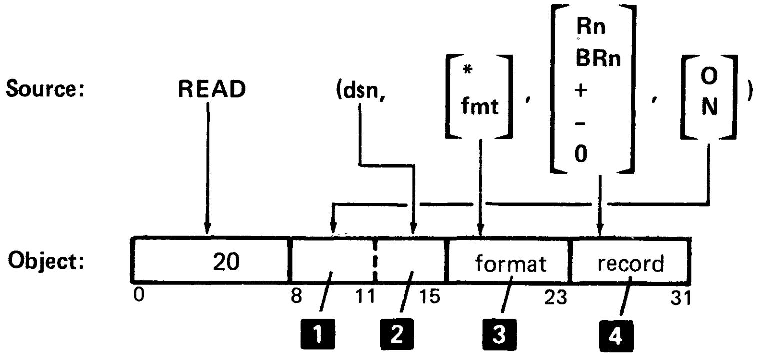
To round the result, a 5 is used with the same sign as the sign that is in the shift register; the 5 is added to the last byte of data that is shifted out of the result register.

2 Shift decimal register address: The address of the decimal register that contains the data to shift. The contents of this register remain unchanged.

3 Shift count: The number of bytes to shift the data. The shift count can be specified as a constant (hex 0-F) followed by hex 0 or as a decimal register that contains the shift count in the digits portion of the low-order byte of the register.

The bytes of the shift register are shifted right the number of bytes indicated by the shift count, and the shifted result is placed into the result register. The high-order bytes of the shifted result contain zeros (hex F0s) for the number of bytes shifted, and the units position of the shifted result retains the zone of the original contents of the shift register. The result is rounded by adding 5 of like sign to the last byte shifted out of the right end of the result register.

Read a Record from a Data Set (READ)



1 Bits:

8 0 = Sequential record access method
1 = Relative record or key access methods
9 0 = Overlap mode (O specified)
1 = Nonoverlap mode (N specified)
10 Not used, always zero
11 Not used, always zero

2 Data set number: The number (hex 1-F) of the data set to be read.

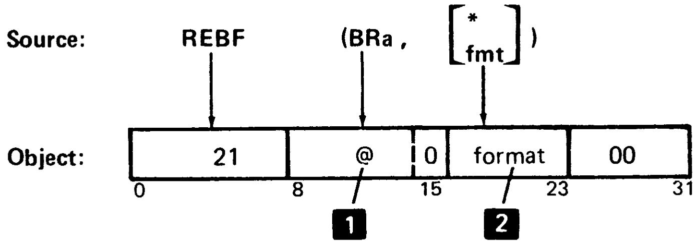
3 Format number: The number (hex 01-FE) of the format to use. If no format number is specified, this is hex FF. For a data directed read, this is hex 00.

4 Record to read: The current record number in the IOB is set to the record number that is to be read. The location of the record to be read can be:

- Rn for the address of the decimal register that contains the key to the record.
- BRn for the address of the binary register that contains the relative record number of the record.
- Hex 07 (– specified) for reading the previous record.
- Hex 08 (0 specified) for reading the current record.
- Hex 09 (+ specified) for reading the next record.

The current record number in the IOB is set to the record number that is to be read. The specified record is read from diskette and put into the logical I/O buffer. If formatting (fmt) is specified, data is formatted, moved from the logical I/O buffer, and put into the storage indicated by the format.

Formatted Read to Storage (REBF)

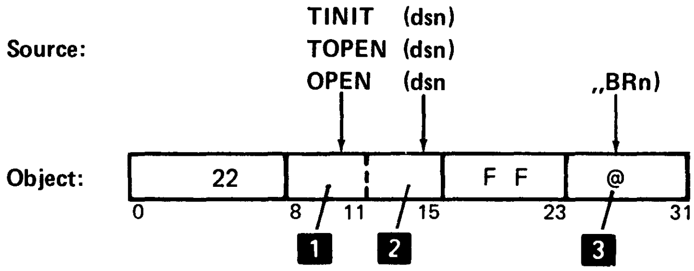


1 Read address: The address of the binary register that contains the address of the leftmost byte of data to read into the storage.

2 Format number: The number (hex 01-FE) of the format to use. For data directed formatting, the * is specified and bits 16 through 23 contain hex 00.

Data is moved into the register specified by the DCLBL parameters of the format. The number of bytes moved is determined by the LEN parameter, with editing controlled by the EDIT parameter of the format.

Open a Data Set or Initialize Communications (OPEN/TOPEN/TINIT)



1 Bits:

8	0 = TINIT, or OPEN for a diskette or printer. 1 = TOPEN
9-11	Always 100

2 Data set number: The number (hex 1-F) of the data set to be accessed.

3 Binary register address: The ID option for OPEN where the binary register contains the storage address of the owner ID information. This optional owner ID is stored on the volume label and is compared with the owner ID in storage. If the diskette is secure, the ID information in storage must match the owner ID on the volume label in order for the OPEN operation to execute. If the diskette is not a secure diskette, the binary register is ignored. The owner ID information may be up to 14 characters long; if less than 14 characters are used, the owner ID must be followed by a blank (hex 40). If the binary register is omitted, or if this is a TINIT or TOPEN instruction, this byte contains hex 00.

Note: Two commas must precede the binary register if it is included. If the register is omitted, the commas are also omitted.

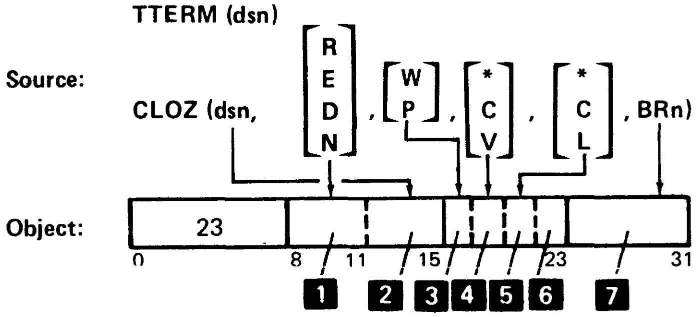
TINIT establishes the communications link and begins the line connection for communications.

TOPEN sets the open flag in the communications IOB to indicate that the IOB is open.

OPEN sets the open flag in the diskette or printer data set IOB to indicate that the data set is open. It adds the address of the data set IOB to the IOB chain, and validates the .DATASET parameters in the IOB.

When the open has completed, the data set's HDR1 label will be located in the first 128 bytes of the physical buffer except for pointer I/O and SCS data sets that have the SW or ERS parameter specified in the .DATASET control statement. For a label update data set, the VOL1 label will be saved instead of the HDR1 label. The op code byte in the data set IOB is replaced with hex 00. If there is an external status for insufficient physical buffer size (3430), or two physical buffers specified with unequal sizes (3435), or if any group 7 warning message is presented, the minimum number of 128-byte blocks required for sufficient buffer size is placed into hex 78 of the data set IOB. If any other external status occurs, this number is not placed into the IOB.

Close a Data Set or Terminate Communications (CLOZ/TTERM)



1 Close option for CLOZ

Bits: For a printer, bit 8 = 0 and bits 9-32 are ignored.
0100 = No label update, N specified; HDR1 label is not updated.
0101 = Normal close, no option specified.
0110 = Close and erase, E specified; EOD is set to BOE.
1100 = Close and release, R specified; EOE is set to EOD-1.
1110 = Close and delete, D specified; label is marked deleted.
0100 = TTERM

2 Data set number: The number (hex 1-F) of the data set to be accessed.

3 Write protect option for CLOZ. This affects the write-protect position on the HDR1 label.

Bits 16 and 17:
00 = Leave write protect as is, no option specified
01 = Clear write protect, W specified
10 = Set write protect, P specified
00 = TTERM

4 Verify and copy option for CLOZ. This affects the verify/copy position on the HDR1 label.

Bits 18 and 19:
00 = Leave verify and copy as is, no option specified
01 = Clear verify and copy, * specified
10 = Set verify, V specified
11 = Set copy, C specified
00 = TTERM

5 Multivolume option for CLOZ. This affects the multivolume positions on the HDR1 label.

Bits 20 and 21:
00 = Leave multivolume as is, no option specified
01 = Clear multivolume, * specified
10 = Set continued volume, C specified
11 = Set last volume, L specified
00 = TTERM

6 Bits 22 and 23: Not used, always 00.

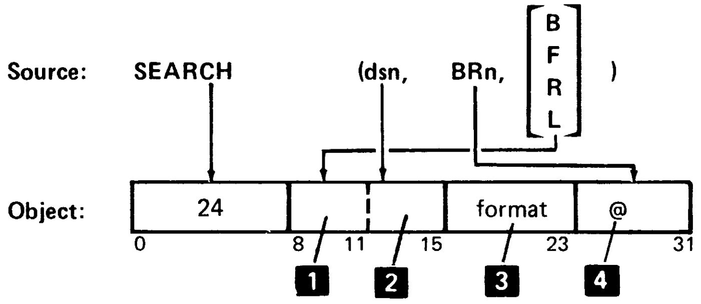
7 Multivolume number for CLOZ: The address of the binary register that contains the volume number, or hex 00 if the multivolume option is not specified. Hex 00 is also for TTERM.

The TTERM instruction terminates the logical connection between the application program and the communications access method.

The CLOZ instruction removes the data set IOB from the IOB chain and resets the open-flag in the IOB. If any records have been added, the EOD is updated as appropriate. Any functions specified in the operand fields of the CLOZ instruction are performed. When the CLOZ is completed, the op code in the IOB is reset to 0.

For an erase type data set, the block length, record length, and EOD are updated on the HDR1 label to the values in the IOB.

Search a Data Set (SEARCH)



1 Type of search

Bits 8 through 11:
0100 = Binary search, B specified
0101 = Forward search, F specified
0110 = Reverse search, R specified
1110 = Logical record search, L specified

2 Data set number: The number (hex 1-F) of the data set that the diskette microprocessor is to search for a specified record. When the record is found, it is placed in the I/O buffer.

3 The assembler sets this byte to hex FF; however, you can change it to a format number, or to hex 00 for data directed formatting.

4 Parameters' address: The address of a binary register that contains the address of the search parameters. The search parameters must be prepared and stored in main storage before the SEARCH instruction is issued. The format of the search parameters is described following the purpose statement.

The search operation searches a data set for a record that agrees with the mask specifications. If a match is found, the matching record is placed into the logical record buffer and the search ends. If no match is found, the contents of the logical buffer depend on the type of search performed.

A binary search operation searches for the relative record position within the data set of a logical record that matches the mask. If a match is not found, the record in the relative record position following the position where the record would have been located is placed into the logical buffer and an external status (3702) is reported. If the record would have been beyond EOD, an external status (3703) is reported, and the last record is placed into the logical buffer.

A sequential search operation searches a data set for a record that matches one or more mask specifications. Multiple mask specifications include the relational operators. AND and OR, with AND having priority over OR. If no match is found, the last logical record (for a forward search) or the first logical record (for a reverse search) is placed into the logical record buffer and an external status (3702) is reported.

The format of the search parameters is as follows:

For a Binary Search

Byte	**Contents**
0-1	Length of the mask
2-3	Field position in which to begin search
4-n	Mask

Only one mask specification may be used.

Example: The following mask specification uses a binary search to search a data set for a record containing 137 in position 15.

$$\text{X'}\overbrace{\text{0003}}^{\textbf{Length}}\underbrace{\text{000F}}_{\textbf{Position}}\overbrace{\text{F1F3F7}}^{\textbf{Mask}}\text{'}$$

For a Forward Search, Reverse Search, or Logical Record Search

Byte	**Contents**
0-1	Length of the mask
2	Relative and logical operators. The 5280 does not check bits 0 and 1 when it processes the first mask specification. However, every following mask specification must have either bit 0 or bit 1 (but not both) turned on. Each mask specification can have one, and only one, of bits 2-7 turned on. If more than one is on, an external status (3417) is presented.

Bit	*Meaning if 1*
0	Logical AND
1	Logical OR
2	LT (less than)
3	GT (greater than)
4	LE (less than or equal)
5	GE (greater than or equal)
6	EQ (equal)
7	NE (not equal)

Byte	Contents
3-4	Field position in which to begin search.
5-6	Field position in which to end search.
7-n	Mask.

The mask specification can be repeated from byte 0. Follow the mask in the last specification with X'0000' to indicate the end.

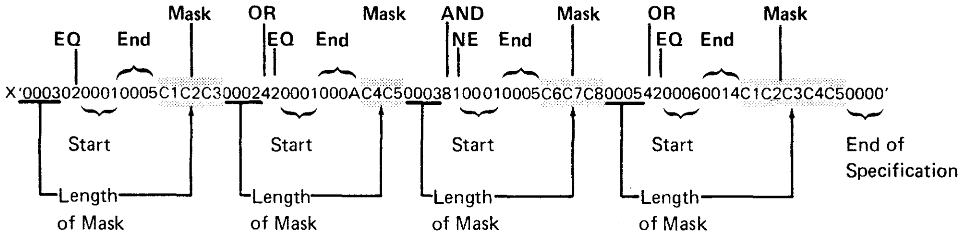
Example of a Forward Search:

The following mask specifications search a data set for a record that satisfies *one* of the following three conditions:

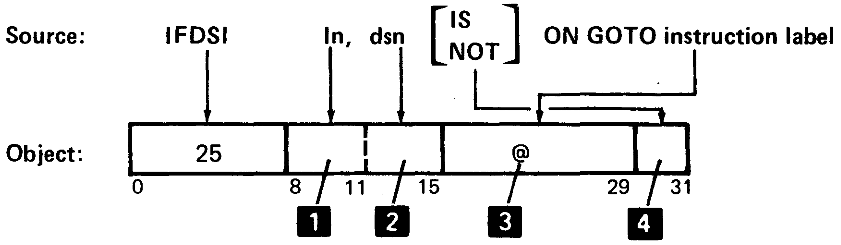
1. Contains 'ABC' in positions 1-5.

2. Contains 'DE' in positions 1-10 AND 'FGH' is not in positions 1-5.

3. Contains 'ABCDE' in positions 6-20.



Test Data Set Status Indicators (IFDSI)



1 Data set status indicator number: The number of the status indicator (hex 0-F) to be tested. Indicator numbers 0-7 test bits 0-7 of IOB byte 0. Indicator numbers 8-F test bits 0-7 of IOB byte 13.

2 Data set number: The number (hex 1-F) of the data set to be accessed.

3 Branch address: The address of the instruction the microprocessor branches to if the diskette data set status conditions are met.

4 Bits 30 and 31:
00 = IS—branch to the instruction 3 if the specified indicator 1 is on.
01 = NOT—branch to the instruction 3 if the specified indicator 1 is not on.

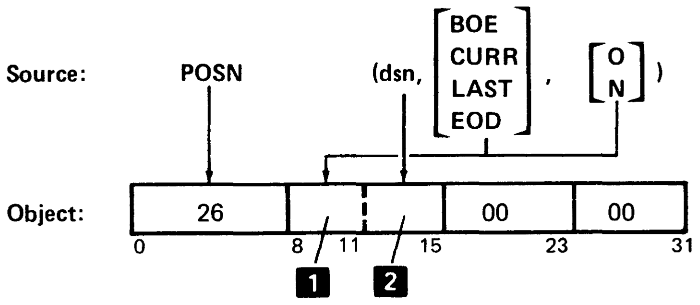
The microprocessor branches to the branch address if:

- The specified indicators are on and IS is specified.

- The specified indicators are off and NOT is specified.

- This instruction does not implicitly check for external status.

Position Diskette (POSN)



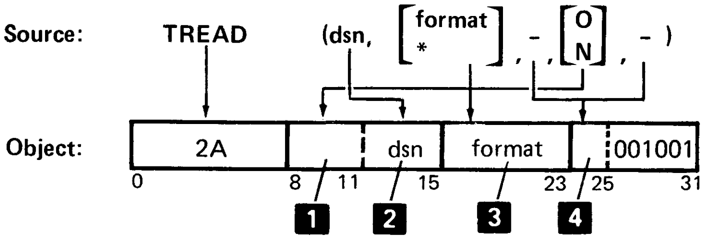
1 Record pointer: Position the data set pointer by the diskette microprocessor to the record specified or read a specified record into the buffer.

Bit Settings	Meaning
0x00	Set the current record counter to zero, BOE specified.
0x01	Read a new copy of the current record from diskette, CURR specified.
0x10	Set the current record counter to the number of the last logical record in the data set, and read it from the diskette, LAST specified.
1x00	Set the current record counter to the record number following the last record, EOD specified.
x = 0	Overlapped (O specified).
x = 1	Nonoverlapped (N specified).

2 Data set number: The number (hex 1-F) of the data set to be accessed.

This operation modifies the contents of the current record counter. If CURR or LAST is specified, the logical record indicated is read into the physical buffer.

Read from Communications (TREAD)



1 Bits:

8 Not used, always zero
9 0 = Overlap mode (O specified)
1 = Nonoverlap mode (N specified)
10 Not used, always zero
11 Not used, always zero

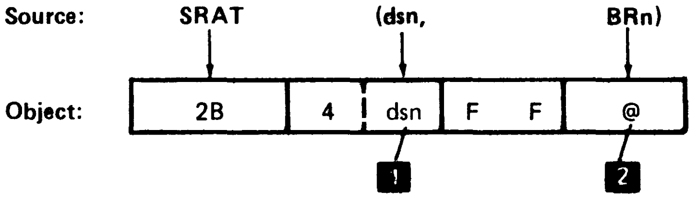
2 Data set number: The number (hex 1-F) of the data set to access. This number is assigned by the DSN parameter of the .COMM control statement.

3 Format number: The number (hex 01-FE) of the format to use. If an asterisk (*) is coded, this will be hex 00 for data directed formatting. If no format entry is coded, this will be hex FF.

4 Bits:

24 0 = Read the next logical record.
1 = Read the entire block (minus sign coded).
25 0 = Read data.
1 = Return control immediately with status if data is not available for the read.

Search Resource Allocation Table (SRAT)



1 Data set number: The number (hex 1-F) of the data set IOB to access.

2 Binary register address: The binary register will be loaded with the physical device address of the data set, which the 5280 finds in the resource allocation table.

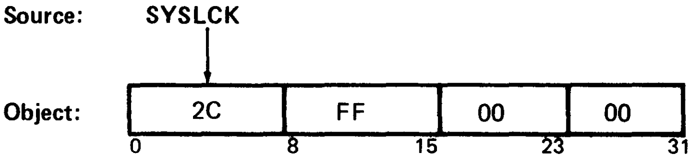
This instruction searches the resource allocation table within the partition to find the physical address of the logical device ID. The logical ID is stored in the data set IOB. If the physical address is found, it is stored in the specified register.

I118 is set on if one of the following is true:

- No logical device identifier is present in the set IOB.
- No match is found in the resource allocation table.
- No resource allocation table is available.

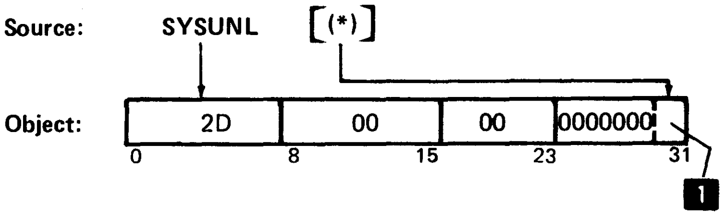
When I118 is set on, the physical address that is currently stored in the IOB and logical I/O table is placed in the specified binary register.

System Lock (SYSLCK)



This instruction sets a bit in the partition IOB. This flag will signal the main microprocessor to ignore all hardware attentions such as time-out attention and keyboard attention. The main microprocessor will not exit the partition to execute instructions in another partition until the flag is turned off via a SYSUNL instruction.

System Unlock (SYSUNL)

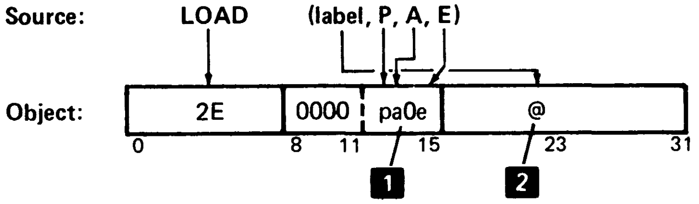


1 Partition exit option:

0 = Exit partition immediately; * not specified.
1 = Execute instructions for the normal time limit, then exit partition; * specified.

This instruction turns off the system lock bit to allow the main microprocessor to resume normal operation. It may also be used to relinquish the remaining time in a time slice.

Load a Partition (LOAD)



1 Load parameters:

Bits:

12	0 = Load a full partition; P is not specified. 1 = Load a partial overlay; P is specified.
13	0 = Do not attempt a background attach; A is not specified. 1 = Attempt a background attach; A is specified.
14	Not used, always 0.
15	0 = System is providing error handling; E is not specified. 1 = Program is providing error handling; E is specified.

2 Load parameters address: The address of data area that contains the load parameters, or all zeros if the data area label is not specified. If the label is omitted the operator will be prompted to enter the load parameters from the keyboard.

This instruction loads a partition according to the load parameters. The load parameters may be entered from the keyboard or may be read from a data area. If the parameters are to be read from a data area, they must be stored in the following format:

1. Partition number; 2 bytes in length. The partition number may contain: (a) the number (hex 0-7) in the first byte and blank (hex 40) in the second byte, (b) the 2-byte logical ID assigned to the partition in the resource allocation table, or (c) two blanks (hex 40) if the current partition is to be reloaded.

2. Device address; 4 bytes in length. The device address may contain: (a) the 4-byte physical address of the device that contains the data set to load, or (b) the 2-byte logical device ID assigned to the device in the resource allocation table, followed by two blanks (hex 40).

3. Start address; 2 bytes of hex digits, used only for a partial overlay. The address must be on a 256-byte boundary and must be greater than hex 100.

4. Data set name; up to 32 bytes in length. The data set name may include a volume ID if volume checking is desired. The volume ID may be up to 6 alphameric characters long, preceded by an asterisk and followed by a period. The name of the data set follows the period if the volume ID is included. The name may be up to 8 alphameric characters long for an H, I, or basic exchange data set. For an E exchange data set the name may be up to 17 bytes long, consisting of one or more simple names of up to 8 alphameric characters each, and with each simple name separated by a period. No blanks are allowed within a data set name, but the data set name must end with a blank.

If a partial overlay is loaded, the load parameters must include the relative address where the overlay begins. The original contents of the partition remain unchanged except in the area of the overlay. The first 8 bytes of a partial overlay contain information added by the assembler. The first 2 bytes contain the length of the overlay, the next 2 bytes contain the last 2 bytes of the overlay name, and the remaining 4 bytes are reserved for a patch log. The last 2 bytes of the program name are replaced with the second 2 bytes of the overlay.

If an error occurs during a load, error recovery can be handled by the system or by the application program.

If an error occurs and the application program is handling error recovery, the main microprocessor places an error code into a system binary register (BR16) and returns control to the first instruction following the load instruction. If the load operation is successful, the main microprocessor returns control to the second instruction following the load instruction.

If an error occurs and the system is handling error recovery, the system sends a message to the screen and waits for the operator to press the Reset key. After the reset, error recovery depends on the kind of load being performed as follows.

- If the standard load processor from the common functions area was performing the load, the load prompt is redisplayed with the load parameters previously entered. The operator then rekeys the correct information.

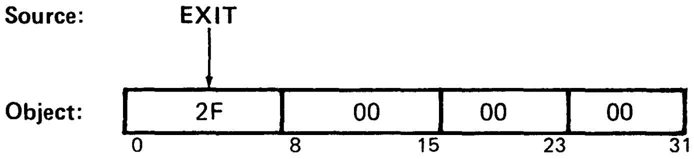
- If a program instruction was reloading the same partition and the standard load prompt is available in the common functions area, the standard load prompt is displayed. The operator then enters the load parameters.

- If a program instruction was reloading the same partition and no standard load prompt is available, the load cannot be retried. The main microprocessor issues an exit instruction and goes to the next partition.

- If a program instruction was loading another partition, the load is not retried. Control returns to the instruction following the load instruction.

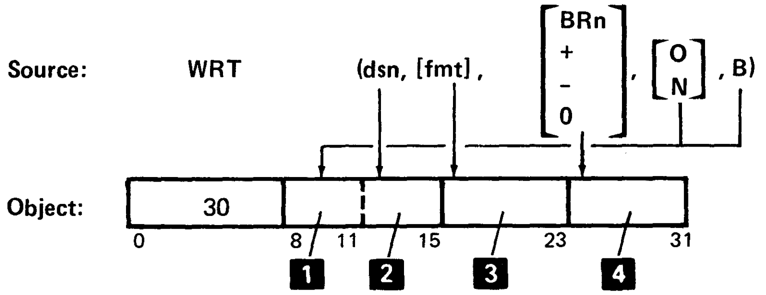
Do not put error recovery procedures in a storage area that is to be overlayed with a partial overlay.

Exit a Partition (EXIT)



This instruction detaches a partition if it was attached to a keyboard, closes all open data sets, and executes a system unlock operation in case the partition was locked when the exit instruction was issued (see op code 2D). If the exit instruction is issued in a background partition, bit 1 of byte 1 of the partition IOB pointer in the system control block is turned on to make the partition available to be loaded. This bit must be on for the partition to be loaded by another partition. If the exit instruction is issued in a foreground partition, a flag is set in the partition IOB (bit 6 byte 2B) to indicate that the partition is available to be loaded; the bit in the partition IOB pointer is not turned on, so keystrokes can be processed in the exited partition.

Write a Record to a Data Set (WRT)



1 Bits:

8 0 = Sequential record access method. For a printer, bit 8 must be 0 and bits 10-23 are ignored.
1 = Relative record access method.

9 0 = Overlap mode.
1 = Nonoverlap mode.

10 Not used, always zero.

11 0 = I/O buffer is not blanked.
1 = I/O buffer is blanked at the start of the operation (B is specified) if edit formatting is specified.

2 Data set number: The number (hex 1-F) of the data set to be written.

3 Format number: The number (hex 01-FE) of the format to use. If no format number is used, this will be a hex FF.

4 Record to write: The location of records to write can be:

- BRn for the address of a binary register. If the data set is an SCS data set, the register contains the address of an area where SCS command characters are stored. If the data set is not an SCS data set, the register contains the record number.

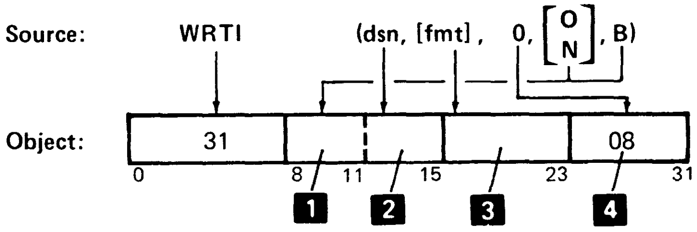
- Hex 07 (- specified) for writing the previous record.

- Hex 08 (0 specified) for writing the current record. For a printer, it must be hex 08.

- Hex 09 (+ specified) for writing the next record.

This instruction writes the contents of the logical buffer into the specified record position of the physical buffer. The contents of the physical buffer may be written to the diskette. If an edit format is specified, data is moved into the logical buffer as indicated by the edit format before it is written to the physical buffer. If this instruction is issued when the current record counter is at EOD, the record is written into the EOD space and the EOD and current record counter are incremented; otherwise, the current record number is never changed by a write instruction.

Insert a Record into a Data Set (WRTI)



1 Bits:

8	Not used, always zero.
9	0 = Overlap mode (O specified). 1 = Nonoverlap mode (N specified).
10	Not used, always zero.
11	0 = I/O buffer is not blanked. 1 = I/O buffer is blanked at the start of the operation (B is specified) if a format is specified.

2 Data set number: The number (hex 1-F) of the data set to be accessed.

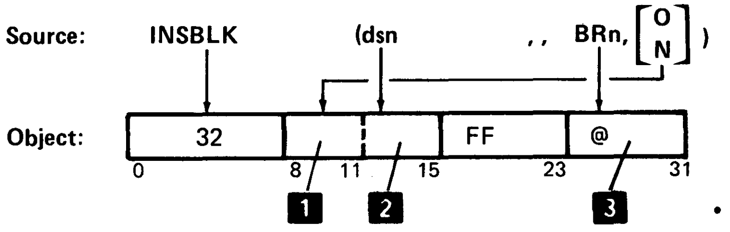
3 Format number: The number (hex 01-FE) of the format to use. If no format is specified, this will be hex FF.

4 Record to access: Always the current record (hex 08).

This instruction writes the current logical record to the physical buffer, into the current record position. The record that was in the current record position, and all records beyond the inserted record, are moved down one position until EOD or a deleted record is encountered. If the record is inserted as the last record in the data set, this instruction acts as a write instruction (op code 30).

Note: Two physical buffers and one logical buffer must be available for this instruction.

Insert a Block of Records into a Data Set (INSBLK)



1 Bits:

8 Not used, always 0.
9 0 = Overlapped mode (O specified).
1 = Nonoverlapped mode (N specified).
10-11 Not used, always 00.

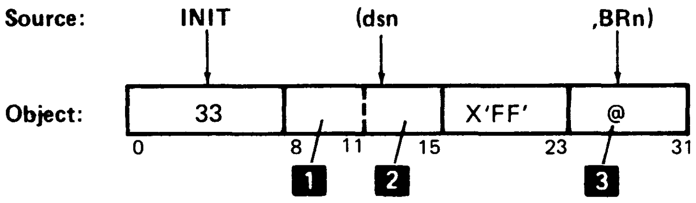
2 Data set number: The number (hex 1-F) of the data set to access.

3 Records to insert: The address of the binary register that contains the number of logical records to be inserted. Two commas must precede the binary register in the source instruction.

The records from (and including) the current record to the end of the data set are moved down to make room for the specified number of records to be inserted. The inserted records are treated as deleted records and may be written with the WRTI instruction (op code 31). The current record counter is modified to point to the first inserted record.

Note: Two physical buffers and a logical buffer must be available for this instruction.

Initialize a Diskette (INIT)



1 0100 (Always nonoverlapped mode.)

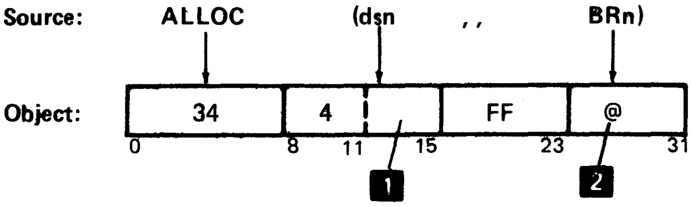
2 Data set number: The number (hex 1-F) of the data set to be accessed.

3 Parameters' address: The address of the binary register that contains the address of the initialization parameters.

This instruction initializes the diskette with information from the data set IOB. The data set IOB must have previously been opened as a write-only label update data set (TYPE = INI). The initialization parameters must be stored in a data area before the initialization instruction is issued. The format of the initialization parameters is:

Bytes	Bits	Information
1	0	Head number
	1-7	Track number
2	0	0 = FM (1 or 2) 1 = MFM (2D)
	1	0 = 1-sided 1 = 2-sided
	2-7	Number minus 1 of 128-byte blocks that make up the sector size.
3-28		Sequence of sector numbers. If byte 3 = hex FF, the track specified by byte 1 is flagged as a defective track.

Allocate a Data Set (ALLOC)



1 Data set number: The number (hex 1-F) of data set to allocate.

2 Parameters' address: The address of the binary register that contains the address of the allocate parameters. The binary register must be preceded by two commas in the source instruction.

This instruction is always executed in nonoverlapped mode. For a printer, this instruction is executed as an open instruction. For diskette, when the ALLOC operation is executed the data set is allocated in the physical space following the last valid data set existing on the diskette, provided sufficient extent and label space exists. A data set cannot be allocated between existing data sets and always originates on a physical track/sector boundary.

The data set HDR1 label is placed in the first deleted HDR1 label space. If there are no deleted HDR1 label spaces, the allocation cannot take place, and an external status (3229) is presented. The HDR1 information is taken from the data set IOB and from the parameter string in storage. The binary register (BRn) in the ALLOC instruction contains the address of the *fifth* byte of the parameter string. The format of the parameter string is as follows:

Byte	Meaning
1	Data set exchange type. A hex number that corresponds to the appropriate exchange type: 00 = Basic exchange 01 = H exchange 02 = I exchange (this is the type normally used) 03 = E exchange, unblocked and unspanned 04 = E exchange, blocked and unspanned 05 = E exchange, blocked and spanned
2-4	The number of logical records to allocate. Hex 000000 allocates the maximum number of records that can be placed on the remaining diskette space.
5	The first of up to 14 characters of an optional owner identification, required for allocating on a secure diskette. The address stored in the binary register always points to this byte. If the owner identification is omitted, the address points to the end blank.

Byte	Meaning
end	The last byte in the parameter string must always be a blank (hex 40) unless a 14-character owner ID is specified.

Note: This parameter string can also be used to open a data set on a secure diskette; the OPEN instruction does not use the bytes before the fifth byte.

The information that is taken from the data set IOB is as follows:

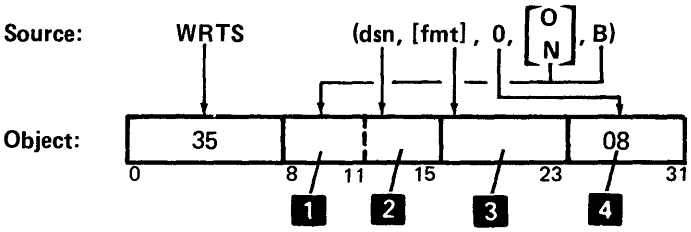
Parameter	Explanation
DATA set name (NAME)	The data set name is mandatory for allocating a diskette data set. It is optional for a printer.
Logical record length (RECL)	If this option .DATASET parameter is omitted, the length is set to equal to block size.
Block size (BSIZ)	Except for blocked and spanned data sets, the block size must equal, or be a multiple of, the logical record length. For blocked and spanned data sets, BSIZ is an optional parameter; if specified it must equal sector size, and if omitted the 5280 sets it to sector size.
Delete Character (DFLG)	Delete flag; the character that is placed in the HDR1 label during the allocate, and which will be used to indicate a deleted record. Optional for I and E exchange; ignored for basic and H exchange. Valid characters can be A-Z, 0-9, or one of the following symbols: . , - / % # @ : $ &.

During the allocation operation, the data set organization byte of the HDR1 label is set to blank (hex 40) for basic and H exchange data sets. It is set to D for I and E exchange data sets. It is invalid to allocate a data set with the ALLOC instruction when the data set type is label update.

Upon completion of the ALLOC operation, the allocated data set is also opened. The op code in the data set IOB is replaced with hex 00. Upon completion of the ALLOC, or if an external status for insufficient physical buffer size (3430) or for two physical buffers specified with unequal sizes (3435) occurs, or if any group 7 warning message is presented, the minimum number of 128-byte blocks required for sufficient buffer size is placed into hex 78 of the data set IOB. If any other external status occurs, this number is not placed into the IOB.

The HDR1 label is placed into the first 128 bytes of the physical buffer except for pointer I/O and SCS data sets that have the SW or ERS parameters specified in the .DATASET control statement.

Delete a Record from a Data Set (WRTS)



1 Bits:

8 Not used, always zero.
9 0 = Overlap mode (O specified).
1 = Nonoverlap mode (N specified).
10 Not used, always zero.
11 0 = I/O buffer is not blanked.
1 = I/O buffer is blanked at the start of the operation (B is specified) if a format is specified.

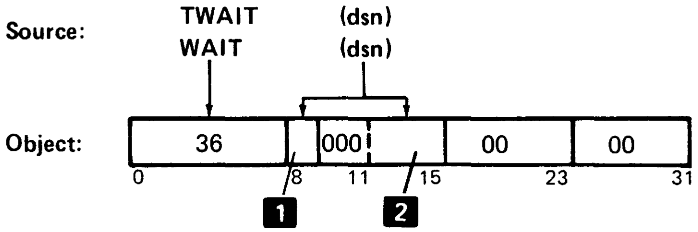
2 Data set number: The number (hex 1-F) of the data set to be accessed.

3 Format number: The number (hex 01-FE) of the format to use. If no format was specified, this is hex FF.

4 Record to access: Always the current record (hex 08).

When this instruction is executed, the record indicated by the current record counter is written as for the write instruction (op code 30). In addition, the record is flagged as deleted. For a basic or H exchange data set, a special address mark is used to flag a deleted record. For an I or E exchange data set, the delete character in the data set IOB is used to flag a deleted record.

Wait for I/O Completion (WAIT/TWAIT)



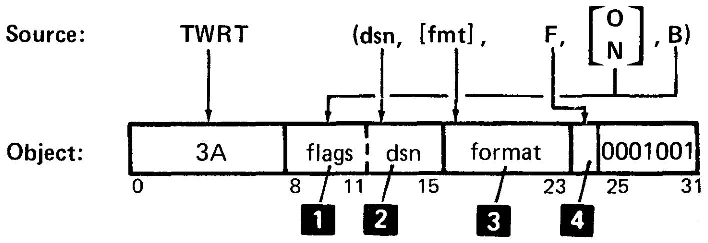
1 Bit 8:

0 = Data set number 2 was specified.
1 = No data set number was specified.

When the main microprocessor executes a wait instruction, it waits until all I/O operations are complete for the specified data set before executing the next sequential instruction. If no data set number is specified, all data sets are checked for completed I/O operations.

2 Data set number: The number (hex 0-F) of the data set to check for completed I/O operations. If data set number zero is specified, it indicates the keyboard/display IOB.

Write to Communications (TWRT)



1 Bits:

8		Not used, always zero.
9	0 =	Overlapped mode (O specified).
	1 =	Nonoverlapped mode (N specified).
10		Not used, always zero.
11	0 =	The I/O buffer is not blanked.
	1 =	The I/O buffer is blanked at the start of the operation (B is coded) if a format is specified.

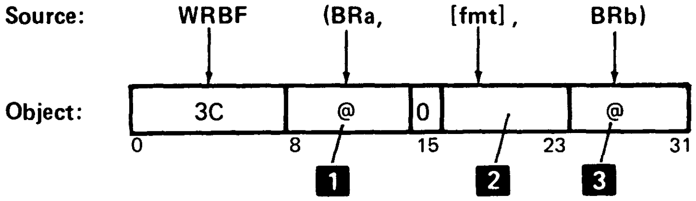
2 Data set number: The number (hex 1-F) of the data set to access.

3 Format number: The number (hex 00-FE) of the format to use. If no format is specified this is hex FF.

4 Bit 24:
0 = Normal write
1 = Final write (F coded) in an SNA application

This instruction transmits a record from the data set specified to the host system. If an edit format is specified, data is placed into the logical buffer as indicated by the format.

Formatted Write from Registers (WRBF)



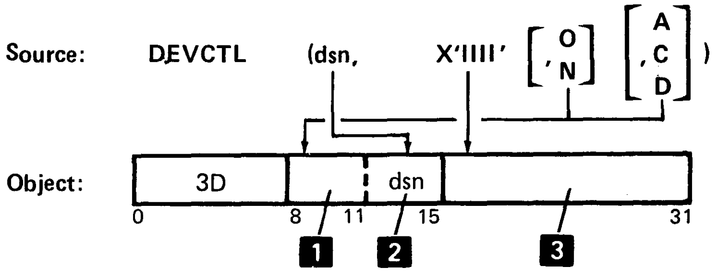
1 Write address: The address of the binary register that contains the address of the leftmost byte of the data area to write into.

2 Format number: The number (hex 01-FE) of the format to use. This is hex FF if no format is specified.

3 Blank option: The address of the binary register that contains the number of bytes that are blanked before formatting begins. If BRb is not specified, no bytes are blanked, and this will contain hex FF.

This instruction moves the data indicated by the format, or blanks if no format and a binary register is specified, into the data area pointed to by the write address.

Device Control (DEVCTL)



1 Bits:

8	0 = A omitted (device dependent)
	1 = A specified (device dependent)
9	0 = Overlap mode (O specified)
	1 = Nonoverlap mode (N specified)
10	0 = C omitted (device dependent)
	1 = C specified (device dependent)
11	0 = D omitted (device dependent)
	1 = D specified (device dependent)

2 Data set number: The number (hex 1-F) of the data set IOB to access.

3 Control parameters: 2 bytes of hex digits that specify the control operation. The hex digits and operations depend upon the different devices.

This instruction is intended for diagnostics use only.

For diskette device control there are write-defective-sector or diagnostic operations.

Diagnostic operations are used for reading or writing data in diskette microprocessor or adapter registers. (These registers are not the decimal or binary registers used in an application program.) If A is specified when writing registers, the data to be written is taken from the binary register specified by bits 24-31. If A is specified when reading registers, the data that is read is placed into the binary register specified by bits 24-31. If A is not specified when writing registers, the data to be written is taken from bits 24-31. If A is not specified when reading registers, the data is read into bits 24-31.

Bits 16-31 have the following meaning:

Bits	**Meaning**
16	0 = Read register
	1 = Write register
17	0 = Diskette microprocessor register
	1 = Adapter register
18-19	00 = Diagnostic command
20-23	0 = Register 16
	1 = Register 17
	2 = Register 18
	3 = Register 19
	4 = Register 20
	5 = Register 21
	6 = Register 22
	7 = Register 23
	8 = Register 24
	9 = Register 25
	A = Register 26
	B = Register 27
	C = Register 28
	D = Diskette microprocessor register 13[1]
	E = Diskette microprocessor register 25[1]
	F = Diskette microprocessor register 26[1]
24-31	Binary register address if option A is specified; immediate data if option A is not specified.

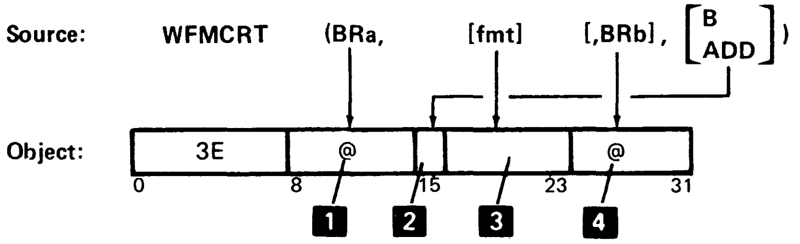
[1] These specifications always indicate a diskette microprocessor register regardless of what bit 17 indicates.

Write-defective-sector is used for marking the sector specified by the current record pointer as a defective sector. This instruction can only be used in a data set where a sector is also a logical record. Write-defective-sector is specified by setting both bits 18 and 19 to 1.

For printer device control, bits 16 through 31 are:

Hex Digits	Option	Operation
FF00	A	Wrap test: The POR wrap test is run once each time this instruction is executed. Any errors encountered are reported.
FE00		Line quality check: This test performs a single poll command without looking for a response.
0Dxx	A	Read external register 13: This operation reads the contents of register 13 into the fourth byte of the instruction (xx).
19xx	A	Read external register 25: This operation reads the contents of register 25 into the fourth byte of the instruction (xx).
1Axx	A	Read external register 26: This operation reads the contents of register 26 into the fourth byte of the instruction (xx).
8Dxx		Write external register 13: This operation writes the contents of the fourth byte of this instruction (xx) into register 13.
99xx		Write external register 25: This operation writes the contents of the fourth byte of this instruction (xx) into register 25.
9Axx		Write external register 26: This operation writes the contents of the fourth byte of this instruction (xx) into register 26.

Formatted Write to the Screen (WFMCRT)



1 Screen address: The address of the binary register that contains the row number to begin the write.

2 Bit 15:
0 = Data between fields is blanked; B is specified.
1 = Data between fields remains on the screen; ADD is specified.

3 Format number: The number (hex 01-FE) of the format to use.

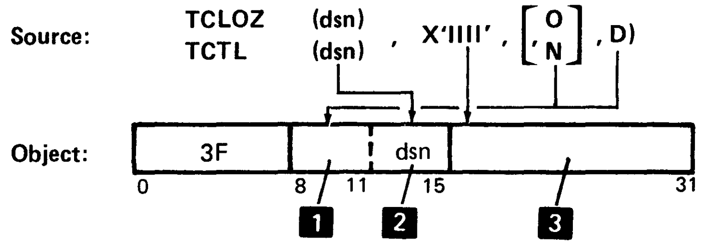
4 The address of the binary register that contains the number of bytes to blank or add in the screen buffer before formatting begins.

Data is moved to the screen, beginning at column 1 of the row specified by the low-order byte of the screen address register. Data is moved from the locations specified by the labeled edit format, for the number of bytes specified by the format. The format also specifies any punctuation that should appear on the screen, such as a dollar sign, decimal point, or minus sign. The format must not use more than 200 screen positions. If row 0 is specified, data is moved to the status line; if row 1 is specified, data is moved to the extra line.

If the binary register 4 is included, the contents of this register are taken as the number (1-200) of screen positions to alter before the formatted data is moved to the screen. If B is coded in the source instruction, all characters on the screen between the data fields that are defined in the edit format are blanked for the number of bytes specified in the binary register. If ADD is coded in the source instruction, only the fields that are defined in the edit format are changed on the screen; the characters between the edit format fields remain on the screen for the number of bytes specified by the binary register. If the binary register and B/ADD are omitted, and if the edit format does not account for all of the positions on the screen within the edit format, the results are unpredictable.

The fields of the format must be in the order of their appearance on the screen.

Communications Close or Device Control (TCLOZ/TCTL)



1 Bits:

8	0 =	TCLOZ
	1 =	TCTL
9	0 =	Overlap mode (O specified).
	1 =	Nonoverlap mode (N specified).
10		Not used, always zero
11	0 =	Normal operation
	1 =	Diagnose operation, D specified (on TCTL only)

2 Data set number: The number (hex 1-F) of the data set to access.

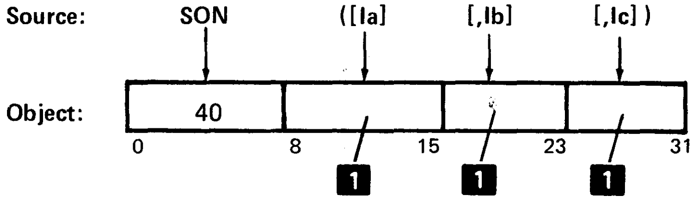
3 Data type: Hex 0000 for TCLOZ. A hex constant data for TCTL.

The TCLOZ instruction is used with BSC. It closes the specified IOB and signifies the end of communications operations.

The TCTL instruction performs the control operation specified by the hex constant, as follows:

Constant	**Operation Valid for BSC**
0100	Write status
0300	Transmit EOT
0400	Transmit RVI
0500	Transmit header (SOH-heading-STX)
0600	Transmit header (SOH-heading-ETB)
0700	Transmit header (SOH-heading-ITB)
0800	Transmit header (SOH-heading-STX-ETX)
0900	Execute wrap test
0A00	Transmit online test message
0B00	Received online test message
0001	Set compression (Expand blank-compressed data)
0002	Reset compression (Do not expand blank-compressed data)
0003	Set transparent mode on
0004	Reset transparent mode off
0005	Set trace on
0006	Reset trace off
	Operation valid for SNA
0007	Transmit signal command to the host
0001	Cancel
0002	Chase
0003	LU Status
0004	Request shutdown
0005	Positive response
0006	Negative response
0008	Shutdown complete

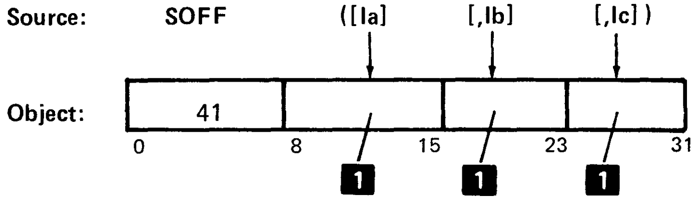
Set Indicator On (SON)



1 Indicator numbers: The numbers of specified indicators that are set on. An indicator number from hex 00-FE can be specified. If no indicator is specified, the contents are hex FF.

This instruction sets the specified indicators on. When the main microprocessor encounters the first byte that contains hex FF, it stops checking for more indicators.

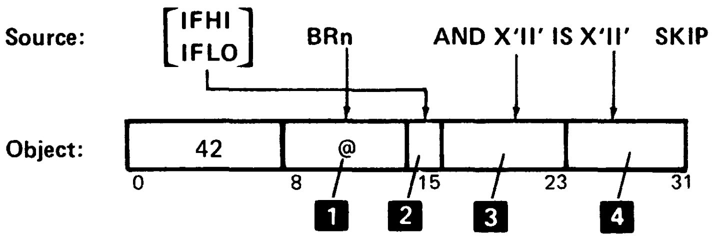
Set Indicator Off (SOFF)



1 Indicator numbers: The numbers of specified indicators that are set off. An indicator number from hex 00-FE can be specified. If no indicator is specified, the contents are hex FF.

This instruction sets the specified indicators off. When the main microprocessor encounters the first byte that contains hex FF, it stops checking for more indicators.

Skip on AND, Exclusive-OR Mask (AND)



1 Test register address: Test the binary register at this address.

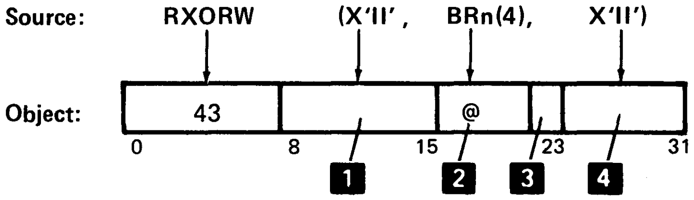
2 Test mask:
  0 = Mask the leftmost byte
    (IFHI specified) in the binary register.
  1 = Mask the rightmost byte
    (IFLO specified) in the binary register.

3 AND mask: 2 hex digits that AND with the specified test mask byte of a binary register.

4 Exclusive-OR mask: 2 hex digits that exclusive-OR with the result of the AND operation.

This instruction applies the AND mask against the specified test mask byte, then applies the OR mask against the result of the AND operation. If the result of both operations is zero, the main microprocessor skips the next sequential instruction. If the result is not zero, the next sequential instruction is executed. The register contents remain unchanged.

Skip on Exclusive-OR, AND Mask (RXORW)



1 Exclusive-OR mask: Two hex digits that exclusive-OR the byte specified by the address in the binary register.

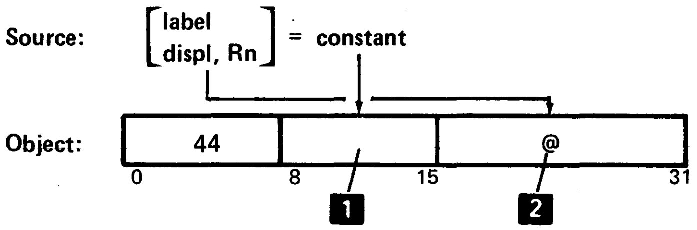
2 Test register address: The address of the binary register that contains the address of the byte to test.

3 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

4 AND mask: Two hex digits that are ANDed with the original contents of the byte specified by the address in the binary register.

This instruction applies the exclusive-OR mask against the byte in the binary register. Then the microprocessor ANDs the original contents of the byte with the AND mask. If the result of the AND operation is 0, the next sequential instruction is skipped. If the result is not 0, the storage position is restored to its original value and the next sequential instruction is executed.

Constant Insert (= constant)

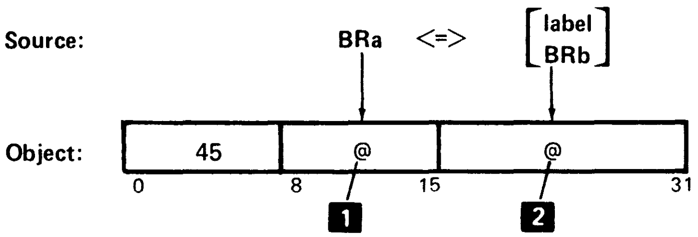


1 Constant: The binary representation of the constant to insert.

2 Insert address: The address of the byte in storage, or the byte in a decimal register, where the constant is inserted.

This instruction inserts the specified 1-byte constant into the indicated byte.

Exchange Binary Register Contents (<=>)

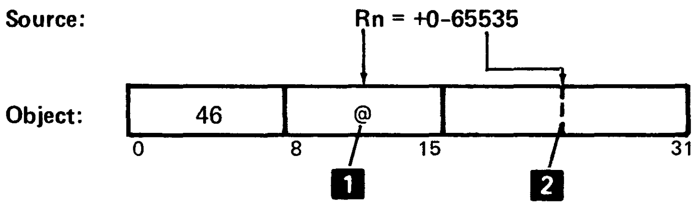


1 Binary register address: The address of the binary register that exchanges contents with 2.

2 Binary register or storage address: The address of the area that exchanges contents with 1.

This instruction exchanges the contents of the two specified data areas.

Immediate Load of Positive Constant into Decimal Register (Rn = +n)

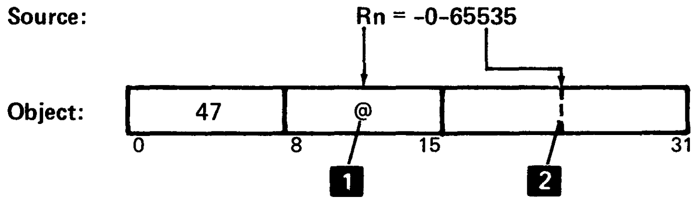


1 Decimal register address: The address of the decimal register that is loaded with the constant.

2 Constant: The 2-byte constant (hex 0-FFFF) that is converted to the decimal EBCDIC and placed into the decimal register.

This instruction places the constant into the decimal register. The constant is padded on the left with hex zeros (F0).

Immediate Load of Negative Constant into Decimal Register (Rn = -n)

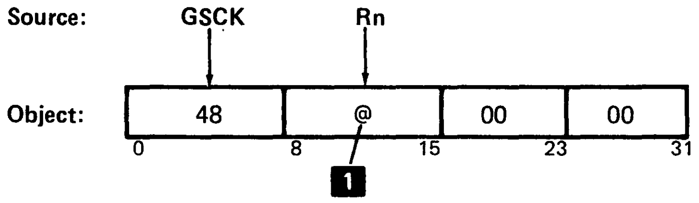


1 Decimal register address: The address of the decimal register that is loaded with the constant.

2 Constant: The 2-byte constant (hex 0-FFFF) that is converted to the decimal EBCDIC and placed into the decimal register.

This instruction places the constant into the decimal register. The constant is padded on the left with hex zeros (F0). The zone of the rightmost byte in the register is changed to hex D.

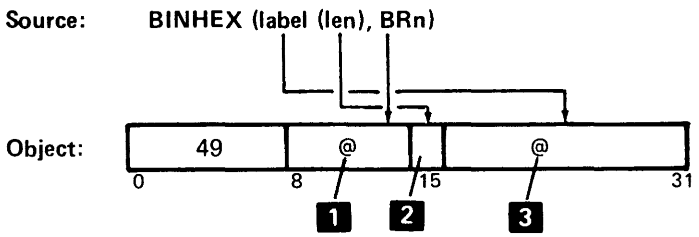
Generate Self-Check Number (GSCK)



1 Decimal register address: The address of the decimal register or decimal double register that contains data to which the self-check digit (from the algorithm defined in the .SELFCHK control block) is added.

This instruction uses the self-check control block to generate a self-check number from the foundation characters contained in the decimal register(s), and inserts the self-check number into the register(s) as specified by the self-check control block.

Convert Binary to EBCDIC (BINHEX)



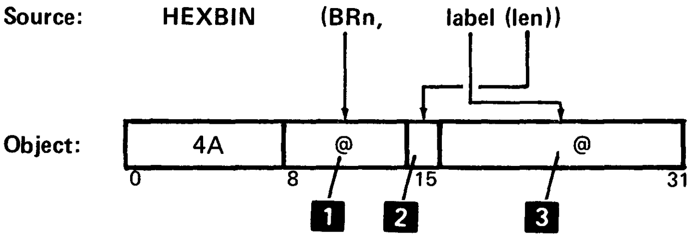
1 Binary register address: The address of the binary register that contains the binary data to convert to EBCDIC.

2 Bit 15:
0 = Data area length is 4 bytes.
1 = Data area length is 2 bytes.

3 Data area address: The address of the data area where the converted data is stored upon completion of this operation.

This instruction converts the contents of the low-order byte of the specified register from binary to 2 bytes of EBCDIC, or the contents of the 2-byte binary register to 4 bytes of binary register to 4 bytes of EBCDIC. The result is stored in the specified data area. Each half-byte is converted into EBCDIC hex characters 0-9, A-F.

Convert EBCDIC to Binary (HEXBIN)



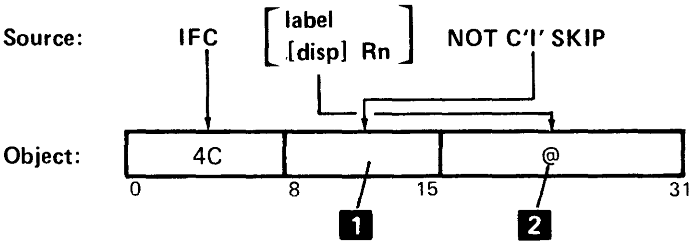
1 Binary register address: The address of the binary register where the converted data is stored upon completion of this operation.

2 Bit 15:
0 = Data area length is 4 bytes.
1 = Data area length is 2 bytes.

3 Data area address: The address of the data area that contains the EBCDIC data to convert to binary.

This instruction converts the contents of the specified data area from 2 bytes of EBCDIC to 1 byte of binary and places it in the low-order byte of the specified register, or from 4 bytes of EBCDIC to 2 bytes of binary and places it in the specified binary register. If the characters are not A-F or 0-9, an error results.

Skip If Not Equal (IFC NOT)

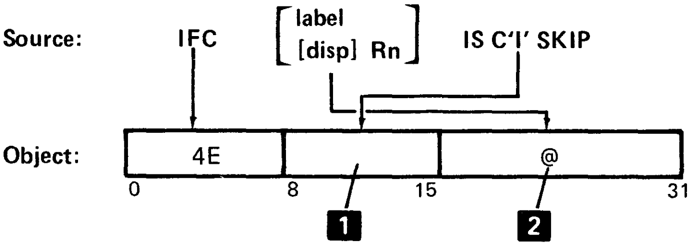


1 Test character: The binary representation of the byte of hex, binary, character, or decimal test data that is compared to the test byte.

2 Test byte address: The address of the byte of data to compare to the test character.

If the test byte is not equal to the test character, the microprocessor skips the next sequential instruction; otherwise, it executes the next sequential instruction.

Skip If Equal (IFC IS)

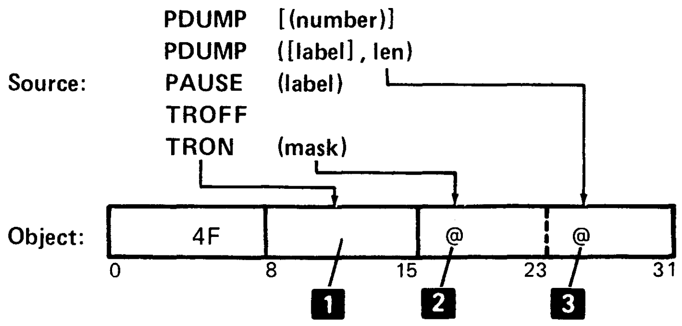


1 Test character: The binary representation of the byte of hex, binary, character, or decimal test data that is compared to the test byte.

2 Test byte address: The address of the byte of data to compare to the test character.

If the test byte is equal to the test character, the microprocessor skips the next sequential instruction; otherwise, it executes the next sequential instruction.

Debugging Aids (PDUMP/PAUSE/TROFF/TRON)



1 Hex 10 = PDUMP (label, len)
08 = PDUMP (number) or PDUMP
20 = PAUSE
40 = TROFF
80 = TRON

2 For PDUMP (label, len): The address, divided by 256, of where to start the dump. If no address (label) is specified, this is hex 00; the dump starts at the beginning of the partition.

For PDUMP (number): The partition number of the partition to dump. If no partition is specified, this is hex FF and the current partition is dumped.

For TRON: The trace options.

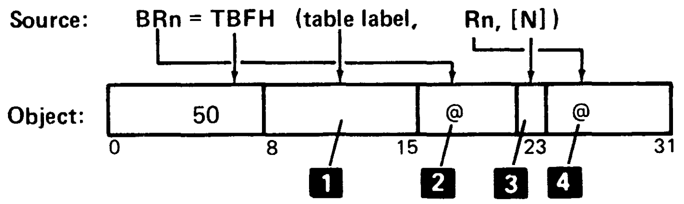
3 For PDUMP (label, len): The number of 256-byte blocks to dump.

For PDUMP (number): All zeros.

For TRON: All zeros.

2 For PAUSE: The address of where to stop the program.
and
3 For TROFF: All zeros.

Search Ordered Table for Higher or Equal Entry (TBFH)



1 Table: The index into the system table that contains the address and parameters for the table to be searched.

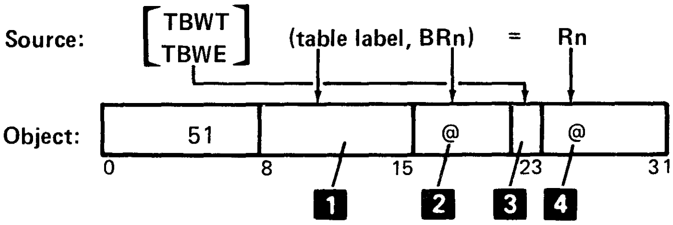
2 Index register address: The address of the binary register into which the table index where the index of the higher or equal entry is placed upon completion of this operation.

3 Bit 23:
0 = Begin the search in the table with the first entry (N not specified).
1 = Begin the search in the table with the next entry after the entry in the index register (N specified).

4 Search argument address: The address of the decimal register that contains the search argument.

The labeled table is searched for an entry equal to or higher than the contents of the decimal register. The search ends when the first higher or equal entry is found or when the last table entry has been searched. If an equal or higher entry is found, the index of that entry is placed into the binary register. If no equal or higher entry is found, the binary register remains unchanged and I125 and I127 are set on.

Write Table Entry (TBWT/TBWE)



1 Table: The index into the system table that contains the address and parameters for the table to be written into.

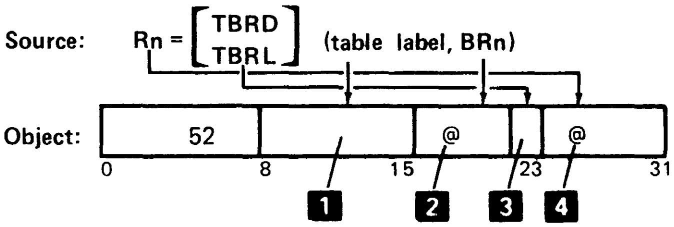
2 Index register address: The address of the binary register that contains the index into the table.

3 Bit 23:
0 = Write the entry to the table at the index contained in the index register (TBWT specified).
1 = Extend the table and add the entry at the end of the table (TBWE specified).

4 Argument address: The address of the decimal register that contains the argument to be written.

An entry is written into the table at either the end of the table for a TBWE instruction, or at a specified location into the table for a TBWT instruction.

Read Table Entry (TBRD/TBRL)



1 Table: The index into the system table that contains the address and parameters for the table to be read.

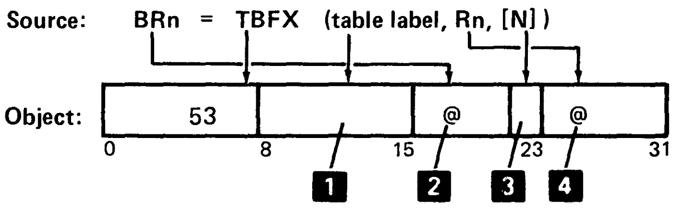
2 Index register address: The address of the binary register that contains the index into the table.

3 Bit 23:
 0 = Read the entry in the table at the index contained in the index register (TBRD specified).
 1 = Read the last entry in the table (TBRL specified).

4 Argument address: The address of the decimal register where the table argument is placed upon completion of this operation.

An entry is read from the table and placed into the argument address.

Search Unordered Table for Equal Entry (TBFX)



1 Table: The index into the system table that contains the address and parameters for the table to be searched.

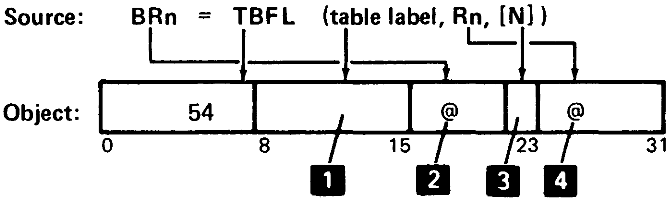
2 Index register address: The address of the binary register into which the table index where the index of the equal entry is placed upon completion of this operation.

3 Bit 23:
0 = Begin the search in the table with the first entry (N not specified).
1 = Begin the search in the table with the entry after the entry in the index register (N specified).

4 Search argument address: The address of the decimal register that contains the search argument.

The labeled table is searched for an entry that is equal to the search argument. If an equal entry is found, the index for that entry is placed in the binary register. If no equal entry is found, the binary register remains unchanged and I125 and I127 are set on.

Search Reverse Ordered Table for Lower Entry (TBFL)



1 Table: The index into the system table that contains the address and parameters for the table to be searched.

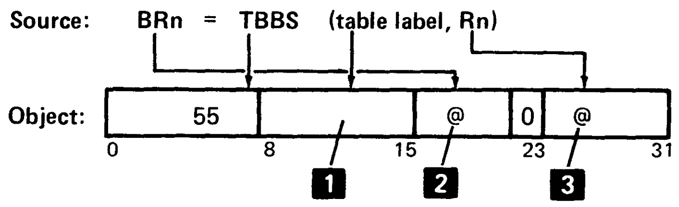
2 Index register address: The address of the binary register into which the table index of the lower entry is placed upon completion of this operation.

3 Bit 23:
0 = Begin the search in the table with the first entry (N not specified).
1 = Begin the search in the table with the entry before the entry in the index register (N specified).

4 Search argument address: The address of the decimal register that contains the search argument.

The table is searched for an entry that is lower than the search argument. If a lower entry is found, the index of that entry is placed into the binary register. If no lower entry is found, the binary register remains unchanged and I125 and I127 are set on.

Search Table Using Binary Search (TBBS)



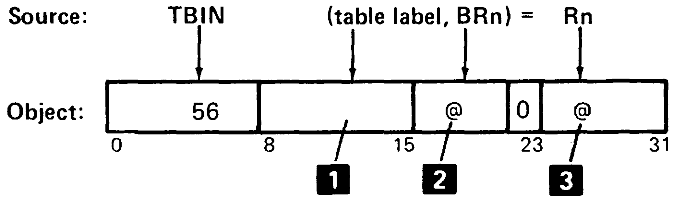
1 Table: The index into the system table that contains the address and paran, eters for the table to be searched.

2 Index register address: The address of the binary register into which the table index of the equal entry is placed upon completion of this operation.

3 Search argument address: The address of the decimal register that contains the search argument.

The labeled table is searched for an entry equal to the search argument. If an equal entry is found, the index of that entry is placed into the binary register and I103 is set on. If no equal entry is found, the binary register remains unchanged and I125 and I127 are set on.

Insert Table Entry (TBIN)



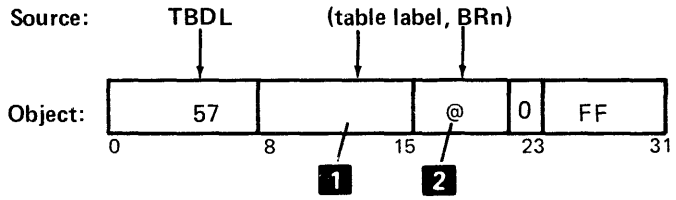
1 Table: The index into the system table that contains the address and parameters for the table to be modified.

2 Index register address: The address of the binary register that contains the table index where the entry is inserted.

3 Argument address: The address of the decimal register that contains the argument to insert.

The argument is inserted into the table at the table index specified in the binary register. All entries below the inserted entry are moved downward.

Delete Table Entry (TBDL)

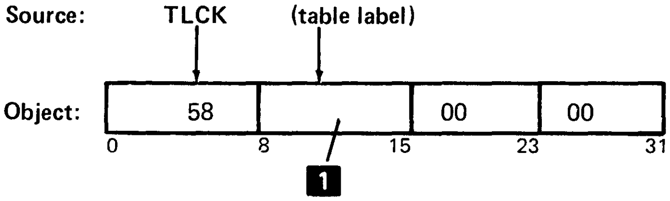


1 Table: The index into the system table that contains the address and parameters for the table to be modified.

2 Index register address: The address of the binary register that contains the table index where the entry is deleted.

The entry in the labeled table is deleted and all other entries move up to replace the deleted entry.

Lock Shared Table (TLCK)

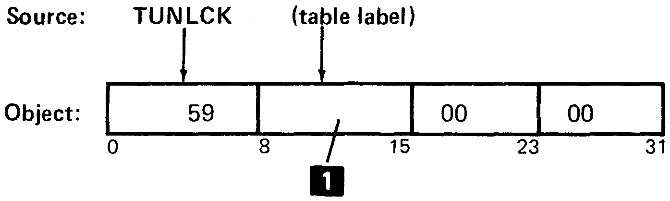


1 Table: The index into the system table that contains the address and parameters for the table to be locked.

The specified table is locked for exclusive use by the program that issues the TLCK instruction.

This instruction can be used only with tables in the common area.

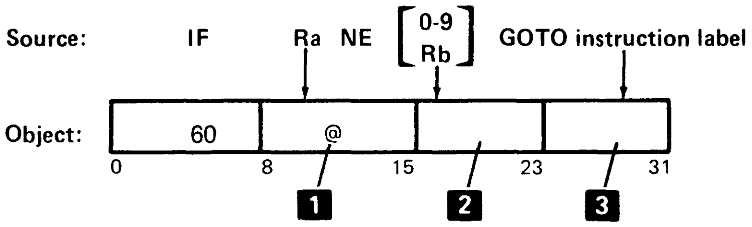
Unlock Shared Table (TUNLCK)



1 Table: The index into the system table that contains the address and parameters for the table to be unlocked.

This instruction frees the table that was locked by the TLCK instruction.

Compare Decimal for Not Equal (IF Rn NE)



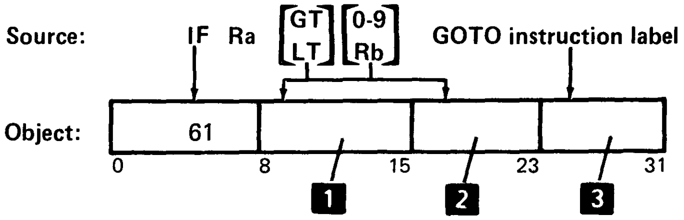
1 Test register: The address of the decimal register that contains the data to compare.

2 Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data. If a constant is used, the bytes on the left are padded with blanks (hex 40s) before the compare.

3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instruction from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the test register is not equal to the compare data.

Compare Decimal for Greater Than or Less Than (IF Rn GT/LT)



1 *GT.* The test register: The address of the decimal register that contains data to compare.

LT. Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data. If a constant is used, the bytes on the left are padded with blanks (hex 40s) before the compare.

2 *GT.* The compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data. If a constant is used, the bytes on the left are padded with blanks (hex 40s) before the compare.

LT. The test register: The address of the decimal register that contains the data to compare.

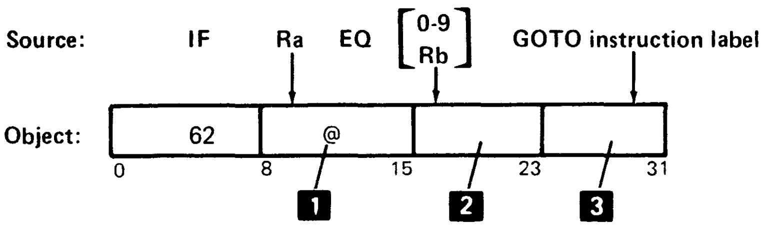
3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The content of the test register is greater than the compare data and GT is specified.

- The content of the test register is less than the compare data and LT is specified.

Compare Decimal for Equal (IF Rn EQ)



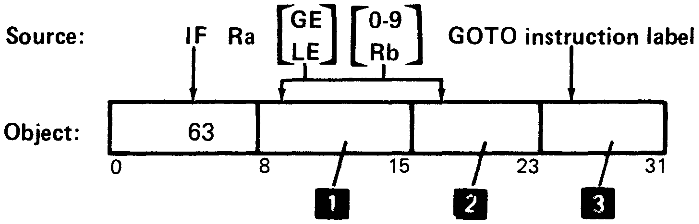
1 Test register: The address of the decimal register that contains the data to compare.

2 Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data. If a constant is used, the bytes on the left are padded with blanks (hex 40s) before the compare.

3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the test register is equal to the compare data.

Compare Decimal for Greater or Equal, or Less Than or Equal (IF Rn GE/LE)



1 *GE.* The test register: The address of the decimal register that contains data to compare.

LE. Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data. If a constant is used, the bytes on the left are padded with blanks (hex 40s) before the compare.

2 *GE.* The compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register (RB) that contains the compare data.

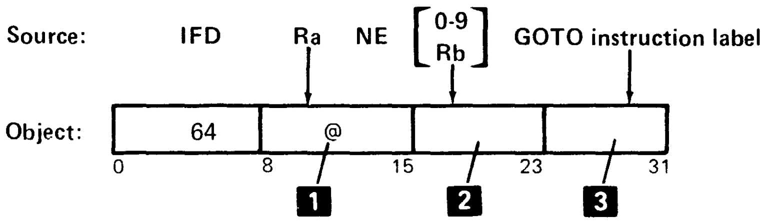
LE. The test register: The address of the decimal register that contains the data to compare. If a constant is used, the bytes on the left are padded with blanks (hex 40s) before the compare.

3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The content of the test register is greater than or equal to the compare data and GE is specified.
- The content of the test register is less than or equal to the compare data and LE is specified.

Compare Decimal Digits for Not Equal (IFD Rn NE)



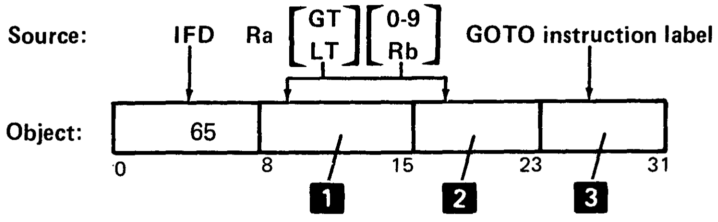
1 Test register: The address of the decimal register that contains the data to compare.

2 Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data.

3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the digit portion and the units zone (sign) of the content of the test register is not equal to the digit portion of the compare data. If the zone portion of the rightmost byte of a decimal register contains hex D, the contents of the register are negative. If it is not hex D, the contents of the register are positive.

Compare Decimal Digits for Greater or Less Than (IFD Rn GT/LT)



1 *GT.* The test register: The address of the decimal register that contains data to compare.

LT. Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data.

2 *GT.* The compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data.

LT. The test register: The address of the decimal register that contains the data to compare.

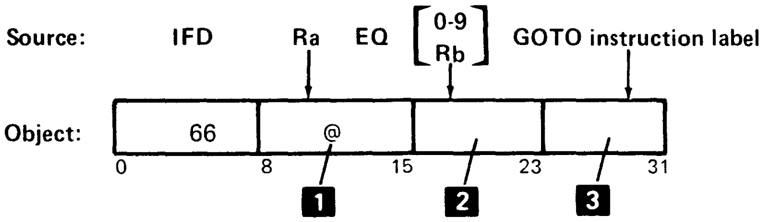
3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The digit portion of the content of the test register is greater than the digit portion of the compare data and GT is specified.

- The digit portion of the content of the test register is less than the digit portion of the compare data and LT is specified. If the zone portion of the rightmost byte of a decimal register contain hex D, the contents of the register are negative. If it is not hex D, the contents of the register are positive.

Compare Decimal Digits for Equal (IFD Rn EQ)



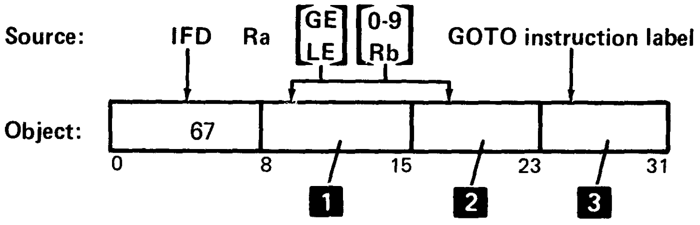
1 Test register: The address of the decimal register that contains the data to compare.

2 Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data.

3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the low-order byte of the test register is not equal to the compare data. If the zone portion of the rightmost byte of a decimal register contains hex D, the contents of the register are negative. If it is not hex D, the contents of the register are positive.

Compare Decimal Digits for Greater or Equal, or Less Than or Equal (IFD Rn GE/LE)



1 *GE.* The test register: The address of the decimal register that contains data to compare.

LE. Compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data.

2 *GE.* The compare data: The constant (hex 0-9) followed by hex 0, or the address of the decimal register that contains the compare data.

LE. The test register: The address of the decimal register that contains the data to compare.

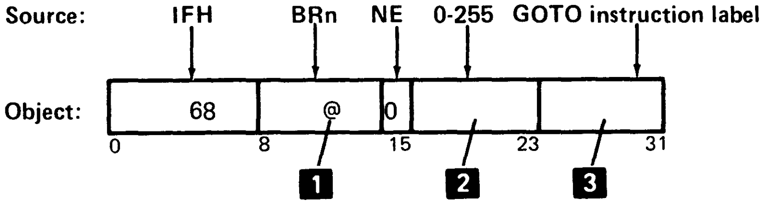
3 Branch instruction: The number minus 1 of 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The digit portion of the content of the test register is greater than or equal to the digit portion of the compare data and GE is specified.

- The digit portion of the content of the test register is less than or equal to the digit portion of the compare data and LE is specified. If zone portion of the rightmost byte of a decimal register contains hex D, the contents of the register are negative. If it is not hex D, the contents of the register are positive.

Compare Binary Half-Register for Not Equal (IFH BRn NE)



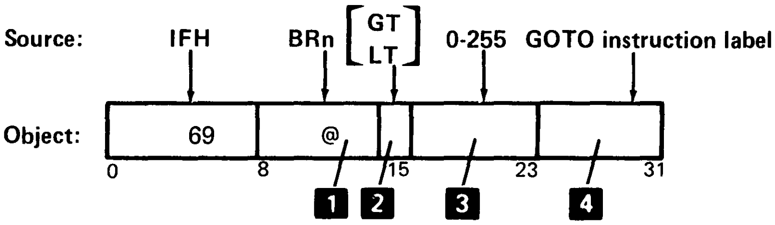
1 Test register: The address of the binary register that contains data to compare.

2 Compare constant: Hex 00-FF.

3 Branch instructions: The number minus 1 of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the low-order byte of the test register is not equal to the compare constant.

Compare Binary Half-Register for Greater or Less Than (IFH BRn GT/LT)



1 Test register: The address of the binary register that contains data to compare.

2 Bit 15:
0 = GT.
1 = LT.

3 Compare constant: Hex 00-FF

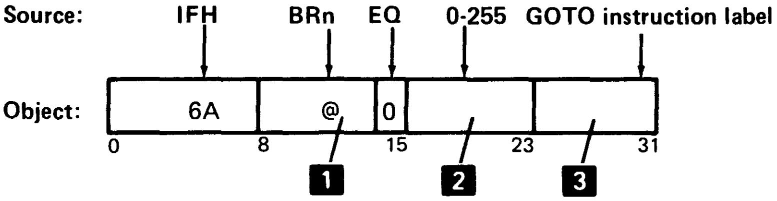
4 Branch instructions: The number minus 1 of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The content of the low-order byte of the test register is greater than the compare constant and GT is specified.

- The content of the low-order byte of the test register is less than the compare data and LT is specified.

Compare Binary Half-Register for Equal (IFH BRn EQ)



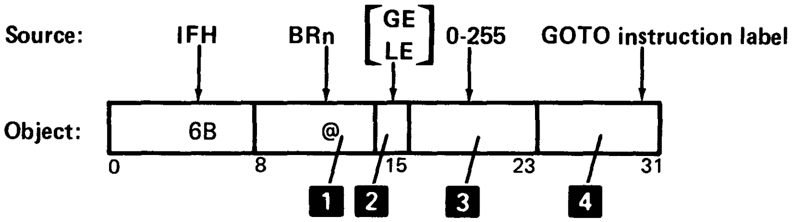
1 Test register: The address of the binary register that contains data to compare.

2 Compare constant: Hex 00-FF.

3 Branch instructions: The number minus 1 of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the low-order byte of the test register is equal to the compare constant.

Compare Binary Half-Register for Greater or Equal, Less or Equal (IFH BRn GE/LE)



1 Test register: The address of the binary register that contains data to compare.

2 Bit 15:
0 = GE.
1 = LE.

3 Compare constant: Hex 00-FF.

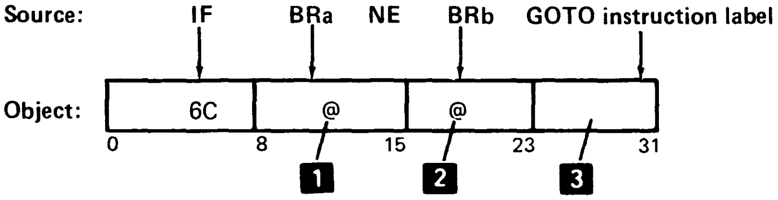
4 Branch instructions: The number minus 1 of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instruction from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The content of the low-order byte of the test register is greater than or equal to the compare data and GE is specified.

- The content of the low-order byte of the test register is less than or equal to the compare constant and LE is specified.

Compare Binary for Not Equal (IF BRn NE)



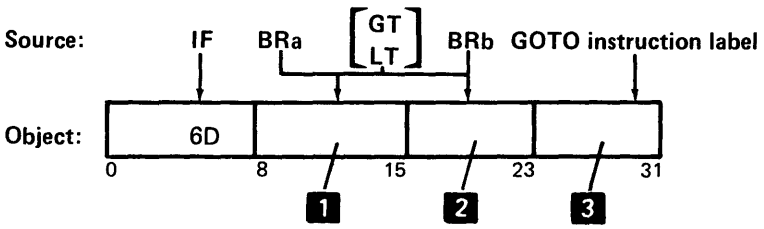
1 Test register: The address of the binary register that contains the data to compare.

2 Compare register: The address of the binary register that contains the compare data.

3 Branch instructions: The number, minus 1 of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the test register is not equal to the compare data.

Compare Binary for Greater or Less Than (IF BRn GT/LT)



1 *GT.* The test register: The address of the binary register that contains the data to compare.

LT. The compare register: The address of the binary register that contains the compare data.

2 *GT.* The compare register: The address of the binary register that contains the compare data.

LT. The test register: The address of the binary register that contains the data to compare.

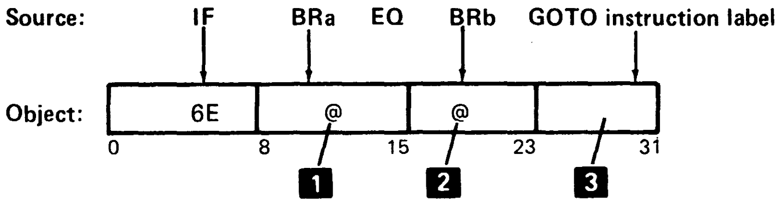
3 Branch instruction: The number minus 1 of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instructions from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The content of the test register is greater than the compare data and GT is specified.

- The content of the test register is less than the compare data and LT is specified.

Compare Binary for Equal (IF BRn EQ)



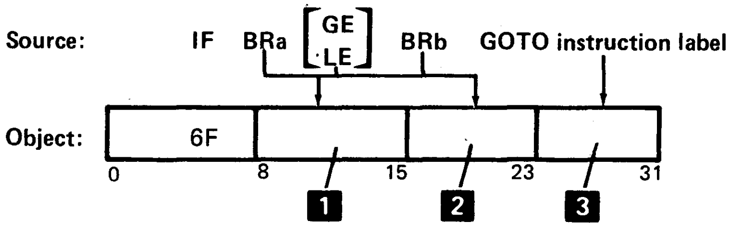
1 Test register address: The address of the binary register that contains the data to compare.

2 Compare register: The address of the binary register that contains the compare data.

3 Branch instructions: The number minus 1 (hex 00-7F) of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instruction from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if the content of the test register is equal to the compare data.

Compare Binary for Greater or Equal, or Less or Equal (IF BRn GE/LE)



1 *GE.* The test register: The address of the binary register that contains the data to compare.

LE. The compare register: The address of the binary register that contains the compare data.

2 *GE.* The compare register: The address of the binary register that contains the compare data.

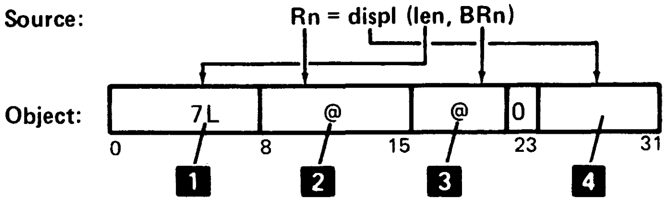
LE. The test register: The address of the binary register that contains the data to compare.

3 Branch instruction: The number minus 1 (hex 00-7F) of the 4-byte object code instructions from the next sequential instruction to skip if the branch is taken (±128 object code instruction from the next instruction). If bit 23 is 1, the number is a negative displacement in twos complement form.

The branch is taken if:

- The content of the test register is greater than or equal to the compare data and GE is specified.
- The content of the test register is less than or equal to the compare data and LE is specified.

Load Decimal Register from Base-Displacement Address (Rn = D(L, BRn))



1 Length: The number of bytes minus 1 (hex 0-F) of data to load.

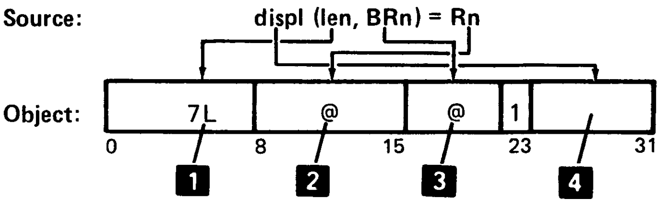
2 Load register address: The address of the decimal register where data is loaded.

3 Base address register: The address of the binary register that contains the base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the bytes to load begin.

The decimal load register is filled with blanks (hex 40s). Then the microprocessor adds the displacement (if any exists) to the base address register contents and loads the data at that address to the specified decimal register. The data is right-justified in the register.

Store Decimal Register into Base Displacement Address (D(L,BRn) = Rn)



1 Length: The number of bytes minus 1 (hex 0-F) of data to store.

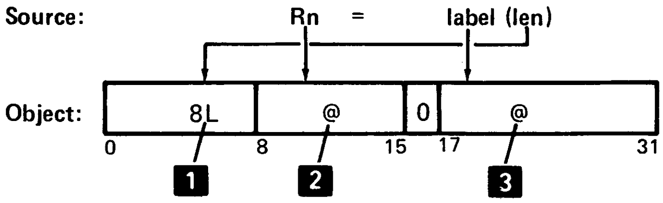
2 Store register address: The address of the decimal register where data is stored.

3 Base address register: The address of the binary register that contains the base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the bytes to store begin.

The microprocessor adds the displacement (if any is specified) to the base address register contents and stores the contents of the specified decimal register at that address. Data is taken from the rightmost bytes of the register.

Load Decimal Register from Labeled Storage (Rn = label(L))



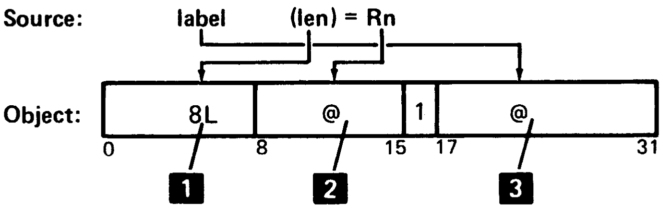
1 Length: The number of bytes minus 1 (hex 0-F) of data to load.

2 Load register address: The address of the decimal register where data is loaded.

3 Storage address: The storage address (hex 0000-7FFF) of data to load.

The microprocessor loads the specified decimal register with blanks (hex 40s), then loads it with data from the specified storage address. Data is right-justified in the register.

Store Decimal Register into Labeled Storage (label (len) = Rn)



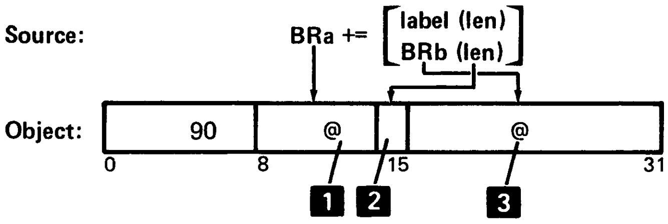
1 Length: The number of bytes minus 1 (hex 0-F) of data to store.

2 Load register address: The address of the decimal register where data is stored.

3 Storage address: The storage address (hex 0000-7FFF) of data to store.

The microprocessor stores the data in the specified decimal register at the specified storage address. Data is taken from the rightmost bytes of the register.

Binary Add (BRn +=)



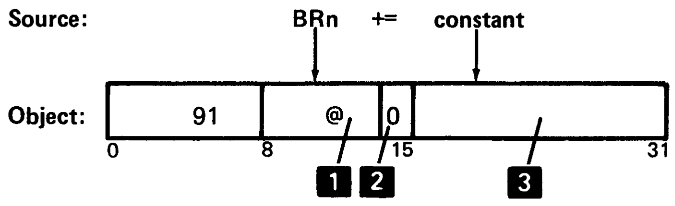
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 2 is added to factor 1, and the result is placed in the factor 1 register.

Binary Add Immediate Data (BRn +=)



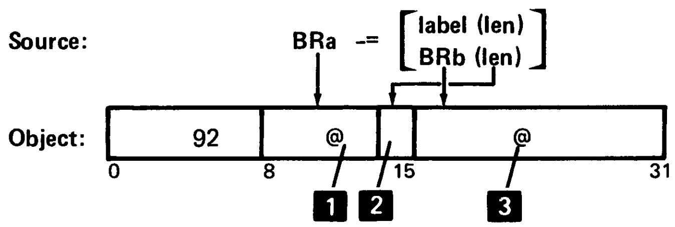
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

The factor 2 immediate data is added to factor 1, and the result is placed into the factor 1 register.

Binary Subtract (BRn -=)



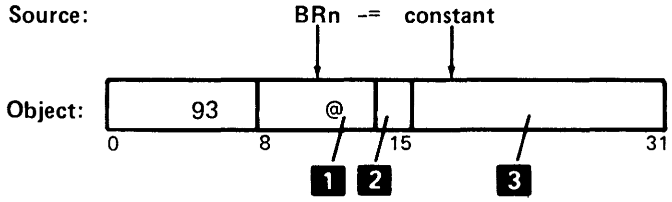
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 2 is logically subtracted from factor 1, and the result is placed in the factor 1 register.

Binary Subtract Immediate Data (BRn -=)



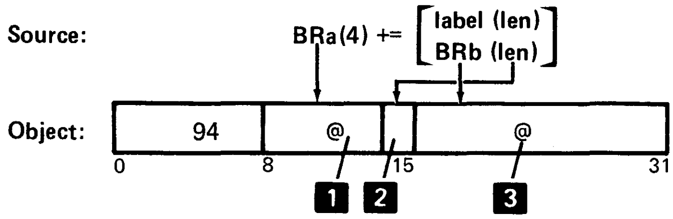
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

The factor 2 immediate is subtracted from factor 1, and the result is placed in the factor 1 register.

Binary Double Register Add (BRn(4) +=)



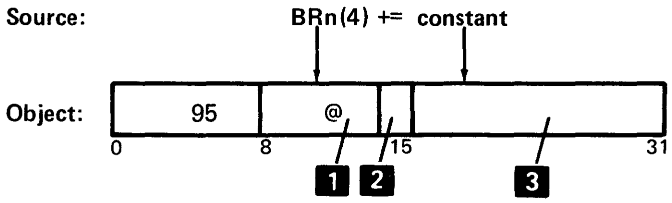
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 2 is added to factor 1, and the result is placed in the factor 1 register.

Binary Double Register Add Immediate Data (BRn(4) +=)



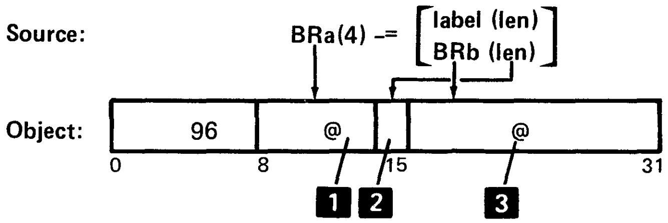
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

The factor 2 immediate data is added to factor 1, and the result is placed in the factor 1 register.

Binary Double Register Subtract (BRn(4) -=)



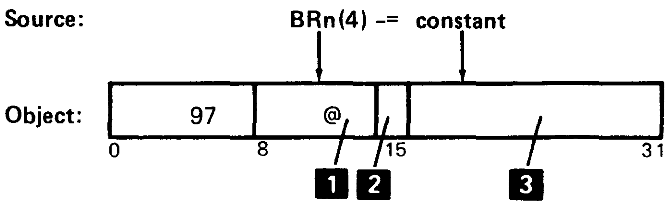
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 2 is logically subtracted from factor 1, and the result is placed in the factor 1 register.

Binary Double Register Subtract Immediate Data (BRn(4) -=)



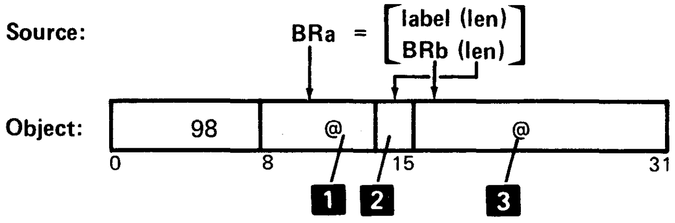
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

The factor 2 immediate data is subtracted from factor 1, and the result is placed in the factor 1 register.

Binary Register Load or Copy (BRn=)



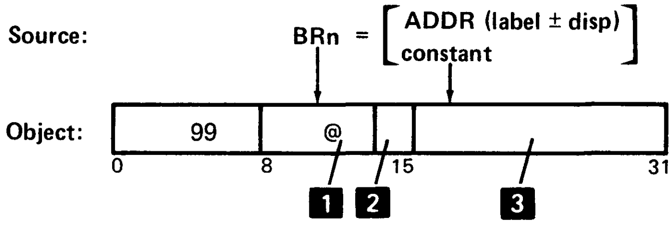
1 Result/factor 1: The address of the binary register that contains factor 1 and the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1. (The leftmost register is set to zeros, and the rightmost byte is loaded.)

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 1 is loaded with factor 2.

Binary Register Load Immediate Data or Address (BRn = C'II'/ADDR)



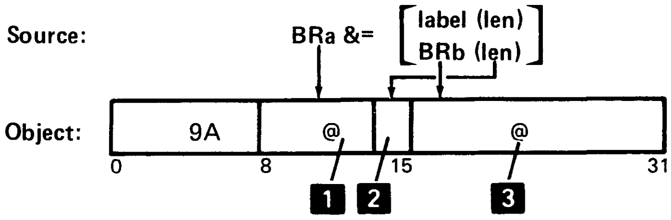
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, character constant, or a storage address.

The factor 2 constant or address is loaded into the factor 1 register.

Binary AND (BRn &=)



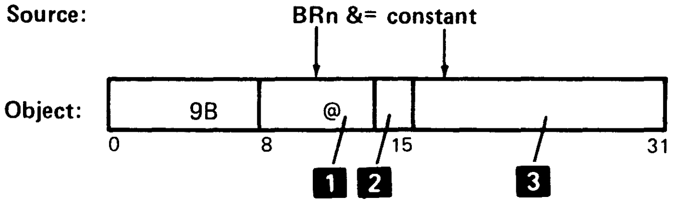
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1. (The leftmost byte of the register is set to zeros.)

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or the labeled (label) area that contains factor 2.

Factor 2 is logically ANDed with factor 1, and the result is placed in the factor 1 register.

Binary AND with Immediate Data (BRn &=)



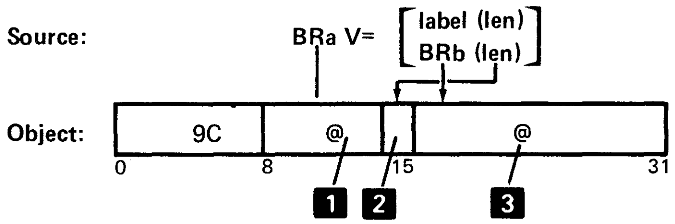
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

The factor 2 immediate data is ANDed with factor 1, and the result is placed in the factor 1 register.

Binary OR (BRn V=)



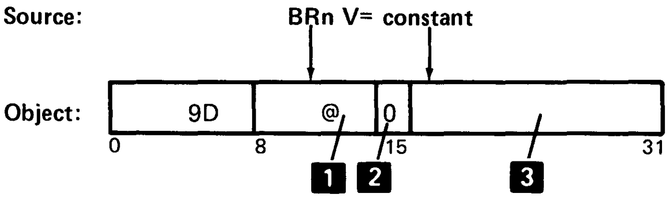
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 2 is logically ORed with factor 1, and the result is placed in the factor 1 register.

Binary OR with Immediate Data (BRn V=)



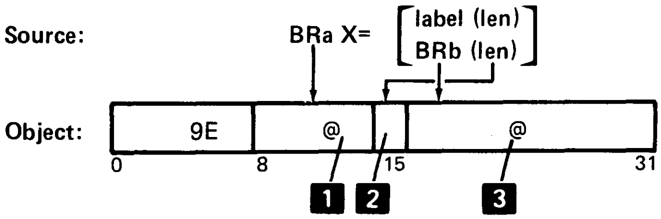
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

Factor 2 is logically ORed with the factor 1, and the result is placed in the factor 1 register.

Binary Exclusive OR (BRn X=)



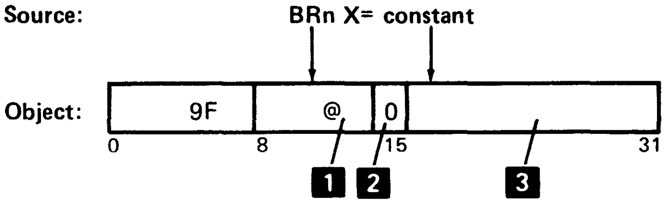
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 2 is logically exclusive ORed with factor 1, and the result is placed in the factor 1 register.

Binary Exclusive OR with Immediate Data (BRn X=)



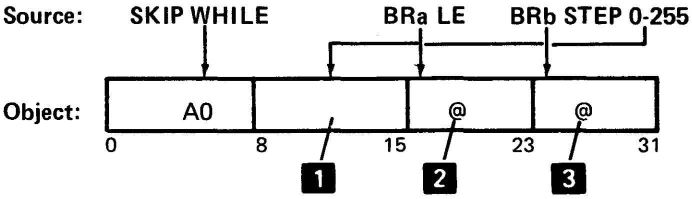
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15 = 0.

3 Factor 2: The binary representation (hex 0000-FFFF) of the binary, hex, decimal, or character constant.

Factor 2 is logically exclusive ORed with factor 1, and the result is placed in the factor 1 register.

Skip While Index Low or Equal Limit (SKIP WHILE)



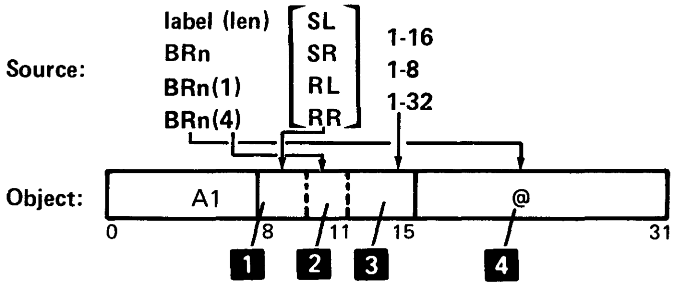
1 Increment value: The number (hex 00-FF) that is added to the contents of the test register.

2 Test register address: The address of the binary register that contains the value that is incremented and compared with the limit value.

3 Limit register address: The address of the binary register that contains limit value.

The increment value is added to the contents of the test register. The result is placed into the test register and then compared with the value in the limit register. If the value in the test register is less than or equal to the value in the limit register, the microprocessor skips the next sequential instruction.

Binary Register Shift or Rotate (SL/SR/RL/RR)



1 Shift or rotate, bits 8 and 9:
00 = SL (shift left)
10 = SR (shift right)
01 = RL (rotate left)
11 = RR (rotate right)

2 Register type, bits 10 and 11:
00 = Binary half-register (BRn(1)) of 1 byte
01 = Binary full register (BRn) of 2 bytes
10 = Binary double register (BRn(4)) of 4 bytes, with the high-order bit of the shift/rotate count = 0
11 = Binary double-register (BRn(4)) of 4 bytes, with the high-order bit of the shift/rotate count = 1

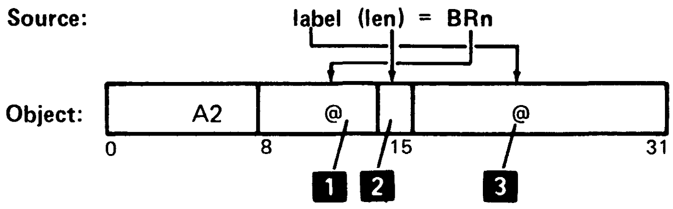
3 Shift or rotate count: For a full register, the number minus 1 (hex 0-F) of bits to shift/rotate. For a half register, the number minus 1 (hex 0-7) of bits to shift/rotate. For a double register, the low-order 4 bits of the number minus 1 (hex 00-1F) of bits to shift/rotate.

4 Result register address: The address of the binary register or labeled area that contains the data to shift/rotate and that will contain the shifted/rotated data.

The contents of the result register is shifted or rotated as specified. Shift operations move the contents of the register out of one end of the register and set the bits from which data was shifted to zero.

Rotate operations move the contents of the register out of one end and into the other end of the register.

Store Binary Register Contents (label = BRn)



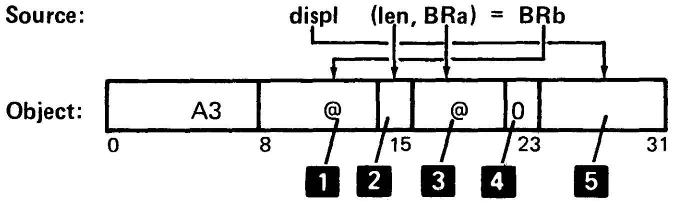
1 Binary register address: The address of the binary register that contains data to be stored at the storage address.

2 Storage location length:
0 = Storage location length is 2 bytes.
1 = Storage location length is 1 byte. (The rightmost byte of the binary register is stored.)

3 Storage address: The address of the storage location where the contents of the binary register are stored.

The contents of the binary register are stored in the labeled area.

Store Binary Register Contents, Indexed (D(L,BRa) = BRb(L))



1 Binary register address: The address of the binary register that contains data to be stored.

2 Storage location length:
0 = Storage location length is 2 bytes.
1 = Storage location length is 1 byte. (The low-order byte of the binary register, BRb, is stored.)

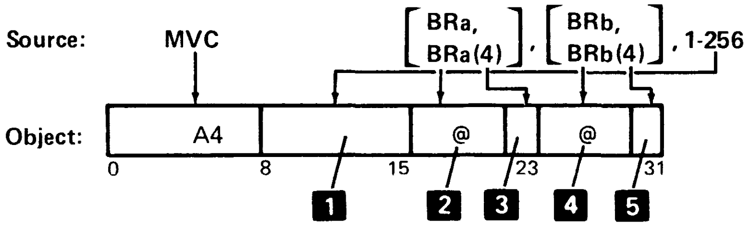
3 Base register address: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Displacement: The number of bytes (hex 00-FF) from the base address where the contents of the binary register are stored.

The displacement is added to the base address, and the contents of the binary register are stored in the resulting address.

Move Characters (MVC(BRn) / MVC(BRn(4))



1 Length of move: The number minus 1 (hex 00-FF), of bytes to move from register to register.

2 To register address: The address of the binary register (BRa), or the rightmost register of a double register (BRa(4)), that contains the address of the storage location into which data is moved.

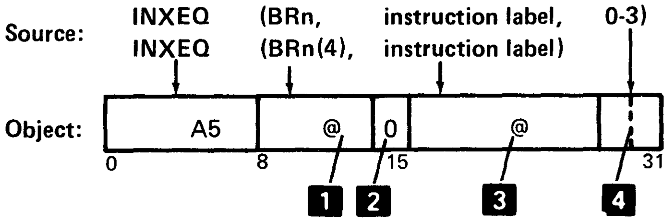
3 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

4 From register address: The address of the binary register (BRb), or the rightmost register of a double-register (BRb(4)), that contains the address of the storage location from which data is moved.

5 Addressing bit:
0 = BRb contains a 16-bit address.
1 = BRb(4) contains a 20-bit address of a storage location outside the partition.

The characters are moved from left to right, into the area specified. Either the to register (BRa) or the from register (BRb) must be a double binary register.

Indirect Instruction Execution (INXEQ)


Source:
INXEQ (BRn, instruction label, 0-3)
INXEQ (BRn(4), instruction label)
Object:
A5
@
0
@
0
8
15
31
1
2
3
4

1 Instruction modifier address: The address of the single binary register (BRn), or the leftmost register of a double register (BRn(4)), that contains the data needed to modify the instruction.

If a single binary register is specified, then the contents of the low-order byte of the 2-byte register are logically ORed with the contents of the specified byte of the instruction.

If a double binary register is specified, then the contents of all 4 bytes of the register are ORed with the contents of all 4 bytes of the instruction, except that bits 30 and 31 are ignored.

2 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

3 Instruction address: The address of the instruction to modify and execute.

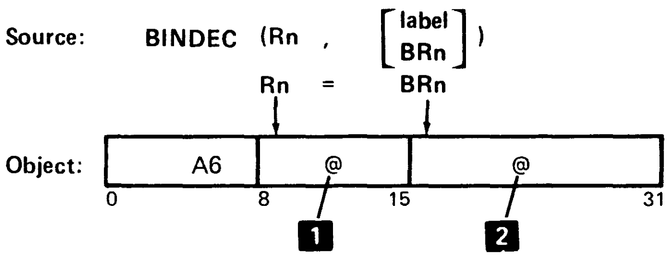
4 Instruction byte modifier, bits 30 and 31:
11 = Modify byte 0 of the instruction (op code)
00 = Modify byte 1 of the instruction
01 = Modify byte 2 of the instruction
10 = Modify byte 3 of the instruction

The specified instruction is modified as indicated, and then the modified instruction is executed. Control then returns to the instruction following the INXEQ instruction unless the modified instruction causes a branch. If a skip instruction is modified, and the modified instruction causes a skip, the instruction skipped is the instruction following the INXEQ instruction. The object code of the modified instruction is not changed.

If a short branch instruction is modified with INXEQ, the displacement is calculated from the INXEQ instruction rather than from the branch instruction. No additional validity for valid addresses is made with the INXEQ instruction.

If an INXEQ instruction is in the common area, the executed instruction is also in the common area.

Convert Binary to Decimal (Rn = BRn or BINDEC)

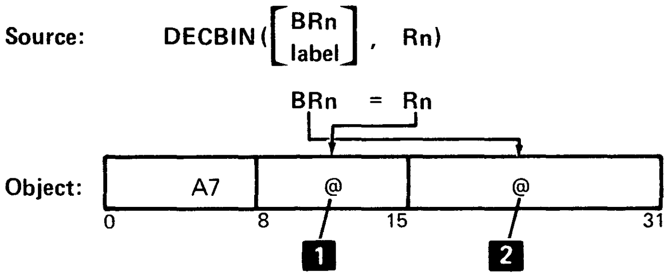


1 Decimal register address: The address of the decimal register that will contain the result of the binary to decimal conversion.

2 Binary register address: The address of the binary register or labeled area that contains the data to convert to decimal.

The contents of the binary register or labeled area are converted to decimal and placed into the decimal register.

Convert Decimal to Binary (BRn = Rn or DECBIN)

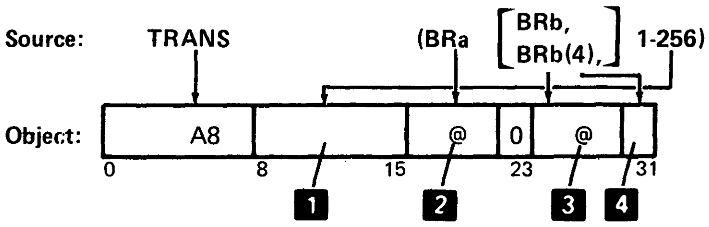


1 Decimal register address: The address of the decimal register that contains the data to convert to binary.

2 Binary register address: The address of the binary register or labeled area that will contain the result of the decimal to binary conversion.

The contents of the decimal register are converted to binary and placed into the binary register or labeled area.

Translate (TRANS)



1 Length: The number minus 1 (hex 00-FF) of bytes to translate.

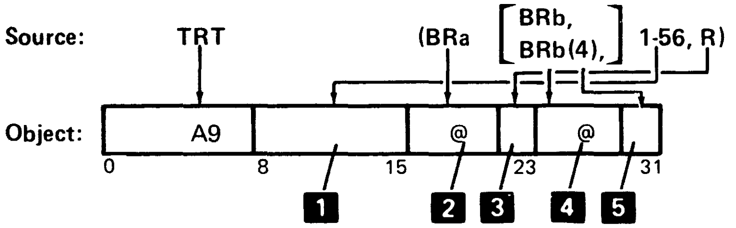
2 Data to translate address: The address of the binary register that contains the address of the data to translate.

3 Translate table address: The address of the binary register (BRb), or of the rightmost register of a double binary register (BRb(4)), that contains the translate table address.

4 Addressing bit:
0 = BRb contains a 16-bit address.
1 = BRb(4) contains a 20-bit address of a translate table outside the partition.

The data is translated, character by character, through the specified 256-byte translate table. The EBCDIC representation of the character is used as a displacement between 0 and 255 into the translate table. The character at that displacement into the translate table replaces the original character.

Translate and Test (TRT)



1 Length: The number minus 1 (hex 00-FF) of bytes to test.

2 Data to test address: The address of the binary register that contains the address of the data to test.

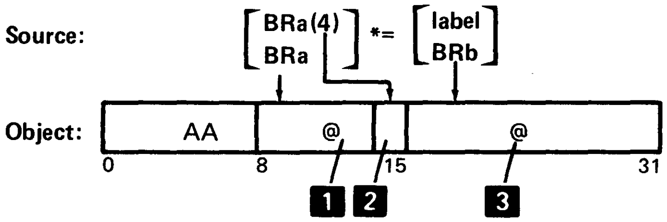
3 Scanning bit:
0 = Forward scanning (R not specified)
1 = Reverse scanning (R specified)

4 Translate table address: The address of the binary register (BRb), or of the rightmost register of a double binary register (BRb(4)), that contains the translate table address.

5 Addressing bit:
0 = BRb contains a 16-bit address.
1 = BRb(4) contains a 20-bit address of a translate table outside the partition.

The data is translated, character by character, through the specified 256-byte translate table. The EBCDIC representation of the character is used as a displacement between 0 and 255 into the translate table. If the character at that displacement in the translate table is zero, the next character is translated until the first nonzero translation is found or until all the characters have been tested. When the first nonzero translation is found, binary register BR16 is set to the address of the tested character, the low-order byte of binary register BR17 is set to the nonzero translate table entry, and the operation ends. If no nonzero translation is found BR16 and BR17 contain zeros when the operation is completed. The original characters are not changed.

Binary Multiply, Single or Double Register (BRn *= or BRn(4)*=)



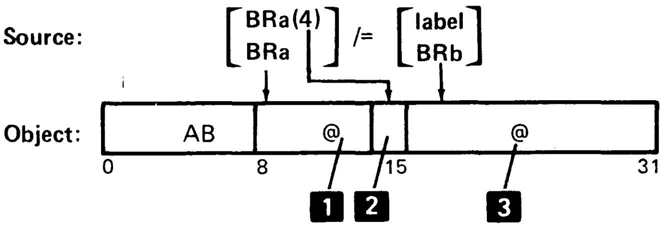
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Single register result.
1 = Double register result. (The result/factor 1 address is the address of the leftmost register.)

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 1 is multiplied by factor 2, and the result is placed in the factor 1 register. For a double register multiply, the first register contains factor 1 and both registers will contain the result.

Binary Divide, Single or Double Register (BRn /= or BRn(4) /=)



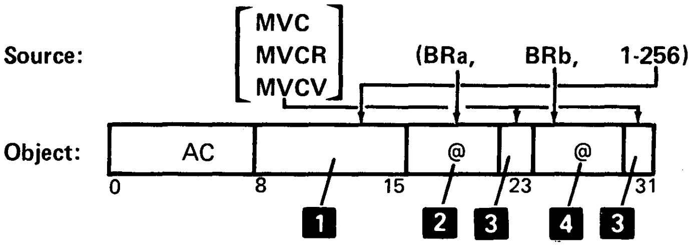
1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction. (Factor 1 is always 16 bits, even if a double binary register is specified.)

2 Bit 15:
0 = Single register result.
1 = Double register result. (The result/factor 1 address is the address of the leftmost register.)

3 Factor 2: The address of the leftmost byte of the binary register (BRb) or labeled (label) area that contains factor 2.

Factor 1 is divided by factor 2, and the result is placed in the factor 1 register. For a double register divide, the remainder is in the rightmost register, and the result is in the leftmost register. No remainder is provided unless a double binary register is used.

Move Characters Within a Partition (MVC/MVCR/MVCV)



1 Length: The number minus 1 (hex 00-FF) of bytes to move.

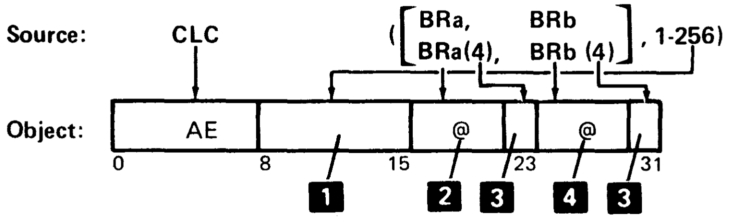
2 Move to address: The address of the binary register that contains the address of storage of where the data is moved to.

3 Bits 23 and 31:
00 = Move characters, left to right (MVC).
10 = Move characters, right to left (MVCR).
11 = Move characters, reverse fill (MVCV).

4 Move from address: The address of the binary register that contains the address of storage of where the data is moved from.

The characters are moved as specified from the from address to the to address.

Compare Character Strings (CLC)



1 Length: The number minus 1 (hex 00-FF) of bytes to compare.

2 Character string 1 address: The address of the single binary register (BRn), or of the rightmost register of a double binary register (BRn(4)), that contains the address of string 1.

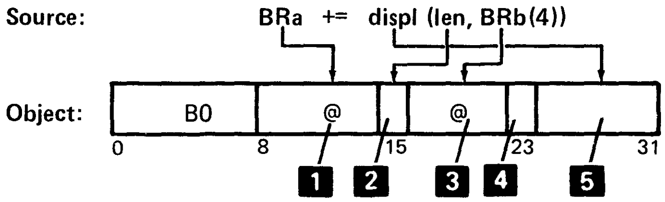
3 Bits 23 and 31:
0 = BRn contains a 16-bit address.
1 = BRn(4) contains a 20-bit address.

4 Character string 2 address: The address of the single binary register (BRn), or of the rightmost register of a double binary register (BRn(4)), that contains the address of string 2.

The microprocessor compares the two character strings and sets one of the following indicators on:

Indicator	Meaning
1101	Character string 1 is greater than character string 2.
1102	Character string 1 is less than character string 2.
1103	Character string 1 is equal to character string 2.

Binary Register Add with Base Displacement Address (+=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

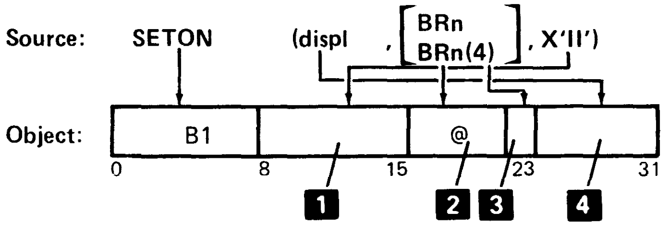
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor 2 displacement is added to the contents of the base register, then the data at the resulting address is logically added to factor 1.

Set Bits On with Mask (SETON)



1 Mask constant: A 1-byte constant to OR with the byte at the base displacement address.

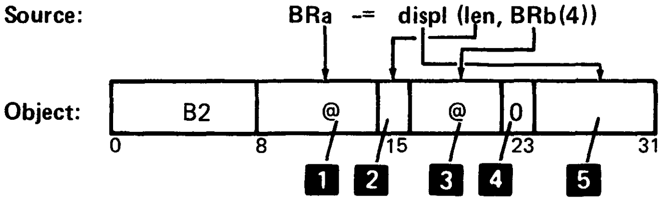
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit base address.
1 = BRn(4) contains a 20-bit base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the byte, with the bits to set on, is stored.

The displacement is added to the contents of the base address register, then the data at the resulting address is logically ORed with the mask constant. The result is stored at the original storage location.

Binary Register Subtract with a Displacement Address (-=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

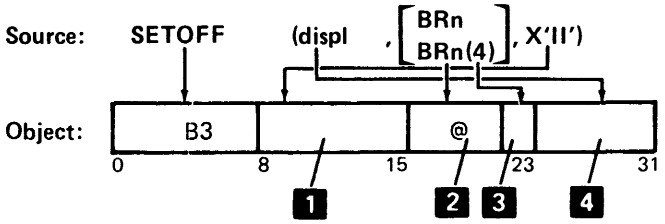
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

Factor 2 is subtracted from factor 1, and the result is placed in the factor 1 register.

Set Bits Off with Mask (SETOFF)



1 Mask constant: A 1-byte constant to convert to the ones complement, then AND with the byte at the base displacement address.

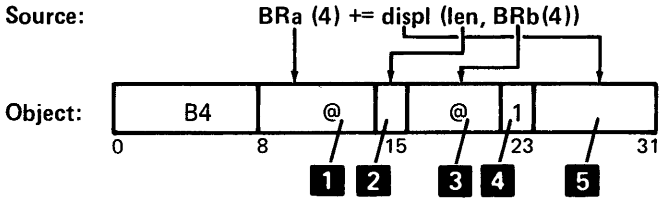
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit base address.
1 = BRn(4) contains a 20-bit base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the byte to mask is stored.

The displacement is added to the contents of the base address register, and then the data at the resulting address is logically ANDed with the ones complement of the mask constant. The result replaces the original data in storage.

Binary Double-Register Add with a Base Displacement Address (+=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

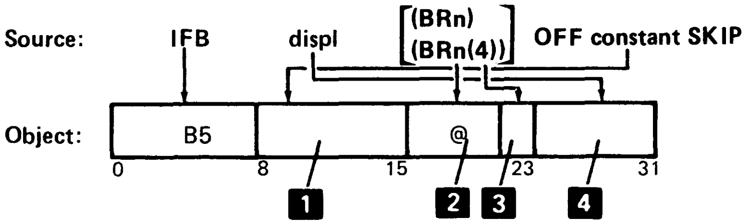
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor 2 displacement is added to the contents of the base register, then the data at the resulting address is logically added to factor 1.

Skip if Bits are OFF (IFB OFF)



1 Mask constant: A 1-byte constant that specifies the bits to test in the byte at the base displacement address.

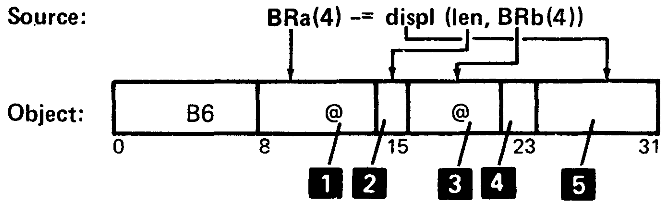
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit base address.
1 = BRn(4) contains a 20-bit base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the byte to test is stored.

The displacement is added to the contents of the base address register, then the ones complement of the data at the resulting address is tested with the mask constant. If any of the test bits are off, the next sequential instruction is skipped.

Binary Double Register Subtract with a Base Displacement Address (-=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

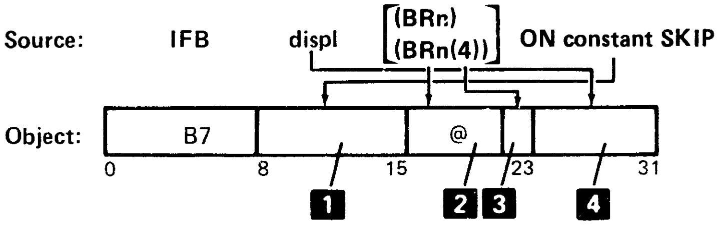
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor 2 displacement is subtracted from the contents of the base register, then the data at the resulting address is logically subtracted from factor 1.

Skip If Bits are On (IFB ON)



1 Mask constant: A 1-byte constant that specifies the bits to test in the byte at the base displacement address.

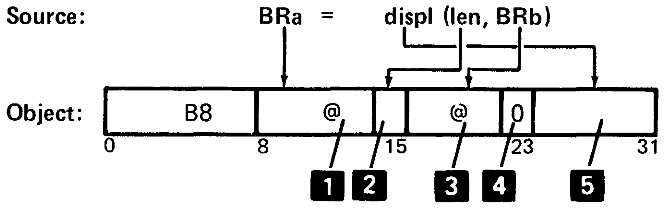
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit address.
1 = BRn(4) contains a 20-bit address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the byte to test is stored.

The displacement is added to the contents of the base address register, then the data at the resulting address is tested with the mask constant. If any of the test bits are on, the next sequential instruction is skipped.

Binary Register Load from a Base Displacement Address (=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2
1 = Length of factor 2 is 1.

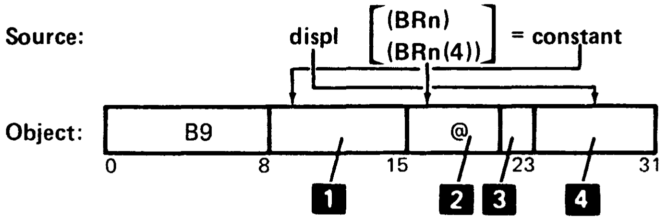
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor displacement is added to the base address register contents, and factor 2 is loaded from that address to the specified binary register.

Insert Constant Into a Base Displacement Address (= constant)



1 Constant: A 1-byte constant to insert into the base displacement address.

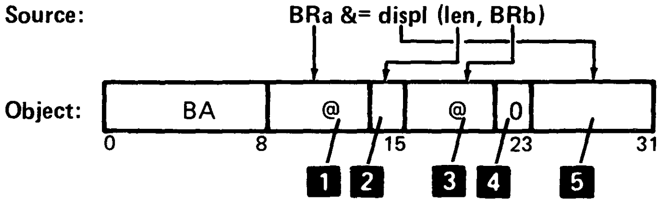
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit base address.
1 = BRn(4) contains a 20-bit base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the character is inserted.

The displacement is added to the base address, and the constant is loaded into the resulting address.

Binary Register AND with Base Displacement Address (&=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

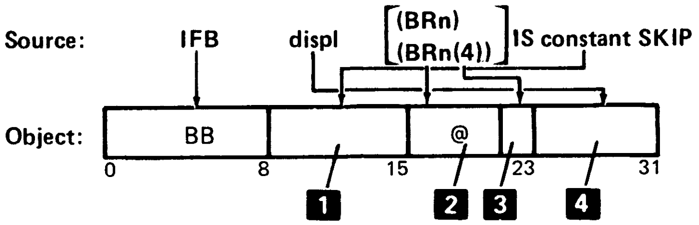
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor 2 displacement is added to the base address register contents, factor 1 is ANDed with the contents of the resulting address, and the result is placed into the factor 1 register.

Skip if Byte Equals Constant (IFB IS)



1 Constant: A 1-byte constant that is compared with the contents of the byte at the base displacement address.

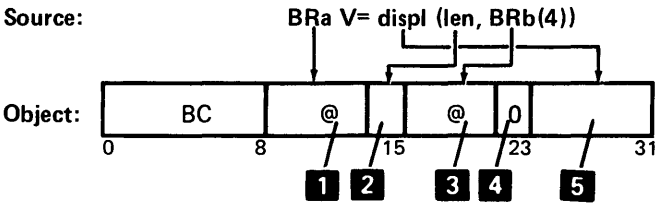
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit base address.
1 = BRn(4) contains a 20-bit base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the byte to compare with the constant is stored.

The displacement is added to the contents of the base address register, the contents of the resulting address is compared with the constant, and the next instruction is skipped if they are equal.

Binary Register OR with a Base Displacement Address (V=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

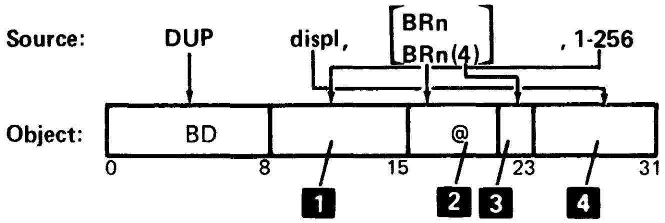
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor 2 displacement is added to the base address register contents, factor 1 is ORed with factor 2, and the result is placed in the factor 1 register.

Duplicate a Character at Base Displacement Address (DUP)



1 Length: The number minus 1 (hex 00-FF) of times to duplicate the byte at the base displacement address.

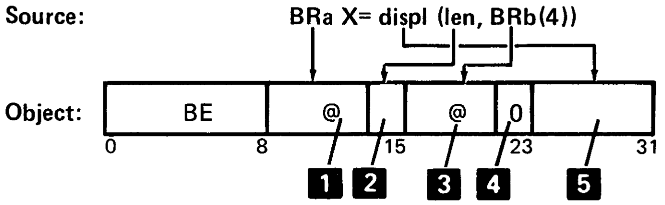
2 Base address register: The address of the single binary register (BRn), or of the rightmost register of a double register (BRn(4)), that contains the base address.

3 Addressing bit:
0 = BRn contains a 16-bit base address.
1 = BRn(4) contains a 20-bit base address.

4 Displacement: The number of bytes (hex 00-FF) from the base address where the byte to duplicate is stored.

The displacement is added to the contents of the base address register and the contents of the resulting address is duplicated into the succeeding bytes.

Binary Register Exclusive OR with a Base Displacement Address (X=)



1 Result/factor 1: The address of the binary register that contains factor 1 and will contain the result of this instruction.

2 Bit 15:
0 = Length of factor 2 is 2.
1 = Length of factor 2 is 1.

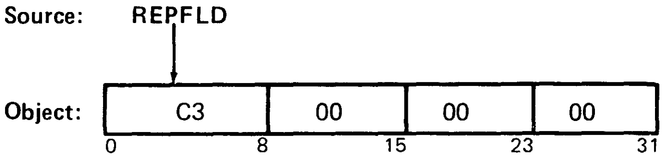
3 Base address register: The address of the binary register that contains the base address.

4 Address bit:
0 = BRa contains a 16-bit address.
1 = BRa(4) contains a 20-bit address of a storage location outside the partition.

5 Factor 2 displacement: The number of bytes (hex 00-FF) from the base address where factor 2 is stored.

The factor 2 displacement is added to the contents of the base address register, factor 1 is exclusively-ORed with the factor 2, and the result is placed in the factor 1 register.

Replace Field on Screen (REPFLD)



When the REPFLD instruction is executed, the main microprocessor does the following:

- Stores the keyboard operation code C3 and the operation parameters in the keyboard/display IOB starting at hex displacement 1F.

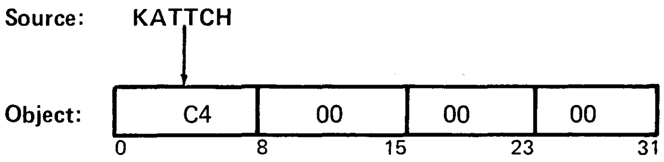
- Moves the contents of register BR19, BR20, and BR21 into the op code instruction to use as parameters. (During keyboard/display external status, BR19 holds the address of the current field in the I/O buffer; BR20 holds the address of the current field in the refresh buffer in keyboard/display storage; and BR21 holds the character set definition, the length minus 1 of the current field, and character set information about the last field processed.

- Notifies the keyboard/display microprocessor of the service request (keyboard operation). The keyboard/display microprocessor then moves the data, specified in the operation parameters, from main storage into the keyboard/display storage main refresh buffer. The bytes are translated through the display translate table; EBCDIC values between hex 20 and 2F are changed to hex 1F and displayed as solid rectangles. The codes in main storage remain unchanged.

- If the signed numeric bit is on in parameter 3 (from BR21) and the rightmost byte moved is D0-D9, a minus sign is displayed in the sign position of the field (to the right of the rightmost byte). If the rightmost byte is not D0-D9, a blank is displayed in the sign position.

  If the character set bits indicate a numeric only or digits only field and the signed numeric bit is not on, and if the rightmost byte moved is D0-D9, the negative graphic corresponding to the digit is displayed in the rightmost position of the field.

Keyboard Attach (KATTCH)

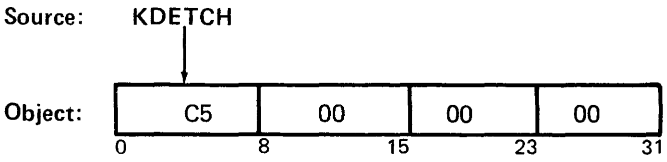


The KATTCH instruction provides temporary control of a keyboard/display unit, attaching the partition to its associated keyboard. This instruction is in effect until a KDETCH instruction is executed. If the attach is successful, the next sequential instruction is skipped.

This operation will fail if:

- There is an outstanding keystroke error
- There is an outstanding request for software error mode (KERRST)
- There is an outstanding ENTR
- Another partition is attached

Keyboard Detach (KDETCH)

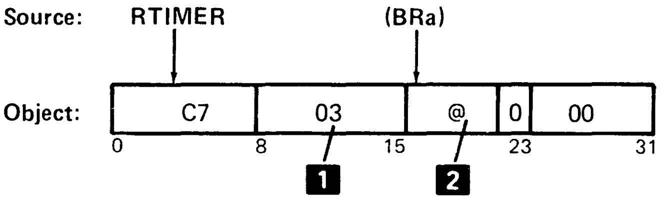


The KDETCH instruction detaches the keyboard/display unit from the current partition. If the detach is successful, the next sequential instruction is skipped.

This operation will fail if:

- There is an outstanding keystroke error
- There is an outstanding request for software error mode (KERRST)
- There is an outstanding ENTR

Read Elapsed Time Counter

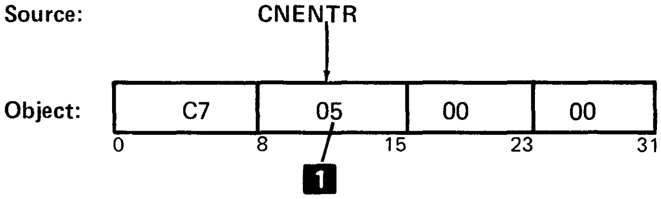


1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 To address: The address of the binary register that contains the main storage address where the timer value is to be stored.

This instruction stores the timer value into a 3-byte storage area. The high-order 2 bytes are taken from a 2-byte counter in the system control block (see *Elapsed Time Counter* in Chapter 1). These 2 bytes of the count indicate the number of 1.6 seconds that have elapsed since power on. The low-order byte is taken from a keyboard/display timer. Bits 0-3 of the low-order byte are always zero. Bits 4-7 indicate the number of tenths of a second since the last count indicated in the high-order 2 bytes.

Cancel Current Enter Command (CNENTR)

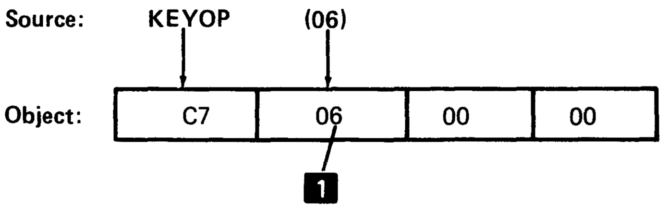


1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

This instruction cancels the current ENTR command. The end of screen format control string functions are performed, and data is no longer accepted from the keyboard. On the status line, the counters, insert mode symbol, keyboard shift, and hex display position are set to blanks. In the IOB, the command op code is set to zeros.

If this operation is issued in an external status subroutine during the processing of a nonoverlapped ENTR command, the return issued in the subroutine is made to the interrupted ENTR if the interrupted ENTR was not made complete by the external status condition. The ENTR is reissued and processing begins at the start of the screen format control string.

Release Character and Field Edits (KEYOP)



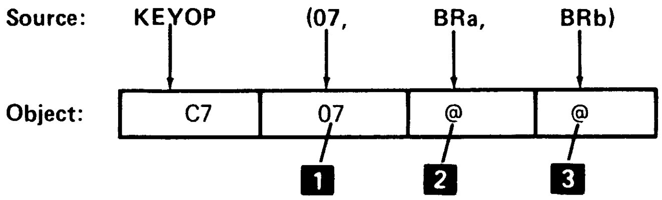
1 Keyboard operation number: The number that the main microprocessor stored in the keyboard/display IOB starting at hex displacement 1F.

The following character and field edit checks are discontinued for the current field:

- Character set check
- Data required
- Blank check
- Mandatory enter
- Mandatory fill

The checks are discontinued only until the field is exited in the forward or backward direction. If the same field is later advanced or backspaced into, the checks will be in effect.

Change Row Attribute (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Row: The address of the binary register that contains the low-order byte, the number of the row on the screen that is effected.

3 Masks: The address of the binary register that contains two 1-byte masks to be used for control information.

In keyboard/display storage, there is a 1-byte attribute specification for each row on the screen. This attribute specification determines how the row is displayed. The format of the attribute specification is as follows:

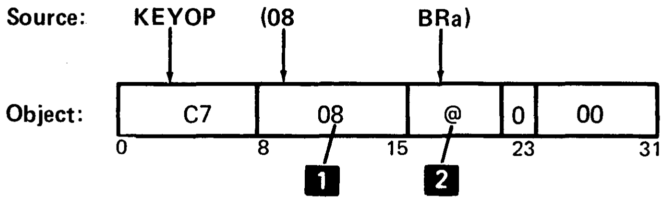
Bit	Meaning
0-1	01 = No system indicator 10 = Dash 11 = Solid rectangle
2	Valid row starting attribute. This bit must be 1 for bits 3-7 to be valid
3	Column separators are displayed
4	Blink the row[2]
5	Underscore the row[1,2]
6	High intensity[1]
7	Reverse image[1]

When this keyboard operation is executed, the attribute specification for the row is ANDed with the mask in the high-order byte of the binary register that holds the masks. The result of the AND is then exclusively-ORed with the mask in the low-order byte of the register. The attribute specified with bits 3-7 stays in effect until the next row starting attribute or character attribute.

[1] If bits 5, 6, and 7 equal 111, the display of the row is inhibited.

[2] These attributes remain in effect until any attribute is encountered.

Change Screen Position Pointer (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Screen position pointer modifier: The address of the binary register that contains the modifier.

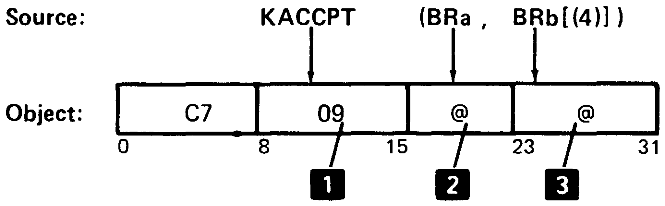
The contents of the screen position pointer are replaced with the modifier. The binary register that holds the modifier contains the row number in the leftmost byte and the column number in the rightmost byte.

If this operation is performed prior to an ENTR command and the format control string for the ENTR specifies that the format should be continued at the current screen position, the format will be initialized at the position specified by this operation rather than at row 2, column 1.

If this operation is performed during the processing of an ENTR command (for example, during an RG exit), all screen definitions such as fields and prompts encountered after this operation is executed originate from the position specified by this operation. The cursor is not moved over intervening fields and prompts; it causes them to be displaced on the screen.

Note: It is not recommended to use this operation during the processing of an ENTR. No checking is made on the specified screen position.

Accept Keystrokes and Store (KACCPT)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Length: The address of the binary register that contains the main storage address where the keystrokes are stored.

3 Length and options: If a 2-byte binary register is specified, this byte contains the address of the binary register. The register contains the information described in bytes 0 and 1 below. The keystrokes are not displayed as they are entered.

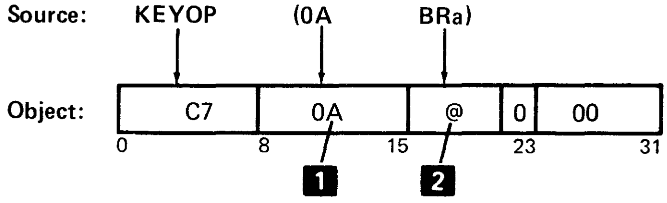
Byte	Bit	Meaning
0		Option Flags:
	0 = 1	The keyboard sounds a response click for each keystroke.
	1-4	Not used.
	5 = 1	The monocase function is enabled; keystrokes are converted to their uppercase equivalent as they are entered.
	6-7	Keyboard Shift Flags: 00 for Alpha shift. 01 for Num shift. 10 for Katakana shift. 11 is invalid.
1		Number minus 1 of keystrokes to accept.
2		Row number where keystroke display begins.
3		Column number where keystroke display begins.

The scan code and its EBCDIC translation are stored for each keystroke accepted from the keyboard. The codes are stored in pairs. For multiple keystrokes the scan code and EBCDIC are stored sequentially in the order they are entered.

The keystrokes are not applied to any outstanding ENTR command. If a shift key is pressed during this operation, the keyboard shift is changed but the scan code and EBCDIC for the shift key are not stored; the shift key does not effect the keystroke count. If a function key is pressed during this operation, the scan code and EBCDIC are stored but the function is not performed and external status does not result. If a command key sequence is entered during this operation, the codes are stored and external status does not result except if the Cmd key is followed by the C key. In this case, the codes for the Cmd key are stored and then the function for the Cmd, C key sequence is performed; the KACCPT operation is made complete regardless of the keystroke count.

The keyboard must be attached when this operation is performed.

Pass Scan Code to Keyboard (KEYOP)



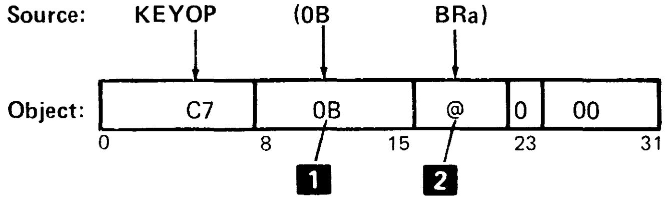
1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Scan code address: The address of the binary register that contains the main storage address of the scan code.

When this operation is executed, the specified scan code is passed to the keyboard/display associated with the partition. The scan code is processed as though it originated from the keyboard.

The keyboard must be attached when this operation is performed.

Pass EBCDIC to Keyboard (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

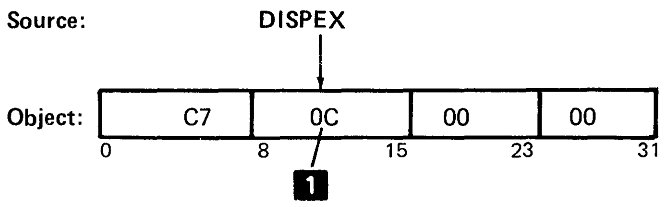
2 EBCDIC Code address: The address of the binary register that contains the ,main storage address of the EBCDIC code.

When this operation is executed, the specified scan code is passed to the keyboard/ display associated with the partition. If the EBCDIC corresponds to a data key or function key, it is processed as though it originated from the keyboard. The scan byte in the IOB is set to zeros.

Note: 29 (clear screen) and 2A (clear status line) are ignored because they are not function key EBCDICs. These functions can be performed with keyboard operation 11.

The keyboard must be attached when this operation is performed.

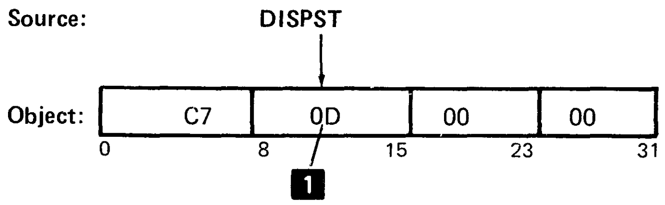
Display Extra Line (DISPEX)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display !OB starting at hex displacement 1F.

The instruction displays the extra line, replacing the display of the status line. The row starting address for the status line is set to the address of the extra line in the keyboard/display storage main refresh buffer area. The status line information is not available when using this instruction.

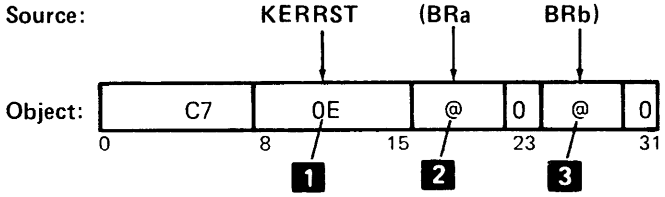
Display Status Line (DISPST)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

The instruction displays the status line, replacing the display of the extra line. The row starting address for the extra line is set to the address of the status line in the keyboard/display storage main refresh buffer area.

Request Keyboard Error Mode (KERRST)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/CRT IOB starting at hex displacement 1F.

2 Message: The address relative to the start of the partition of the binary register that contains the main storage address of the message to move to the status line refresh buffer.

3 Attribute mask and control information: The address of the binary register that contains the attribute mask in byte 0 and control information in byte 1.

Byte	Bit	Meaning if 1
0		Attribute Mask:
	0-2	Reserved
	3	Column separators displayed
	4	Blink
	5	Underscore[1]
	6	Highlight[1]
	7	Reverse image[1]
1		Control Information:
	0	0 = Do not check for display of status line. 1 = Display status line if it is not currently displayed.
	1	Start in column 1. (If bit 1 = 0, start in column 3.)
	2-7	Message length minus 1, up to 63. If 63 is specified, it indicates 0 bytes.

[1] If bits 5, 6, and 7 equal 111, data will not be displayed.

This operation places the keyboard in software error mode. When the keyboard/display is in software error mode, all data keys, function keys, and command key sequences are ignored. However, if the KEYOP instruction for operation hex 11 (perform keyboard function) is issued, the function is performed as long as the keyboard is in an appropriate state.

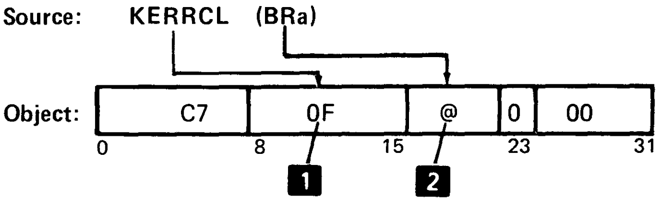
Bits 3-7 of the attribute mask are exclusively-ORed with bits 3-7 of the row attribute byte (which determines the display of the row) for the top row of the screen. If the status line is not displayed on the screen, the extra line will have the indicated attributes.

Bytes are moved from the address specified to the status line. The bytes are translated through the display translate table, and attributes are translated and passed. The bytes moved from storage overwrite the original status line data, and the original status line data destroyed.

If the status line is currently being displayed when this instruction is executed the indicated message is displayed in column 1 or column 3, according to byte 1, bit 1 of the control information. If the status line is not being displayed, the message is not displayed unless byte 1, bit 0 is 1.

This operation is invalid if the keyboard/display is already in software error mode, or if issued from an unattached background partition.

Reset Keyboard Error Mode (KERRCL)

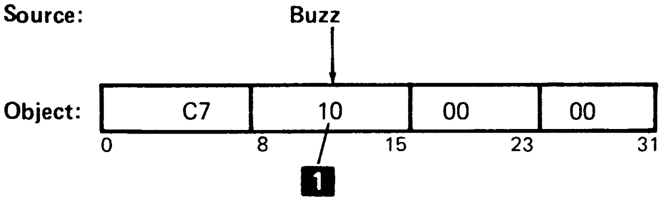


1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Attribute mask and control information: The address of the binary register that contains the attribute mask in byte 0 and control information in byte 1. See keyboard operation 0E for the format.

This operation takes the keyboard/display out of software error mode. It is valid only after a KERRST operation, and only when issued from an attached partition. When this operation is executed, if an ENTR command is outstanding and bits 2-7 of the control information do not equal zero, the field shift, hex display, current position counter, insert mode, and positions remaining in current field counter are restored in the status line. An attribute change is allowed, as for KERRST. Bits 2-7 of the control information specify the number minus 1 of positions to replace with blanks when the KERRCL operation is executed.

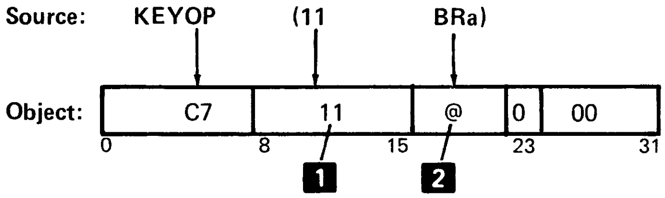
Sound Buzzer (BUZZ)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

This instruction sounds the alarm on the keyboard associated with the partition. The duration of the alarm is approximately 180 milliseconds.

Perform Keyboard Function (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Function address: The address of the binary register that contains, in the rightmost byte, the EBCDIC code for a function.

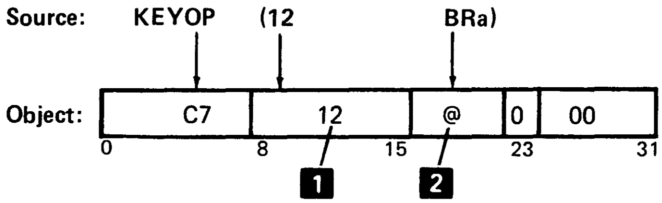
When this operation is executed, the function specified by the function EBCDIC is performed, with the following exceptions:

- The keyboard bit map is not checked to determine if the application program normally handles the function.

- If the keyboard is in software error mode, the function is executed if the keyboard is in an appropriate state. If the function is 29 (clear screen) or 2A (clear status line), the function is executed regardless of the state of the keyboard. If a function EBCDIC other than hex 01 through 2C is specified, an invalid operation external status condition occurs. The keyboard must be attached when this operation is performed.

See Appendix C for a list of the function codes.

Allocate Keyboard/Display Storage (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Length address: The address of the binary register that contains, in the rightmost byte, the length in K-bytes to allocate for each area, as follows:

Bits	Meaning
0-1	Number of K bytes (in binary) to allocate to section F (unit 1).
2-3	Number of K bytes (in binary) to allocate to section B (unit 2).
4-5	Number of K bytes (in binary) to allocate to section 7 (unit 3).
6-7	Number of K bytes (in binary) to allocate to section 3 (unit 4).

This instruction should be issued only at IPL time to allocate keyboard/display storage. The storage address range for one to three K bytes is as follows if the specified amount of storage is available:

Binary Specification	Number of K Bytes	Address Range in Keyboard/Display Storage
01	1	xC00 through xFFF
10	2	x800 through xFFF
11	3	x400 through xFFF

Where x is hex F, B, 7, or 3 for keyboard/display units 1, 2, 3, or 4 respectively.

If 3 K of storage is specified for a section and only 1 or 2 K is available, the storage is allocated beginning at x400. If 2 K of storage is specified for a section and only 1 K of storage is available, the storage is allocated beginning at x800.

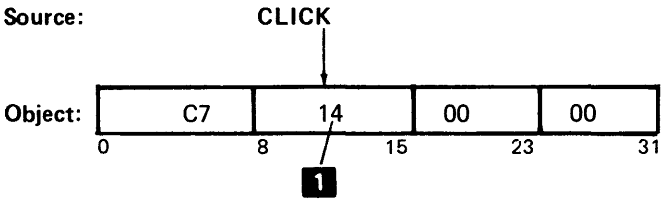
If the amount of storage specified for allocation to sections F, B, 7, and 3 is less than the total amount available, the remaining storage is allocated in section 0 starting at address hex 0000 to a maximum of 4 K bytes.

Notes:

1. Certain 5280 models have keyboard/display storage that is not dynamically allocatable. On these models, execution of this instruction does not change the storage allocation.
2. Regardless of how the allocation is specified, the hardware will not allocate storage in a section if that section does not have a corresponding display attachment.

See Chapter 3 for more information about keyboard/display storage.

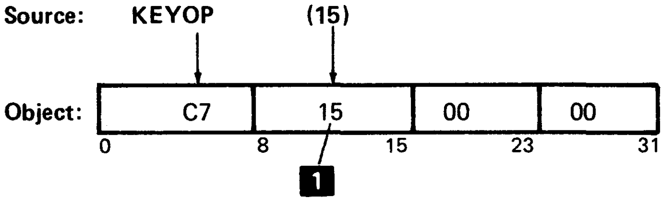
Click Keyboard (CLICK)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

This instruction clicks the keyboard associated with the partition.

Open Keyboard/Display (KEYOP)



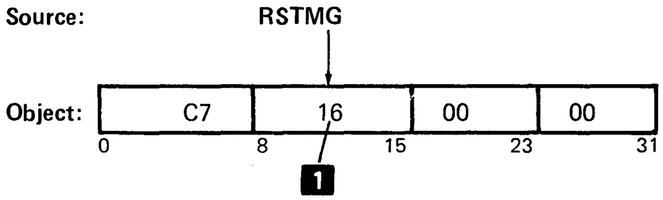
1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

This instruction initializes the keyboard/display unit.

- The clear screen function (29) is performed.
- The clear status line function is performed.
- The cursor is erased from the screen.
- The blink attribute for the top line displayed on the screen is cleared unless a keystroke error or software error mode is outstanding.

This operation is performed automatically during a load operation; it should not normally be issued by an application program. If this operation is issued from an unattached partition, an external status condition for invalid operation occurs.

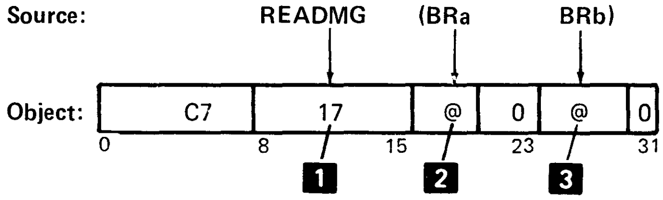
Reset Magnetic Stripe Reader (RSTMG)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

This instruction resets the magnetic stripe reader to read data from a badge.

Read Magnetic Stripe Reader (READMG)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 To address: The address of the binary register that contains the main storage address within the partition where data is read into from the magnetic stripe reader buffer.

3 Length: The address of the binary register that contains the number, minus 1, of bytes to read.

When a badge is inserted into a magnetic stripe reader, the badge characters are read into a buffer in the reader. External status condition 11 occurs in the partition associated with the reader. After badge data is read into the buffer, no other badge data is accepted until the buffer data is read with the READMG instruction or until the reader is reset with the RSTMG instruction. After the execution of the READMG instruction, the reader is automatically reset to enable the reader to accept another badge.

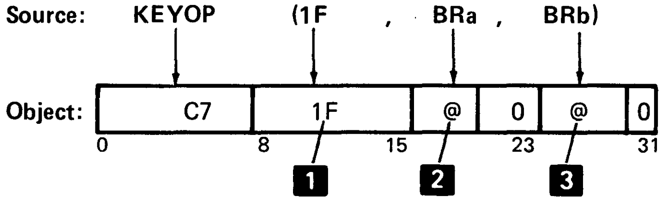
The magnetic stripe data consists of a string of from 3 to 128 characters. The first character must be a start of message (SOM) control character. The next-to-the-last character must be an end-of-message (EOM) control character. The last character must be a longitudinal redundancy check (LRC) control character of even parity for the entire data group. Any character can be placed in the other positions except an EOM character.

The reader control characters and data characters are as follows:

Bits	**Meaning if 1**
0	Device flag: A magnetic stripe reader is installed on the system.
1	Error flag: One of the following conditions has occurred: ● Parity error ● LRC error ● EOM missing ● Improper badge insertion or removal ● Speed error ● Buffer address overflow
2	LRC control character.
3	Parity bit: Odd parity for bits 4-7.
4-7	Data or control character: If hex 0 through 9, a data character. If hex B, a SOM control character. If hex F, an EOM character.

If any byte has an error, the error flag is set in all bytes.

Device Control Read (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Command address: The address of the binary register that contains, in the low-order byte, the attachment command.

3 To address: The address of the binary register that contains the main storage address where external register 5 contents are stored.

When this instruction is issued, external register 5 is set to 0, the command is loaded into external register 13, and external register 5 contents are sorted at the main storage address specified. (For some EAR commands the IOD contents remain unchanged.) The command must be one of the following:

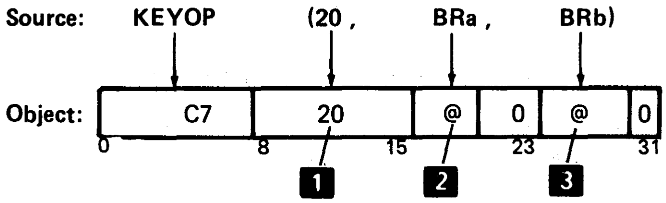
Hex Value	Command
41, C1, 45, C5	Read keyboard data
51, D1, 55, D5	Read keyboard status
61, E1, 65, E5	Activate keyboard click
43, C3, 47, C7	Activate keyboard buzz
49, C9, 4D, CD	Magnetic strip read data
69, E9, 6D, ED	Magnetic strip error reset
4C, CC	Read extended sense register
4E	Read interval timer
7A	Read mar hi
6A	Read mar lo
CA, EA, DA, FA	Enable translation
4B, 5B, CB	Keyboard/display storage read
FF	Power on reset
4F, CF	Read error sense

This operation should be issued *only* for diagnostic programming in a dedicated mode. Following this operation, the keyboard/display microprocessor ignores all keystrokes, the internal timer, parity errors, and the extended sense register until an ENTR command or keyboard operation other than 1F, 20, 21, or 22 is executed. When a diagnostic operation is issued, there is a change in external status 13 before the operation is executed. The contents of external register 13 depends on the keyboard/operation is executed. The contents of external register 13 depends on the keyboard/display unit associated with the partition that issued the instruction as follows: for unit 1, 40; for unit 2, C0; for unit 3, 44; for unit 4, C4. Register 25 and bit 7 of register 26 should not be altered by the application program.

Notes:

1. The external registers are used by the microprocessors; external registers are not binary or decimal registers located within a main storage partition.
2. The execution of a diagnostic operation will cause a change to external register 13 and bit 7 of external register 26 before the operation is performed.
3. External register 25 should not be altered because it is used to determine which partitions are serviced by the microprocessor.
4. Bit 7 of external register 26 should not be altered because it has status information required by the microprocessor.

Device Control Write (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Command address: The address of the binary register that contains, in the low-order byte, the attachment command.

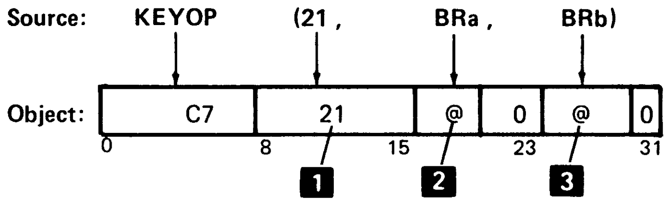
3 From address: The address of the binary register that contains the main storage address of data to write into the external register, XR5.

When this instruction is executed, external register 5 is loaded with the data at the specified main storage address, external register 13 is loaded with the command. The command must be one of the following:

Hex Value	Command
5A	Load mar hi
4A	Load mar lo
6B, 7B, EB	Keyboard/display storage write
48	Load configuration register
C8	Load sum register
5F	Load diagnostic control register

This operation should be issued *only* for diagnostic programming in a dedicated mode. Following this operation, the keyboard/display microprocessor ignores all keystrokes, the interval timer, parity errors, and the extended sense register until an ENTR command or keyboard operation other than 1F, 20, 21, or 22 is executed. When a diagnostic operation is issued, there is a change in external register 13 before the operation is executed. The contents of external register 13 depends on the keyboard/display unit associated with the partition that issued the instruction, as follows: for unit 1, 40; for unit 2, C0; for unit 3, 44; for unit 4, C4. Register 25 and bit 7 of register 26 should not be altered by the application program. See notes under keyboard operation hex 1F.

Keyboard/Display External Register Read (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Register address: The address of the binary register that contains, in the low-order byte, the external register to read.

3 To address: The address of the binary register that contains the main storage address to which the contents of the external register is read.

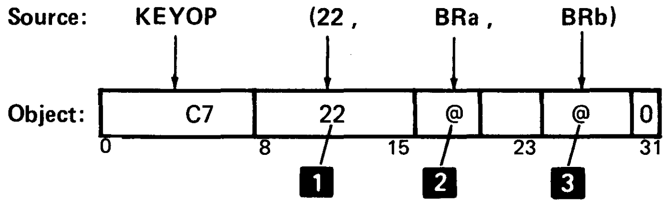
The contents of the low-order byte of the binary register 2 indicates the external register to read into the main storage address, as follows:

00 = External registers 5, 13, 25, and 26
01 = External register 5
02 = External register 13
03 = External register 25
04 = External register 26

This operation should be issued *only* for diagnostic programming in a dedicated mode. Following this operation, the keyboard/display microprocessor ignores all keystrokes, the internal timer, parity errors, and the extended sense register until an ENTR command or keyboard operation other than 1F, 20, or 22 is executed. When a diagnostic operation is issued, there is a change in external status 13 before the operation is executed. The contents of external register 13 depends on the keyboard/display unit associated with the partition that issued the instruction, as follows: for unit 1, 40; for unit 2, C0; for unit 3, 44; for unit 4, C4. Register 25 and bit 7 of register 26 should not be altered by the application program.

See notes under keyboard operation hex 17.

Keyboard/Display External Register Write (KEYOP)



1 Keyboard operation number: The number that the main microprocessor stores in the keyboard/display IOB starting at hex displacement 1F.

2 Register address: The address of the binary register that contains, in the low-order byte, the external register to write.

3 From address: The address of the binary register that contains the main storage address where the data to write into the external register is contained.

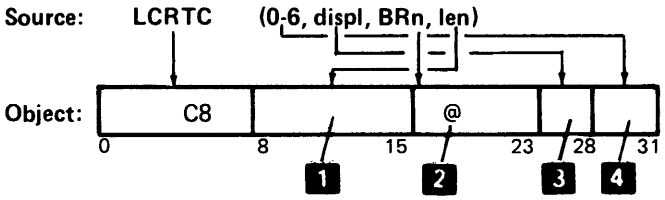
The contents of the main storage address are copied into the external register specified by the low-order byte of the binary register, as follows:

01 = External register 5
02 = External register 13
03 = External register 25
04 = External register 26

The operation should be issued *only* for diagnostic programming in a dedicated mode. Following this operation, the keyboard/display microprocessor ignores all keystrokes, the interval timer, parity errors, and the extended sense register until an ENTR command or keyboard operation other than 1F, 20, 21, or 22 is executed. When a diagnostic operation is issued, there is a change in external status 13 before the operation is executed. The contents of external register 13 depends on the keyboard/display unit associated with the partition that issued the instruction, as follows: for unit 1, 40; for unit 2, C0; for unit 3, 44; for unit 4, C4. Register 25 and bit 7 of register 26 should not be altered by the application program.

See the notes under keyboard operation hex 1F.

Load Keyboard/Display Control Area (LCRTC)



1 Length: The number minus 1 (hex 00-FF) of bytes to load into the keyboard/display area from main storage.

2 From address: The address of the binary register that contains the main storage address within the partition where data is moved from.

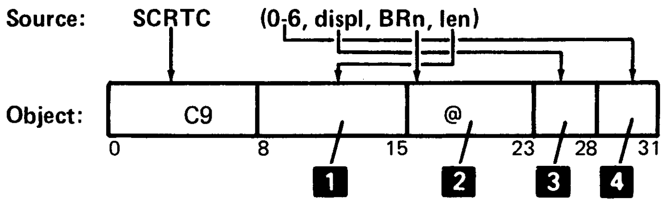
3 Displacement: The number of bytes, divided by 8 (hex 00-1F) into the keyboard/display control area where the loading of bytes begins.

4 Control area: The number (hex 0-6) of the control area to load. Control areas are defined as follows:

0 = Validity table
1 = Display control
2 = Storage area
3 = Scan code translate table
4 = Display translate
5 = Katakana translate
6 = Diacritic translate table

This instruction loads the specified storage area into keyboard/display storage. See Chapter 3 for a description of each area.

Store Keyboard/Display Control Area (SCRTC)



1 Length: The number minus 1 (hex 00-FF) of bytes to load into main storage from the keyboard/display area.

2 To address: The address of the binary register that contains the main storage address where data is stored.

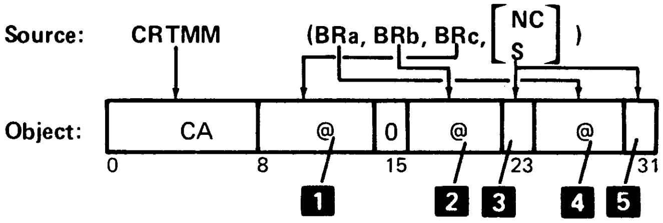
3 Displacement: The number of bytes, divided by 8 (hex 00-1F), into the keyboard/display control area where bytes are moved from.

4 Control area: The number (hex 0-6) of the control area to move bytes from. Control areas are defined as follows:

0 = Validity table
1 = Display control
2 = Storage area
3 = Scan code translate table
4 = Display translate
5 = Katakana translate
6 = Diacritic translate table

This instruction copies the specified storage area from keyboard/display storage to the main storage location specified.

Move Characters to Screen (CRTMM)



1 Length: The address of the binary register that contains the following:

Bit 0: 0 = BRa contains a screen row and column specification, with the row in the high-order byte and the column in the low-order byte.
1 = BRa contains an absolute address in keyboard/display storage. This specification is used for diagnostics.

Bits 1-15: The number minus 1 (hex 0000-7FFF) of bytes to move.

2 From address: The address of the binary register that contains the main storage address where data is moved from.

3 Bit:
0 = S is omitted.
1 = S is specified.

4 To address: The address of the binary register that contains the screen row and column, or the absolute address of keyboard/display storage to which data is moved.

5 Bit:
0 = NC is specified.
1 = NC is omitted.

The bytes are moved from main storage to the specified location. If the location is an absolute address, no checking is done to ensure that it is a valid address. If the location is a row and column, and if the move would extend into the keyboard/display control area (starting at XEA0), or if the column specification is 0, an external status for invalid operation occurs. If the move extends out of the refresh buffer and not into the control area, no external status occurs. No checking is done to assure that the move does not extend into tables stored in the keyboard/display storage, or into the refresh buffer for another screen.

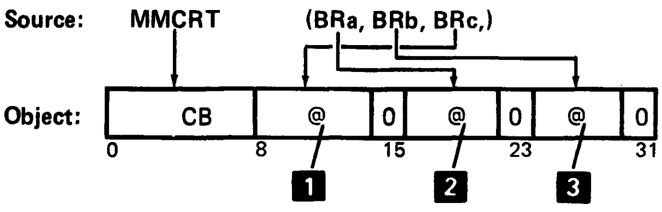
If NC and S are omitted, the bytes are translated through the display translate table before being placed in the refresh buffer. EBCDIC values from hex 20 through hex 2F are translated to display attributes and moved to the refresh buffer; the display attributes effect the display of the screen.

If NC is specified, the bytes are not translated through the display translate table before being placed into the refresh buffer.

If S is specified, the bytes are translated through the display translate table. However, EBCDIC values between hex 20 and 2F are changed to hex 1F and displayed as solid rectangles. The codes in main storage remain unchanged.

If row 0 is specified, the move is to the status line. If row 1 is specified, the move is to the extra line in the screen refresh buffer.

Move Characters from Screen (MMCRT)



1 Length: The address of the binary register that contains the following.

Bit 0:
0 = BRa contains a screen row and column specification with the row in the high-order byte and the column in the low-order byte.
1 = BRa contains an absolute address in keyboard/display storage. This specification is used for diagnostics.
Bits 1-15:
The number minus 1 (hex 0000-7FFF) of bytes to move.

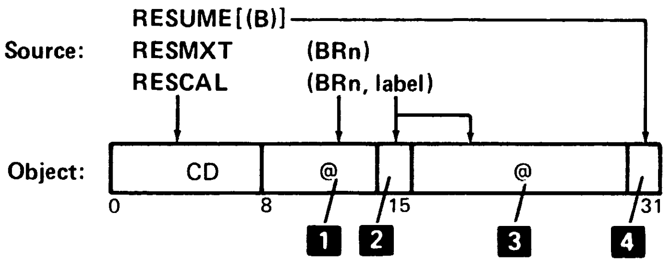
2 To address: The address of the binary register that contains the main storage address relative to the beginning of the partition to which data is moved.

3 From address: The address of the binary register that contains either the row and column, or the absolute address of keyboard/display storage where data is moved from.

The bytes are moved from keyboard/display storage to the main storage address within the partition. If the from address specifies row 0, the move is from the status line. If it specifies row 1, the move is from the extra line in the screen refresh buffer. If the from address specifies an absolute address that is outside the keyboard/display storage area, an external status for keyboard/display storage parity error occurs. If the from address specifies a row and column that extends into the keyboard/display control area (starting at XEA0), or if the column is 0, an external status for invalid operation occurs.

If the row and column specification extends into tables or another screen refresh buffer, no error occurs.

Resume Data Entry (RESUME/RESMXT/RESCAL)



1 Index address for RESMXT: The address of the binary register that contains the index for an indexed return.

Index address for RESCAL: The address of the binary register that contains either of the following:

- The index into the label table for a subroutine call.
- The index for an indexed subroutine call.

If BRn is not specified on either the RESMXT or the RESCAL instruction, the index address is all zeros.

If RESUME is specified, the index address is all zeros.

2 Bit 15:
0 = RESUME is specified.
1 = RESMXT is specified.
OR
0 = RESCAL is specified and the address at **3** is a subroutine address.
1 = RESCAL is specified and the address at **3** is a table address.

3 Table address or subroutine address for RESCAL. If RESUME or RESMXT is specified, this address is all zeros.

4 Bit 31 for RESUME:
0 = The cursor is repositioned forward (B is not specified).
1 = The cursor is repositioned backward (B is specified).

Bit 31 for RESMXT: 0.

Bit 31 for RESCAL: The last bit of either the table address or the subroutine address.

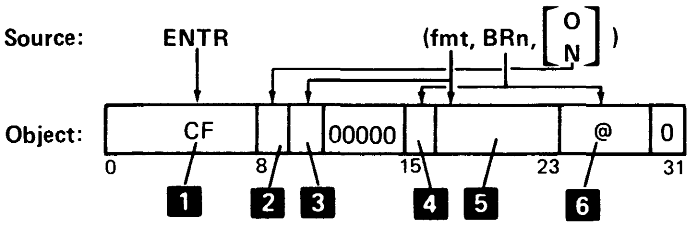
This instruction is included in external status subroutines to unlock the keyboard to allow key entry under the interrupted ENTR command.

For RESUME, the keyboard is unlocked and the interrupted ENTR is resumed. If B is coded, after an external status 04 or 05 condition the cursor is repositioned to the manual field preceding the RG specification in the screen format control string.

For RESCAL, the keyboard is unlocked and the interrupted ENTR is resumed. At the same time a subroutine is called and executed. If a label is specified with no binary register, the call is made to the label. If a subroutine label is specified with a binary register, the contents of the register are added to the subroutine with a binary register, the contents of the binary register are taken as an index into the label table. The call is made to the address in the label table at the index.

For RESMXT, the keyboard is unlocked and the interrupted ENTR is resumed. In addition, the external status bit in the IOB is turned off. The external status subroutine is terminated, BR18 is decremented by 2, and return is made to the address in the partition subroutine stack pointed to by BR18. If a binary register is included, the contents of the register are added to the address pointed to by BR18, and return is made to the resulting address.

Enter (ENTR)



1 Op code: The op code is stored in the command operation code byte of the keyboard/display IOB by the main microprocessor.

2 Bit 8:
0 = Overlap mode (O specified)
1 = Nonoverlap mode (N specified)

3 Bit 9:
0 = The format is contained in the partition.
1 = The format is contained in the common area.

4 Bit 15:
0 = The current record buffer address and the previous record buffer address are not alternated (BRn is not specified).
1 = The current record buffer address and the previous record buffer address are alternated (BRn is specified).

5 Format: The number (hex 00-FF) of the screen format to use.

Note: The source instruction specifies a label that the assembler converts to the number of the index where the format address is located in the screen format system table.

6 Previous record buffer address: The address of the binary register that the system loads with the address of the buffer that contains the previous record.

When the main microprocessor encounters an ENTR command, it places the command op code in the keyboard/display IOB. If the binary register is specified, the main microprocessor exchanges the contents of the current record buffer address and the previous record buffer address in the IOB, and places the address of the buffer that contains the previously entered record in the binary register. The main microprocessor places the screen format number into the keyboard/display IOB at hex displacement 09 and 0A. If overlapped I/O is specified, the main microprocessor continues executing instructions following the ENTR command while the keyboard/display microprocesor processes the screen format control string. If nonoverlapped I/O specified, the main microprocessor waits until the keyboard/display microprocessor has finished processing the screen format control string before it executes instructions following the ENTR command.

The keyboard/display microprocessor uses the screen format number in the keyboard/display IOB as an index into the screen format system table. If the system table is within the partition, the address of the system table is found in the keyboard/display IOB. If the system table is in the common area, the address of the system table is found in the system control block. The keyboard/display microprocessor takes the address at the index into the system table and stores it in the keyboard/display IOB at hex displacement 09 and 0A. While the keyboard/display microprocessor is processing the screen format control string, the address of the byte currently being processed is maintained in this IOB location. When the keyboard/display microprocessor finishes the control string or encounters a condition that requires the main microprocessor, it reports an external status condition.

This page intentionally left blank

Chapter 5. Diagnostic Aids

Diagnostic aids include the display/alter function and the dump and trace functions. In addition, several instructions are intended for diagnostic use. See Chapter 4 under *Device Control (DEVCTL),* opcode 3D, and under *Keyboard Operations (KEYOP),* opcode C7 for the following operations:

Keyboard Operation	**Description**
12	Allocate keyboard/display storage
1F	Device control read
20	Device control write
21	Keyboard/display external register read
22	Keyboard/display external register write

DISPLAY/ALTER FUNCTION

The display/alter function is a diagnostic tool that allows you to examine and alter the contents of main storage or keyboard/display storage, move the contents of main storage to keyboard/display storage and move the contents of keyboard/display storage to main storage.

You can use only keyboard 0 (the keyboard attached to partition 0) to execute display/alter functions; however, during power-on checkout, you can start display/alter from any keyboard. If you do start display/alter during power-on checkout, power-on checkout and IPL do not continue when you terminate display/alter.

To use the display/alter functions, the keyboard/display MPU must be operational. While you are using display/alter, no other keyboard/display operations can be performed. *Thus, you will hold up the rest of the system while using display/alter.*

Keyboard/display storage for keyboard 0 must be allocated to addresses FE00 through FFFF. Normally, keyboard/display storage is allocated by the configuration utility.

Other system conditions during display/alter functions are as follows:

- Magnetic stripe reader and elapsed time counter functions are not operational.
- Parity errors in main storage and keyboard/display storage are not detected.
- Status line data is removed from the secondary display of a dual display station.

How to Start the Display/Alter Function

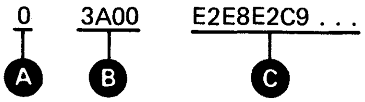
During Power-on Checkout and IPL: Press the L key on any keyboard while the cursor is moving through the power-on checkout display.

After Power-on Checkout and IPL: You must use keyboard 0; press the Cmd key, then the L key (the keyboard buzzes).

After you have started the display/alter function, if you are using a proof keyboard or a dual display, press and hold the shift key and press the C key (the keyboard buzzes). Release the shift key, then enter one of the following:

01—Nonproof keyboard and dual display
10—Proof keyboard and single display
11—Proof keyboard and a dual display

A line of data is displayed on the bottom of the screen as follows:



```
0   3A00   E2E8E2C9 . . .
```

A Main storage page number of data displayed, or an asterisk (*) if keyboard/display storage data pointer is displayed; set to 0 when the display/alter is started.

B Address of the first byte of data displayed. The address is set to 0000 when the display/alter is first started.

C Data: Displayed in eight 4-byte groups.

Pointers Maintained for Display/Alter

Three pointers are maintained for the display/alter function. The pointers indicate: (1) the main storage address (the address of the data being displayed on the screen), (2) the base address, and (3) the keyboard/display storage address.

How to Terminate Display/Alter

To terminate the display/alter function, press and hold the numeric shift key on the data entry/proof keyboard or the upper shift key on the typewriter keyboard and press the E key. Then press the Error Reset key.

How to Select and Use the Display/Alter Functions

When you use the display/alter functions on a typewriter keyboard, use the numeric key pad to enter digits 0 through 9.

To select a function, press and hold the Num (Numeric Shift) key on a data entry/proof keyboard or the ⇧ (Upper shift) key on the left of the typewriter keyboard, then press a key 0 through 9 or A through F to select the desired function. (When you select a function the keyboard buzzes.) Then release the shift key and enter the required parameters for the function (if parameters are required). If you press any key other than 0-9 or A-F, unpredictable results occur.

The following chart shows the options available with the display/alter function. Following the chart is a more complete description of several of the options.

Press this key to select the option.	**Parameters**	**Option Function**
0		Display main storage; display is not updated if data changes. See *Display Main Storage.*
2	aabbccdd	Search storage, where aabbccdd is the data to be found. See *Search Storage.*
3	dddd	Display the main storage at a displacement from the base address, where dddd is the displacement. See *Display Main Storage.*
·4	aaaa	Display main storage at a specified address, where aaaa is the address. Display is updated as main storage changes. See *Display Main Storage.*
5		Increment main storage address; the address in the main storage address pointer is incremented by 16, and the data at that address is displayed. See *To scan the main storage display* under *Display Main Storage.*
6		Decrement main storage address; the address in the main storage address pointer is decremented by 16, and the data at that address is displayed. See *To scan the main storage display* under *Display Main Storage.*
7	dd...	Alter main storage, where dd is one hexadecimal character to replace the data in main storage. See *Alter Main Storage.*
A	00	Test a byte; the keyboard buzzes when the byte at the current main storage address changes. See *Test for a Change in a Byte or a Bit.*
A	nnxx	Test a bit, where nn and xx are masks to test the byte. See *Test for a Change in a Byte or a Bit.*
B	1p0	Display the beginning of a partition, where p is the partition number. See *Display the Beginning of a Partition or of an IOB.*

Press this key to select the option.	Parameters	Option Function
B	1pd	Display the beginning of an IOB, where p is the partition number and d is the data set number. See *Display the Beginning of a Partition or of an IOB.*
B	0@@	Display the beginning of an IOB chain, where @@ is the low-order hexadecimal address of the IOB pointer. See *Display the Beginning of a Partition or of an IOB.*
C	pd	Accept keystrokes, where p=1 for a proof keyboard and p=0 for a typewriter or data entry keyboard, and d=1 for dual screen and d=0 for a single screen. See *How to Start the Display/Alter Function,* earlier in this section.
D	p	Set page number, where p is the page number to use for the current main storage address. See *To set the page number* under *Display Main Storage.*
E	Reset	Terminate display/alter. See *How to Terminate Display/Alter.*
F	0@@@@	Set keyboard/display address pointer, where @@@@ is the address. See *Move Keyboard/Display Storage.*
F	1	Display keyboard/display address. See *Move Keyboard/Display Storage.*
F	2	Display main storage address. See *Move Keyboard/Display Storage.*
F	3	Move keyboard/display storage to main storage. See *Move Keyboard/Display Storage.*
F	4	Move main storage to keyboard/display storage. See *Move Keyboard/Display Storage.*
F	5	Increment keyboard/display address, move to main storage, and display. See *Move Keyboard/Display Storage.*
F	6	Decrement keyboard/display address, move to main storage, and display. See *Move Keyboard/Display Storage.*

Display Main Storage

The display main storage function allows you to display main storage at a specified address or at a specified displacement from the base address. The base address can be set to the beginning of a partition or to the beginning of an IOB. When the display/alter function is first started, the base address is set to page 0 address 0000.

To display main storage at a specified address on the current 64 K page, press and hold the shift key and press the 4 key. Release the shift key and enter the hexadecimal address of the storage to be displayed. For example:

```
4 0100
```

causes 32 bytes of data to be displayed starting at the address 0100 within the current 64 K byte page.

Once the data is displayed, the system can alter the data at that location in main storage. The displayed data is updated to show the change until you press the shift key. If you do not want the display to reflect changes being made to the data, hold the shift key and press 0. Release the shift key when you want the display to stop changing.

To display main storage at a specified displacement from the base address, press and hold the shift key and press the 3 key, then enter the hexadecimal displacement value. For example:

```
3 0010
```

displays the data beginning with the base address plus 0010.

The display is updated to show any change in main storage data.

To set the page number of the current main storage address, press and hold the shift key and press the D key. Release the shift key and enter p, where p is the number of the 64 K byte page.

To scan the main storage display, press and hold the shift key and press the 5 key to scan forward or the 6 key to scan backward. Each time the 5 key is pressed the address of the displayed data is incremented 16 bytes. Each time the 6 key is pressed the address of the displayed data is decremented 16 bytes. If you hold down the 5 key or the 6 key, the address of the displayed data is automatically incremented or decremented until you release the key.

Alter Main Storage

The alter storage function allows you to alter main storage beginning at the currently displayed address. The format of the alter storage function is:

```
7dd
```

where dd is one hex character that replaces the character in main storage.

For each additional hex character entered, the storage position altered is automatically incremented one position. When 16 bytes have been altered, the displayed address is incremented 16 bytes.

Display the Beginning of a Partition or of an IOB

To display the beginning of a partition:

1. Press and hold the shift key and press the B key. Then release the shift key.

2. Enter:

   ```
   1p0
   ```

   where p is the number of the partition to be displayed. The beginning of the partition is displayed, and the base address is set to the beginning of the partition.

To display the beginning of a device IOB using an IOB pointer address:

1. Press and hold the shift key and press the B key. Then release the shift key.

2. Enter:

   ```
   0 @@
   ```

   where @@ is the low-order hexadecimal address of the device IOB pointer that points to the device IOB to be displayed.

The first IOB on the chain is displayed. You can increment through the chain by pressing the 0 key to display the next IOB on the chain.

To display the beginning of a device IOB using a data set number:

1. Press and hold the shift key and press the B key. Then release the shift key.

2. Enter:

   ```
   1pd
   ```

   where p is the number of the partition that contains the IOB chain to be displayed, and d is the data set number for the IOB to be displayed.

The base address is set to the beginning of the IOB currently displayed.

Move Keyboard/Display Storage

This function allows you to move data from keyboard/display storage to main storage or from main storage to keyboard/display storage. You can also display data moved from keyboard/display storage.

To start this function, press and hold the shift key and press the F key, then release the shift key and enter a number (0-6) to select the desired function as follows:

Number	Function
0@@@@	Set the keyboard/display address pointer, where @@@@ is the hexadecimal address to place in the pointer.
1	Display the current keyboard/display address set by the 0@@@@ function.
2	Display the'address in the main storage address pointer.
3	Move 32 bytes of keyboard/display storage data into main storage, using the addresses in the main storage address pointer and the keyboard/display address pointer.
4	Move 32 bytes of main storage data into keyboard/display storage, using the address in the main storage address pointer and the keyboard/display address pointer.
5	Increment the keyboard/display storage address by 16, move 32 bytes of keyboard/display storage data into main storage, and display the 32 bytes of data.
6	Decrement the keyboard/display address by 16, move 32 bytes of keyboard/display storage data into main storage, and display the 32 bytes of data.

Search Storage

The search storage function allows you to search storage for specified data. To start this function, press and hold the shift key and press the 2 key. Then enter the EBCDIC data to be found as follows:

aabbccdd

where aabbccdd is the data to be found.

The search begins with the address displayed and continues until the data is found or until 4 K (hex 1000) bytes of storage have been searched. If the data is found, the address of the first byte of the data is displayed along with 32 bytes of data beginning with the first byte. If the data is not found in the 4 K bytes of storage, the address displayed is incremented by hex 1000.

For example, assume the address displayed is 4000 and the data to be found is D4C8D7C8. If the data is found at address 4C00, the displayed result is:

0 4C00 D4C8D7C8 XXXXXXXX . . .

If the data is not found, the displayed address is:

0 5000 XXXXXXXX XXXXXXXX . . .

If you enter hex 00 for dd or for cc and dd, the hex 00 is not included in the search. For example, if you enter D4C80000, a match occurs when the data D4C8 is found.

Test for a Change in a Byte or a Bit

To determine when the value at the displayed address changes, press and hold the shift key and press the A key. Then enter 00. The keyboard buzzes the first time the data changes.

To determine when a bit(s) turns on or off in a byte at a specified address, press and hold the shift key and press the A key. Then enter the following:

nnxx (nn must not equal 00)

where nn is a byte to be logically ANDed with the byte at the displayed address, and xx is a byte to be logically exclusive ORed with the result of the AND operation. If the result of the operation is zero, the keyboard buzzes. (The byte is not changed in storage.) Thus, you can determine when a certain bit turns on or off. For example, to determine if bit 4 of the byte at the displayed address turns on, you would enter 0808 as shown below:

xxxx 1xxx	This bit is the bit to be tested.
0000 1000	AND the value with this byte.
0000 1000	This is the result of the AND operation if the bit is on.
0000 1000	Exclusive OR the result with this byte.
0000 0000	The keyboard buzzes when the result is zero to indicate that the bit did turn on.

To determine if the same bit goes off, you would enter 0800; that is, exclusive OR the result of the AND operation with 00.

DUMP AND TRACE CONSOLE FUNCTIONS

With the dump console function, you can dump any portion or all of main storage. There are two options with this function: (1) dump storage by page number, beginning address, and number of blocks, and (2) dump an entire partition by partition number.

With the trace console functions, you can trace program execution of specified instructions and display the results, or write the results to a diskette or printer. You can also display or write the contents of storage. You can use trace options to display information on the status line.

To write dump or trace output to a diskette or printer, a program must be loaded that sets up an IOB for data set 15. (When using trace options, data set 15 is not required unless the results are to be written to diskette or printer.) If you are using an assembler program, data set 15 must be set up by a .DATASET control statement. Following is an example of a .DATASET control statement, and .DC control statements required by the .DATASET statement:

```
.DATASET  LABEL=DMPTRIOB  DSN=15  DEV=X'8000' PBI=BUF256
          LBUF=BUF128  RECL=128  NAME=DTNAME  TYPE=SW,SHRW
          ELAB=ERRRTN;
.DC  LABEL=DTMANE    TYPE=STOR    LEN=9    INIT='DUMP0000';
.DC  LABEL=BUF256    TYPE=STOR    LEN=256  BDY=128;
.DC  LABEL=BUF128    TYPE=STOR    LEN=128;
```

If you are using a DE/RPG program, you can set up data set 15 by setting bit 0 in byte 0100, using the patch function. If you set this bit, you do not have to recompile; however, you must use a larger partition.

If the dump or trace data is to be written to a diskette data set, you must have allocated a data set that is large enough to contain the data to be dumped.

Before you start the dump or trace function, you must open data set 15 (if data set 15 is required for your output). Data set 15 can be opened by calling the CFDUMPTR routine. If you use CFDUMPTR the name of data set 15 must be DUMP0000. The CFDUMPTR routine is called by pressing the Cmd key, then shift, then the Dump Trace Open key when using one of the following programs:

SYSSORT
SYSMERGE
SYSCOPY
SYSPRINT
SYSKEU
DE/RPG program that has been modified as described above.

When the CFDUMPTR routine is called, the following prompt is displayed:

Dump/Trace file open
Enter device address:______

Enter the device address for the printer or diskette to indicate the destination of the dump or trace output, and press the Enter key. You can now use the dump or trace function, as described below.

Dump Function

To dump storage by page number, address, and number of blocks, do the following:

1. Press the Cmd key.
2. Press the C key.
3. Press the D key.
4. Enter the following to specify the main storage data to dump:

   P@@BB

   where:

   P = The page in main storage from which the data is dumped.
   @@ = The first 2 hex digits of the address (hex 00-FF) at which the dump begins (the low-order address is always hex 00).
   BB = The number (hex 00-FF) of 256-byte blocks of data to dump.

To dump the data from an entire partition, do the following:

1. Press the Cmd key.
2. Press the C key.
3. Press the P key.
4. Enter the partition number, 0 through 7.

When dumping to data set 15, data set 15 is not closed when the dump is completed until: (1) the partition is exited, or (2) the partition is reloaded if the partition exit operation calls the standard load processor, or (3) the application program explicitly closes the data set.

Trace Function

The trace function traces the execution of specified instructions and displays and/or writes the trace output after each instruction is executed. The trace output is in the following format:

```
P  @@@@  xx  R
```

where:

P = The partition number of the last instruction traced. If the last instruction traced is in the common function area of main storage, an asterisk is displayed in this field.

@@@@ = The relative address of the last instruction traced.

xx = The op code of the last instruction traced.

R = The result of the last instruction traced. The length of this field varies from 0 to 16 positions depending upon the type of instruction traced.

For branch instructions, the result field contains the address of the next instruction to execute if the branch is successful.

For table and binary instructions, the result field contains the specified binary result register.

For decimal instructions, the result field contains the specified decimal result register.

For all other instructions, the result field is blank.

To start the trace function, press the Cmd key, then the C key, then enter the following:

Tnss

where:

T = The uppercase letter T to select the trace function.

n = 0-To display trace output on the screen only.
4-To write trace output to data set 15.

s = 01-To trace branching instructions only.
02-To trace decimal arithmetic instructions only.
03-To trace branching instructions and decimal instructions.
04-To trace nonbranching instructions only.
05-To trace all instructions except binary and decimal.
06-To trace nonbranching instructions and decimal instructions.
07-To trace all instructions except binary instructions.
08-To trace binary instructions.
09-To trace branching instructions and binary instructions.
10-To trace decimal and binary instructions.
11-To trace decimal instructions, binary instructions, and branching instructions.
12-To trace binary instructions and nonbranching instructions.
13-To trace all instructions except decimal instructions.
14-To trace binary instructions, decimal instructions, and nonbranching instructions.
15-To trace all instructions.

Note: To cancel trace output to data set 15, specify trace output to the display screen only (option T0ss). This puts the trace program in address-stop mode. Then press the uppercase letter C to cancel address-stop mode. This does not close data set 15.

ADDRESS-STOP MODE

Address-stop mode causes the 5280 to stop executing program instructions when it reaches the instruction at a specified address. To start address-stop mode or alter the address-stop, enter the following:

A@@@@

where:

@@@@ = The address of the instruction at which to stop, relative to the start of the the partition.

When the 5280 stops program execution at the selected address, the following functions can be requested.

Main Storage Display

You can request main storage display only after setting single instruction mode by trace (T0ss) or after stopping on address-stop. When you request the main storage display, the contents of 16 bytes of main storage are displayed on the status line. The specified address, relative to the start of the partition, is displayed in positions 41 through 44 of the status line, followed by the byte at the selected address and the following 15 bytes. To select the main storage display, enter:

```
M@@@@
```

where:

@@@@ = The address of the first byte to display.

Forward Scroll

You can request the forward scroll function only after the main storage display function. The forward scroll function replaces the 16 bytes being displayed on the status line with the next sequential 16 bytes of main storage. To select forward scroll, enter the uppercase letter F.

Backward Scroll

You can request the backward scroll function after the main storage display function. The backward scroll function replaces the 16 bytes being displayed on the status line with the preceding 16 bytes of main storage. To select backward scroll, enter the uppercase letter B.

Replace Main Storage

During main storage display, you can replace the byte of storage at the address displayed in position 41-44 of the status line.

To replace the byte at the displayed address, enter:

```
Rdd
```

where:

dd = Two digits to store in main storage.

After the two digits are stored in the main storage byte, the address is incremented by one and the contents of the next 16 bytes are displayed on the status line.

Single Instruction

When you request the single instruction function, the 5280 executes the next instruction and stops. After the instruction is executed, trace information is displayed on the status line, beginning in position 41. To select the single instruction function, enter the uppercase letter S.

Loop

When you request the loop function, the 5280 executes the program instructions until it again reaches the original address-stop address. It stops at the address-stop address, and trace information is displayed on the status line, beginning in position 41. To select the loop function, enter the uppercase letter L.

Main Storage Dump

You can request a main storage dump function while the 5280 is operating in address-stop mode. The dump function is requested the same as a normal dump (Dp@@BB), that is, the contents of storage are written to data set 15. (See *Dump Function*.) When the dump is completed, you may continue with other address-stop mode functions. Data set 15 is not closed until the partition is exited, or until the partition is reloaded if the exit operation calls the standard load processor, or until the application program explicitly closes the data set.

Trace

You can request a trace while the 5280 is operating under address-stop mode. The trace function is requested the same as a normal trace (T4ss). (See *Trace Function*.) The 5280 executes the remaining program instructions and does not stop, but the trace information continues to be written to data set 15.

Cancel Address-Stop

You can cancel address-stop mode by keying an uppercase letter C. The 5280 will execute the remaining program instructions with no stops and no trace output.

This page intentionally left blank

Chapter 6. Keyboard Functions

Keyboard functions may be initiated by function keys or by program instructions. Each function is assigned an EBCDIC value between hex 00 and hex 3F. See Appendix C for a list of these EBCDIC values. When a keyboard function is initiated, the EBCDIC for that function is placed into the keyboard/display IOB, at relative address hex 47.

Certain functions are normally processed by the 5280, but may be processed by an application program subroutine. Other functions are always processed by the 5280, and others are always processed by an external status subroutine for external status condition 1. Many functions that are processed by the 5280 must first be enabled by the application program, which must set flags in the keyboard/display IOB.

KEYBOARD FUNCTION CONTROL

The 5280 performs automatic functions and maintains certain function control. The application program must enable the automatic functions by setting flags in the keyboard/display IOB. The keyboard function control flag bytes are maintained by both the 5280 and the application program. The flag bytes are located at relative address hex BE and BF, as follows:

Byte	**Bit**	**Meaning if 1**
X'BE'	0	Keyboard is in enter mode.
	1	Keyboard is in update mode.
	2	Keyboard is in rerun mode. (See BF, bit 6.)
	3	Keyboard is in verify mode.
	4	An application program must not change this bit.
	5	An application program must not change this bit.
	6	Keyboard is in display mode.
	7	Fixed prompts are not displayed.
X'BF'	0	*Modified data bit* is set to 0 by the 5280 when the current field is entered, and set to 1 by the 5280 if data is entered into the field. When the field is exited, the 5280 ORs this bit with the modified data indicator that is assigned to the field.
	1	An application program must not change this bit.
	2	An application program must not change this bit.

Byte	Bit	Meaning if 1
	3	*Auto-dup/skip enable* bit must be maintained by the application program. While this bit is 1, the 5280 automatically processes fields defined as auto-skip (AS) or auto-dup (AD). When this bit is 0, these fields are treated as manual fields. When this bit is 1, a field defined as main storage store (MS) is stored; otherwise, it is not stored.
	4	*Auto-enter enable* bit must be maintained by the application program. When the bit is 1, the 5280 automatically performs a record advance when the operator enters the last manual position of a record format. If bit is 0, the 5280 puts the keyboard in the awaiting record advance state after the operator enters the last position of the record.
	5	*Alternate record advance* bit must be maintained by the application program. If this bit is 1, when the operator presses the Rec Adv key the 5280 ignores all remaining fields and format specifications. If this bit equals 0, the 5280 processes the remaining fields and format specifications.
	6	*Rerun/display enable* bit must be maintained by the application program. This bit is 1 and the 5280 is processing in rerun mode, display is enabled.
	7	An application program must not change this bit.

FUNCTIONS NORMALLY HANDLED BY THE 5280

The following function descriptions detail how each function is initiated and how the 5280 processes the function. The function descriptions pertain to all modes of entry unless a mode is specifically mentioned. Most functions are processed differently for verify mode; the descriptions for verify mode follow the general descriptions of each function.

Alpha Shift Function

The alphabetic shift function is initiated when the operator presses the Alpha key. The Alpha key is on the data entry and proof keyboards, and is valid at all times. While the Alpha key is pressed, the lower character on the key top is selected for any data key.

Character Advance Function

The character advance function is initiated when the operator presses the → (Character Advance) key, and is valid only while an ENTR command is being processed.

In enter, update, special verify and field correct modes, when the → (Character Advance) key is pressed the cursor moves to the next position within the current field. If the → (Character Advance) key is pressed when the cursor is in the rightmost position of the field or when awaiting field advance, a field advance is performed. The contents of the positions the cursor moves through remain unchanged. If the character advance key is pressed when the cursor is in the last position of the record, an error occurs unless the auto-enter flag is on. If the auto-enter flag is on a record advance is performed.

In verify mode, the → (Character Advance) key is not valid except when the system is awaiting field exit or record advance, or when the cursor is in a position other than the rightmost position of a right-to-left field. If the system is awaiting field exit, a field advance is performed. If the system is awaiting record advance, an error occurs unless the auto-enter flag is on. If the flag is on, a record advance occurs. If the cursor is in any position other than the rightmost position of a right-to-left field, the → (Character Advance) key is processed as for enter mode except that any character advanced over is blanked on the screen and must be reverified before the field is exited.

Character Backspace Function

The character backspace function is initiated when the operator presses the ← (Character Backspace) key, and is valid only when an ENTR command is being processed.

In enter, update, and special verify modes, when the ← (Character Backspace) key is pressed the cursor normally moves back to the previous position within the field. If the system is awaiting field exit, the cursor remains in the same position but the awaiting field exit condition is cleared. The cursor stops blinking; the operator can enter another character into the position. If the system is awaiting record advance, the condition is cleared and the cursor is positioned to the last manual input position of the record. If the key is pressed when the cursor is in the leftmost position of a field, the cursor moves to the rightmost position of the previous input field. Any automatic fields, display attribute specifications, or prompts that the cursor moves through are processed, and fields with RG (return to program) exits specified cause external status condition 5. If the mode is special verify, the fields are blanked, including the field the cursor was in when the key was pressed.

If the cursor is in the first input position of the record when the ← (Character Backspace) key is pressed, screen format specifications between the first input position and the start of the screen format are processed in the backward direction, and then in the forward direction. No record backspace function occurs.

In field correct mode, if the cursor is in a field position other than the leftmost position when the ← (Character Backspace) key is pressed, the backspace is processed as described for enter mode. If the cursor is in the leftmost position of the field, the backspace is processed as described above and, in addition, the field is blanked on the screen and the mode returns to Verify mode.

In verify mode, for all fields except a right-to-left field, when the ← (Character Backspace) key is pressed the backspace is processed as described above and, in addition, any position the cursor backspaces through is blanked on the screen. If the field is a right-to-left field, the ← (Character Backspace) key is not valid unless the system is awaiting a field exit or record advance; then the field is blanked on the screen, the awaiting field exit or record advance condition is cleared, the cursor remains in the rightmost position of the field, and the entire field must be reverified.

Clear Screen Function

The clear screen function is initiated by the KEYOP instruction (op code C7) for keyboard operation hex 11. This function is always handled by the 5280. The 5280 fills the screen, but not the status line, with blanks.

Clear Status Line Function

The clear status line function is initiated by the KEYOP instruction (op code C7) for keyboard operation hex 11. This function is always handled by the 5280. The 5280 fills the status line except the first position with blanks. The partition number in the first position is not cleared.

If this function is performed when an ENTR command is being processed, the status line counters and the field shift position will not be completely updated until the cursor enters the next field in the screen format.

The Command Key

The command function is initiated when the operator presses the Cmd (Command) key. The Cmd key is a prefix key, valid at all times. When the Cmd key is pressed, the 5280 sets a flag in the keyboard/display IOB. When the next key is pressed, except for the Shift key, Reset key, the Hex key, Console key, or another Cmd key, external status is posted. If the keystroke following the Cmd key is lowercase, external status condition 2 is posted. If the keystroke following the Cmd key is uppercase, external status condition 3 is posted. After the Cmd key has been pressed, the Reset key will clear the fact that the Cmd key has been pressed.

Cursor Movement

Cursor movement is initiated when the operator presses one of the cursor movement keys. Cursor movement keys, which are located on the left of the keyboard, can move the cursor to the right (→), to the left (←), up (↑) or down (↓). Or the New-line key at the right of the keyboard (↵) can move the cursor to the first position of the next line. The cursor movement keys are valid only while an ENTR command is being processed in enter, update, verify, or special verify modes, and the current screen format definition is for a format level zero field. (Field correct mode is not valid for a format level zero field.) If a cursor movement keystroke moves the cursor out of the format level zero field, an error occurs.

In verify mode, the cursor movement keys to move right, down or to the next line are not valid. The keys to move the cursor left and up are valid, and blank the screen positions through which the cursor moves.

Note: The → (Cursor Right) and ← (Cursor Left) keys are normally redefined in the scan code translated table to invoke the character advance and character backspace functions, respectively.

Delete Function

The delete function is initiated when the operator presses the Del (Delete) key. The Del key is valid only when an ENTR command is being processed. The Del key is not valid in a blank check field or in a mandatory fill field if the mandatory fill check is enabled.

In enter, update, special verify, and field correct modes, when the Delete key is pressed the character above the cursor is deleted. All characters within the field to the right of the cursor are shifted one position to the left. A blank is inserted at the end of the field. The cursor position does not change.

If the cursor is within a picture check subfield when the delete key is pressed, the delete function treats the subfield as a field. If the cursor is within a field defined as format level zero, the delete function treats the total number of 1-byte alphameric fields as one field.

In verify mode, the Del key is not valid.

Duplicate Function

The duplicate function is initiated when the operator presses the Dup (Duplicate) key. The Dup key is valid only when an ENTR command is being processed, and when the Dup enable flag is set to zero. The Dup key is never valid if the system is awaiting field exit or record advance. How the Dup key is processed depends upon the current field definition.

In enter, update, special verify, and field correct modes, if the field definition does not specify a main storage duplicate field, data is duplicated into the current field from corresponding field positions in the previous record buffer. The duplication begins at the current cursor position and continues to the end of the field. If the field is a right-to-left field, the duplication begins at the current cursor position and continues to the leftmost position of the field. A field advance function is then performed.

If the field definition specifies a main storage duplicate field, data is duplicated into the field as described above, except that the data is duplicated from the main storage location specified in the format.

If the field definition specifies a format level zero, only the current position is duplicated from the previous record.

No character set checking is performed on data duplicated into the current field.

In verify mode, automatic verification is performed on the data from the current cursor position to the end of the field, or, for a right-to-left field to the leftmost position of the field. If the field is a main storage duplicate field the data in the field in the current record buffer is verified with the corresponding data in the main storage location. If the field is not a main storage duplicate field, the data is verified with the corresponding data in the previous record buffer. If the verification is successful, a field advance function is performed.

If the field definition specifies format level zero, the data at the current field position in the current record buffer is compared with the corresponding position in the previous record buffer.

If the verification is not successful, a verify mismatch error occurs. The cursor stops at the position where the mismatch occurred, the entire field is displayed on the screen. If the operator presses the Reset key, and then presses the Dup key again, the character is replaced with the corresponding character from the previous record buffer or the main storage location. The automatic verification then continues to the end of the field unless the field is a format level zero field.

Field Advance Function

The field advance function is initiated when the operator presses the →| (Field Advance) key and is valid only when an ENTR command is being processed.

In enter and update modes, when the →| (Field Advance) key is pressed the 5280 checks the characters that have been entered into the field to make sure they satisfy the attributes (except character set) that are specified in the screen format definition for the field. If they do not, an error occurs. If they do, the cursor moves to the first position of the next input field; the first position is the leftmost position in a left-to-right field and the rightmost position in a right-to-left field. Intervening automatic fields, prompts, and display attributes are processed. Fields with RG (return to program) exits specified cause external status condition 4. If the Field Advance key is pressed while the system is awaiting field exit, the awaiting field exit condition is cleared and the field advance is performed. If processing is complete on the last input field of the screen format and the auto-enter flag is on, a record advance is performed. Otherwise, the system sets the awaiting record advance condition.

If the →| (Field Advance) key is pressed while the system is awaiting record advance, an error occurs unless the auto-enter flag is on. If the auto-enter flag is on, a record advance is performed.

If the field is a format level zero field, the field advance is processed as a character advance.

In special verify mode the field advance is processed as described above except that the characters in the field are not checked to make sure they satisfy the attributes specified in the screen format.

In field correct mode, the checks are made as in enter mode. If the checks are successful, the field is blanked on the screen, the cursor moves to the first position of the field, and the mode returns to verify mode. The field can then be verified.

In verify mode, the →I (Field Advance) key is valid only after a constant insert verify error or when the system is awaiting field exit or record advance. If awaiting record advance or field exit, the field advance key is processed as for enter mode except that the attribute checks are ignored. If the →I (Field Advance) key is pressed after a constant insert verify error has occurred, after the operator has pressed the Reset key the constant in the current record buffer remains unchanged and a field advance is performed.

Field Backspace Function

The field backspace function is initiated when the operator presses the I← (Field Backspace) key, and is valid only when an ENTR command is being processed.

In enter, update, and special verify modes, if a I← (Field Backspace) key is pressed while the system is awaiting the field exit or record advance, the condition is cleared and the cursor is repositioned to the first position of the field.

If the I← (Field Backspace) key is pressed when the cursor is in any field position other than the first position of the field, the cursor is repositioned to the first position of the field.

If the I← (Field Backspace) key is pressed when the cursor is in the first position of the field, the cursor is repositioned to the first position of the preceding input field. Intervening automatic fields, prompts, and display attributes are ignored. Intervening RG (return to program) exit specifications are posted with the external status condition 5. If the mode is special verify, the data field in which the cursor was positioned when the key was pressed is blanked, and intervening automatic fields are blanked.

If the I← (Field Backspace) key is pressed when the cursor is in the first position of the record, format specifications between the first position and the start of the screen format are processed in the backward direction, then in the forward direction. A record backspace is not performed.

If the field definition specifies format level zero, the field backspace is processed as a ← (Character Backspace) key.

In field correct mode, if the system is awaiting field exit, the I← (Field Backspace) key is processed as described for enter mode.

If the I← (Field Backspace) key is pressed when the cursor is in any position other than the first position, the key is processed as described for enter mode. If the key is pressed when the cursor is in the first position of the field or of the record, the key is processed as described for enter mode except that the entire field is blanked on the screen and the mode returns to verify mode.

In verify mode, if the I← (Field Backspace) key is pressed while the system is awaiting field exit or record advance in a right-to-left field, the condition is cleared, the field is blanked on the screen, the cursor remains in the rightmost position of the field and the entire field must be reverified.

If the I← (Field Backspace) key is pressed under any other condition, the key is processed as described for enter mode except that any data character that is backspaced over is blanked on the screen, and reverification is required in order to advance over the position.

If the field definition specifies a format level zero, the I← (Field Backspace) key is processed as for aI← (Character Backspace) key.

Field Correct Function

The field correct function is initiated when the operator presses the lowercase Field Corr (Field Correct) key, and is valid only in verify mode. It is not valid for a format level zero field.

If the Field Corr key is pressed when the cursor is in an automatic field, verification of the field is not done automatically.

When the Field Corr key is pressed and a constant insert verify error has not occurred, the cursor is repositioned at the first input position of the current field and the field is filled with blanks in the current record buffer and on the screen. The operator can enter data into the field as in enter mode. Auto dup and auto skip functions are not performed. Character set and field edit checks are performed. When the field is exited in the forward direction, the cursor is repositioned to the first position and the mode returns to verify mode.

Following a constant insert verify error, the operator can press the Reset key, then press the Field Corr key. The constant insert data from the main storage location specified in the screen format is placed into the current record buffer. This data overwrites the data in the buffer. A field advance is then performed.

Field Exit Function

The field exit function is initiated when the operator presses the Field Exit or the Field+ key, and is valid only when an ENTR command is being processed.

In enter, update, special verify, and field correct modes, if the key is pressed while the system is awaiting field exit, the awaiting field exit condition is cleared and a field advance is performed. If the key is pressed while the system is awaiting record advance, an error occurs unless the auto-enter flag is on. If the auto-enter flag is on, a record advance is performed.

- Right-Adjust Field: For a right-adjust field (that is not signed numeric), the right-adjust function is performed before the field advance occurs. The data in the field to the left of the cursor is right-adjusted on the screen and in the current record buffer. The leftmost positions are filled with alphabetic or numeric fill characters, according to the field definition. The zone of the rightmost byte in the buffer is not changed. If the key is pressed when the cursor is in the leftmost position of the field, the entire field is filled with the fill characters.

- Signed Numeric Field: If the field is a signed numeric field, the right-adjust function is performed as described above for a right-adjust field, except that the rightmost position of the field on the screen is blank, which represents a positive sign. The data in the field is right-adjusted to the position to the right of the blank.

- All Other Fields: When a Field Exit key is pressed when the cursor is in any other field, and the system is not awaiting field exit or record advance, it is processed as for the skip function.

  In verify mode, if a Field Exit key is pressed while the system is awaiting record advance, it is processed as described for enter mode. If the system is not awaiting record advance, the verify function is determined by the field definition.

- Right-Adjust Field or Signed Numeric Field: If the field is specified as right-adjust or signed numeric, the key may be pressed only when the cursor is in the leftmost position of the field or when the system is awaiting field exit. When the key is pressed when the cursor is in the leftmost position, the field is verified for the appropriate fill characters. When the key is pressed when the system is awaiting field exit, and if the rightmost character of the field has been completely verified, a field advance is performed. If only the digit portion of the rightmost character has been verified, the zone portion is verified against hex F. If it does not match hex F, a verify sign mismatch error occurs. If the operator presses the Reset key and then presses the Field Exit key again, the zone of the rightmost character is changed to hex F. If the field is signed numeric, the minus sign on the screen is replaced by a blank. If the field is not signed numeric and not numeric shift, the negative numeric graphic in the rightmost position is replaced with the routine numeric graphic. A field advance is then performed.

- All Other Fields

  In all other fields the verify action of the key is as for the Skip key.

Field Exit Minus Function

The field exit minus function is initiated when the operator presses the Field- key, and is valid only when an ENTR command is being processed.

In enter, update, special verify, and field correct modes, if the key is pressed while the system is awaiting record advance an error occurs unless the auto-enter flag is on. If the auto-enter flag is on, a record advance is performed. If the system is not awaiting record advance, the processing of the Field- key depends upon the field definition.

- Digits Only Right-Adjust or Numeric Only Right-Adjust: When the system is not awaiting field exit, the data to the left of the cursor is right-adjusted to the rightmost position of the field on the screen and in the current record buffer. The leftmost positions of the field are filled with the appropriate fill characters. After the right-adjust, if the rightmost character of the data is not a digit (0-9) an error occurs. Otherwise, the negative graphic for the digit is placed into the rightmost field position on the screen, and a hex D is placed into the zone portion of the rightmost byte in the current record buffer. Then a field advance is performed. If the Field- key is pressed when the cursor is in the leftmost position of the field, the function is processed as described above except that the field is filled with the appropriate fill character before the negative graphic and the hex D zone are processed. If the system is awaiting field exit, the function is processed in the same way except that no right-adjust occurs.

- Digits Only or Numeric Only (Not Right-Adjust): For a digits-only or numeric-only field that is not right-adjust, and while not awaiting field exit, the positions to the right of the cursor except for the rightmost position are filled with blanks on the screen and in the current record buffer. The negative zero graphic is placed in the rightmost field position on the screen, a negative zero (hex D0) is placed into the rightmost byte in the buffer, and a field advance is performed. If awaiting field exit, the key is processed in the same way except that an error occurs unless the rightmost data character in the field is a digit (0-9). The zone of the digit is set to hex D in the current record buffer and the negative zero graphic is displayed on the screen.

- Numeric Field: For a numeric field, the Field- key is valid only if the field exit- flag (bit 0 of byte hex 3D into the IOB) is set to 0. If the field exit-flag is zero, the function is processed as for a numeric-only field except that the negative graphic is not displayed on the screen. If the field exit- flag is not zero, the Field- key is not valid in the field; an error occurs if it is pressed while the cursor is within the field.

- Signed Numeric Field: If the system is not awaiting field exit, the data to the left of the cursor is right-adjusted to the next to the rightmost position of the field on the screen and to the rightmost position of the field in the current record buffer. The leftmost positions are filled with the appropriate fill character. After the right-adjust, if the rightmost character is not a digit (0-9) an error is posted. Otherwise, the zone of the rightmost digit is set to hex D in the current record buffer, a minus sign is displayed on the screen in the sign (rightmost) position of the field, and a field advance is performed. If the system is awaiting field exit, the processing is the same except that no right-adjust is performed. If the key is pressed while the cursor is in the leftmost position of the field, the processing is the same except that the field is filled with the appropriate fill character before the sign is processed.

- All Other Fields: The Field- key is not valid for any other field definition for these modes.

In verify mode, if the system is awaiting record advance when the Field- key is pressed, an error occurs unless the auto-enter flag is on. If the auto-enter flag is on, a record advance is performed. If the system is not awaiting record advance, the function is processed depending upon the field definition.

- Signed Numeric, Digits Only Right-Adjust, Numeric Only Right-Adjust: For a signed numeric, digits-only right-adjust, and numeric-only right-adjust field, the Field- key is valid only when the cursor is in the leftmost position of the field or when the system is awaiting field exit.

  Awaiting Field Exit: If the key is pressed when the system is awaiting field exit, the zone portion of the rightmost byte in the current record buffer is verified for a hex D. If the zone is not a hex D, a verify sign mismatch error occurs. If the operator presses the Reset key, and then presses the Field- key again, the zone is changed to hex D in the buffer and a field advance is performed.

  If the field is signed numeric and the rightmost byte is hex D0-D9 the negative graphic for the digit is displayed in the rightmost field position on the screen, or if it is hex DA-DF, a blank is displayed in the rightmost field position on the screen; a field advance is then performed.

  If the field is numeric-only or digits-only and the rightmost byte is hex D0-D9, the negative graphic for the digit is displayed in the rightmost field position on the screen, or if it is hex DA-DF, no change is made on the screen; a field advance is then performed.

  Leftmost Position: If the Field- key is pressed when the cursor is in the leftmost position of the field, all field positions except the rightmost position are verified for the appropriate fill character. The digit portion of the rightmost byte in the buffer is verified for the digit portion of the appropriate fill character. The zone portion is verified for a hex D. If the verification is successful, the rightmost field position on the screen displays as described above. If the zone of the rightmost character is not hex D, an error occurs and the zone may be changed to hex D as described above. If verification other than sign verification fails, an error occurs and the operator must press the Reset key, then reenter the field positions from the error keystroke to the end of the field.

- Digits Only or Numeric Only (Not Right-Adjust): If the field is digits-only or numeric-only but right-adjust is not specified, and if the system is awaiting field exit the Field- key is processed as for a digits-only or numeric-only right-adjust field. If the system is not awaiting field exit when the Field- key is pressed, the positions to the right of the cursor except for the rightmost field position are verified for blanks. If a nonblank character is encountered, the cursor stops at that position, the remainder of the field is displayed, and a verify mismatch error is reported. If the operator presses the Reset key and then presses the Field- key again, a blank replaces the character at that position and the blank verification continues. The rightmost byte of the field in the buffer is verified for a negative zero (hex D0). If it is not a negative zero, the character is displayed on the screen and a verify mismatch error is reported. If the operator presses the Reset key and then presses the Field- key again, a negative zero is placed into the rightmost byte in the buffer and displayed in the rightmost field position on the screen. After the rightmost position is successfully verified, a field advance is performed.

- Numeric Field: For a numeric field, the Field- key is valid only if the field exit minus flag (bit 3 of byte hex 3D into the IOB) is zero. The function is processed as described above for the numeric-only field except that the negative graphic is not displayed.
- The Field- key is not valid for any other field definition.

Field Exit Minus/Dash Function

The field exit minus/dash function is initiated when the operator presses the lowercase dash key on the data entry keyboard.

In all modes, if the cursor is positioned within a field in which the field- key is allowed [a signed numeric, numeric only, digits only, and (if the field- key enable flag is 0) a numeric field], a field minus function is performed. Otherwise, this key is processed as a dash/minus data key.

Hex Function

The hex function is valid only when an ENTR command is being processed. It is not valid for a hex field or when the system is awaiting field exit or record advance.

In all modes, the hex function is selected with a command key sequence from the operator or by the application program issuing a keyboard operation (KEYOP) for a keyboard function. When the hex function is selected, the keyboard is placed in hex mode. The next two keystrokes must be 0-9 or A-F, and are combined to make one EBCDIC value. This EBCDIC value is then processed as a data character. It is not necessary to use the shift key to select the hex characters.

If the operator presses the Reset key after the Cmd key and the Hex key have been pressed, or after the first of the two hex character keystrokes has been pressed, no data is processed and hex mode is cleared. If a key other than the 0-9 or A-F key is pressed following the Cmd key and the Hex key, an error occurs. The operator must press the Reset key; hex mode is cleared and no data is processed.

Insert Function

The insert function is initiated when the operator presses the Ins (Insert) key. The Ins key is valid only when an ENTR command is being processed. The Ins key is not valid in a field defined as mandatory fill.

In enter, update, special verify, and field correct modes, when the Ins key is pressed the keyboard is placed in insert mode. The insert mode symbol is displayed in position 14 of the status line. When the operator presses a data key, the data character is inserted into the field in the current cursor position. All field positions to the right of the cursor, and the cursor and character above the cursor, are shifted one position to the right. If the character that would be shifted out of the end of the field is not blank, an error occurs. If the cursor is in the rightmost position of a field when an attempt is made to insert a character, an error occurs. Any attempt to exit a field while in insert mode causes an error. The operator can cancel insert mode by pressing the Reset key.

If the cursor is within a picture check subfield when the Ins key is pressed, the insert function treats the subfield as a field. If the cursor is within a field defined as format level zero, the insert function treats the total number of 1-byte alphameric fields as one field.

If the first of two hex digits has been entered into a position of a hex field when the Ins key is pressed, the keyboard is placed into Insert mode but the one previously entered hex digit is lost.

In Verify mode, the Insert key is not valid.

Katakana Alphameric Lowershift

Alphameric lowershift is initiated when the operator presses the Alphameric lowershift key (ALPH SHIFT) on Katakana keyboards, and is valid at all times. If a key is pressed while the Alphameric lowershift key is held down, the bottom left symbol on the keytop is selected. The Alphameric uppershift key overrides the Alphameric lowershift key.

Katakana Alphameric Uppershift

Alphameric uppershift is initiated when the operator presses the Alphameric uppershift key (NUM SHIFT) on Katakana keyboards, and is valid at all times. If a key is pressed while the Alphameric uppershift key is held down, the top left symbol on the keytop is selected.

Katakana Shift Lock Function

The shift lock function is initiated when the operator presses the Shift lock key (Lock) on Katakana keyboards and is valid at all times. If the Shift lock key is held down while the Alphameric lowershift key is released, the keyboard is locked to the Alphameric shift. If the Shift lock key is held down while the Katakana lowershift key is released, the keyboard is locked to the Katakana shift. The lock status is overridden by the Alphameric uppershift key, the Alphameric lowershift key, the Katakana uppershift key, and the Katakana lowershift key. The lock status is cleared when the Alphameric lowershift key or the Katakana lowershift key is pressed.

Katakana Lowershift

Katakana lowershift is initiated when the operator presses the Katakana lowershift key (Kata shift) on Katakana keyboards, and is valid at all times. If a key is pressed while the Katakana lowershift key is held down, the bottom right symbol on the keytop is selected. The Alphameric uppershift, Alphameric lowershift, and Katakana uppershift key override the Katakana lowershift key.

Katakana Uppershift

Katakana uppershift is initiated when the operator presses the Katakana uppershift key (SYM SHIFT) on Katakana keyboards, and is valid at all times. If a key is pressed while the World Trade uppershift key is held down, the top right symbol on the keytop is selected. The Alphameric uppershift and Alphameric lowershift keys override the Katakana uppershift key.

Record Advance Function

Record advance is initiated when the operator presses the Enter or Rec Adv (Record Advance) key, and is valid only when an ENTR command is being processed.

In enter, update, and special verify modes, unless the alternate record advance is enabled, when the Rec Adv key is pressed the current field and all remaining input fields in the format are processed as though the →| (Field Advance) key were pressed for each field. Edit checks except the character set checks are performed on the input fields. Intervening automatic fields, prompts, display attributes, and RG (return to program) exit specifications are processed. When the last field has been exited, the record advance function is performed. The current ENTR command is made complete, and external status condition 6, for record advance, occurs. The record advance function can be initiated by pressing the Rec Adv key or by pressing a key that causes a field exit to be performed on the last input field of the record format when the auto-enter switch is on.

If the Rec Adv key is pressed when the alternate record advance is enabled, the format specifications from the current point to the end of the format are not performed, but the record advance function is performed.

In field correct mode, the record advance function is processed as a field advance.

In verify mode, if the Rec Adv key is pressed, the remainder of the record format is verified. Input fields are verified as though the Skip key were pressed for the field except when: a verify mismatch error occurs, and the operator presses the Reset key and then presses the Rec Adv key. In this case the nonblank character is replaced with a blank and blank verification continues to the end of the record. Auto-duplicate fields, auto-skip fields, and constant insert fields are verified as described under the duplicate function, skip function, and insert function.

Record Backspace Function

Record backspace is initiated when the operator presses the Home (Record Backspace) key, and is valid only when an ENTR command is being processed. If the Home key is pressed while the system is awaiting field exit or record advance, the condition is cleared before the record backspace function is performed.

In enter and update modes, when the Home key is pressed the cursor is repositioned to the first position of the record. The format specifications between the position of the cursor when the Home key was pressed and the start of the format are processed in the backward direction. RG (return to program) exit specifications cause external status 5. The format specification between the start of the format and the first manual field are processed in the forward direction. RG exit specification cause external status 4.

If the cursor is in the first position of the record when the Home key is pressed, the current ENTR command is made complete and format specifications between the position on the cursor when the Home key was pressed and the start of the format are processed in a backward direction. RG (return to program) exit specifications cause external status condition 5. After return is made from any specified external status 5 subroutine, external status condition 7 occurs.

For special verify mode, the Home key is processed as described above except that the fields on the screen are blanked as they are backspaced through.

For field correct mode, the Home key is processed as described for enter mode, except that all data on the screen is blanked as it is backspaced through, and requires reverification. The mode returns to verify mode.

For verify mode, the Home key is processed as described for the enter mode, except the data on the screen is blanked as it is backspaced through and requires reverification.

Reset Function

The reset function is initiated when the operator presses the Reset key. The Reset key is valid at all times.

In all modes, when the Reset key is pressed following a keystroke error, the blinking stops and the operator may continue keying. The Reset key also cancels hex mode and insert mode, and resets the Cmd key so the following keystroke is not treated as part of a command key sequence.

Shift Function

The shift function is initiated when the operator holds down the uppershift or numeric shift key, and is valid at all times. While the key is held down, the keyboard is in uppershift. Keys that are pressed while the keyboard is in uppershift select the character, symbol, or function on the upper half of the keytop. When the shift key is released, the shift of the keyboard returns to the shift specified in the screen format control string for the current field.

Shift Lock Function

The shift lock function is initiated when the operator presses the ⬇ (Shift Lock) key. The ⬇ (Shift Lock) key is on typewriter keyboards only. It is valid at all times to lock the keyboard into the uppershift. Except for keys that require special shifting, when the keyboard is in uppershift, the uppercase character on the keytop is selected when a data key is pressed. The shift lock is cleared when the ⬆ (Shift) key is pressed.

Skip Function

The skip function is initiated when the operator presses the Skip key. A Skip key is valid only when an ENTR command is being processed.

In enter, update, special verify, and field correct modes, if the Skip key is pressed while the system is awaiting field exit, the field advance function is performed. If the Skip key is pressed while the system is awaiting record advance, an error occurs unless the auto-dup/skip switch is on. If the auto-dup/skip switch is on, a record advance function occurs.

If the system is not awaiting field exit or record advance, the positions from the cursor to the rightmost field position are filled with blanks. If the field is a right-to-left field, the positions from the cursor to the leftmost field position are filled with blanks. Character set checks are not performed for the blanks. A field advance function is performed.

If the field definition specifies format level zero, the Skip key fills the current position with a blank. The cursor then moves to the next position.

In verify mode, if the system is awaiting field exit or record advance, the function is processed as for enter mode. If not awaiting field exit or record advance, the positions from the cursor to the rightmost field position are verified for blanks. In a right-to-left field the positions from the cursor to the leftmost field position are verified for blanks. If a nonblank character is encountered, the cursor stops at that position and the skip action is terminated. The remainder of the field is displayed, and a verify mismatch error is reported. If the operator presses the Reset key and then presses the Skip key again, a blank replaces the character and the blank verification continues. After the last position has been successfully verified, a field advance function is performed.

If the field definition specified format level zero, the Skip key verifies the current position for a blank, and the cursor moves to the next position.

Conversion Table

Byte				Byte				Byte			
0123		4567		0123		4567		0123		4567	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0	0	0	0	0
1	1 048 576	1	65 536	1	4 096	1	256	1	16	1	1
2	2 097 152	2	131 072	2	8 192	2	512	2	32	2	2
3	3 145 728	3	196 608	3	12 288	3	768	3	48	3	3
4	4 194 304	4	262 144	4	16 384	4	1 024	4	64	4	4
5	5 242 880	5	327 680	5	20 480	5	1 280	5	80	5	5
6	6 291 456	6	393 216	6	24 576	6	1 536	6	96	6	6
7	7 340 032	7	458 752	7	28 672	7	1 792	7	112	7	7
8	8 388 608	8	524 288	8	32 768	8	2 048	8	128	8	8
9	9 437 184	9	589 824	9	36 864	9	2 304	9	144	9	9
A	10 485 760	A	655 360	A	40 960	A	2 560	A	160	A	10
B	11 534 336	B	720 896	B	45 056	B	2 816	B	176	B	11
C	12 582 912	C	786 432	C	49 152	C	3 072	C	192	C	12
D	13 631 488	D	851 968	D	53 248	D	3 328	D	208	D	13
E	14 680 064	E	917 504	E	57 344	E	3 584	E	224	E	14
F	15 728 640	F	983 040	F	61 440	F	3 840	F	240	F	15
6		5		4		3		2		1	

Hexadecimal Addition Table

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

EBCDIC CHARTS FOR PRINTABLE CHARACTERS

Second Hex Digit ↓ / First Hex Digit →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						&	-						{	}	\	0
1							/		a	j	~		A	J		1
2									b	k	s		B	K	S	2
3									c	l	t		C	L	T	3
4									d	m	u		D	M	U	4
5									e	n	v		E	N	V	5
6									f	o	w		F	O	W	6
7									g	p	x		G	P	X	7
8									h	q	y		H	Q	Y	8
9									i	r	z		I	R	Z	9
A					¢	!	¦	:								
B					.	$	,	#								
C					<	*	%	@								
D					(	)	_	'								
E					+	;	>	=								
F					|	¬	?	"								

IBM 5256 STANDARD CHARACTER SET

LANGUAGE: US AND CANADA

Second Hex Digit ↓ / First Hex Digit →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0						&	-	ø	Ø	°	µ	¢	{	}	\	0
1						é	/	É	a	j	‾	£	A	J		1
2					â	ê	Â	Ê	b	k	s	¥	B	K	S	2
3					ä	ë	Ä	Ë	c	l	t	₧	C	L	T	3
4					à	è	À	È	d	m	u	ƒ	D	M	U	4
5					á	í	Á	Í	e	n	v	§	E	N	V	5
6					ã	î	Ã	Î	f	o	w	¶	F	O	W	6
7					å	ï	Å	Ï	g	p	x		G	P	X	7
8					ç	ì	Ç	Ì	h	q	y		H	Q	Y	8
9					ñ	ß	Ñ	`	i	r	z		I	R	Z	9
A					[	]	¦	:	«	ª	¡	¬	-	≥	²	³
B					.	$	,	#	»	º	¿	|	ô	û	Ô	Û
C					<	*	%	@	ð	æ	Ð	≠	ö	ü	Ö	Ü
D					(	)	_	'	≤	¸	↑	¨	ò	ù	Ò	Ù
E					+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
F					!	^	?	"	±	¤		‗	õ	ÿ	Õ	

IBM 5256 STANDARD CHARACTER SET

LANGUAGE: INTERNATIONAL

This page intentionally left blank

Appendix B. SCS Control Characters

Printer formatting can be accomplished via an assembler or DE/RPG program or via communications. To use the SCS characters supported by communications, see the description under the desired communications access method: SNA, BSC, or MRJE. To use an assembler or DE/RPG program to format your printed output, code these following SCS characters that your 5256 or 5225 printer supports. Code the control characters in the printer output data stream intended for the printer. The format of the data stream is:

CC Data CC Data CC Data

where CC is the control characters.

The following table describes the general functions provided by the printer control characters. A detailed description of each control character follows the table.

SCS Control Character	Hex Code	Function
Bell	2F	Bell; sound bell on printer
CR	0D	Carrier return
FF	0C	Forms feed
Fmt	2B . . . 2BC1nnhh 2BC2nnvv 2BC8nngguu	Format Horizontal (SHF) Vertical (SVF) Graphic error (SGEA)
IRS	1E	Interchange record separator
LF	25	Line feed
NL	15	Next line
NUL	00	No operation
PP	34 . . . 34C0nn 34C8nn 34C4nn 344Cnn	Print position Horizontal absolute Horizontal relative Vertical absolute Vertical relative

- *Bell*

Function: This control character stops printing, sounds the audible alarm, if installed, and turns on the Attention indicator.

Code: X'2F'

Results: When the printer microprocessor detects this control character, it:

1. Allows all preceding data to be printed and all preceding control characters to be executed
2. Turn the Ready indicator off
3. Turns the Attention indicator on
4. Sounds the audible alarm, if installed
5. Stops printing
6. Stops formatting
7. Returns an unavailable status to the controller

- *CR (Carrier Return)*

Function: This control character performs a carrier return to the first print position on the same line.

Code: X'0D'

Results: The horizontal print position logically moves to the first print position on the same line. If it already is at the first print position, no operation occurs.

- *FF (Forms Feed)*

Function: This control character moves the paper to the next logical page as specified by the Set Vertical Format control character (see *Fmt*) in this topic.

Default: 1 logical page = 1 logical line.

Code: X'0C'

Results: The print position moves to the first logical print line and first logical print position of the next logical page.

- *Fmt (Format)*

Function: This control character defines data formatting for a specified length (provided in the parameter).

Default: Logical line length = 132 character positions; logical page length = 1 line.

Format of this control character:

Code	Set Type	Associated Parameters
X'2B'	Start of formatted data stream. Must include: SHF, SVF, or SGEA. (See *note.*)	Length of formatted data stream.

Note: The following table shows the various set types and their associated parameters.

Set Types Available for Use with the Format (Fmt) Printer Control Character

Set Type	Format	Values of Parameters	Description of Set Type
SHF (set horizontal format)	C1nnhh	nn = number of bytes in the SHF string.	
		hh = maximum horizontal print position (greater than or equal to 1 and less than or equal to 132). The default is 132.	Sets the maximum print position (MPP), which is the value of the print line length.
SVF (set vertical format)	C2nnvv	nn = number of bytes in the SVF string.	
		vv = maximum number of lines on a page greater than or equal to 1 and less than or equal to 255). The default is a page length of one line.	Sets the maximum print line (MPL) on the logical page; it overrides the physical device logical page.
SGEA (set graphic error action)	C8nnggxx	nn = number of bytes in the SGEA string. See note.	
		gg = unprintable character option. 01=No stop, no status. 02=Defaults to 01. 03=Stop, hard error status. Unit not available. 04=Defaults to 03. The default for xx is 01.	Sets the way the printer will respond when it encounters an unacceptable symbol in the data stream. **Note:** nn must be at least 1 and not greater than 3 for the SGEA set type.

The following table shows the characteristics of the SHF and SVF set types.

Valid Values for the SHF and SVF Set Types

Set Type Code	**Parameters**	**Results (MPL and MPP)**	**Error**
SHF 2BC1nnhh	nn=00	MPP=132	Invalid SCS parameter
	nn=01	MPP=132	None

	nn=02
	hh=00
		MPP=132	None

	nn=02
	hh=1-84
		MPP=1-132 as specified	None

	nn=02
	hh=85-FF
		MPP=132	Invalid SCS parameter

	nn=03-FF	MPP=132	Invalid SCS parameter
SVF 2BC2nnvv	nn=00	MPL=1	Invalid SCS parameter
	nn=01	MPL=1	None

	nn=02
	vv=00
		MPL=1	None

	nn=02
	vv=1-FF
		MPL=1-255 as specified	None

	nn=03-FF	MPL=1	Invalid SCS parameter

- *IRS (Interchange Record Separator)*

Function: This control character does the same thing that NL does.

Code: X'1E'

- *LF (Line Feed)*

Function: This control character moves the paper one line without altering the print position.

Code: X'25'

Results: Moves the paper logically to the same print position on the following line. If you use this control character on the last line of a page, it will move the print position to the first line of the next page.

- *NL (New Line)*

Function: This control character moves the paper to the next line.

Code: X'15'

Results: The print position moves to the first print position on the next line if it is not coded on the last line of the page. If you code this on the last line, it moves the paper to the first print position on the first line of the next page.

- *NUL*

Function: No-op

Code: X'00'

Results: No characters are printed and no functions are performed.

- *PP (Print Position)*

Function: This control character moves the logical print position as determined by the associated parameters.

Restrictions: The absolute parameters (see the following explanation) must be equal to or less than the page length. If the absolute horizontal parameter is less than the current print position, the printer microprocessor treats it as a separate line and inserts a CR control character in the printer data stream. If the absolute vertical parameter is less than the current line number, the microprocessor treats it as a new page. If both are equal, no operation is performed. Relative values must indicate a move to but not past the end of the line or page. A value of 0 is not valid, and no operation is performed.

Code and Format:

X'34'	Function Parameter	Value Parameter
	(Hex)	(Decimal)
	Note 1	*Note 2*

The results are determined by the parameters as described in the following notes.

Notes:

1. The following table shows the types of moves available and indicates what the PP CC accomplishes for each type:

Function	Function Parameter (Hex)	Value Parameter (Decimal)
Absolute horizontal move	C0	Numeric value of horizontal position (less than or equal to the end of the line)
Absolute vertical move	C4	Numeric value of vertical position (less than or equal to the end of page)
Relative horizontal move	C8	Numeric value of horizontal movement from the present position (less than or equal to the end of the line).
Relative vertical move	4C	Numeric value of vertical movement from the present position (less than or equal to the end of the page).

2. The following table shows the relationships of the parameters:

Function	Value Parameter (nn)	Results
Absolute horizontal move (X'34C0nn')	00	No-op; the current print position is unchanged; no error.
	00<nn≤132	The print position becomes the value of nn.
	nn>max PP	Error; invalid SCS parameter.
Absolute vertical move (X'34C4nn')	00	No-op; the current print position is unchanged; no error.
	current PP ≤nn≤max PP	The print position becomes the value of nn and remains on the same logical page.
	0<nn<current PP	The print position becomes the value of nn and goes to the next logical page.
	nn>max PP	Error; invalid SCS parameter.

Function	Value Parameter (nn)	Results
Relative horizontal move (X'34C8nn')	00	No-op; the current print position is unchanged; no error.
	nn+current PP ≤max PP	The new print position is equal to the current print position plus the value of nn.
	nn+current PP >max PP	Error; invalid SCS parameter.
Relative vertical move (X'344Cnn')	00	No-op; the current print position is unchanged; no error.
	nn+current PP ≤max PP	The print position becomes the value of the current print position plus the value of nn.
	nn+current PP >max PP	Error; invalid SCS parameter.

This page intentionally left blank

Appendix C. Keyboard Functions: EBCDIC Codes and Bit Numbers

The EBCDIC is the code that is placed into byte hex A7 of the keyboard/display IOB. The bit number is the number used for the TRAP parameter of the .KBCRT control statement.

EBCDIC	Bit Number	Key	Description
X'00'	–	–	Invalid scan code generated from translate table or hardware. An error code is presented to the operator.
X'01'	0	Cmd	Command key prefix to select command function.
X'02'	1	Cmd	Shifted command key.
X'03'	2	–	Keyboard overrun; keyboard has lost two keystrokes due to hardware keystroke buffer overrun.
X'04'	3	–	Invalid keystroke; the code is generated directly from the scan code translate table or the World Trade translate table.
X'05'	4	Reset	Reset function; reset error condition, or reset Hex key command, or insert function.
X'06'	5	Ins	Insert function; initiate character insert.
X'07'	6	Del	Delete function; initiate character delete.
X'08'	7	Alpha	Alpha shift, with the Alpha key pressed.
X'09'	8	⬆ or Num	Numeric shift, with the ⬆ (Shift) key or Num key pressed.
X'0A'	9	⬇	Shift lock, with the ⬇ (Shift Lock) key pressed.
X'0B'	10	Num Shift	Katakana numeric shift, with the Num Shift key pressed.
X'0C'	11	Alpha Shift	Katakana alphabetic shift, with the Alpha Shift key pressed.

EBCDIC	Bit Number	Key	Description
X'0D'	12	Kata Shift	Katakana shift, with the Kata Shift key pressed.
X'0E'	13	Sym Shift	Katakana uppershift; with Sym. Shift key pressed.
X'0F'	14	Lock	Katakana shift lock; with the Lock key pressed.
X'10'			Cursor right (not used for normal IPL).
X'11'			Cursor left (not used for normal IPL).
X'12'	17	↑	Move cursor up; valid only for format level zero.
X'13'	18	↓	Move cursor down; valid only for format level zero.
X'14'	19	↵	New line; moves cursor to the first position on the next line; valid only for format level zero.
X'15'	20	Field Exit, Field+	Field exit function.
X'16'	21	Field–	Field exit minus function.
X'17'	22	Skip	Skip function.
X'18'	23	Alpha	Alpha shift, with the Alpha key released.
X'19'	24	⬆ or Num	Numeric shift, with the ⬆ (Shift) key or Num key released.
X'1A'	25	⬇	Shift lock, with the ⬇ (Shift Lock) key released.
X'1B'	26	Num Shift	Katakana numeric shift, with the Num Shift key released.
X'1C'	27	Alpha Shift	Katakana alphabetic shift, with the Alpha Shift key released.
X'1D'	28	Kata Shift	Katakana shift, with the Kata Shift key released.
X'1E'	29	Sym Shift	Katakana uppershift, with the Sym Shift key released.

EBCDIC	Bit Number	Key	Description
X'1F'	30	Lock	Katakana shift lock; with the Lock key released.
X'20'	31	Dup	Duplicate function.
X'21'	32	→I	Field advance function.
X'22'	33	I←	Field backspace function.
X'23'	34	Unshifted Corr	Field correct function.
X'24'	35	Enter/ Rec Adv	Record advance function.
X'25'	36	Home	Record backspace function.
X'26'	37	→	Character advance function.
X'27'	38	⟵	Character backspace function.
X'28'	39	Hex key	Hex command function key.
–	40		Keystroke error, detected and normally handled by the keyboard/display.
X'29'	–	No key is associated with this function.	Clear screen function; blanks all positions on the screen except the status line.
X'2A'	–	No key is associated with this function.	Clear status line function; blanks all positions on the status line.
X'2B'	–	No key is associated with this function.	Keystroke with this EBCDIC is ignored.
X'2C'	43	=	Field-/dash combination key.

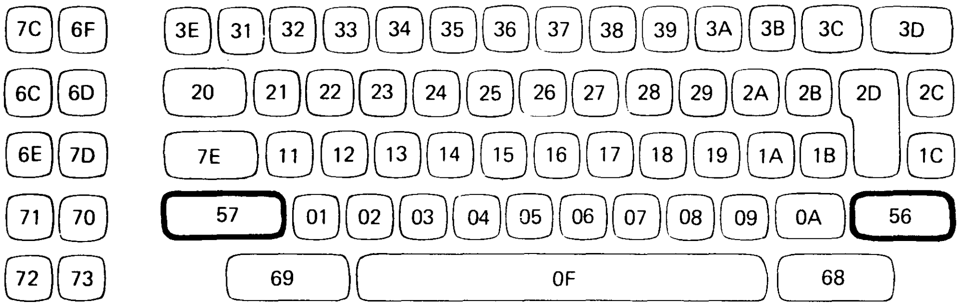
Functions from 2D-3F are handled by your program, with external status condition 1 subroutines.

EBCDIC	Bit Number	Key	Description
X'2D-32'			Not assigned; you may assign these codes to special functions for your applications.
X'33'		Sel Fmt	Select format function.
X'34'		Dup Skip	Switches the auto dup/skip flag.
X'35'		Auto Enter	Switches the auto enter flag.
X'36'		Cncl	Cancel function; defined and processed by your program.
X'37'		Page Fwd	Page forward function; to read the next record without writing out the current record, processed by your program.
X'38'		Next Fmt	Next format function; to allow the operator to exit a repetitive format, processed by your program.
X'39'		Prnt	Print function; to initiate output from the printer.
X'3A'			Not used.
X'3B'		Erase Inpt	Erase input function.
X'3C'		Corr	Record correct function; initiated by the shifted Corr key.
X'3D'		Sys Req	System request function.
X'3E'		Attn	Attention function.
X'3F'		Help	Help function; to request a help message.

Appendix D. Keyboard Scan Codes

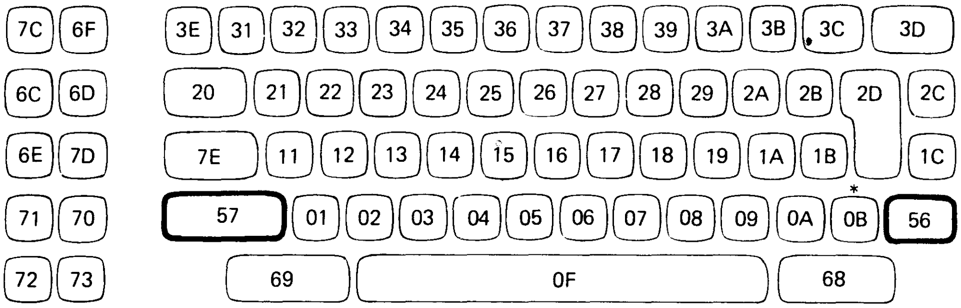
The keyboard scan codes for the different keyboards are shown on the individual keytops. The make/break keys generate a scan code of X'5n' when they are pressed, as illustrated in the following illustrations. The make/break keys generate a scan code of X'Dn' when they are released. For example, the shift key shown in the first illustration generates a scan code of X'57' when it is pressed and generates a scan code of X'D7' when it is released.

The 66-Key Data Entry Keyboard and Data Entry with Proof Keyboard


7C 6F 3E 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D
6C 6D 20 21 22 23 24 25 26 27 28 29 2A 2B 2D 2C
6E 7D 7E 11 12 13 14 15 16 17 18 19 1A 1B 1C
71 70 57 01 02 03 04 05 06 07 08 09 0A 56
72 73 69 0F 68

□ — A make/break key

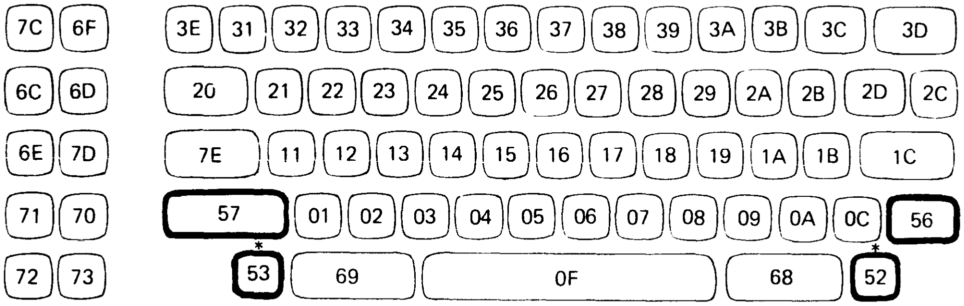
The 67-Key Data Entry Keyboard and Data Entry with Proof Keyboard (Not used in the US)


7C 6F 3E 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D
6C 6D 20 21 22 23 24 25 26 27 28 29 2A 2B 2D 2C
6E 7D 7E 11 12 13 14 15 16 17 18 19 1A 1B 1C
71 70 57 01 02 03 04 05 06 07 08 09 0A 0B 56
*
72 73 69 0F 68

*
□ — Not present on the 66-key keyboard

□ — A make/break key

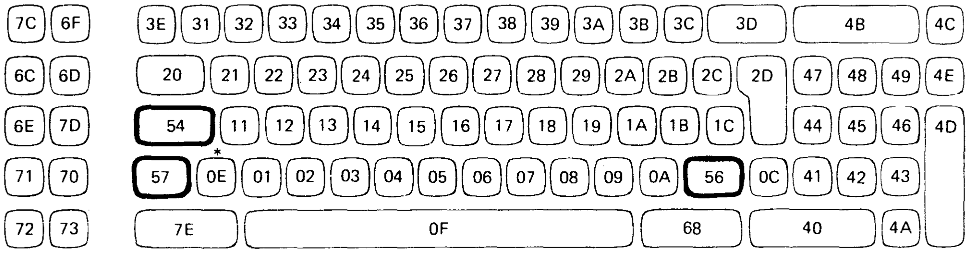
The 69-Key Data Entry Keyboard and Data Entry with Proof Keyboard (Katakana only)


7C 6F 3E 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D
6C 6D 20 21 22 23 24 25 26 27 28 29 2A 2B 2D 2C
6E 7D 7E 11 12 13 14 15 16 17 18 19 1A 1B 1C
71 70 57 01 02 03 04 05 06 07 08 09 0A 0C 56
72 73 * 53 69 0F 68 * 52

*□ — Not present on the 67-key keyboard

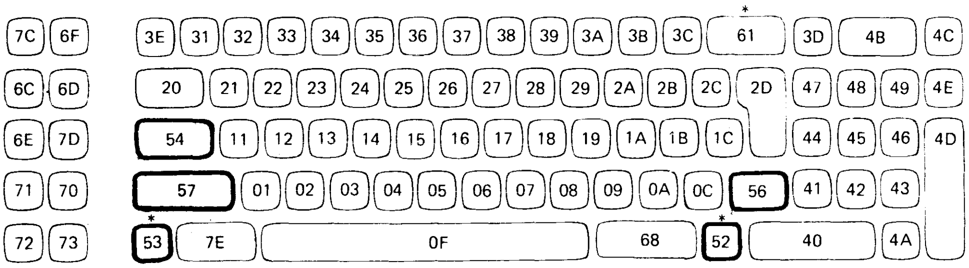
▢ — A make/break key

The 83-Key Typewriter Keyboard


7C 6F 3E 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 4B 4C
6C 6D 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 47 48 49 4E
6E 7D 54 11 12 13 14 15 16 17 18 19 1A 1B 1C 44 45 46 4D
71 70 57 * 0E 01 02 03 04 05 06 07 08 09 0A 56 0C 41 42 43
72 73 7E 0F 68 40 4A

▢ — A make/break key

The 85-Key Typewriter Keyboard (Katakana only)


7C 6F 3E 31 32 33 34 35 36 37 38 39 3A 3B 3C * 61 3D 4B 4C
6C 6D 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 47 48 49 4E
6E 7D 54 11 12 13 14 15 16 17 18 19 1A 1B 1C 44 45 46 4D
71 70 57 01 02 03 04 05 06 07 08 09 0A 0C 56 41 42 43
72 73 * 53 7E 0F 68 * 52 40 4A

*□ — Not present on the 83-key keyboard

▢ — A make/break key

DISKETTE VOLUME LABEL

The diskette volume label identifies the volume, owner, security and sequence of the physical records on the tracks of the specified volume. The volume label is located at track 00, head 0, record 7, on each diskette. The following table shows the format of the diskette volume label (VOL1).

Decimal Position — **Description**

01-03 Volume label identifier; must be C'VOL'.

04 Volume label number; must be C'1'.

05-10 Volume identification field; up to 6 alphameric characters.

11 Accessibility indicator; a blank in this field permits access to this diskette. Any other character requires additional owner ID information to permit access.

12-24 Not used.

25-37 System code; not supported on the 5280.

38-51 Owner identification field; up to 14 alphameric characters used to access secure diskette.

52-64 Not used.

65 Volume Label Extension Indicator:

Character	*Meaning*
Blank or 0	No additional cylinders allocated.
1-9	Number of additional cylinders allocated; valid only on diskette 2D.

66-71 Not used.

72 Surface Indicator:

Character	*Meaning*
Blank	1 surface, FM recording (diskette 1)
2	2 surfaces, FM recording (diskette 2)
M	2 surfaces, MFM recording (diskette 2D)

Decimal Position	Description
73	Extent arrangement indicator (not used by the 5280).
74	Special requirements indicator (not used by the 5280).
75	Not used.
76	Physical Record (sector) Length Indicator:

Character	*Meaning*
Blank	128-byte sectors
1	256-byte sectors
2	512-byte sectors
3	1024-byte sectors

77-78	Physical record sequence code (see *Physical Record Sequence Code*).
79	Not used.
80	Label standard version; W indicates standard IBM labels are used. Anything else is invalid on the 5280.
81-128	Padding.

Physical Record Sequence Code (Position 77-78): Indicates how the physical records are sequenced on the diskette. This field contains either blanks or the characters 01 through 13. Blanks or a 01 indicates the sectors are physically sequential. Otherwise, this field is used as an increment to determine the next physical sectors.

26 Sectors Per Track

When this field contains:	Blank	01	02	03	04	05	06	07	08	09	10	11	12	13
The sequencing is:	1	1	1.	1	1	1	1	1	1	1	1	1	1	1
	2	2	3	4	5	6	7	8	9	10	11	12	13	14
	3	3	5	7	9	11	13	15	17	19	21	23	25	2
	4	4	7	10	13	16	19	22	25	2	2	2	2	15
	5	5	9	13	17	21	25	2	2	11	12	13	14	3
	6	6	11	16	21	26	2	9	10	20	22	24	26	16
	7	7	13	19	25	2	8	16	18	3	3	3	3	4
	8	8	15	22	2	7	14	23	26	12	13	14	15	17
	9	9	17	25	6	12	20	3	3	21	23	25	4	5
	10	10	19	2	10	17	26	10	11	4	4	4	16	18
	11	11	21	5	14	22	3	17	19	13	14	15	5	6
	12	12	23	8	18	3	9	24	4	22	24	26	17	19
	13	13	25	11	22	8	15	4	12	5	5	5	6	7
	14	14	2	14	26	13	21	11	20	14	15	16	18	20
	15	15	4	17	3	18	4	18	5	23	25	6	7	8
	16	16	6	20	7	23	10	25	13	6	6	17	19	21
	17	17	8	23	11	4	16	5	21	15	16	7	8	9
	18	18	10	26	15	9	22	12	6	24	26	18	20	22
	19	19	12	3	19	14	5	19	14	7	7	8	9	10
	20	20	14	6	23	19	11	26	22	16	17	19	21	23
	21	21	16	9	4	24	17	6	7	25	8	9	10	11
	22	22	18	12	8	5	23	13	15	8	18	20	22	24
	23	23	20	15	12	10	6	20	23	17	9	10	11	12
	24	24	22	18	16	15	12	7	8	26	19	21	23	25
	25	25	24	21	20	20	18	14	16	9	10	11	12	13
	26	26	26	24	24	25	24	21	24	18	20	22	24	26

15 Sectors Per Track

When this field contains:	Blank	01	02	03	04	05	06	07
The sequencing is:	1	1	1	1	1	1	1	1
	2	2	3	4	5	6	7	8
	3	3	5	7	9	11	13	15
	4	4	7	10	13	2	4	7
	5	5	9	13	2	7	10	14
	6	6	11	2	6	12	2	6
	7	7	13	5	10	3	8	13
	8	8	15	8	14	8	14	5
	9	9	2	11	3	13	5	12
	10	10	4	14	7	4	11	4
	11	11	6	3	11	9	3	11
	12	12	8	6	15	14	9	3
	13	13	10	9	4	5	15	10
	14	14	12	12	8	10	6	2
	15	15	14	15	12	15	12	9

8 Sectors Per Track

When this field contains:	Blank	01	02	03	04
The sequencing is:	1	1	1	1	1
	2	2	3	4	5
	3	3	5	7	2
	4	4	7	2	6
	5	5	2	5	3
	6	6	4	8	7
	7	7	6	3	4
	8	8	8	6	8

DISKETTE HEADER LABEL (HDR1)

Diskette 1: The HDR1s on diskette 1 are located on track 0, head 0, on sectors hexadecimal 08 through 1A.

Diskette 2: The HDR1s on diskette 2 are located on cylinder 0, head 0, on sectors hex 08 through 1A and on cylinder 0, head 1, on sectors 01 through 1A.

Diskette 2D: The HDR1s on diskette 2D are located on cylinder 0, head 0, on sectors 08 through 1A and on cylinder 0 head 1 on sectors 01 through 1A. In addition, nine additional cylinders can be allocated on diskette 2D for HDR1 labels on sectors 01 through 1A. On cylinder 0, head 1 and on additional index cylinders there are 2 labels per sector. The following table shows the format of the diskette header label (HDR1).

Decimal Position **Description**

01-03 Header label identifier; must be C'HDR'.

04 Header label number; must be C'1'.

05 Not used.

06-22 Data set identifier; user name for the data set. It must be 1 to 17 characters. The first character must be in position 6 and must be alphabetic. No blanks are allowed between characters. Duplicate names are not permitted on the same diskette. For basic, H, and I exchange only 8 characters can be used. The names ERRORSET and SYSAREA are reserved for special use.

23-27 Block length; a numeric value (1 to 16 256) specifying the maximum numbers of characters per block. At label creation, the contents must be entered. Blocks must begin on physical record boundaries. For basic exchange, 1 to 128. For H exchange, 1 to 256. For I exchange, the block length must equal the physical record length.

28 Record Attribute:

Character	*Meaning*
Blank	Records are unblocked, unspanned
B	Records are blocked, unspanned
R	Records are blocked, spanned

29-33 Beginning of extent (BOE); the address of the first sector of the data set. Positions 29 to 30 contain the cylinder number, position 31 contains the header number, and positions 32 to 33 contain the sector number.

Decimal Position **Description**

34 Physical record length; must be the same as position 76 of the volume label.

Blank = 128-byte records.
1 = 256-byte records.
2 = 512-byte records.
3 = 1024-byte records.

35-39 End of extent (EOE); the address of the last sector reserved for this data set, using the same format as BOE.

40 Record Block Format:

Character	*Meaning*
Blank or F	Fixed-length records in the fixed blocks.
V	Record length is variable; not supported on the 5280.

41 Bypass Indicator:

Character	*Meaning*
B	Not to be exchanged or copied
Blank	Can be exchanged or copied

42 Data set security; data set cannot be accessed if this byte contains other than a block.

43 Write protect; if this position contains a P, the data set can only be read. This field must be blank to allow both reading and writing.

44 Exchange Type Indicator:

Character	*Meaning*
Blank	Basic exchange for diskette 1 and 2, formats 1 and 4.
H	H exchange for diskette 2D, format 7.
E	No summation of attributes exists.
I	I exchange.

Decimal Position **Description**

45 Multivolume Data Set Indicator:

Character	*Meaning*
blank	The data set is complete on this diskette.
C	The data set is continued on another diskette.
L	This is the last volume of a multivolume data set.

46-47 Volume sequence number; specifies the sequence of volumes in a multivolume data set. The sequence must be consecutive, beginning with 01 (to a maximum of 99). Blanks indicate that volume sequence checking is not to be performed on this volume and all subsequent volumes of a multivolume data set.

48-53 Creation date; may be used to record the date the data set was created. The format is digits representing YYMMDD, where YY is the low-order 2 digits of the year, MM is a 2-digit representation of the month, and DD is a 2-digit representation of the day of the month. Blanks indicate that the creation date is not significant.

54-57 Logical record length; 1-9999. Blank indicates that the logical record length equals the block length.

58-62 Offset to next record space; indicates the starting position for the next sequential record relative to the end of the last block before EOD (end of data) and contains blanks or a decimal value to be used as a negative displacement. Blanks mean zero displacement from the next block (starts at EOD address). This field is used only in conjunction with blocked records.

63-66 Not used.

67-72 Expiration date; may be used to contain the date the data set (and the data set label) may be deleted. The format is the same as creation date (positions 48-53). All blanks indicate the data set is expired. All 9s indicate the data set will never expire.

73 Verify/copy indicator; V indicates that the data set has been verified. C indicates that the data set has been copied. Blank indicates that it has been neither verified nor copied.

74 Data set organization; blank indicates that the data set organization is sequential. D indicates the sequential relocation is not allowed.

Decimal Position **Description**

75-79 End of data address (EOD); identifies the address of the next unused sector within the data set extent, using the same format as BOE. If this field is the same as BOE, the extent contains a null data set. If this field contains the address of the next block beyond the extent (for unblocked, unspanned records), the entire extent has been used. For blocked or spanned records, this field must be used with offset to next record space (positions 58-62) to determine the end of actual data recorded.

80 Not used.

81 Destination selection code (not used by the 5280).

82-95 Not used.

96-108 System code; identifies the operating system that created the data set label for this data set. (For the 5280 system, this field will be IBM5280.)

109-110 File application type (not used by the 5280).

111-118 Not used.

119 Data header/trailer label indicator (for I and E exchange only):

Blank Indicates that the data set uses no data labels.
F Indicates that one or more data header labels are stored in the beginning of the data set.
E Indicates that one or more data trailer labels are stored at the end of the data set.
B Indicates that one or more data header labels are stored at the beginning of the data set and one or more data trailer labels are stored at the end of the data set.

120-121 Number of data header labels (I and E exchange only). Can have values 01 through 99.

122 Number of data trailer labels (I and E exchange only). Can have values 1 through 9.

123 Record delete indicator (for I ad E exchange only); the character used to indicate deleted records. This character appears in the last position of a logical record to indicate that it is deleted. Valid characters are A-Z, 0-9, or one of the following symbols: . , - / % # @ : $ &

124-128 Not used.

This page intentionally left blank

access method: A technique for moving data between main storage and an input/output device.

active data set: A data set being used by a program.

adapter: The part of an attachment that is needed to electrically or physically fit a device to another system component.

address: A name, label, or number that identifies a register, location in storage, or any other data source.

alphabetic characters: Letters and other symbols, excluding digits, used in a language.

alphabetic field: One or more alphabetic characters of related information in a record.

alphabetic shift: A control (attribute or key) for selecting the alphabetic character set in an alphameric keyboard.

alphameric characters: Same as alphabetic characters, with the addition of digits 0 through 9.

alphameric field: One or more alphameric characters of related information in a record.

arithmetic expression: An expression that contains arithmetic operations and that can be reduced to a single numeric value. An arithmetic expression is evaluated from left to right with multiplication and division preceding addition and subtraction.

alternate record advance: A function that causes the system to stop processing the current record and ignore any specifications between the cursor position and the end of the record when the Enter or Rec Adv key is pressed.

apostrophe: This character (') is used to enclose character strings such as 'NUMBER'. Two consecutive apostrophe characters are used to form an apostrophe in a character constant such as 'DRIVERS''S LICENSE'.

application: A unit of work for which the system will be used. For example, this unit of work can consist of entering data from source documents to do payroll for a small company.

application program: A program that processes user data to perform a particular data processing task; for example, inventory control or payroll.

arithmetic expression: An expression that contains arithmetic operations and that can be reduced to a single numeric value. An arithmetic expression is evaluated from left to right with multiplication and division preceding addition and subtraction.

ASCII: American National Code for Information Interchange. The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

assembler: A computer program that prepares an object program from a source program written in a symbolic source language.

assembler language: A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

attachment: An entire device or feature as attached to a processing unit, including any required adapters.

attention line: A hardware attention used by the microprocessors to communicate with each other.

attribute: A characteristic. For example, attributes of a data set include record length, label, and creation date. Attributes of a displayed field could include high intensity, reverse image, and column separators.

attribute byte: A control position that describes attributes to the system.

auto enter: A record advance function is automatically performed when the operator enters the last position of a record.

auto dup: Automatic duplication. (1) The process of automatically copying the contents of a field in a previous record or a storage area into the corresponding positions of the current record. (2) The process of automatically verifying the contents of a field in the current record with the contents of the corresponding positions of a previous record or a storage area.

auto record advance: Automatic record advance. A movement forward to the next sequential record without manual intervention when the current record is completely entered and the auto rec adv switch is on.

auto skip: Automatic skip. In enter mode, if the auto skip/dup switch is on, the process of automatically filling an auto skip field with blanks and advancing to the next field. In verify mode, the process of verifying that all the positions in the field are blank.

auto verify: Automatic verify. In verify mode, auto dup fields are checked against the same fields in the previous record. See *auto dup*, 2.

auxiliary duplication: The process of copying or verifying data from a named storage location into a field. For assembler programs, this is called main storage duplication.

awaiting field exit: The state of the keyboard when the operator has entered the last position of a field that is defined as a field exit required field.

awaiting record advance: The state of the keyboard when the operator has entered the last position of a record with a key other than the Record advance key, and the auto-enter function is not enabled.

background job: A job that is run in a partition which does not have immediate access to a keyboard/display unit.

base displacement addressing: An addressing method that involves setting up a base address from which other addresses can be calculated.

basic data exchange: A diskette data exchange that uses 128-byte sectors and allows only one record per sector. The logical record length must be $\leq$ 128 bytes and is unblocked and unspanned. The basic data exchange formats allow you to exchange data between 5280 and other systems that use the basic data exchange format.

binary: Base 2 arithmetic.

binary register: A 2-byte register in partition storage which contains binary notation and is used for binary arithmetic/logical operations.

binary search: At each step of the search the set of items is partitioned into two equal parts so that the search starts at the middle.

blank check: A check of a field to ensure that there are no blank characters (hex 40) in the field.

blank fill: To fill a field with blank characters (hex 40).

block: (1) A set of things, such as words, characters, or digits, handled as a unit. (2) A collection of contiguous records recorded as a unit. Each block can contain one or more records.

blocking: Combining two or more records into one block.

boundary alignment: The positioning of data areas such as registers or blocks, on an appropriate boundary for that type of data.

bps: Bits per second.

branch instruction: An instruction that changes the sequence in which the instructions in a computer program are executed. Execution of instructions continues at the address specified in the branch instruction.

BSC: Binary synchronous communications.

buffer: An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. See also *physical buffer, logical buffer, current record buffer,* and *refresh buffers.*

CAM: Communications access method.

CCR: Communications configuration record.

CCU: Communications configuration utility.

character constant: Any combination of characters, including blanks, enclosed in apostrophes.

collating sequence: The order each character holds in relation to other characters according to the bit structure.

column separators: A display screen attribute that shows vertical lines preceding each position of a field on a display. These lines do not occupy positions on the display. For example, ABC.

command keys: The 14 keys on the top row of the data station keyboard that are used with the Cmd key to request functions.

comments: Words or statements in a program that serve as documentation rather than instructions to an assembler or compiler.

common area: The first part of main storage that contains the system control area, common functions, global tables (such as ASCII and error recording), and so on. Depending upon the common function option selected, this area can be 6 K, 14 K, or 16 K. This area is not available for user programs.

common functions: A set of IBM-supplied programs in the common area that is used by programs executing in any partition.

communications access method (CAM): A 5280 program that provides the necessary link between a communications program and the communication line. It performs functions such as data formatting and link protocol.

communications adapter: A hardware feature that enables the 5280 to become a part of a data communications network.

communications configuration record: A record that describes the communications environment. This record is created by the communications configuration utility.

communications control block pointer: Contains the address of the communications control block (CCB) and flags.

concurrent: Pertaining to the occurrence of two or more activities within a given interval of time.

configuration: The group of machines, devices, features, and programs that make up a data processing system.

constant: A data item that does not change during the execution of a program. This item represents itself and is actually used in processing rather than being a field name representing the data. For example, 'cost' is a name representing a field containing data that changes. The constant 100 is actual data used that does not change.

control block: A storage area used by a program to hold control information.

controller: A device that controls operation of one or more input/output devices; for example, a 5285 Programmable Data Station.

copy: To read data from a source, leaving the source data unchanged, and to write the same data elsewhere in a physical form that may differ from that of the source.

counter: A register or storage location used to accumulate the number of occurrences of an event.

current record buffer: The I/O buffer that holds the current record during data input via a keyboard.

cursor: A movable horizontal line (underscore) on a display screen, used to indicate where the next character entered by the operator will appear. It blinks when no additional entry is allowed and the system is awaiting the Enter key.

cylinder: The tracks that can be accessed without repositioning the diskette drive access mechanism.

data-directed format selection: Format selection is determined by the data contained in the record.

data exchange: The ability to exchange diskettes and the data recorded on a diskette data set with a system or device that is different from the one recording the data.

data required: A field attribute that indicates an operator must enter at least one nonblank character into the displayed field.

data set: An organized collection of related data records treated as a unit and existing on a diskette.

data set label: A 128-byte area on the diskette index cylinder that describes a data set.

data set name: The name associated with a data set. The first character must be alphabetic, and the remaining characters can be any combination of alphabetic or numeric characters. Blanks cannot appear between characters in a name.

data stream: Data transferred by stream-oriented transmission, as a continuous stream of data elements in character form.

data table: A table defined by the .TABLE control statement.

decimal register: A 16-byte register wherein data is stored in EBCDIC or signed decimal numbers and is used for arithmetic/logical operations.

default value: A value automatically chosen by the system when a value is not specified by the user.

DE/RPG: Data Entry with RPG Subroutines. A 5280 program product that provides a means for writing programs that provide the function required for a specific job.

device address: Four hex characters used to identify a 5280 I/O device such as a diskette drive or printer.

device microprocessors: The microprocessors that control I/O devices, such as the keyboard/display microprocessor, diskette microprocessor, printer attachment microprocessor, and communications microprocessor. The device microprocessors are controlled by the main microprocessor.

diacritic: A modifying mark that changes the phonetic value of a character. When you enter a diacritic from the keyboard, the cursor does not advance until another character is entered to combine with the diacritic.

diacritic table: A table in keyboard/display storage that defines diacritic characters and valid diacritic-character composites for graphic display.

direct access method: An access method for processing files by specifying the address (record number) or key value of each record to be accessed.

direct addressing: A method of addressing in which the addressed storage location contains the desired data. See also *indirect addressing.*

direct by key access method: An access method for processing index data files by specifying the key associated with each record to be accessed. The current key specified need not have any relative sequence with the last key or next key to be specified.

diskette attachment: Controls the function for up to four diskette drives and includes the hardware adapter, a microprocessor, and read only storage (ROS).

diskette drive: The mechanism used to read and write diskettes.

diskette head: The device that moves a diskette past a read/write mechanism.

diskette labels: The header (HDR1) and volume (VOL1) labels that are recorded on a diskette to describe the data sets on the diskette.

displacement: The number of bytes from the beginning of a partition or block to the beginning of a particular data area.

display/alter: A diagnostic function that allows data in storage to be displayed and changed.

display attributes: The characteristics assigned to a field record that control the way the data is displayed.

display mode: The mode in which the prompts, display attributes, and the contents of the current record buffer are displayed, but the cursor is not displayed and no data can be entered. This mode is used to inspect prompts and display attributes of a screen format.

double register: Two decimal or binary registers used together as one data area. In a source program, the leftmost register is referenced, followed by the length in parentheses (4 for binary, 32 for decimal).

dup: Abbreviation for duplicate.

EBCDIC (extended binary-coded decimal interchange code): A character set containing 256 eight-bit characters.

edit format: A description of a record that is read from a diskette, written to a diskette or printer, or moved from one storage location to another. An edit format is set up by a FMT series of control statements in an assembler program, and defines the fields, punctuation, data types, and other editing requirements of the record.

ELAB/ETAB: Parameter in the .COMM and .DATASET control statements that specifies the name of a routine (ELAB) or table (ETAB) to be used to handle error or external status conditions.

enter mode: The mode in which the operator initially enters data through a display station. Some editing and interaction may occur. See also *verify mode; update mode.*

E-type data exchange: A diskette data exchange format that uses blocked and spanned, blocked and unspanned, or unblocked and unspanned records. Block size can be up to 16 256 bytes.

extent: A continuous space on a diskette that is occupied by or reserved for a particular data set.

external register: A register that is used by a microprocessor. External registers are not located within main storage and are not used by an application program.

extra line: Row 1 of the screen refresh buffer, which can be displayed on the top row of the screen in place of the status line.

field: One or more bytes of related information in a record.

field attribute: See *attribute.*

field correct mode: The mode in which the operator can enter data into a field during verification.

field separator: A blank character position preceding every field of an enter record. This position is required for the attribute byte.

fixed position prompt: A user-written message that appears on a specified row of the display screen. Contrast with *standard position prompt.*

foreground job: The keyboard/display unit is immediately available to the partition where the job is being executed.

format control string: The object code generated by source program edit format or screen format specifications.

format level: The identification associated with a format.

format 0 (zero): A screen format for display stations that allows entering information on an unformatted display.

formatted diskette: A diskette on which track and sector control information has been written but which may or may not contain data.

global load: A load operation that uses the standard load prompt. A global load is initiated by the system when the load parameters are not specified for a LOAD instruction in an assembler program, and when an error occurs when using the Standard Load Processor.

global table: A table in the common area. The first two global tables are the error recording tables. If the ASCII translate table is selected during system configuration, the ASCII translate table is another global table.

HDR1 label: Control information written on a diskette index that describes a data set on the diskette.

hex: Hexadecimal. A number system using 16 symbols: 0-9, A-F each representing 4 bits (one-half byte).

host computer: The primary or controlling computer in a data communications system.

H-type data exchange: A diskette data exchange format that uses 256-byte sectors. It allows only one record per sector. The logical record length must be 256 bytes; it is unblocked and unspanned. The H-type exchange allows you to exchange data between 5280 and other systems that use the H-type data exchange format.

ID: Identification.

index data set: A data set in which the keys from another data set and their record position within that data set are recorded. When index data sets are used, the following access methods can be used: sequential; direct by relative record number; and direct by key value.

index register: A register whose contents can be added to or subtracted from the operand address before or during execution of a computer instruction.

indexed address: An address that is modified by the content of an index register before or during the execution of an instruction.

indexed instruction: An instruction that requires address modification before the data byte is fetched from storage.

indirect addressing: A method of addressing in which the addressed storage location contains the address of the desired data. See also *direct addressing.*

initial program load (IPL): A sequence of events that loads the system programs and prepares the system for execution of jobs.

input data set: A set of records a program uses as source information.

input/output control block (IOB): A data area that may be used to pass the required information from the calling program to the input/output supervisor for data operations.

input record: A data record that is transferred to computer storage for processing.

insert field: A field not present in the enter record, but which will be inserted by the system and will be present in the output record.

insert mode: The mode, initiated by the Ins key, in which the operator can insert characters into a field at the current cursor position. The cursor, the character above the cursor, and all characters to the right of the cursor are shifted to the right.

instruction: A statement that specifies an operation to be performed by the computer and the locations in storage of all data involved in that operation.

IOB: Input/output control block.

IOB pointer: A 4-byte block in the system control area that contains the address of a device IOB and other information (such as, if the device is installed).

IPL: Initial program load.

I-type data exchange: A diskette data exchange format that uses 128-, 256-, 512-, or 1024-byte sectors. All records in a data set must be the same length. All records in the data set are blocked and spanned. The I-type exchange allows you to exchange data between the 5280 and other systems that use the I-type data exchange.

keyboard bit map: Control bits in the keyboard/display IOB that correspond to functions that are totally or partially processed by the keyboard/display microprocessor. An application program written in assembler language can set these bits to indicate that the corresponding function is to be handled by the application program.

keyboard/display storage: An area of control storage separate from main storage, which provides control information and refresh areas for processing keystrokes and for displaying characters on the screen.

keyword: A word, coded in source statements, that represents specific attributes and functions, and that is usually accompanied by a string of one or more parameters.

label table: A table of addresses set up by the .LABTAB control statement, and used for indexed branches and indexed subroutine calls.

label update: A data set type that allows the labels on the diskette index to be updated.

logical buffer: An I/O buffer that contains a logical record, used to block and deblock logical records in the physical buffer.

logical record: A record independent of its physical environment. Portions of the same logical record may be located in different physical records, or several logical records or parts of logical records may be located in one physical record, depending on the exchange type being used.

Magnetic Stripe Reader feature: Allows use of the 5280 system only after a valid badge (operator ID) is read by an attached magnetic stripe reader.

main microprocessor: The microprocessor that processes the object code instructions and controls the device microprocessors.

main storage: (1) General purpose storage of a computer. (2) Storage that can be addressed by programs, from which instructions can be executed, and from which data can be loaded directly into registers.

main storage duplication field: See *auxiliary duplication.*

main storage store field: A field that is automatically stored from the current record buffer into a main storage location.

make/break key: A key that generates a scan code when the key is pressed, and another when the key is released.

mandatory enter: A field attribute that indicates an operator must enter at least one character into the displayed field.

mandatory fill: A field attribute that indicates an operator must enter all or none of the displayed field.

mask: A pattern of characters that is used to control the retention or elimination of another group of characters.

microprocessor save area: Bytes marked as a microprocessor save area contain information that depends upon the operation being executed, and is therefore unpredictable. An application program must not change these bytes.

mode: The operational category of a data station. Modes for the 5280 include: enter, update, verify, rerun, rerun/display, field correct, and special verify.

multinational character set: The 188-character (or 184-character) display and printer character set available with the 5280.

multiprogramming: The concurrent execution of 2 or more programs (up to the maximum number of partitions) in which each program appears to be the only program in the system. Programs can have exclusive use of data sets and/or system I/O resources or can share them, depending upon the application requirements.

multivolume data set: A data set that extends beyond the boundaries of a single data set. It can be extended on the same diskette or on another diskette.

nest: To embed subroutines or data in other subroutines or data at a different hierarchical level such that the different levels of routines or data can be accessed or executed in a reentrant fashion.

nondisplay: A field attribute that prevents display of data. It can be used for fields containing confidential information.

null character: The hexadecimal character 00.

numeric fields: A field that contains one or more numeric characters. Valid numeric characters are the digits 0-9 and + (plus sign), - (minus sign), . (decimal point), blank, and , (comma).

numeric shift: A control (attribute or key) for selecting the numeric character set in an alphameric keyboard.

object code: The 4-byte instructions from the compiler or assembler that are machine executable. The first byte of the object code contains the operation code.

object program: A set of instructions in machine language (object code). The object program is produced by the assembler from the source program.

offset: The distance from the beginning of a register or record to the beginning of a particular field.

overlay: (verb) The act of one module being called on top of another to use the same space.

output data set: A data set containing the data that results from processing.

packed data field: One byte is used to store two numeric digits. Bits 0 through 3 for one digit and bits 4 through 7 for the other.

packed decimal format: Each byte within a field represents two numeric digits except the rightmost byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111. Contrast with zoned decimal format.

packed field: A field that contains data in the packed decimal format.

pad: To fill unused positions in a field with dummy data, usually zeros or blanks.

parameter: A field of a control statement or instruction.

partition: An area of 5280 storage in which a program can execute.

partition IOB: A control block that is stored in the first 128 bytes of a partition, and which describes the partition and the program that is loaded into the partition.

partition pointer: Contains the address of the beginning of a partition. The partition pointer also contains flags to indicate the status of the partition (such as whether the partition is a foreground or background partition).

partition stack pointer: See *stack pointer.*

physical buffer: An I/O buffer that contains a physical record.

physical record: A record whose characteristics depend on the manner or form in which it is stored, retrieved, or moved. A physical record may consist of all or part of a logical record, or more than one logical record.

port: An access point for receiving or transmitting data.

production statistics: Statistics related to activities occurring during key entry operation.

program listing: A computer printout that gives information about the source program, such as source statements, diagnostic messages, indicators used, storage addresses of fields and constants used.

program product: An IBM-written, licensed program for which a monthly charge is made. A program product performs functions related to processing user data.

prompt: A message issued by a program that requests either information or an operator action to continue processing.

reformatting: The rearrangement of an addition or elimination of fields in a record.

refresh: The continuous redisplaying of data on the display screen to prevent the data from fading out.

refresh buffers: Areas in keyboard/display storage that are used to refresh each row of display characters on the screen. The refresh area for the status line is in an area separate from the refresh area for the other rows on the screen.

relative addressing: A means of addressing instructions and data areas by designating their location in relation to the location counter or to some symbolic symbol. Relative addresses of areas within a partition are relative to the beginning of the partition.

relative record number: A number that specifies the location of a record in relation to the beginning of the data set.

rerun mode: An operational mode that allows the application program to perform the return-to-program (RG) exits within a screen format control string in a rapid fashion without operator intervention. No status line information, prompts, data, or display attributes appear on the display screen.

rerun/display mode: An operational mode that is the same as rerun mode except that the status line information, prompts, data, and display attributes appear on the screen so the operator can inspect the rerun data.

resource allocation table: A table in storage that is used to assign a logical device ID (a name) to a physical device.

return-to-program exit: See *RG exit.*

reverse image: The display attribute that causes characters to be displayed as dark characters on a light background.

RG exit: A user exit that interrupts the processing of a screen format to give controi to a user's routine.

right adjust: The placement of data in a register or field, or the shifting of the contents of a register or field, so that the least significant byte at the right end of the data is placed into the rightmost position of the register or field.

right justify: The adjustment of positions of characters so that the rightmost character entered is at the extreme right of a field.

SCP: See *system control programs.*

screen format: A description of a record that is entered via the keyboard/display. A screen format is set up by a SFMT series of control statements, and defines the fields, prompts, control specifications, and display attributes of the record.

screen format control string: The object code that is generated by a series of SFMT control statements.

SCS: SNA standard character string.

SCS conversion data set: A data set that has SCS conversion specified in the .DATASET control statement that defined the data set. The system automatically inserts SCS control characters into an SCS conversion data set.

SCS data set: A data set that contains SCS control characters. Contrast with *SCS conversion data set.*

SDLC: Synchronous data link control.

search argument: The data to be compared to specific parts of a record for a data set search operation, or to table entries for a table search operation.

sector: An area on a diskette track reserved to record a unit of data.

security: Prevention of access to or use of all or part of data or programs without authorization.

self-check field: A field, such as an account number, consisting of a base number and a self-check digit or digits. For data entry applications, the self-check digit or digits entered by the operator is compared to the self-check digit or digits computed by the system. If the operator makes a mistake when entering (keying) a self-check field, an error message is displayed.

sequential access method: An access method in which records are accessed in the order in which they occur in the file. Contrast with *direct access method.*

sequential by key: A method of data set processing that accesses records in the order in which a keyed or indexed data set is arranged.

SNA: Systems network architecture.

source program: A set of instructions that represents a particular job as defined by the programmer. These instructions are written in a programming language, such as DE/RPG.

spanned record: (1) A record that crosses a block boundary. (2) A record that is stored in more than one block.

stack pointer: The binary register (BR18) used for subroutine calls and returns. During a subroutine call, the stack pointer contains the address of the next available entry in the subroutine stack; during a subroutine return it contains the address of the last entry in the subroutine stack.

standard load prompt: The screen format stored in the common area that is used to prompt for load parameters during a global load or by the Standard Load Processor.

standard position prompt: A user-written message that can appear in any position on the display screen. Contrast with *fixed position prompt.*

status line: Usually, the first line on a display screen. This line provides operational information.

stripped zone: See *packed data field.*

subroutine stack: A table of return addresses used for subroutine returns.

subroutine stack pointer: See *stack pointer.*

Synchronous data link control (SDLC): A discipline for managing synchronous, transparent, serial-by-bit information transfer over a communications line.

syntax: (1) The structure of expressions in a language. (2) The rules governing the structure of a language.

system configuration: A process that specifies the various components and devices that form a particular operating system. System configuration combines user-specified options and parameters with IBM programs to produce a system having the desired form and capacity.

system control block: 256 bytes starting at address X'00'. This area contains information such as the address of each partition, device IOB pointers, system flags, machine storage size, and so on.

system control programming: IBM-supplied programs that are on a diskette. These programs are included with each system and allows the operator to configure the system, IPL the system, and recover from power failures.

system table: A table set up and used by the system to store the addresses of screen formats, edit formats, prompts, data tables, and duplicate or store fields.

system network architecture (SNA): A total description of logical structure, formats, protocols, and operation sequences for transmitting information throughout a communications network.

system use only: Bytes marked system use only should never be changed by an application program.

timeout: A time interval during which a station waits for a certain operation to occur. Some timeouts are automatic hardware functions and some are program functions.

trace: A record of the execution of a computer program; it exhibits the sequence in which the instructions were executed.

track: A circular path on the surface of a diskette upon which information is magnetically recorded and from which recorded information is read.

update mode: The mode in which the operator selects certain records for review and correction.

verify: To determine whether a transcription of data or other operation has been accomplished accurately.

verify bypass field: A field that was entered, but does not need to be verified.

verify mode: The mode in which the operator rekeys data from a source document that has already been keyed in order to check that the data has been entered correctly.

VOL1 label: Control information written on a diskette index that describes the volume, owner, security, and sequence of the physical records on the tracks of the specified volume.

zero fill: To fill with the numeric value zero.

zero suppress: The elimination of preceding zeros in a number. For example, 0057 becomes 57 when zero suppressed.

zone: The high-order 4 bits of a byte.

zoned decimal format: Representation of a decimal value by 1 byte per digit. Bits 0 through 3 of the rightmost byte represent the sign; bits 0 through 3 of all other bytes represent the zone portion; bits 4 through 7 of all bytes represent the numeric portion. For example, the decimal value +123 is represented as 1111 0001 1111 0010 1111 0011. Contrast with packed decimal format.

zoned field: A field that contains data in the zoned decimal format.

Index

IBM 5280
Distributed Data
System
Functions Reference Manual

GA21-9353-0

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

● No postage necessary if mailed in the U.S.A.

Name ______________________________

Company or
Organization ______________________________

Address ______________________________

____

Fold and tape   Please do not staple   Fold and tape

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape   Please do not staple   Fold and tape

IBM

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

IBM 5280 Distributed Data System Functions Reference Manual (File No. S5280-01) Printed in U.S.A. GA21-9353-0

GA21-9353-0