GC33-2008-0

1401/1440/1460 OS Emulator on Models 135/145/155

Systems

Reference

Program Number 360C-EU-735

OS Release 20

Third Edition (January, 1972)

This is a major revision of, and makes obsolete, GC27-6945-1 and Technical Newsletters GN27-1346 and GN33-7015. This edition describes version 2, modification 0, of the 1401/1440/1460 emulator and applies to Releases 20 and 20.1 of IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Before using this publication in connection with the operation of IBM Systems, consult the latest IBM System/360 and System/370 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM branch offices.

There is a reader's comment form at the back of this publication. If the form has been removed, or if it is more convenient for you, you may address comments directly to:

IBM France
Programming Publications
Department 841
06 - La Gaude
France

This publication describes the 1401/1440/1460 OS Emulator, an "integrated emulator" executed under control of OS MVT on System/370 Models 145 and 155, or of OS MFT on Models 135, 145, and 155. The publication contains the information you need to plan and use the emulator.

The information given here is that needed by (1) planners of hardware and programming installations, (2) programmers who prepare jobs to be run using the emulator, and (3) machine operators who actually run the jobs.

In addition to information on the emulator itself, this publication contains a description of three utility programs distributed with the emulator: two tape formatting programs and a disk formatting program. These programs are used to make original data formats compatible with System/370 data formats.

ORGANIZATION OF THE PUBLICATION

"What is Emulated and What is not Emulated" describes the 1401, 1440, and 1460 systems that are emulated. It also describes the minimum System/370 required, including main storage, auxiliary storage, and I/O devices.

"Emulator-1400 Differences" gives restrictions and limitations that you must consider when emulating a given system.

"Processing Data" shows how the emulator processes data on tape, disk, and unit record devices.

"Generating the Emulator" gives the information needed to generate an emulator for your 1401, 1440, or 1460 system.

"Executing the Emulator" shows how to use OS JCL statements and emulator control statements for an emulator job. It also gives information on cataloguing jobs and handling errors.

"Communicating with the Emulator" describes the commands and control statements the operator can use to communicate with the emulator.

Appendix A gives the correspondence between character codes used by the emulator and those used by OS and by 1401, 1440, and 1460 systems.

Appendix B contains the rules of syntax and the conventions used to describe emulator control statements and commands.

Appendix C describes a sample program that can be used to test the emulator. It also gives examples of JCL statements and emulator control statements used to execute the emulator.

Appendix D describes the tape formatting programs, used to convert tape files. Both the tape preprocessor program and the tape postprocessor are described in detail.

Appendix E describes the disk formatting program, used to prepare System/370 disk packs to receive 1401, 1440, or 1460 disk files.

Appendix F gives the information you need in order to add an emulator routine you have written.

Appendix G contains the messages issued by the emulator, the tape formatting programs, and the disk formatting program. Messages issued by these programs are not in Messages and Codes, GC28-6631, but there is a section reserved for them. You can remove the appendix and put it in Messages and Codes.

Following Appendix G is a glossary of terms used in this publication.

REFERENCES

In addition to the information in the reference manuals of the emulated system (1401, 1440, or 1460), you should be familiar with the information in:

IBM System/360 Principles of Operation, GA22-6821

IBM System/370 Principles of Operation, GA22-7000

IBM System/360 Operating System:

Supervisor Services, GC28-6646

Data Management Services, GC26-3746

The following publications provide additional material on subjects discussed in this publication.

IBM System/360 Operating System:

Job Control Language Reference, GC28-6704

MFT Guide, GC27-6939

MVT Guide, GC28-6720

Tape Labels, GC28-6680

Operator's Reference, GC28-6691

System Generation, GC28-6554

Storage Estimates, GC28-6551

Programmer's Guide to Debugging, GC28-6670

System Programmer's Guide, GC28-6550

Utilities, GC28-6586

The Program Logic Manual (PLM) that describes the logic of this program is 1401/1440/1460 OS Emulator on Models 145/155, GY33-7011.

FIGURES

The integrated emulator combines a compatibility feature (a set of microprogrammed instructions) with a complementary program that is executed under control of an operating system. The complementary program is the 1401/1440/1460 emulator program, which is referred to as the "emulator" throughout this publication. Using the emulator, programs written for 1401, 1440, and 1460 Data Processing Systems can be run on the System/370 Models 135, 145, and 155.

The predecessor of the integrated emulator is the stand-alone emulator. A stand-alone emulator has the same characteristics as an integrated emulator, except that it is not executed under control of an operating system.

Throughout this publication, the term "1400" is used when discussing characteristics common to 1401, 1440, and 1460 Data Processing Systems. The term "partition" is used to identify either an MFT partition or an MVT region.

The emulator helps you convert from a 1401, 1440, or 1460 Data Processing System to the System/370. Most 1400 programs can be executed under control of OS without reprogramming if they were written according to programming principles in IBM reference manuals. Because some 1400 features and operations are not emulated, certain programming restrictions and limitations must be considered when planning.

All 1400 card, tape, and disk systems are emulated, except for the 1401 Model G. Special consideration must be given to tape and disk programs because their data formats are different from System/370 data formats. Data files may need to be converted before they are used with the emulator.

Tape files can be used in 1400 format or in the standard operating system data format. The format used by the operating system is the spanned variable-length record format, henceforth called spanned format. Disk files created by other emulators or by 1400 systems must be converted before they are used with the emulator.

The emulator is executed as a problem program under control of OS MFT or MVT. Two sets of control statements are required: JCL statements to describe the emulator, and emulator control statements to describe the 1400 program and the devices it uses.

Note: MVT is not available on the Model 135.

The operating system provides data management services and allocates resources and devices. Both 1400 jobs and System/370 jobs are placed in a single input stream and are executed concurrently. Thus, multiprogramming and tasks that run indefinitely, such as telecommunications and graphics tasks, can continue during emulation.

You generate the emulator by assembling a set of macro instructions and then link editing the modules that are created into an emulator load module. The EM1401 macro instruction describes the 1400 system that is to be emulated. The remaining macro instructions are called by the EM1401 macro instruction to create the Stage I and Stage II job streams of emulator generation. You can assemble and catalog any number of emulators. The characteristics of each emulator will depend on your description in the EM1401 macro instruction. For example, you can assemble one emulator for a 1401 with 8,000 positions of core storage and four tape units, and another for a 1460 with 16,000 positions of core storage and six tape units.

THE COMPATIBILITY FEATURE

To emulate a 1400 system, the System/370 processor unit must be equipped with a compatibility feature, which is a set of seven microprogrammed routines. These routines execute the most frequently used instructions and operations of the 1400 system:

- I-Fetch, address translation, indexing, address register storing (Models 135 and 145 only), error checking, address verification, and certain branching operations
- Arithmetic operations (except multiply and divide), including Modify Address
- Comparisons
- CPU data moves

- I/O data moves

- Store A-Register (SAR), on Models 135 and 145

- Store B-Register (SBR), on Models 135 and 145

The emulator uses these routines as well as the instruction set and data management services of the operating system to emulate:

- 1400 CPU operations

- 1400 I/O operations

- Operator services and console operations

The compatibility feature fetches and analyzes each 1400 instruction. It executes some instructions directly; for others, it passes control to the emulator, which emulates the instruction and returns control to the compatibility feature to fetch and analyze the next 1400 instruction.

The compatibility feature makes the emulator program fully relocatable (no fixed addresses), thus allowing the emulator to be executed using multiprogramming. Although no fixed main-storage addresses are used, 1400 storage is located in consecutive locations of System/370 main storage. Because the compatibility feature is reenterable, two or more 1400 emulators can be executed concurrently.

You may use one of three compatibility features. Your choice of feature depends on the System/370 processor unit you have and the 1400 systems you want to emulate:

- Compatibility feature #4457 emulates 1401, 1440, and 1460 systems on the Models 135 and 145.

- Compatibility feature #4458 emulates 1401, 1440, 1460, 1410, and 7010 systems on the Model 145.

- Compatibility feature #3950 emulates 1401, 1440, 1460, 1410, and 7010 systems on the Model 155.

All compatibility features are functionally the same. For more information about the compatibility features and a description of the instructions, see Appendix F.

THE TAPE FORMATTING PROGRAMS

Because of differences in record formats and data representation on 1400 tapes and System/370 tapes, it is often advantageous and sometimes necessary to reformat data on tape before executing a 1400 program. To do this, two tape formatting programs are available: the tape preprocessor program and the tape postprocessor program. These programs are supplied with the emulator, and are run as problem programs under control of OS MFT or MVT.

The tape preprocessor program converts data files in 1400 format to data sets in spanned format. The files are in mixed density or when the records are more than 32,755 bytes long. The tape preprocessor can read physical records from 1 to 200,000 bytes long, and write physical records from 18 to 32,755 bytes long.

The tape postprocessor program converts data sets in spanned format written by the emulator to data files in 1400 format. The data files can be used by 1400 programs executed on 1400 systems. The tape postprocessor can read physical records from 18 to 32,755 bytes long, and write physical records from 1 to 200,000 bytes long.

THE DISK FORMATTING PROGRAM

The track organization of a 1400 disk file must be retained when the file is used with the emulator. The disk formatting program creates a System/370 data set of fixed-length blank records. Each record will hold an entire track of data from the 1400 disk. Once these blank records have been created, a 1400 utility program must be used to write the 1400 tracks onto the formatted System/370 disk pack.

PROCESSING UNITS

1400 systems with 1,400 to 16,000 positions of core storage are emulated.

- For the 1401 Data Processing System, 1401 Models A through F and Model H.
- For the 1440 Data Processing System, 1441 Models A2 through A6
- For the 1460 Data Processing System, 1441 Models B4 through B6

The 1401 Model G processing unit is not emulated because its addressing scheme differs from that of the other processing units.

FEATURES EMULATED

- High-Low-Equal Compare
- Multiply-Divide
- Sense Switches
- Advanced Programming (Indexing, Store Address Register, and Move Record)
- Bit Test
- Expanded Print Edit
- Inverted Print Edit
- Processing Overlap
- Print Storage
- Additional Print Control
- Space Suppression
- Selective Stacker
- Preferred Character Set and Numerical Print (if the System/370 printer has the Universal Character Set)
- 24 Additional Print Positions on the 1443 printer if the corresponding feature is on the System/370 printer

FEATURES NOT EMULATED

CPU Features

- Column Binary
- Binary Transfer
- Compressed Tapes
- Read Punch Release
- Translate (1460)

I/O Features

- Punch Feed Read
- Card Image
- Punch Column Skip
- Selective Tape Listing

1400 FEATURES CONSIDERED STANDARD BY THE EMULATOR

The emulator assumes that certain features of 1400 systems are present in the system to be emulated. These features are:

- High-Low-Equal Compare
- Sense Switches
- Print Storage
- Advanced Programming: for branching operations, the B-address register contains the address of the next sequential instruction instead of blanks (Indexing)

A 1400 program dependent on the absence of one or more of these features must be modified.

DEVICES EMULATED

Unit Record Equipment

- 1402 Card Read Punch (1401, 1460)
- 1442 Card Read Punch (1440)
- 1442 Card Reader (1440)
- 1403 Printer

- 1443 Printer
- 1407 Console Inquiry Station
- 1447 Console

Tape Devices

- 729 Magnetic Tape Unit, Models II, IV, V, and VI
- 7330 Magnetic Tape Unit (1401, 1460)
- 7335 Magnetic Tape Unit (1440)

Disk Devices

- 1301 Disk Storage Drive
- 1311 Disk Storage Drive, Models 11 and 12
- 1405 Disk Storage, Models 1 and 2

DEVICES NOT EMULATED

Unit Record Equipment

- 1402 Card Read Punch (1440)
- 1404 Printer
- 1442 Card Read Punch (1401, 1460)
- 1444 Card Punch
- 1445 Printer
- 1011 Paper Tape Reader
- 1012 Paper Tape Punch
- Optical readers
- Magnetic character readers
- Multiple readers, punches, or printers

Tape Devices

The 7340 Hypertape Drive is not emulated.

Disk Devices

Only one module of the 1301 is emulated during the execution of any one program.

Other Devices

Teleprocessing devices are not emulated.

I/O DEVICE CORRESPONDENCE

A 1400 program executed under control of emulator can request I/O operations on various System/370 devices. Figure 1 shows which devices can be used to emulate 1400 devices.

Figure 2 summarizes the track and cylinder correspondence of 1400 disks and of System/370 disks using the Record Overflow feature.

Figure 3 summarizes the track and cylinder correspondence of 1400 disks and of System/370 disks without the Record Overflow feature.

MINIMUM SYSTEM REQUIRED

To emulate a 1400 system, you must have:

- The 3135 Processing Unit equipped with compatibility feature #4457, the 3145 Processing Unit equipped with compatibility feature #4457 or #4458, or the 3155 Processing Unit equipped with compatibility feature #3950.
- The MFT control program on the Model 135, or the MFT or MVT control program on the Model 145 or 155.
- In addition to the I/O devices needed by the operating system, System/370 I/O devices to emulate the 1400 devices.

MAIN STORAGE REQUIRED FOR THE EMULATOR PARTITION

EMULATOR ROUTINES

The main storage required for emulator routines depends on the options chosen when the emulator is generated, and is given in Figure 4.

Storage needed for optional I/O routines is reserved at emulator generation if you specify the IODVTYP operand of the EM1401 macro instruction (see "Generating the Emulator").

1400 I/O Device[1]	System/370 I/O Device
1402 Card Read Punch 1442 Card Read Punch 1442 Card Punch	Any card reader, card read punch, tape unit, or disk unit accepted by QSAM (if your emulator has the Selective Stacker feature and your 1400 program uses the Select Stacker instruction, only the 2540 Card Read Punch can be used)
1403 Printer 1443 Printer	Any printer, tape unit, or disk unit accepted by QSAM
729 Magnetic Tape Unit, Model II, IV, V, and VI 7330 Magnetic Tape Unit 7335 Magnetic Tape Unit	Any tape unit or disk unit accepted by BSAM
1407 Console Inquiry Station 1447 Console	Any operator's console accepted by the operating system
1301 Disk Storage 1311 Disk Storage Drive, Models 11 and 12 1405 Disk Storage, Model 1 or 2	Any disk unit accepted by BDAM[2]

[1] Programmed reading from more than one reader, printing on more than one printer, or punching on more than one punch is not emulated.
[2] If two or more System/370 disk units emulate a 1400 disk device, all System/370 disk units must be the same type.

Figure 1. I/O Device Correspondence

Emulated Device	1400 Tracks Per System/370 Track			Number of System/370 Cylinders Required		
	2311	2314	3330	2311	2314	3330
1301 Disk: Sector Mode	1.5	3	5.7	634	161	92
1301 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2.5	4.8	744	188	10
1311 Disk: Sector Mode	1.5	3	5.7	64	17	9
1311 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2	4.2	87	22	12
1405 Disk, Model 1	3	6	11	316	82	47
1405 Disk, Model 2	3	6	11	632	163	93

Figure 2. Track and Cylinder Correspondence Using the Record Overflow Feature

Special features are optional and are part of the emulator only if you specify them in the CPUOPTN operand of the EM1401 macro instruction. (For a list of emulated devices and features, see the beginning of this section.)

The minimum storage required for the emulator is 21,354 bytes for a system with only unit record devices, excluding the 1407 or 1447 Console. The maximum storage required for the emulator is 36,996 bytes for a system that includes the entries in Figure 4, with magnetic tape units using both preprocessed and 1400 format tapes and 1301 and 1311 disk units.

Storage required for buffers for each 1400 job is computed separately, and is discussed below.

Emulated Device	1400 Tracks Per System/370 Track			Number of System/370 Cylinders Required		
	2311	2314	3330	2311	2314	3330
1301 Disk: Sector Mode	1	3	5	1000	167	105
1301 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2	4	1000	250	132
1311 Disk: Sector Mode	1	3	5	100	17	11
1311 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2	4	100	25	14
1405 Disk, Model 1	3	6	11	334	84	48
1405 Disk, Model 2	3	6	11	667	167	96

Figure 3. Track and Cylinder Correspondence Without the Record Overflow Feature

EMULATOR ROUTINES	STORAGE (in bytes)[1]
Basic CPU emulation routines	6,446
Operator-emulator communications	8,140
Basic I/O emulation routines (unit record only, excluding routines for selective stacker and 1407 and 1447 Consoles)	4,229
Initialization	2,539
Basic emulator (sum of entries above)	21,354
I/O emulation routines - 1407 or 1447 Console, magnetic tape units, and disk devices	500
1407 or 1447 Console	950
Magnetic tape units[2]	
• Both preprocessed and 1400-format tapes	8,140
• 1400- format tapes only	4,820
Disk devices	
• 1301 and 1311	3,050
• 1405	1,980
Expanded Print Edit	323
Advanced Programming	355
Multiply-Divide	804
Selective Stacker	1,520

[1] Does not include variables such as 1400 storage, OS data management routines, or I/O buffers.

[2] Select one.

Figure 4. System/370 Storage Needed for Emulator Routines

DATA MANAGEMENT ROUTINES

Main storage for data management routines can vary from approximately 2,000 bytes to 10,000 bytes. The 10,000-byte estimate is for all three access methods used by the emulator: the Basic Sequential Access Method (BSAM), the Queued Sequential Access Method (QSAM), and the Basic Direct Access Method (BDAM). The 2,000-byte estimate is for QSAM only. The control program and the number and type of I/O units that are in use at one time also affect the storage required. Estimates of data management storage can be obtained by using the formula given for each access method in <u>Storage Estimates</u>, GC28-6551.

EMULATED 1400 STORAGE

One byte of System/370 main storage is needed to emulate one position of 1400 storage. For example, 16,000 bytes of System/370 main storage are needed to emulate 16,000 positions of 1401 core storage.

BUFFERS

Main storage required for buffers for unit record equipment, tape units, and disk units depends on the number and types of units being emulated.

<u>Unit Record and Tape</u>

The following formula is used to determine the size of buffers for unit record equipment and tape units used by the emulator:

Buffer Size = (B1•L1)+(B2•L2)+...
+(Bn•Ln)+48D+LCHAN

where:

B (B1 through Bn)
is the number of buffers per device. This value is specified in the BUFNO parameter of the DCE operand of the DD statement. If BUFNO is not specified, one buffer is assigned to each tape unit and two buffers are assigned to the printer, to the card punch, and to the card reader. If a 2540 Card Read Punch has the Punch Error Recovery feature, three buffers are assigned instead of two. If the device is a console, B is 1. (There is no DD statement for a console.)

<u>Note</u>: Do not include tape units using a channel buffer.

L (L1 through Ln)
is the length of each buffer in bytes. For the printer, one byte is required for the carriage control character. Do not include the channel buffer.

D
is the number of 1400 tape and unit record devices used by the 1400 program. A maximum of ten devices can be emulated; six tape units, a printer, a card reader, a card punch, and a console.

LCHAN
is the length of the channel buffer.

<u>Example</u>: This example is a calculation of the main storage needed for buffers for a 1400 system having:

- One card reader, with two 80-byte buffers
- One printer, with two 133-byte buffers (including one byte for carriage control)
- A console, with a 126-byte buffer
- Two tape units, each with two 1000-byte buffers

The maximum main storage needed for buffers for this system would be:

(2•80)+(2•133)+(1•126)+(2•1000)
+(2•1000)+(48•5)+0=4792 bytes

For additional information on the size of data records for tape units, see "Processing Data on Tape" in the section "Processing Data."

<u>Disk</u>

If you are emulating disk units, you may share buffers between two or more emulated access mechanisms, or you may assign one buffer to each. You assign shared buffers using two or more DISK emulator control statements with the same buffer number in the BUFID parameter. (For a description of the DISK emulator control statement, see the section "Executing the Emulator.")

For best emulator performance, you should assign one buffer to each access mechanism to be emulated. If main storage is limited, you may assign shared buffers.

The main storage needed for buffers is determined by the number of bytes required for one record after the disk formatting program formats the tracks for the emulated disk unit. Buffer lengths are listed in Figure 5 by disk device. If the buffer is to be shared by two or more devices, the buffer length for the device needing the largest buffer should be used to determine the storage required.

The formula for determining the size of buffers for disk units is:

Buffer Size = (L1+16)+(L2+16)+...
+(Ln+16)+40D

where:

L (L1 through Ln)
is the length in bytes of each buffer.

D
is the number of the 1400 disk devices used by the 1400 program. A maximum of six devices can be specified.

Example: This example gives the main storage needed for buffers for a system with:

- One 1301 Disk Storage, formatted for both track and sector modes, with 2555 bytes per data track
- One 1311 Disk Storage Drive, formatted for sector mode, with 2164 bytes per data track

If there is one buffer per device, the main storage needed for buffers would be:

(2555+16)+(2164+16)+(40•2)=4831 bytes

If one buffer is shared by two devices, the main storage needed for the buffer would be:

(2555+16)+(40•2)=2651 bytes

For additional information on data records for disk devices, see Appendix E.

AUXILIARY STORAGE REQUIRED

The options chosen and the number of emulators generated determine the auxiliary storage required. The auxiliary storage must be in the SYS1.LINKLIB data set, and is shown in Figure 6.

In addition to SYS1.LINKLIB, you need one-tenth of a 2311 track in the SYS1.SVCLIB data set for SVC 88, regardless of the number of emulators generated.

1400 Device	Buffer Length in Bytes
1301 Sector Mode	2,164
1301 Track-Record, or Both Track-Record and Sector Mode	2,555
1311 Sector Mode	2,164
1311 Track-Record or Both Track-Record and Sector Mode	2,992
1405, Model 1 or 2	1,044

Figure 5. Buffers Needed to Emulate 1400 Disk Devices

	Minimum No. of Tracks			Maximum No. of Tracks		
	2311	2314	3330	2311	2314	3330
First emulator, including the tape formatting programs and the disk formatting program	12	6	4	16	8	5
Each additional emulator	6	3	2	10	5	3

Figure 6. Auxiliary Storage Required for the Emulator (SYS1.LINKLIB)

PERFORMANCE INFORMATION

The execution time of an emulator job is affected by:

- The 1400 program:

  A. The mix of 1400 CPU and I/O instructions. CPU instructions, in general, are executed faster than I/O instructions.

  B. The percentage of 1400 instructions completely and partially emulated by the compatibility feature. Using the compatibility feature speeds emulation.

- The emulator program:

  A. Preprocessed tapes (spanned format). Using them is generally faster than using tapes in 1400 format.

  B. Processing overlap. Although it speeds execution of 1400 programs on the 1400 system, it slows emulation.

  C. Assigning a relatively high priority to the emulator job. This reduces interruptions from higher priority partitions.

  D. Blocking data records. This reduces the number of I/O operations.

  E. Specifying two buffers instead of one for tape[1] and unit record devices, and having one buffer per disk device rather than sharing buffers improves performance.

  F. Using I/O devices that have high data transfer rates improves performance.

- The operating system:

  A. Making access methods resident speeds emulation.

  B. Specifying the most efficient set of options at system generation speeds emulation (for example, specifying Multiple Console Support and enough WTO buffers for emulator messages).

TIME-DEPENDENT PROGRAMS

Because 1400 programs are executed under control of the emulator, they are not given control when an interruption occurs. Programs that manipulate data during the time required for mechanical movement of an I/O device, or those that test the first or last character of an input buffer to determine when data has been moved, need special emulator control to be executed properly.

The DILCNT parameter in the emulator control statements for tape and unit record equipment can help run time-dependent programs successfully. DILCNT causes the emulator to execute a given number of 1400 instructions while a 1400 read or write instruction is being executed. This allows you to execute programs that:

- Issue a write instruction for a tape unit and then prepare the area from which the data is to be written

- Issue a read instruction and continue processing the previous record in the same area into which the new record is to be read

An incorrect value assigned to DILCNT could cause unexpected results, or could make it impossible to execute a time-dependent program. If DILCNT is not specified in the emulator control statement, a default value for it is assumed. The emulator does not use DILCNT unless the 1400 program uses processing overlap.

PROGRAMMING DIFFERENCES

When choosing 1400 programs to be executed by the emulator, remember that:

- Programs that depend on the absence of a particular feature may not be executed properly.

- Programs that depend on error conditions may not be executed properly.

----

[1] Using two buffers increases the speed of tape operation when backspace instructions are infrequent. Backspacing operations are most efficient using a single buffer.

- Process checks are not emulated.

- The emulator accepts only the characters in the 64-character BCD set. Invalid characters in input are replaced by asterisks in emulated 1400 storage.

- The result of a multiply or divide operation may not be correct if the A-field and B-field overlap, and if the result alters any portion of the A-field.

- In 1400 systems, wordmarks in the B-field of a Divide instruction were ignored but retained. Under emulation, the first five wordmarks are retained; all others are cleared.

- The indexing feature on the 1400 is considered to be standard for all 1400 branch instructions. The emulated B-address register contains the address of the next sequential instruction instead of blanks.

- If a 1401 system has only 1,400 positions of core storage, the results of an address check caused by storage wraparound will be unpredictable.

- Programs with undetected programming errors may give unpredictable results.

- A substitute blank in 1400 storage is not converted to a blank on tape when emulating write operations for nine-track tapes in even-parity normal mode. A substitute blank on tape is converted to a blank in 1400 storage on all even-parity read operations. In all other cases, the substitute blank and the blank are treated as they were in the 1400 system.

- Programs that depend on the timing of I/O operations may give unpredictable results.

- Programs that depend on sensing the end-of-tape reflective strip are not executed properly if the data is in spanned format. These programs are executed properly when the data is in 1400 format (see "Processing Data").

- Programs that depend on the device-busy indicator being on may not be executed properly.

- When the emulator executes 1400 instructions and the transfer of data stops because the end of core is reached, the B-address register will not contain the end-of-core address as it did in 1400 systems. The B-address register cannot point to a location outside emulated 1400 storage, so it is changed to point to the beginning.

- The Data Converter feature must be installed on the control unit of any 2400-series tape unit that will be used to read or write EBCDIC data on seven-track tapes.

I/O OPERATIONS

This section deals with factors that must be considered before I/O devices are emulated.

TAPE

All read, write, and control operations are emulated. The emulator accepts and produces data in odd, even, and mixed parity on seven-track and nine-track tapes and in tape format on disk.

Occasionally, a 1400 file on one reel of tape does not fit on a System/370 volume after being preprocessed or after a file is created by the emulator. Although the preprocessor and emulator create a second tape when this occurs, any 1400 program that backspaces to rewrite or reread data will be unable to backspace past the beginning of the tape that is currently being used. If this occurs, the tape preprocessor or the emulator must be rerun to re-create the tape at a higher density.

Note: The 1400 program can also backspace when the data is stored on a direct access device (without the Record Overflow feature). Data records that have been split by the Record Overflow feature make backspacing unpredictable.

Tapes in spanned format cannot be used by System/370 programs other than the emulator, because 1400 tapemarks, header labels, and trailer labels are changed into spanned records. To use these tapes with other programs, you must either rewrite the tapemarks and labels or modify your program to recognize them. For additional information, refer to Appendix D and "Processing Data."

When using preprocessed tapes, changing from write to read operations on the same tape is normally not accepted by the operating system, except for the sequence write-backspace-read. Any other sequence (examples are write-write-read-read, write-backspace and rewind-write-read, and write-backspace and rewind-read-read) may cause the operating system to stop the emulator with a condition code of 337. All sequences are executed properly when the records are in 1400 format.

DISK

All disk operations are emulated for:

- Up to five 1311 Disk Storage Drives, Models 11 and 12, and one module of 1301 Disk Storage

- One 1405 Disk Storage, Model 1 or 2

You cannot generate the 1311 or the 1301 in the same emulator with the 1405. Furthermore, only one module of the 1301 can be emulated at a time.

Although all disk operations are emulated, there are some qualifications:

- Only standard sequential sector addresses can be written for the operations Read Disk Track Sectors With Addresses and Write Disk Track Sectors With Addresses.

- Scan Disk operations (File Scan) do not require the Hardware File Scan feature.

- Data written in load mode on a direct access device is not in standard EBCDIC; it must be translated to EBCDIC before being processed by programs other than the emulator (see "Character Codes" in the section "Processing Data").

UNIT RECORD

Operations Emulated

The following instructions are emulated:

- Read a Card

- Punch a Card

- Write a Line

- Write a Line and Suppress Space

- Write Word Marks

- Read from or Write on Console

- Carriage Control

- Read and Punch

- Write and Read

- Write and Punch

- Write, Read, and Punch

- Select Stacker

- Select Stacker and Branch

All overlapped operations are emulated. If a feature is emulated, all operations for that feature are emulated unless otherwise stated.

Operations Not Emulated

The following operations are not emulated:

- Branch on Buffer Busy (1447)

- Punch and Stop (1442 - The punching is emulated, but the emulator does not stop in the middle of a card.)

- Read and Punch Same Card (1442)

- Punch-Column-Skip (1442)

- Branch on Printer Carriage Busy

- Branch on Printer Busy (when 1400 program is not in overlap mode)

Factors to Consider

Certain factors must be considered when unit record operations are emulated:

- The emulator does not use locations 000 and 100 in 1400 storage to control the timing of read and punch operations. The contents of these locations is undisturbed.

- There are differences between the graphics printed by the 1407 and 1447 consoles and those printed on System/370 consoles. These differences are shown in Figure 7. Alphabetic characters can be typed in upper or lower case except for the characters g, p, and x. When the operator types lower case g, p, and x, the emulator transmits ?, !, and ‡ to the 1400 program.

- There are differences between the AN, HN, and PN print chains used on the 1403 and 1443 printers. Figure 8 shows the codes that need to be translated or that have different graphics.

- More than one 1400 data file can be entered on the card reader by reassigning the reader after every file is processed. The operator reassigns the reader by typing the UR emulator control statement on the console. This procedure may be followed to assign more than one output file on the punch or the printer (see "Typing Emulator Control Statements From the Console" in the section "Communicating With the Emulator").

1407 Console Character	1447 Console Character	Corresponding System/370 Character
⌑	⌑	<
(	[	(
<	<	+
‡	‡	|
)	]	)
△	△	¬
=	ᵐ	_ (underscore)
'	\	>
"	⧻	?
¢	ƀ	:
:	:	'
>	>	=
√	√	"
?	?	g
!	!	p
ǂ	ǂ	x
b	blank	space

Figure 7. Differences in Console Graphics

- A groupmark wordmark in 1400 storage always stops input or output on the console. On the 1400 instruction Read from Console, the keyboard does not lock when a groupmark wordmark is found in 1400 storage, but the transfer of data stops and the inquiry-clear indicator is set to show that the length of data was incorrect.

- A Write on Console instruction types up to 50 characters per line until a groupmark wordmark is found in 1400 storage. The message "IIQ003I jobname R" indicates that the output was started by the Write on Console instruction. A hyphen is typed at the end of the line if another line will be typed.

- When emulating the Write on Console instruction in load mode (on the 1407), characters with wordmarks are not printed in red. Instead, an underscore is placed before the character with the wordmark. The load mode blank is typed and printed as an underscore followed by a space, and not as the character b.

- The Read from Console instruction allows the operator to enter data from the console. The message "IIQ002A jobname I" is displayed to show that inquiry requests can be processed. Multiple lines of input can be read from the console when the operator types a hyphen as the last character of data in a line. The hyphen is not included as part of the data (see "Communicating with the Emulator").

- In load mode, typing one or more word separators (underscores) in front of a valid character causes a wordmark to be associated with the character in 1400 storage.

- The control character for console carrier return (]) is emulated, but the character for tabulation ([) is not.

- Card codes accepted by the 1400 card readers and punches are accepted by the emulator. Certain multiple-line print (MLP) codes are accepted and are passed to the 1400 program without the 8-9 punch (see Appendix A).

- Whenever the emulator encounters an error condition, it sets an error indicator and issues a message to the operator. If the job was not terminated by the emulator, the operator may either alter the invalid character or the invalid address and continue the job, or end the job.

- The Write Wordmark instruction is executed as it was on the 1460 system; a wordmark is printed as 1, a groupmark as 2, and a groupmark wordmark as 3.

- If a 1440 program issues a Read instruction to eject the last card when closing a 1442 file, one blank card must be placed after the last data card for each Read instruction.

- If a 1440 program issues a Punch instruction for the 1442, it may issue one or more Read instructions to position blank cards at the punch station. In this case, there must be a data set opened for the reader and enough blank cards at the appropriate spot in the reader data set.

- The channel-1 punches and the length of the carriage control tape on the System/370 printer must correspond to those defined by the CCTL emulator control statement. Punches in other channels need not correspond. The CCTL emulator control statement is defined under "How to Prepare Control Statements" in the section "Executing the Emulator."

- If the emulator is to have the Selective Stacker feature, a module is generated that uses an EXCP macro instruction to emulate the feature. This module uses QSAM to emulate all other unit record operations. On the 1400 system, the pocket selected by the Select Stacker instruction is determined by the last K operation code encountered during the ten-microsecond interval following a read or punch operation; during emulation, however, when STACKER=YES is coded, the pocket is determined by the last K operation code encountered before the next read or punch operation. (When STACKER=YES is coded and no K operation code is encountered, the first pocket is always selected.) Therefore, programs that depend on the ten-microsecond interval may not be executed properly.

			System/370 Printer Graphics	
Card Code	BCD Graphic Symbol	Hexadecimal Code Sent to Printer	AN or PN Chain	HN Chain
12-4-8	⌑)	4C	⌑[1]	)
12-5-8	[	40	blank	blank
12-6-8	<	40	blank	blank
12-7-8	⯒	40	blank	blank
12	& +	50	&	&
11-5-8	]	40	blank	blank
11-6-8	;	40	blank	blank
11-7-8	△	40	blank	blank
0-4-8	% (	6C	%	(
0-5-8	⩘	40	blank	blank
0-6-8	\	40	blank	blank
0-7-8	⧻	40	blank	blank
2-8	ƀ	40	blank	blank
3-8	# =	7B	#	=
4-8	@ '	7C	@	'
5-8	:	40	blank	blank
6-8	>	40	blank	blank
7-8	√	40	blank	blank
12-0	?	50	&	&
11-0	!	60	-	-
0-8-2	⧧	4E	+	+

[1] On the PN chain, card code 12-4-8 is printed as a < rather than a ⌑.

Figure 8. Differences in Printer Graphics

USING MULTIPLE CONSOLE SUPPORT

Multiple Console Support (MCS), an option of OS, lets operators communicate with OS from more than one console. The emulator uses MCS to communicate with one or all of the consoles. All emulator messages to the operator are given a routing code of 12 and a descriptor code of 7. Emulator messages can be sent to one or several consoles simply by permitting consoles to receive messages with a routing code of 12. (All tape formatting program and disk formatting program messages are assigned routing codes of 11 and 12 and a descriptor code of 7.)

Emulator messages are automatically deleted by the operating system if, at the end of the job step, they have not been issued or if a reply is outstanding.

The text field of an emulator message begins with the job name of the emulator job. The job name helps the operator identify the emulator that issued the message when more than one emulator is being executed. For example, message "IIQ061D jobname EMULATOR WAITING" is issued when the emulator cannot continue without information from the operator. If two emulator jobs are being executed at the same time, there may be two IIQ061D messages that have different reply identifications and job names. If two consoles can receive messages with a routing code of 12, there will be two IIQ061D messages on each console, and either operator can respond to either message.

When more than one console receives messages with routing code 12, you can eliminate message duplication by:

- Writing an MCS user-exit routine to change routing code 12 to routing code 13, 14, or 15. You can use the job name to identify the programs that should have their routing code changed. Information on MCS user-exit routines in the <u>System Programmer's Guide,</u> GC28-6550 will help you make the modifications.
- Punching an object deck of emulator module IIQMW, modifying the part that contains routing code 12 to contain the routing code or codes you want, and replacing the module.
- Writing a routine to change the routing code (see Appendix F).

<u>Note</u>: You are modifying Type I coding when you replace module IIQMW with a modified version, and you may be billed for a program maintenance call if an error is caused by the modified code.

This section describes how data is processed on unit record equipment, tape units, disk drives, and operator consoles.

CHARACTER CODES

The emulator uses EBCDIC as the basic character code for all I/O operations. However, it modifies EBCDIC when the 1400 program writes on or reads from nine-track tape in odd parity or disk in load mode.

Data in odd parity is represented on nine-track tape by setting bit 1 to 0 (n0nnnnnn, where n can be 0 or 1). Data in load mode on disk is represented by setting bit 1 to 0 for characters with a wordmark and setting it to 1 for characters without a wordmark.

Note: Data is sent to seven-track tapes in EBCDIC. If the translator is on, the data is translated to BCD. If the data converter is on, the data is segmented into six-bit segments so that it fits on the tape. When the converted data is read, it is reassembled.

DATA FORMATS

The emulator accepts the following data formats:

- Spanned format [1] [3]
- Even-parity BCD[2]
- Odd-parity BCD[2]
- Mixed-parity BCD[2]
- Normal mode, even parity[3]
- Normal mode, odd parity[3]
- Alternate mode[3]
- Normal mode, mixed parity [3]

The parity of the input data must be that indicated by the 1400 instruction; the parity of the output data is determined by the 1400 instruction. When the data is in alternate mode, a parity error cannot occur on input, and parity of the output data is not determined by the 1400 instruction.

All data formats listed above are considered to be 1400 formats, except for spanned format; that is, only spanned format has a block descriptor word (BDW) and a segment descriptor word (SDW). The BDW and SDW are described below.

PROCESSING DATA ON TAPE

The emulator uses the Basic Sequential Access Method (BSAM) to emulate all 1400 tape input, output, and control operations. The System/370 device to be used by BSAM is specified in the JCL for the emulator job. Data files in spanned format can be read from or sent to either a tape unit or a direct access device. Data files in 1400 format should be used only on tape units.

TAPE FILES

In most cases, the emulator accepts tapes in 1400 format. The main difference between tape files on 1400 systems and on System/370 is the restriction on the size of System/370 physical data records. There is no restriction on the size of physical records for 1400 systems (other than the size of core storage), but physical records on System/370 cannot be smaller than 18 bytes[4] or greater than 32,755 bytes. These restrictions are discussed under "Converting Tape Files" in this section.

Also, System/370 programs (including the emulator) do not handle tapes written in more than one density. Mixed-density tapes must be changed to single-density tapes by the tape preprocessor before being used by the emulator.

----

[1] All 1400 tapemarks and header and trailer labels must be written as data records, as is done by the tape preprocessor program.

[2] Data must be on a seven-track tape (Data Translator on).

[3] Data must be on a nine-track tape, a seven-track tape (Data Converter on), or in a sequential data set on disk.

----

[4] The minimum record size to ensure that a record is not discarded as noise is 12 bytes for a read operation and 18 bytes for a write operation (includes the block descriptor word and the segment descriptor word).

1400 Format

When the data is to be used by a program on the 1400 system, it must be on a seven-track tape in BCD. When the data is to be used by a program executed using a stand-alone emulator, CS/30, or CS/40, it can be on seven-track tape in BCD, or on nine-track tape in even-parity normal mode (equivalent to EBCDIC) or in alternate mode.

Note: If a scratch tape is used for a 1400-format output file, you should write a tapemark on the tape before using it for an emulator job.

Normal and Alternate Modes

Nine-track tape output of 1400 stand-alone emulators is similar to the nine-track tape format used with System/370 operations, except that stand-alone emulators use bit 1 of the EBCDIC character as the parity bit. Using bit 1 for parity, stand-alone emulators, CS/30, CS/40, and this emulator can process mixed-parity data on nine-track tapes, a procedure that is not permitted by OS. Each even parity six-bit BCD character is represented by its corresponding EBCDIC character. Bit 1 of the EBCDIC character (which is not used to represent the BCD character) is always one. Each odd parity six-bit BCD character is represented by its corresponding EBCDIC character, but bit 1 is zero.

Example:
Even parity: n1nnnnnn
Odd parity: n0nnnnnn
where "n" may be either 1 or 0.

A tape error is recognized during even-parity operations when bit 1 is 0 and during odd-parity operations when bit 1 is 1. Both even-parity and odd-parity operations are normal mode operations.

The nine-track odd-parity operations of emulators are not compatible with conventional EBCDIC on System/370 tapes. Odd-parity operations are used to make all emulators compatible; for example, when the output from a card-to-tape operation on a 1401 emulator is to be used as input to a 1410 emulator.

So that System/370 programs can handle all tapes produced by the emulator, it converts normal-mode tapes to alternate mode. If the tape is written in alternate mode, bit 1 contains a 1 regardless of parity. No distinction is made between odd-parity and even-parity, and data always appears as if it were written in even-parity normal mode, except that substitute blanks are preserved on odd-parity tape read operations. Alternate mode tapes can be used for both input and output.

Spanned Format

The standard data format for OS is spanned format. The records may be either blocked or unblocked; data blocks must include specific descriptive information. This descriptive information is in two control words, the block descriptor word and the segment descriptor word. (See Data Management Services, GC26-3746 for detailed information on spanned format.)

The block descriptor word (BDW) is the first word of each block, and it defines the length of the block. The format of the BDW is shown in Figure 9.

The segment descriptor word (SDW) is the second word of an unsegmented block or the first word of each segment of a multisegmented block. It defines the length and relative position of the segment. The format of the SDW is shown in Figure 9.

The emulator uses two record formats within the spanned format: VBS (variable blocked spanned) and VS (variable spanned) formats.

VBS format should be used in most cases, because it is more efficient. (Throughput is reduced noticeably when the 1400 program issues frequent backspace instructions for files in VS format.) VBS format cannot be used with channel buffering, but VS format (or 1400 format) can. Both record formats are specified in the RECFM parameter of the DD statement.

An end-of-volume condition is not returned to the 1400 program when the data is in spanned format. OS detects the end-of-tape reflective strip and automatically rewinds and unloads the first volume and asks the operator to mount the next volume.

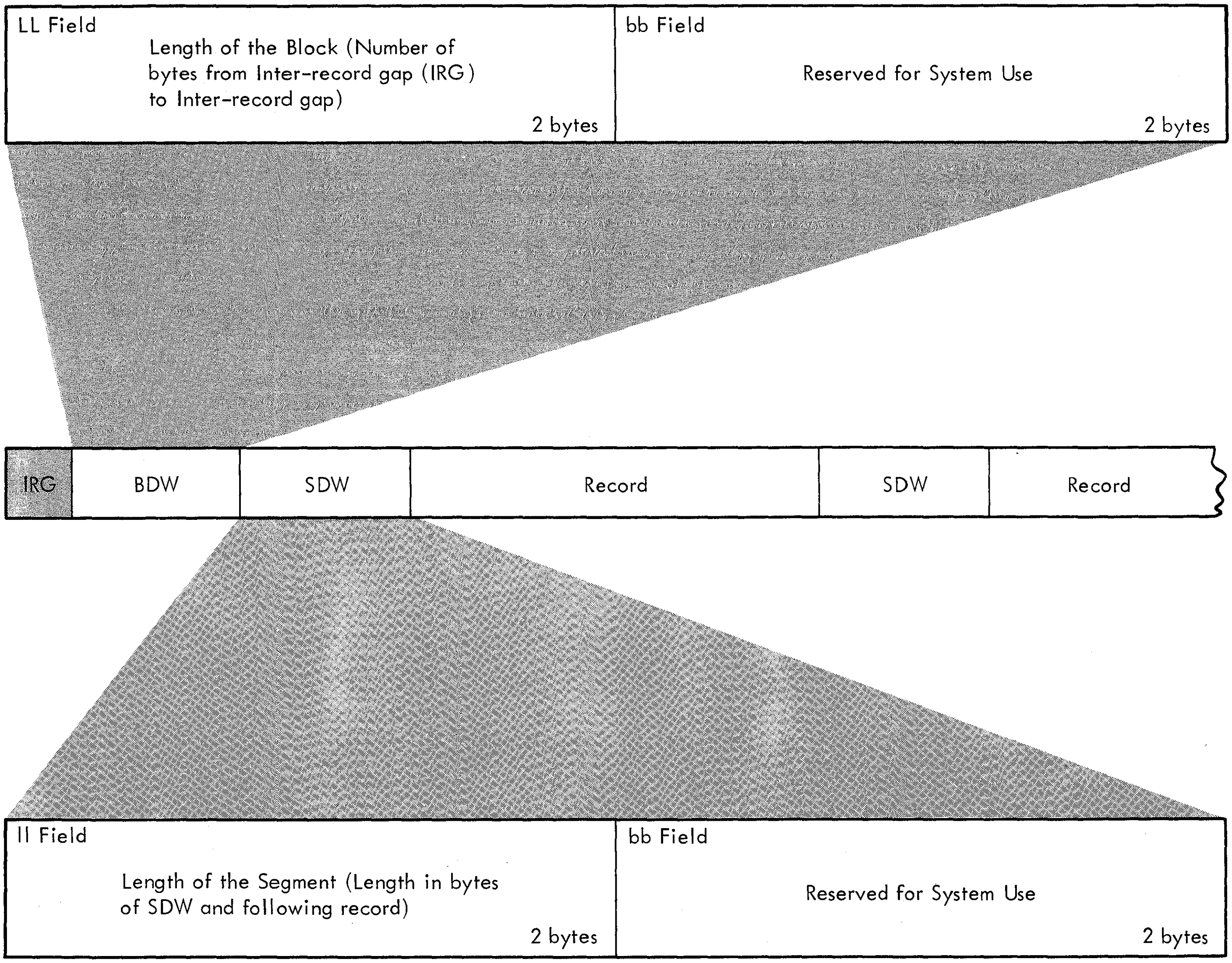

LL Field
Length of the Block (Number of bytes from Inter-record gap (IRG) to Inter-record gap)
2 bytes
bb Field
Reserved for System Use
2 bytes
IRG
BDW
SDW
Record
SDW
Record
ll Field
Length of the Segment (Length in bytes of SDW and following record)
2 bytes
bb Field
Reserved for System Use
2 bytes

Figure 9. Format of the Block Descriptor Word (BDW) and Segment Descriptor Word (SDW)

If there are not enough volumes identified in the DD statement that describes the device, OS abnormally ends the emulator task. Not returning an end-of-volume condition to the 1400 program causes it to be executed improperly when:

- A specific function is started by an end-of-volume condition.
- Write instructions are issued (to fill the tape) until the end-of-tape reflective strip is sensed.

CONVERTING TAPE FILES

Seven-track or nine-track tapes in 1400 format are more easily read by the emulator if they have been converted to spanned format by the tape preprocessor program. Preprocessing tapes saves I/O emulation time by blocking small records and standardizes emulator input and output.

There are three approaches to file conversion:

- One-time conversion. If 1400-format tapes are not needed for a program on a 1400 system, the tapes should be preprocessed so that the more efficient spanned format can be used.
- Staying in 1400 format. If the data format is accepted by the emulator, you may not want to convert. When the data must be converted back to 1400 format after emulation, the efficiency gained by the emulator in having the records in spanned format may be lost by the time spent running the tape preprocessor and tape postprocessor

programs. To determine this, average run times must be computed using both methods.

- Preprocess data - emulate - postprocess data. If the data is not accepted by the emulator, or if the efficiency of the emulator is greatly increased by preprocessing the data, the tape preprocessor program should be used to convert the data. The emulator then executes the 1400 program and the postprocessor program returns the data to its original format so that it may be used on a 1400 system.

Because of the OS limit on physical record size, the emulator does not read or write a physical record larger than 32,755 bytes. Also, OS may not read physical records smaller than 12 bytes or write physical records smaller than 18 bytes. To ensure that all physical records can be read, you should preprocess your data so that there are no physical records smaller than 18 bytes.

To read 1400 physical records from 1 to 17 bytes and from 32,756 to 200,000 bytes, use the preprocessor program to block the short records and segment the long ones. After emulation, use the postprocessor program to unblock the short records and reconstruct the long ones.

Since the emulator does not accept mixed-density tapes, use the preprocessor to create single-density tapes. The preprocessor changes 1400 tapemarks, header labels, and trailer labels to data records. For a System/370 program other than the emulator to use preprocessor output, either:

- Write a program to read and rewrite a preprocessed tape, converting tapemark and label records to standard System/370 tapemarks and labels.

- Incorporate coding in the System/370 program that will recognize and handle tapemark and label records.

Note: BSAM pads any output record that has fewer than 18 bytes with binary zeros. The emulator is not aware that the record has been padded because the BDW is not modified.

EMULATING TAPE OPERATIONS

The emulator uses the BSAM READ, WRITE, and CHECK macro instructions to move data. Records in spanned format are blocked and deblocked in buffers by the emulator.

If double buffers are specified, the emulator issues a READ operation for one buffer while processing records in the other buffer. Double buffers let the emulator read the next record while the 1400 program is using the current one, and to write the old record while the 1400 program is creating the next one. Frequent backspacing of input files by the 1400 program causes the emulator to do more backspacing and reading, which slows down performance when using double buffers. Using single buffers is generally more efficient when backspacing. (The number of buffers is specified in the BUFNO parameter of the DCB operand of the DD statement.)

Moving data during tape emulation and the macro instructions that are used by the emulator are shown in Figure 10.

PROCESSING DATA ON DISK

The emulator uses the Basic Direct Access Method (BDAM) to emulate all 1400 disk operations. The System/370 device to be used by BDAM is specified in the JCL for the emulator job. Data files are written in EBCDIC (move mode only) or a modified form of EBCDIC where bit 1 of the EBCDIC character represents the presence or absence of a wordmark (load mode operations only). The disk formatting program described in Appendix E must be used to create the System/370 data set that holds the data.

DISK FILES

The track organization of 1400 disk files is maintained by transferring all the data on a 1400 track to a record in a System/370 data set. The data set is a sequential data set on any direct access device accepted by OS. The record is a fixed-length record whose size depends on that of the 1400 track that is transferred. There is a one-to-one correspondence between the number of records in the data set and the number of tracks on the 1400 device.

More than one System/370 direct access device can be used to emulate a 1400 disk device. However, the entire data set must be on units of the same type. Instructions on how to create a data set on more than one device are given in Appendix E.

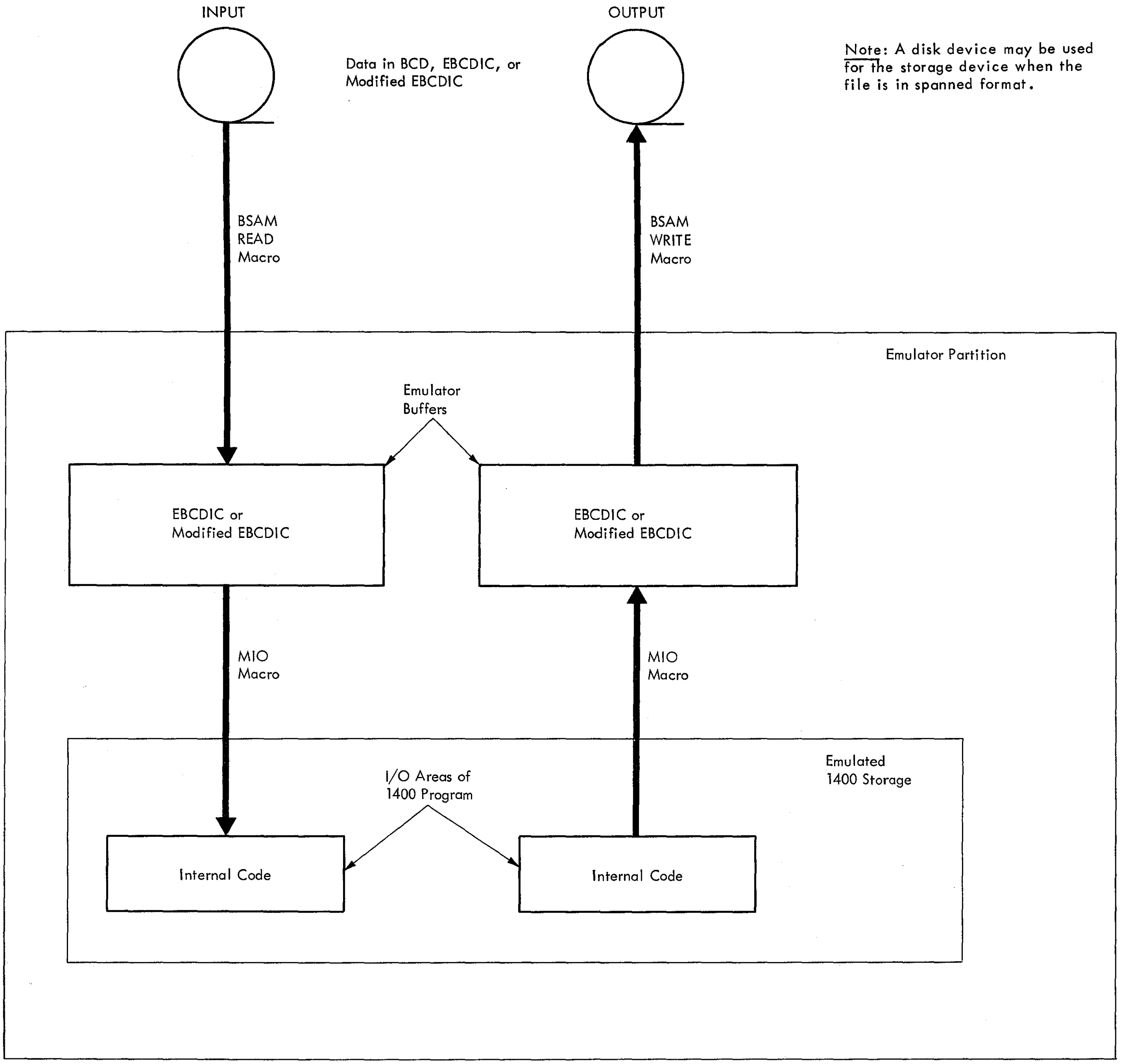

INPUT
OUTPUT
Data in BCD, EBCDIC, or Modified EBCDIC
Note: A disk device may be used for the storage device when the file is in spanned format.
BSAM READ Macro
BSAM WRITE Macro
Emulator Partition
Emulator Buffers
EBCDIC or Modified EBCDIC
EBCDIC or Modified EBCDIC
MIO Macro
MIO Macro
Emulated 1400 Storage
I/O Areas of 1400 Program
Internal Code
Internal Code

Figure 10. Data Movement During Tape Emulation

CONVERTING DISK FILES

To convert 1400 disk files to a format acceptable to the emulator:

1. Dump the contents of the disk onto tape using a 1400 disk-to-tape utility program on the 1400 system or a stand-alone emulator on a System/360. For example, the tape-to-disk utility program 1401-UT-053 could be used for 1401 systems and the tape-to-disk utility program 1440-UT-041 for 1440 systems.

2. Using the disk formatting program described in Appendix E, prepare a previously initialized System/370 direct access storage device to accept the data.

3. Restore the data to direct access storage on the System/370 device using

a 1400 tape-to-disk utility program run under control of the emulator.

EMULATING DISK OPERATIONS

The emulator uses the BDAM READ, WRITE, and CHECK macro instructions to move data. The movement of data during disk emulation and the macro instructions that are used are illustrated in Figure 11. A buffer can be assigned to one device or it can be shared by two or more devices. Figure 11 shows both single and shared buffers.

When 1400 systems write load-mode data on direct access devices, an extra bit is needed in each character to represent wordmarks. The extra bit reduces the number of characters that can be stored on a 1400 track. When emulating load-mode operations, bit 1 of each System/370 byte is used to represent the wordmark.

The emulator maintains fixed sector and track sizes, whether the data is in move mode or load mode. The extra load-mode positions are padded with blanks. Figure 12 shows how the System/370 record is constructed for each 1400 disk device. An emulator control word (ECW) is used to indicate the mode the data is in and the sequence number of the record:

Byte 1, Bit 1	Reserved
Byte 1, Bit 0=0	Move mode
Byte 1, Bit 0=1	Load mode
Bytes 1 & 2, Bits 2-15	Sequence number
Bytes 3 & 4	Unused (Zeros)

The sequence number of the first record in the file is zero.

PROCESSING DATA ON UNIT RECORD EQUIPMENT

The emulator uses QSAM to emulate all 1400 unit record operations except select stacker; to emulate select stacker operations, it uses an EXCP macro instruction. Operations for the 1400 card reader, printer, card punch, and the console are emulated using a comparable System/370 unit record device (if the emulator includes the Selective Stacker feature, select stacker operations can be emulated only by the 2540 Card Read Punch), a magnetic tape unit, a disk unit, and one or more consoles.

Cards may be read from a card reader or from a tape or disk device; cards may be punched on a card punch or put on a tape or disk device; printed output may be sent to a printer or to a tape or disk device. When tape and disk files are used to emulate unit record data, the records in these files must conform to the physical limitations of the unit record data.

EMULATING READER, PUNCH, AND PRINTER OPERATIONS

The emulator uses QSAM GET and PUT macro instructions to move data (except when it is emulating select stacker operations, in which case it uses an EXCP macro instruction). Records are blocked and deblocked in OS buffers by QSAM routines. The emulator issues GET and PUT macro instructions to move records between OS and emulator buffers, and the MIO macro instruction to move data from emulator buffers to emulated 1400 storage. Figure 13 shows this movement of data.

The emulator maintains fixed I/O areas within emulated 1400 storage when executing programs written for 1401 and 1460 systems. These areas are:

- Locations 1 to 80 for the card reader
- Locations 101 to 180 for the card punch
- Locations 201 to 332 (maximum) for the printer

Programs written for 1440 systems do not use fixed I/O areas, so these areas may be anywhere in emulated 1400 storage.

Anticipatory (look-ahead) buffering is standard with QSAM. For this reason, blocking unit record files on tape or disk and specifying more than one buffer per device speeds up unit record emulation. (Blocking and the number of buffers are specified in the BLKSIZE and BUFNO parameters of the DCB operand of the DD statement.)

EMULATING CONSOLE OPERATIONS

The emulator uses the WTO (write-to-operator) and WTOR (write-to-operator-with-reply) macro instructions to communicate with the operator.

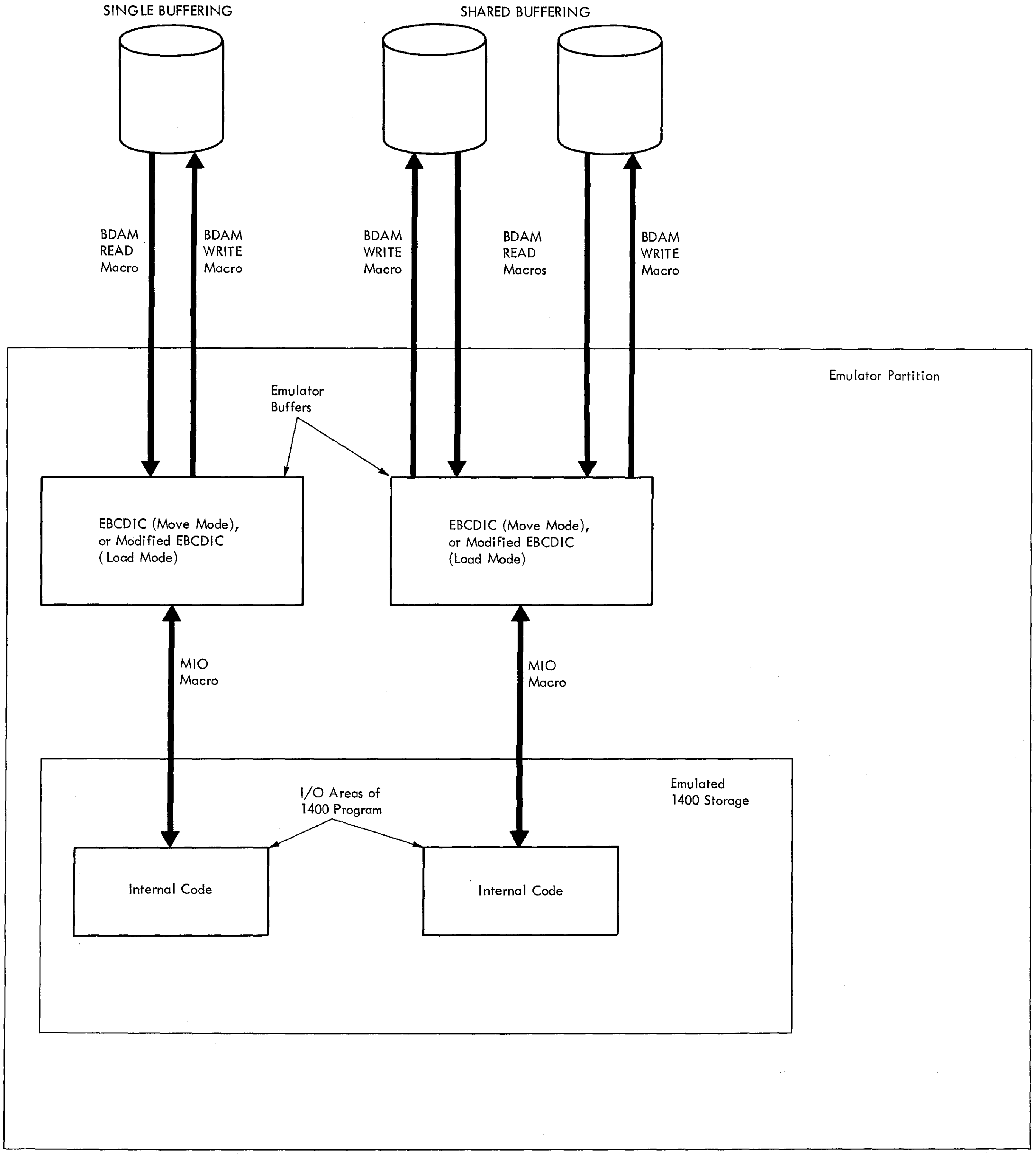
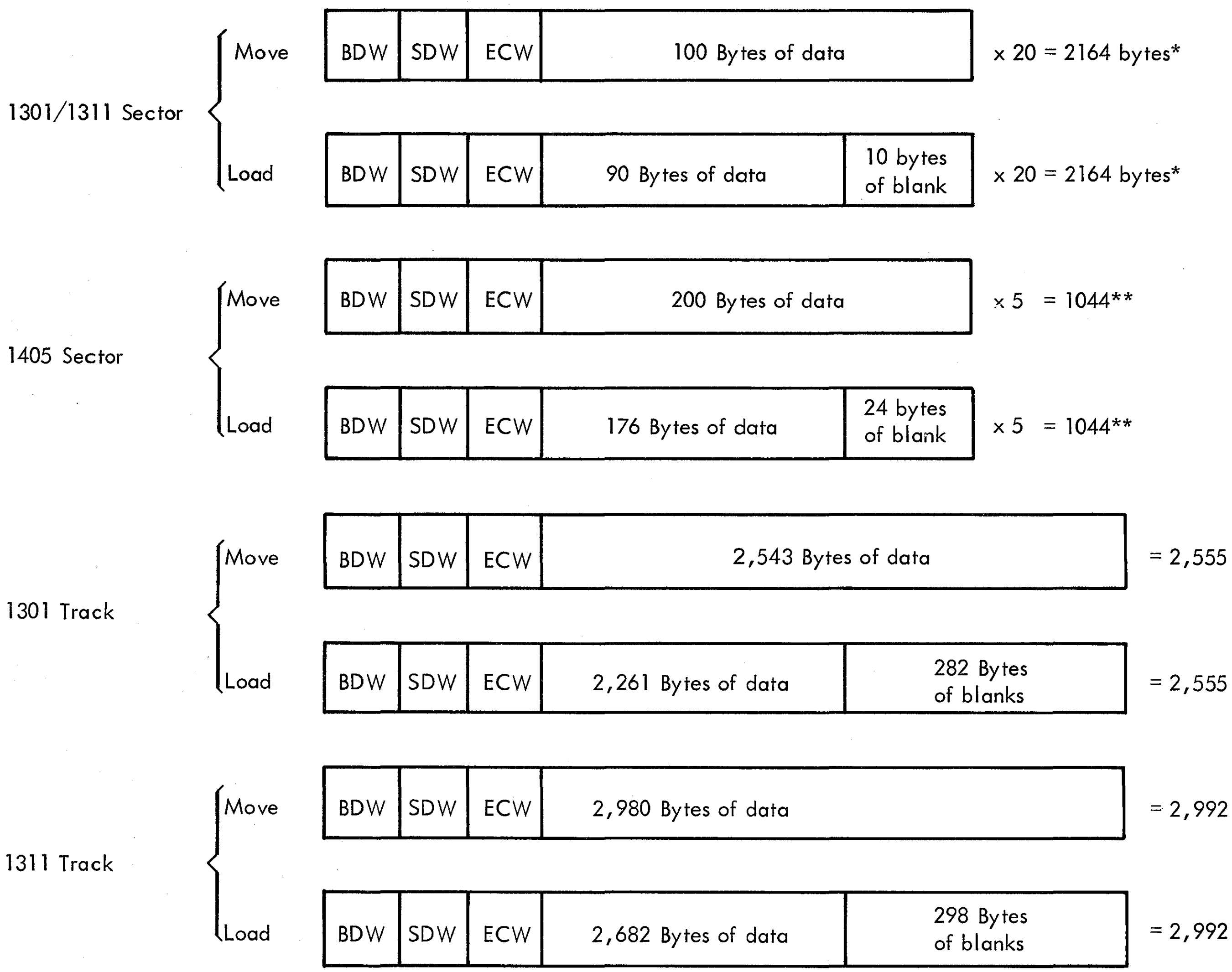


Figure 11. Data Movement during Disk Emulation



* Only the SDW, ECW, and data field are repeated 20 times (100 bytes = 1 1301/1311 Sector)
** Only the SDW, ECW, and data field are repeated 5 times (200 bytes = 1 1405 Sector)

BDW: Bytes one and two = length of physical block; bytes three and four = X'00'.

SDW: Bytes one and two = logical record length; bytes three and four = X'00'.

ECW: Bytes one and two: bit 0 = 1 (load mode), bit 0 = 0 (move mode); bit 1 reserved; bits 2-15 = record count (relative record number of this record in the file, first r ecord in this file has record count = zero).
Bytes three and four = X'00'.

Figure 12. Record Formats for Disk Emulation

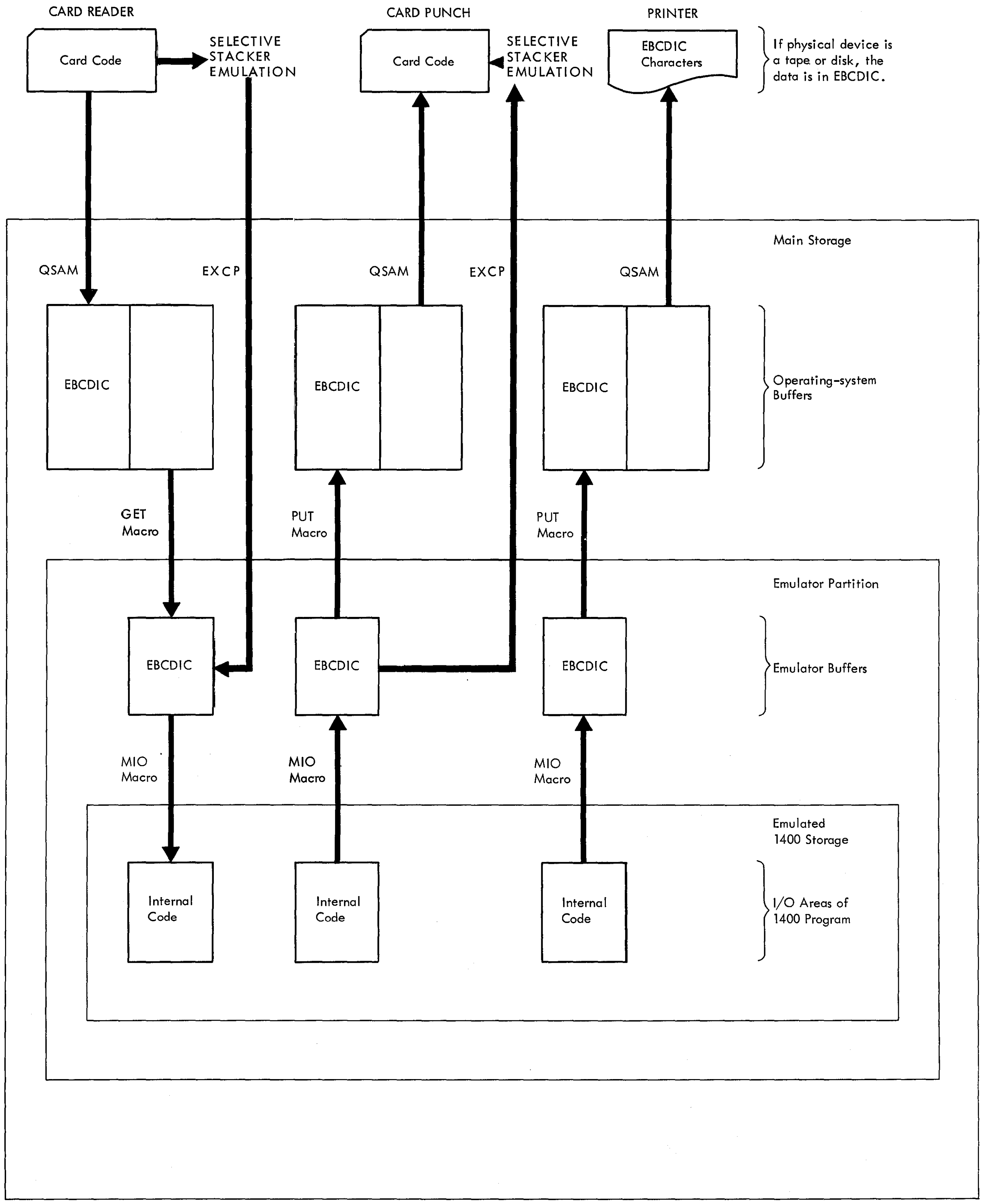


Figure 13. Data Movement during Unit-Record Emulation

Generating an emulator as part of the operating system is done in two steps:

1. Generation of a complete operating system, where the EMULATOR macro instruction includes the operating system routines needed by the emulator.

2. Generation of the emulator, an independent generation that assembles the emulator and the tape and disk formatting programs and link edits them into the operating system.

PREPARING FOR EMULATOR GENERATION

The system used for emulator generation must meet the requirements given below.

MINIMUM SYSTEM/370 REQUIRED

The System/370 used for emulator generation must have:

- Enough main storage to execute the Linkage Editor F program and the Assembler F or Assembler H program

- One operator console

- One direct access storage device in addition to those needed for SYSRES and the SYS1.LINKLIB, SYS1.SYSJOBQE, and SYS1.MACLIB data sets

- One card reader or one magnetic tape drive for input

- One card punch or one magnetic tape drive for intermediate output

- One printer or one magnetic tape drive for diagnostic messages

OPERATING SYSTEM REQUIRED

The emulator is generated under the control of an MFT or MVT control program and is executed as any other job. The operating system must include the following system data sets:

- SYSCTLG

- SYS1.LINKLIB

- SYS1.MACLIB

- SYS1.NUCLEUS

- SYS1.SVCLIB

- SYS1.SYSJOBQE

The operating system can be on any System/360 or System/370 processor unit that can execute the MFT or MVT control program. After emulator generation, the SYS1.LINKLIB data set contains the emulator program.

The SYS1.LINKLIB data set must include an Assembler F or Assembler H program with the alias ASMBLR, a Linkage Editor F program with the alias IEWL, and the IEHDASDR utility. The SYS1.MACLIB data set must be catalogued.

SYSTEM GENERATION PROCEDURES

To include the emulator in the operating system, you must:

- Specify the EMULATOR system generation macro instruction.

- Specify the TYPE parameter of the CTRLPROG macro instruction as either MFT or MVT, and the TIMER parameter of the SUPRVSOR macro instruction as JOBSTEP or INTERVAL. (MVT is available on the Models 145 and 155 only.)

- Allocate sufficient space on SYS1.LINKLIB for each emulator to be generated and for the tape and disk formatting programs. Only one set of tape and disk formatting programs is required.

- Include BDAM, using the ACSMETH operand of the DATAMGT macro instruction, when 1400 disk units are to be emulated.

THE EMULATOR DISTRIBUTION LIBRARY

Emulator routines and the tape and disk formatting programs are distributed on a dump and restore tape for a direct access storage device. Before an emulator can be generated, a direct access volume must be initialized and the distributed tape must be restored to the direct access volume. The tape, and ultimately the direct access volume, is called the Emulator Distribution Library. If you have a system that does not include a tape drive, you must make arrangements for a system with both tape and disk devices to perform the tape-to-disk restore operation.

The IEHDASDR system utility can be used to initialize a direct access volume and to restore the dump and restore tape to that volume. See Utilities, GC28-6586 for a description of this utility and the control statements it requires.

The data sets on the Emulator Distribution Library (EMDLIB) are:

- EMUL.EMMAC - Contains the macro definitions of the emulator generation macro instructions used during Stage I and Stage II.
- EMUL.EMMOD - Contains the load modules that are link edited tc form the generated emulator program, and the load modules of the tape and disk formatting programs that are link edited to SYS1.LINKLIB.
- EMUL.EMSAMP - Contains the sample program used to test the emulator.

EMULATOR GENERATION PROCEDURES

The emulator is made up of modules that can be put together in a number of combinations to meet the needs of your installation. You select the programming options that meet your needs using the parameters of the EM1401 emulator generation macro instruction. Emulator generation is the process of interpreting your selection and building system data sets.

Job control language (JCL) statements and the EM1401 macro instruction are placed in the job stream of the operating system.

Figure 14 illustrates the two stages of emulator generation. During Stage I, the assembler checks the EM1401 macro instruction for errors and uses the parameters to produce a job stream. If the assembler finds errors, it writes error messages and does not produce the job stream.

During Stage II, the assembler and linkage editor process the Stage I job stream to combine the emulator modules and to make the emulator a member of the SYS1.LINKLIB data set. The generated emulator is then ready to be used.

JCL Statements

JCL statements instruct the operating to assemble the EM1401 macro instruction and to produce the Stage I job stream. Figure 15 shows two sample input decks to Stage I of emulator generation. Use the first deck with catalogued procedures and the second deck with standard assemblies.

If Stage I is completed successfully, the job stream produced becomes the input to Stage II of emulator generation.

EM1401 Emulator Macro Instruction

Figure 16 shows the format of the EM1401 emulator generation macro instruction. The macro instruction must be coded according to assembler language rules. The parameters of the macro instruction are:

CPU
specifies the system to be emulated. If omitted, 1401 is assumed.

MODEL
specifies the System/370 CPU. If omitted, the Model 155 is assumed.

EMNAME
specifies the one- to six-character name to be assigned to the emulator load module. Each emulator load module must have a unique name so that you can have different emulator programs with different options. If this parameter is omitted, IIQE14 is assumed.

CORE
specifies the core storage size of the system being emulated (in thousands of 1400 storage positions). CORE=2 must be specified to emulate a system with 1,400 positions of storage.

CPUOPTN
specifies the 1400 features that are to be part of the emulator. More than one feature may be specified. The features are:

MD	Multiply-Divide
EPE	Expanded Print Edit
AP	Advanced Programming
IPE	Inverted Print Edit
PROVP	Processing Overlap (1401 and 1460)

If this operand is omitted, none of the above features are generated.

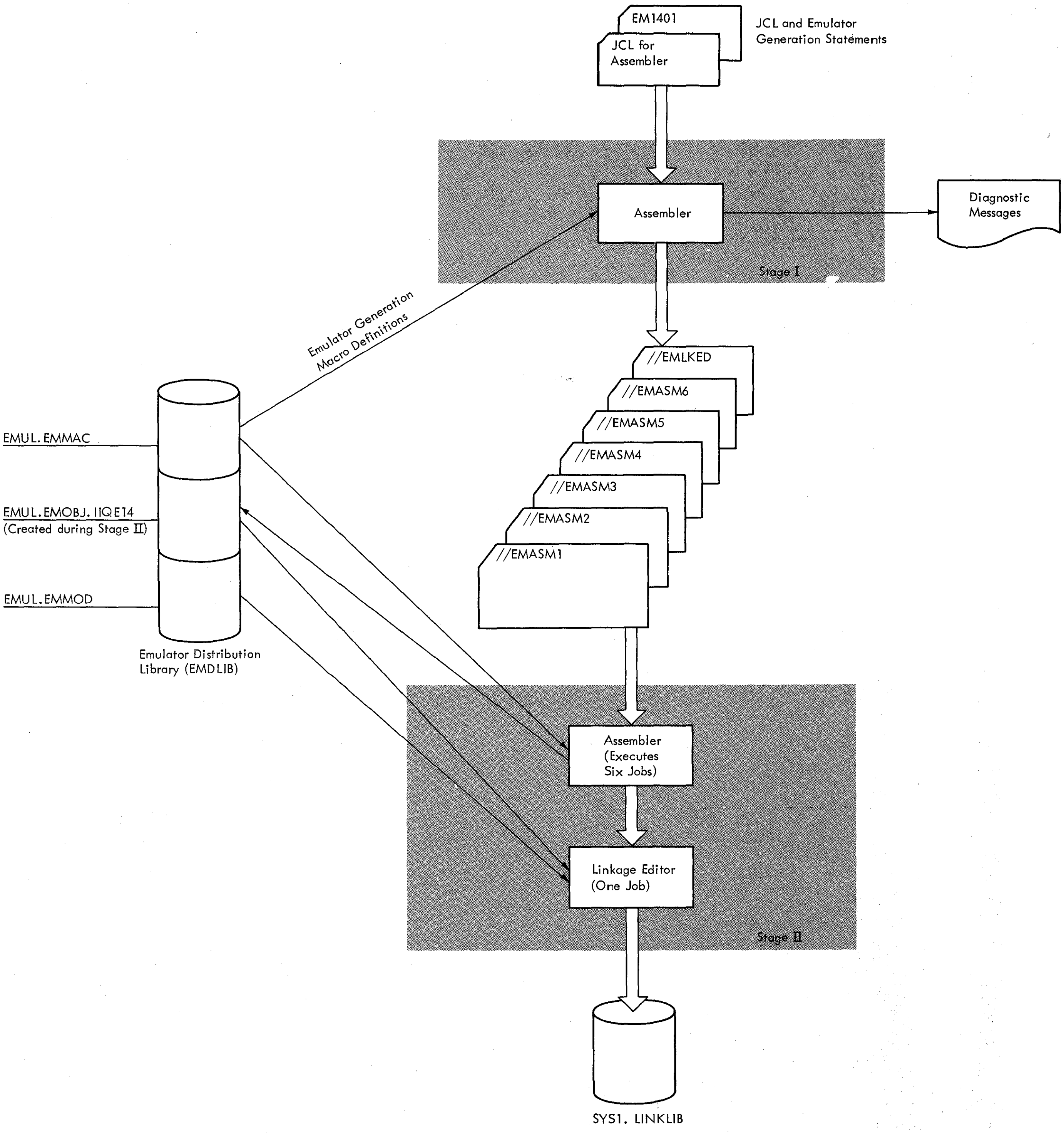

EM1401
JCL for Assembler
JCL and Emulator Generation Statements
Assembler
Diagnostic Messages
Stage I
Emulator Generation Macro Definitions
//EMLKED
//EMASM6
//EMASM5
//EMASM4
//EMASM3
//EMASM2
//EMASM1
EMUL.EMMAC
EMUL.EMOBJ.IIQE14 (Created during Stage II)
EMUL.EMMOD
Emulator Distribution Library (EMDLIB)
Assembler (Executes Six Jobs)
Linkage Editor (One Job)
Stage II
SYS1. LINKLIB

Figure 14. Emulator Generation

EXECUTING A CATALOGUED PROCEDURE

```
//STG1          JOB    ACCT123,PROGRAMMER,MSGLEVEL=(1,1)
//STEP1         EXEC   PROC=ASMFC
//ASM.SYSLIB    DD     DSNAME=SYS1.MACLIB,VOL=(,RETAIN),DISP=SHR
//              DD     DSNAME=EMUL.EMMAC,VOL=SER=EMDLIB,UNIT=2311,       X
//                     DISP=SHR
//ASM.SYSIN     DD     *
                EM1401 CORE=16,CPUOPTN=(MD,EPE,AP) ,                   X
                       IODVTYP=(PPTAPE,NMTAPE,CONSINQ,DISK)
                END
/*
```

EXECUTING A STANDARD ASSEMBLY

```
//STG1          JOB    ACCT123,PROGRAMMER,MSGLEVEL=(1,1)
//STEP1         EXEC   PGM=ASMBLR
//SYSLIB        DD     DSNAME=SYS1.MACLIB,VOL=(,RETAIN),DISP=SHR
//              DD     DSNAME=EMUL.EMMAC,VOL=SER=EMDLIB,UNIT=2311,       X
//                     DISP=SHR
//SYSUT1        DD     DSNAME=&SYSUT1,SPACE=(1700,(400,50)),            X
//                     UNIT=SYSDA
//SYSUT2        DD     DSNAME=&SYSUT2,SPACE=(1700,(400,50)),            X
//                     UNIT=SYSDA
//SYSUT3        DD     DSNAME=&SYSUT3,SPACE=(1700(400,50)),             X
//                     UNIT=SYSDA
//SYSPUNCH      DD     SYSOUT=B
//SYSPRINT      DD     SYSOUT=A
//SYSIN         DD     *
                EM1401 CORE=16,CPUOPTN=(MD,EPE,AP),                    X
                       IODVTYP=(PPTAPE,NMTAPE,CONSINQ,DISK)
                END
/*
```

Note: Underlined values represent variables. All other values must be coded as shown. The continuation characters are in column 72.

Figure 15. Stage I Input Deck

Several features of 1400 systems have been combined under Advanced Programming. The features are slightly different for each system. When you specify AP, you get:

- 1401 - Advanced Programming (feature #1060)
- 1440 - Indexing and Store Address Register (feature #4631)
- 1460 - Indexing and Store Address Register (feature #4631)

Processing Overlap is not a 1440 feature and should not be specified when generating the emulator for that system.

IODVTYP
specifies the type of I/O devices or features that are to be emulated. More than one device can be selected and more than one tape format can be specified. If this parameter is omitted, only unit record equipment is emulated. The devices and features that are emulated are:

PPTAPE tape units reading or writing records in spanned format that have been formatted by the tape preprocessor program or created by an integrated emulator

NMTAPE tape units reading or writing 1400-format records

DISK 1301 and 1311 disk storage drives

1405DISK 1405 disk storage drive (on 1401 processor units only)

STACKER 1402 or 1442 Selective Stacker feature

CONSINQ 1407 or 1447 console; must be specified to execute 1400 console read and write instructions properly

Operation	Operand

EM1401	CPU= $\left\{\begin{array}{c}\underline{1401}\\1440\\1460\end{array}\right\}$
	MODEL= $\left\{\begin{array}{c}135\\145\\\underline{155}\end{array}\right\}$
	EMNAME= $\left\{\begin{array}{c}\text{cccccc}\\\underline{\text{IIOE14}}\end{array}\right\}$
	CORE= $\left(\begin{array}{c}2\\4\\8\\12\\16\end{array}\right)$
	CPUOPTN= ([MD][,EPE][,AP]
	[,IPE][,PROVP])
	IODVTYP= ([PPTAPE][,NMTAPE]
	[$\left\{\begin{array}{l}\text{,DISK}\\\text{,1405DISK}\end{array}\right\}$]
	[,CONSINQ]
	[,STACKER])
	EMVOL= $\left\{\begin{array}{c}\underline{2311}\\2314\\3330\end{array}\right\}$
	FSTEMUL= $\left\{\begin{array}{c}\underline{\text{YES}}\\\text{NO}\end{array}\right\}$

Figure 16. EM1401 Emulator Macro Instruction

This operand is used to include those functions of the emulator that process data from tape and disk. For example, if the PPTAPE and DISK parameters are omitted, the emulator routines that handle preprocessed data and data from disk are omitted, but the tape and disk formatting programs are not. These programs are always in the SYS1.LINKLIB data set for the first emulator generated and are not included for each additional emulator.

EMVOL

specifies the unit for the Emulator Distribution Library (EMDLIB). If omitted, 2311 is assumed.

FSTEMUL

specifies whether the emulator being generated is the first 1400 emulator in the system. Since only one copy of certain routines is needed, this parameter can be used to save auxiliary storage in systems with several emulators. Auxiliary storage cannot be saved when using both the 1401/1440/1460 emulator and the 1410/7010 emulator, since they use different load module names. If FSTEMUL=NO, the tape and disk formatting programs and other common emulator components are omitted. If this parameter is omitted, YES is assumed.

THE JOB STREAM

Producing the Job Stream (Stage I)

Stage I is an assembly of the EM1401 macro instruction. The assembly produces a Stage II job stream that contains a maximum of seven jobs. The first six jobs are assembler jobs, the last one is a linkage editor job. The job stream is punched on cards. The sixth assembler job assembles the routines that emulate 1400 disk devices. It is omitted from the job stream when the IODVTYP operand of the EM1401 macro instruction does not request disk routines.

Restarting Stage I

Stage I is one job, and it can be restarted only at the beginning of the job. Common causes of error during Stage I are:

- Improper allocation of utility data sets caused by incorrect parameters
- Keypunching errors in the input deck
- JCL errors
- Contradictory or invalid parameters in the EM1401 macro instruction

To restart Stage I, correct any errors in the input deck and rerun the job.

Executing the Job Stream (Stage II)

During Stage II, emulator modules are assembled and stored in the Emulator Distribution Library in the data set EMUL.EMOBJ.emname, where emname is the name specified in the EMNAME operand of the EM1401 macro instruction. After all modules have been assembled, they and other modules from the EMUL.EMMOD library are processed by the linkage editor to form the emulator load modules.

Note: If you need accounting information on the JOB card, you must replace the existing cards with your JOB cards. Do not delete any information on the JOB cards when you add your accounting information.

The suggested procedure for executing the job stream is:

1. Execute the assembler jobs. The linkage editor job will not run when

the TYPRUN=HOLD parameter in the JOB statement is coded.

2. If the first assembler job does not run successfully, find the error, change the DISP parameter on the SYSPUNCH DD statement to DISP=(OLD,KEEP), and resubmit the job.

3. If any remaining assembler jobs do not run successfully, find the error and resubmit only those jobs that did not run.

4. When the assembler jobs have been completed, start the linkage editor job by typing "A EMLKED".

The linkage editor job leaves the EMUL.EMOBJ.emname data set on the EMDLIB so that the emulator can be maintained and re-created.

By changing the SYSLMOD DD statement of the linkage editor job, the emulator can be link edited to other libraries. These libraries can be for any system that can execute the emulator (a system with the MFT or MVT control program and an appropriate compatibility feature). The six assembly jobs need not be rerun. The Emulator Distribution Library that contains the output from the assembly jobs is used as input.

If any of the emulator load modules are destroyed during system operation, re-create the emulator by rerunning the linkage editor job.

PUNCHING THE SAMPLE PROGRAM

After Stage II has been completed, but before the Emulator Distribution Library has been removed, the sample program should be punched. The sample program, which tests various components of the emulator, is in the EMUL.EMSAMP data set. The JCL statements to punch the sample program, and the results you should obtain from executing the program are shown in Appendix C.

GENERATING SEVERAL EMULATOR PROGRAMS

More than one emulator can be generated and included in the SYS1.LINKLIB data set by following the procedures for generating a single emulator. To ensure that subsequent emulators are generated without replacing the existing ones, each emulator must be given a unique name in the EMNAME operand of the EM1401 macro instruction. In addition, FSTEMUL=NO must be coded to indicate that this is not the first emulator. Sufficient space must be available in the SYS1.LINKLIB data set for the additional emulators (see Figure 6).

The emulator is loaded, started, and stopped by the operating system. The operating system treats it the same as any other problem program, such as a compiler or a payroll program. JCL statements and emulator control statements are used to control execution of the emulator and the 1400 program.

The emulator initializes its tables and control blocks from information in the emulator control statements. It then loads the 1400 program from the unit specified in the emulator control statements.

The compatibility feature fetches and interprets each 1400 instruction. When it can, the feature executes the instruction itself; if not, it returns control to the emulator program, which emulates the instruction.

The compatibility feature also moves data from System/370 buffers to emulated 1400 storage and back again. Both buffers and emulated 1400 storage are in System/370 main storage; data is in EBCDIC or a modified form of EBCDIC in the buffers and in an internal code in 1400 storage. The emulator uses one instruction of the compatibility feature to translate and move data.

The emulator can execute several 1400 programs in a single OS job step. When a second or subsequent 1400 program is to be executed, all data files except SYSEMCTL are closed, all control blocks and buffers acquired by a GETMAIN macro instruction are released by means of a FREEMAIN macro instruction, emulated 1400 storage is cleared, and any routines you have written that have been loaded are deleted. Note that SYSOUT data files, and in particular the log of console communications on SYSEMOUT, cannot be filled indefinitely and you may have to specify an online printer or SYSEMOUT DD DUMMY.

During the execution of the OS job step, the operator can end either the 1400 program being executed or the whole job step; otherwise, the emulator executes 1400 instructions until the last 1400 program in the job step reaches normal end-of-job or until an unrecoverable error occurs. An unrecoverable error in any 1400 program ends the entire OS job.

JCL STATEMENTS

Each 1400 job or group of jobs must be preceded by JOB, EXEC, and DD job control statements. Examples of these statements are shown in Figure 17.

THE JOB STATEMENT: In MFT, the emulator and the 1400 program are in the same partition. The CLASS parameter in the JOB statement directs the operating system to start emulator jobs in the correct partition. You can reserve one of the 15 available job classes (A-O) for 1400 jobs.

More than one job class can be assigned to the emulator partition. Also, the emulator partition can be redefined to modify the size or job class assignments so that other work can be done in the partition. If emulation jobs have widely varying storage requirements, assigning classes by partition size may use main storage more effectively than assigning them by job type. For more information about assigning job classes to jobs and partitions to job classes, see MFT Guide, GC27-6939.

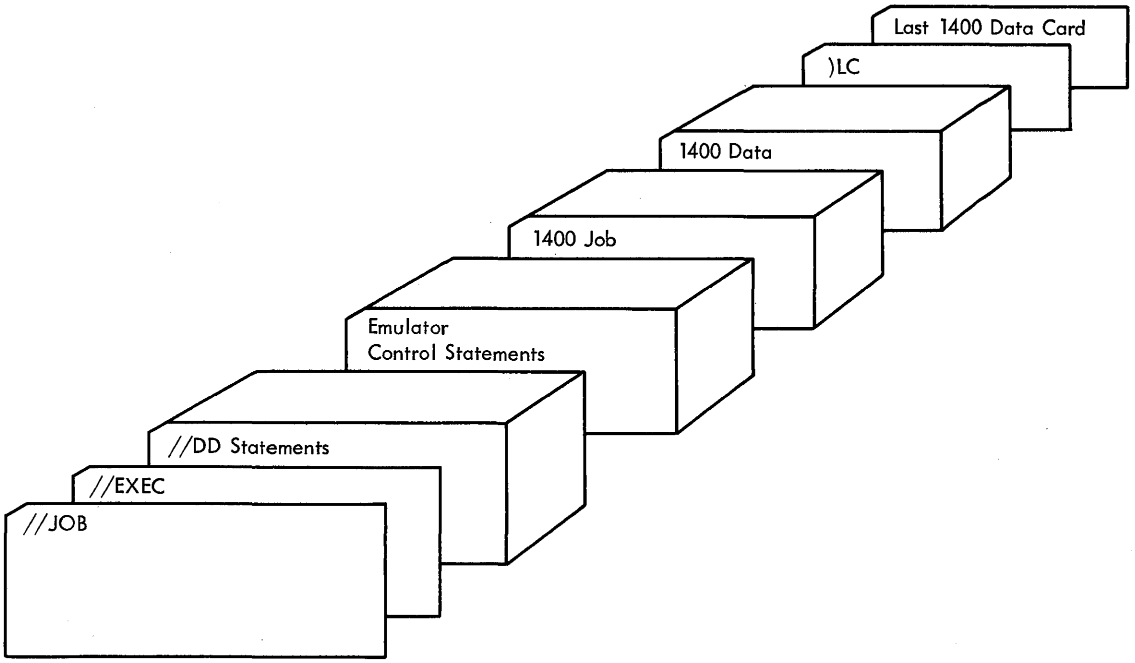
In MVT, the emulator program and the 1400 program are in the same region. The REGION parameter in the JOB statement should indicate the storage needed.

No special JOB statement parameters are needed to execute the emulator, but you should use the MSGLEVEL=(1,1) parameter so that error condition information will be printed. For a description of the parameters of the JOB statement, see Job Control Language Reference, GC28-6704.

THE EXEC STATEMENT: This statement must identify the program name of the emulator. You must assign a program name to each emulator catalogued by the operating system; you assign the name at emulator generation. PGM=IIQE14 should be coded if no program name was assigned. Other operands may be coded in the EXEC statement when required.

DD STATEMENTS: Data definition (DD) statements specify the data sets and devices used by the emulator and the 1400 program. DD statements should be provided as listed below. Factors to be considered are listed under each type of DD statement.

SYSUDUMP DD statement
defines an optional data set used to print the contents of the emulator



```
//EMUL      JOB    MSGLEVEL=(1,1)
//STEP1     EXEC   PGM=IIQE14
//SYSUDUMP  DD     SYSOUT=A
//SYSEMOUT  DD     SYSOUT=A
//TAPE1     DD     DSNAME=ABC,UNIT=2400,DISP=OLD,VOL=SER=140101
//*                BUFNO DEFAULTS TO 1 AND BLKSIZE AND RECFM ARE IN LABEL
//TAPE1A    DD     DSNAME=DEF,UNIT=(2400,,DEFER),DISP=OLD,                      X
//                 VOL=SER=140102,DCB=(BUFNO=2,BLKSIZE=5000,RECFM=VBS)
//TAPE2     DD     UNIT=(2400-2,,DEFER),DCB=(BLKSIZE=1000,                      X
//                 BUFNO=2,DEN=1,TRTCH=ET,RECFM=U),LABEL=(,NL),                 X
//                 DISP=OLD,DSNAME=EFG,VOL=SER=140105
//DISK1     DD     DSNAME=HIJ,UNIT=2314,VOL=SER=140108,DISP=OLD
//PUNCH1    DD     SYSOUT=B
//PRINT1    DD     SYSOUT=A
//SYSEMCTL  DD     *
            EMCTL  SENSE=ABCD,EOJAADDR=10384,OPSERV
            CCTL   CH1=2/68,CH12=62/128
            TAPE   UNIT=12,NAME=TAPE1
            TAPE   UNIT=15,NAME=TAPE2
            DISK   UNIT=16,NAME=DISK1,FROMDEV=1311,BUFID=01,                    X
                   MODE=MIXED
            UR     UNIT=RDR,NAME=SYSEMCTL
            UR     UNIT=PNCH,NAME=PUNCH1
            UR     UNIT=PTR,NAME=PRINT1
            UR     UNIT=CNSL
            LOAD   CARD
Cards containing the 1401 program go here.
The BCD data for the 1401 program goes here.  Start the data in column 1 and go to
column 80, continuing on the next card in column 1.
)LC
The last data card goes here.
```

Figure 17. JCL and Emulator Control Statements - Example 1

partition and related system control areas if the emulator fails to reach normal end-of-job.

SYSEMOUT DD statement
: defines an optional data set used to print messages and operator commands, all emulator dumps, and the control statements in the SYSEMCTL data set. The data set is also used to print the dumps requested by the operator using the DUMP command. The data set can be temporarily stored on tape or disk for offline printing.

SYSEMCTL DD *
: defines a data set that provides all the control information needed by the emulator to execute the 1400 program; this data set is required. Use SYSEMCTL DD DATA if a slash is the first character of any data card in the data set. This DD statement generally includes three items:

- Emulator control statements
- 1400 program
- 1400 data

These items may be catalogued by a previous job step. If they are, the SYSEMCTL DD statement points to the device on which they are catalogued.

Emulator control statements contain information about the 1400 program and identify which System/370 device or unit is to emulate a 1400 device. The 1400 program must be an object deck that can be executed on the 1400 system for which it was originally written. The 1400 data must be in BCD.

Other DD statements are required to define input and output data sets for the emulator. Parameters used in these DD statements are:

DDNAME
: is required; any valid name may be used.

SYSOUT
: is required for offline printing and punching.

DSNAME
: is required for disk units and for tape units if the tape has standard labels.

UNIT
: is required, except for offline punching, printing, and card reading. A specific device can be requested by address, type, or group. Unit affinity, parallel mounting of multiple-volume data sets, and deferred mounting can be specified. Deferred mounting is desirable if specific device addresses are coded and the job can be set up in advance.

For unit record equipment, UNIT must specify an online device if immediate input or output is desired. When the Selective Stacker feature is being emulated, UNIT must specify a 2540 Card Read Punch.

VOLUME
: volume serial numbers must be used for input tapes (even if the tapes are unlabeled), and are generally needed for outout tapes. RETAIN should be coded if output is to be printed by a subsequent job step.

LABEL
: is optional; if not specified, standard operating system labels are assumed. For tapes in 1400 format, LABEL=(,NL) must be specified.

DISP
: the KEEP parameter is required to execute a 1400 program that issues a Rewind and Unload instruction and then uses that file again.

SPACE
: is required for direct access storage devices when creating data sets.

DCB
: is optional, and is used to describe the attributes of the data set. Certain parameters are required when the data set characteristics differ from default values. DCB parameter considerations are:

BLKSIZE
: is required for tape files except when the block size is included in a standard operating system label for input files. It is required for all tapes in 1400 format.

The Record Overflow feature cannot be used when emulating tape files on disk units because the emulator cannot backspace over records split between cylinders. This limits the maximum block size to the data that can be written on one track. We recommend that you specify 3500 bytes for the 2311 and 2314, and either 3100 or 4200 bytes for the 3330.

BUFNO
is optional; it indicates the number of emulator buffers to be used for I/O operations. If not specified, one buffer is reserved for tape and disk units, and two buffers for unit record equipment. Three buffers are reserved for the 2540 Card Read Punch if OS punch error recovery is used.

The maximum number of buffers used by the emulator is one buffer per disk unit and two per tape unit. If BUFNO is greater than the maximum, it is reduced to that number. The number of buffers specified for unit record equipment is not changed.

DEN
is required for tapes when the density is not the default value.

LRECL
is required when blocking records and when a value other than the default value is desired.

OPTCD
is optional; for a validity check for write operations on a direct access storage device, OPTCD=W must be specified.

RECFM
If not specified, input tape files are assumed to be in 1400 format (RECFM=U), and output tape files in VBS record format (RECFM=VBS). RECFM=U is required for output tapes in 1400 format. RECFM=VS (channel buffer) or RECFM=VBS is required for input tapes in spanned format. If a record format other than U, VS, or VBS is specified for a tape, message IIQ094I is issued to indicate that an invalid record format was specified, and the 1400 job is ended. RECFM=FA is required for the printer. The printer buffer contains an ASCII control character (carriage control) in the leftmost byte, but the character is not printed.

Note: The Record Overflow feature may not be used when emulating tape files on disk units because the emulator is unable to backspace across cylinders.

TRTCH
is required for seven-track tapes. TRTCH=ET should be specified if the tape is in 1400 format (the emulator automatically switches parity if necessary); TRTCH=C must be specified if the tape has been preprocessed or created by the emulator in spanned format on a seven-track tape.

Multiple DD statements must be used to define multiple files or data sets on a single device.

Figure 18 shows the DD statement parameters that define unit record equipment. Additional parameters may be coded if values are inadequate.

Note 1: A DD statement is not required for operator consoles.

Note 2: An operating system /* card must not be placed at the end of a job step if the Selective Stacker feature is being emulated.

	Online Processing	Peripheral Processing	
Device	No Blocking	No Blocking	Blocking
Reader	UNIT=00C		
Punch	UNIT=00D	SYSOUT=B[1]	SYSOUT=B,DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)[2]
Printer	UNIT=00E	SYSOUT=A[3]	SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=399)[2]

Notes:
[1] Defaults to RECFM=F and LRECL=80.
[2] The value specified in BLKSIZE must be a multiple of the value specified in LRECL, and cannot exceed 32,760 bytes.
[3] Defaults to RECFM=FA and LRECL=80.

Figure 18. DD Statement Paramaters for Unit Record Equipment

EMULATOR CONTROL STATEMENTS

The emulator needs certain information to execute the 1400 program properly. This information is provided in emulator control statements. Emulator control statements are required to:

- Define each I/O device being emulated
- Indicate 1400 program end-of-job addresses
- Describe how 1400 console sense switches are to be emulated
- Identify the device from which the 1400 program is loaded
- Add your own routines to the emulator
- Send comments to the operator before the 1400 program is executed

Emulator control statements are defined in this section, and their formats are shown in Figure 30. This figure, which is located just before the appendixes, can be folded out and read with the rest of this section. After you become familiar with the definition of each statement and operand, Figure 30 can be used to prepare emulator control statements for your 1400 program.

The input stream may contain emulator commands. The ALTER command can be used to modify an instruction in the 1400 program or to store a value in a data area. The DUMP command can be inserted between any two emulator control statements to record the contents of the emulator partition at the time the command is read. The commands that are not accepted in the input stream are:

- DISPLAY, with the REG, STATUS, TAPE, DISK, UR, or CONFIG operand
- EOJ, ECJ ALL or EOJ A
- SET
- START

Emulator commands are discussed in the section "Communicating with the Emulator." Syntax and coding conventions are given in Appendix B.

When you have data for the 1400 program in the input stream, you must put a)LC card just before the last data card. The)LC card is defined with the emulator control statements in this section.

CAUTION: The)LC and)RC cards differ from the emulator control statements in that they must start in column 1 of the card.

HOW TO PREPARE CONTROL STATEMENTS

CCTL - Carriage Control Information

Use the CCTL statement to emulate the carriage control tape for a 1400 printer. The CCTL statement defines the image and the length of the carriage control tape that was used on the printer when the 1400 program was executed on a 1400 system. Multiple operands may be specified if they are separated by commas. Do not specify the LENGTH operand more than once.

The channel-1 punches and the length of the carriage control tape defined by the CCTL statement must correspond to the channel-1 punches and the length of the carriage control tape on the System/370 printer. The System/370 printer is identified in the UR emulator control statement by the UNIT=PTR operand. Punches in other channels of the carriage control tape do not have to correspond.

CHcc=nnn[/nnn]...
describes how the carriage control tape is to be emulated for the 1400 program. CHcc specifies the channel being emulated. cc is a decimal value from 1 through 12 that identifies a specific channel. nnn is a decimal value from 1 through 132 that identifies the column in which a carriage tape punch is to be emulated. Multiple column numbers can be specified for each channel if they are separated by slashes (for example, CH5=8/96/108).

This operand must be specified and can be specified twelve times in one CCTL statement, but there must be no more than one operand for any one channel. If two or more operands are coded for the same channel, all but the last operand are ignored. Default values for this operand are listed in Figure 30.

LENGTH=nnn
is the length of the carriage control tape, where nnn is a decimal value from 1 through 132. If this operand is omitted, 132 is assumed.

CHBUF - Obtain Channel Buffer

Use the CHBUF statement when you use a channel buffer for tape units. This buffer may be used by one tape unit or by all tape units. The CHBUF statement establishes the size of the channel buffer. The emulator uses the channel buffer if (1) the size of the buffer needed by the tape unit is equal to or less than that of the channel buffer, (2) the data is in 1400 format or VS format, and (3) double buffers are not requested for that tape unit. If the emulator cannot use the channel buffer, a separate buffer is obtained.

The size of the buffer needed by the tape unit is specified in the BLKSIZE parameter of the DCB operand (DD statement). If the value in the BLKSIZE parameter is greater than the size of the channel buffer, a separate buffer is obtained for that tape unit. If the BLKSIZE parameter is omitted, the channel buffer is used, and the block size becomes that of the channel buffer.

Only two record formats can be used with the channel buffer: 1400 format (RECFM=U) and VS format (RECFM=VS). VBS format cannot be used with the channel buffer because the second and subsequent records in the block would be lost.

Note: Using a channel buffer greatly reduces the main storage required for some emulator jobs, but it also increases the time required to run those jobs. Most 1400 programs are I/O-bound, and using a channel buffer makes them even more so. A channel buffer should not be used unless you must reduce the main storage required by the emulator job.

CH1=nnnnn
: specifies the size of the channel buffer. nnnnn is a decimal value from 0 through 32,755. Values above 32,755 cause the emulator to stop.

  Note: When emulating tape files on a disk unit, no records can be written that exceed the track size of the disk unit.

COMMENT - Programmer Comment to Operator

Use the COMMENT statement to give instructions to the console operator. The instructions are displayed on the operator console when the emulator job is initialized. One COMMENT statement must be coded for each line of information to be displayed on the console.

comments
: contains the text that is to be displayed on the operator console. Any EBCDIC character acceptable to the operating system may be entered. The emulator stops when an invalid character is found.

  The emulator types up to 52 characters of comments on the operator console for each COMMENT statement. If the comments field exceeds 52 characters, the field is truncated after the 52nd character, and the remaining characters are lost.

  The comments field becomes the text field of a WTO (write-to-operator) macro instruction. A routing code of 12 and a descriptor code of 7 are assigned to the WTO.

DISK - 1400 Disk Information

Use the DISK statement to identify a 1400 disk device. One DISK statement must be coded for each disk device used by the 1400 program. Multiple operands may be specified if they are separated by commas, but each operand may be specified only once.

If the first volume must be removed for a subsequent volume to be mounted, the operator must identify the second and any subsequent volumes by typing the DISK statement at the console. A DD statement must be included in the JCL to define each volume used.

UNIT=cm
: identifies the channel (c) and module (m) of the 1400 disk being emulated. This operand must be specified. c must be 1; m must be one of the following values:

  0 - for a 1301
  0 - for a 1405
  0, 2, 4, 6, or 8 - for a 1311

NAME=ddname
: is the name of the DD statement that describes the System/370 device that is to emulate the 1400 device. This operand is used to connect emulator control information provided in the DISK statement with system control information provided in JCL statements.

  Although the NAME operand is optional, the ddname associated with the first file on the device should be specified. If the ddname is not specified, the emulator asks the operator to assign one during execution. The operator must then type the DISK statement at the console.

FROMDEV
: is the type of disk device to be emulated. This operand must be specified. There is no default value.

1400 Device and Mode	MODE Operand Needed	Buffer Size
1301 Sector Mode	SECTOR	2,164 Bytes
1301 Track-Record, or Both Track-Record and Sector Mode	MIXED	2,555 Bytes
1311 Sector Mode	SECTOR	2,164 Bytes
1311 Track-Record, or Both Track-Record and Sector Mode	MIXED	2,992 Bytes
1405	SECTOR	1,044 Bytes

Figure 19. Comparison of Buffer Sizes and Mode Operands

BUFID=nn
identifies a buffer when buffers are shared, where nn is a two-digit buffer identification from 00 through 99. If two or more DISK statements have the same buffer identification, the devices described by those statements share that buffer. The first DISK statement read by the emulator determines the size of the buffer (see the MODE operand). Omit this operand if the device is not sharing a buffer.

MODE
is the data format being used on the 1400 device. MIXED indicates that track-record mode or both track-record and sector modes are used. SECTOR indicates that only sector mode is used. If MODE is not specified, sector mode is assumed. Use Figure 19 to determine which MODE operand to use.

CAUTIONS:

- Use the same mode in the DISK statement as was used when the volume was formatted by the disk formatting program.

- The emulator determines the size of the disk input or output buffer from the MODE operand; therefore, the operand must indicate the largest buffer used on the disk volume (or subsequent volume if one volume is removed and another mounted). The operator cannot increase the buffer size when he enters the DISK statement from the console, because the first DISK statement defines the size of the buffer.

- If a buffer is shared, the size of the buffer is determined by the MODE operand of the first DISK statement read; it cannot be changed by another DISK statement.

EMCTL - Emulator Control Information

Use the EMCTL statement to indicate when the 1400 program will reach normal end-of-job, how the sense switches on the 1400 console are to be emulated, and whether communication between the operator and the emulator is desired.

All operands of the EMCTL statement are optional, but at least one operand must be coded if the statement is used. Multiple operands may be coded if they are separated by commas, but each operand may be coded only once.

Whenever a Halt or Halt and Branch instruction is encountered, the contents of the emulated A-address, B-address, and I-address registers are compared with information in the EMCTL statement. The emulator tests the registers in the order:

A-address register (AAR)
B-address register (BAR)
I-address register (IAR)

The results of the possible combinations are listed in Figure 20.

Only one address need be specified when one address is used for all halts in the 1400 program. When there is more than one halt address, one operand should be specified for each.

This statement has three operands when decimal addresses of locations in emulated 1400 storage are specified. The first position of 1400 storage is location 0. Addresses outside the boundaries of 1400 storage are never placed in the IAR, AAR, or BAR, and need not be specified.

EOJAADDR=nnnnn
is the contents of the emulated AAR when the 1400 program reaches normal end-of-job. nnnnn must be a decimal value. Whenever the 1400 program issues a Halt or Halt and Branch instruction, the emulator compares the

contents of the AAR with the address specified in the EOJAADDR operand. If the addresses match and the EOJBADDR and EOJIADDR operands were not specified, the emulator ends the 1400 job. Figure 20 shows the interaction of EOJAADDR with other operands.

MESSAGE	EOJIADDR	EOJAADDR	EOJBADR
IIQ023	=[1]	any combinations	
	≠[1] or not specified	=[1]	not specified
		not specified	=[1]
		=[1]	=[1]
IIQ021 IIQ022	all other combinations		

[1] indicates that the contents of the emulated register and the specified operand (nnnnn) are equal (=) or different (≠).

Figure 20. Combinations of EMCTL Operands

EOJBADDR=nnnnn

is the contents of the emulated BAR when the 1400 program reaches normal end-of-job. nnnnn must be a decimal value. Whenever the 1400 program issues a Halt or Halt and Branch instruction, the emulator compares the contents of the BAR with the address specified in the EOJBADDR operand. If the addresses match and the EOJAADDR and EOJIADDR operands were not specified, the emulator ends the 1400 job. Figure 20 shows the interaction of EOJBADDR with other operands.

EOJIADDR=nnnnn

is the contents of the emulated IAR when the 1400 program reaches normal end-of-job. nnnnn must be a decimal value. Whenever the 1400 program issues a Halt or Halt and Branch instruction, the emulator compares the contents of the IAR with the address specified in the EOJIADDR operand. These addresses are compared after the AAR and BAR have been compared. If the addresses match, the emulator program ends the 1400 job. Figure 20 shows the interaction of EOJIADDR with other operands.

SENSE[=abcdefg]

turns on the indicated sense switches. All switches are turned on by specifying SENSE with no sub-operands. If you want one or more (but not all) switches turned on, code the SENSE operand using capital letters for the switches to be turned on (for example, SENSE=BFG). All other switches are turned off.

OPSERV

indicates that you want communication between the emulator and the operator. The OPSERV operand causes the emulator to issue the WTOR message:

IIQ001I jobname OPERATOR SERVICES AVAILABLE

If the OPSERV operand is omitted, the WTOR message is not issued and the operator has no way of entering emulator commands or control statements except on request from the emulator or at a 1400 halt.

Omitting this operand slightly increases emulator performance. However, the operator cannot interrupt the 1400 program.

This operand should be coded until the 1400 program runs properly. It can then be omitted for those 1400 programs that do not need to issue message IIQ001I.

When several 1400 programs are being executed in a single OS job step, replying to this message during the execution of a 1400 program for which OPSERV is not specified has no effect.

LOAD - Load 1400 Program

Use the LOAD statement to indicate the type of device used to load the 1400 program. The LOAD statement is required. It must be the last emulator control statement before the 1400 program; that is, it follows all the emulator control statements entered into the input stream except the)LC card.

CARD

indicates that the 1400 program was loaded from the card reader. The emulator uses the ddname from the UR emulator control statement for the card reader to locate the System/370 device that contains the 1400 program. If there is no UR statement for the card reader, the emulator stops.

TAPE

indicates that the 1400 program was loaded from a tape unit. The emulator uses the ddname from the TAPE emulator control statement with UNIT=11 to locate the System/370 device that contains the 1400 program. If there is no TAPE statement with UNIT=11, the emulator stops.

TAPE - 1400 Tape Information

Use the TAPE statement to identify the 1400 tape units to be emulated. One TAPE statement must be coded for each tape unit used by the 1400 program. If the 1400 program uses multiple-volume files, only the first file can be identified using the TAPE statement. Subsequent files must be identified by the operator after the first file has been processed.

Multiple operands may be specified if they are separated by commas, but each operand may be specified only once.

UNIT=cu
identifies the channel number (c) and unit number (u) of the 1400 tape unit being emulated. This operand must be specified. c must be 1; u is a value from 1 through 6.

NAME=ddname
identifies the DD statement that describes the System/370 device that is to emulate the 1400 tape unit. This operand is used to connect emulator control information in the TAPE statement with system control information provided in JCL statements.

Although the NAME operand is optional, the ddname associated with the first file on the unit should be specified. If the ddname is not specified, the emulator asks the operator to assign one during execution. The operator must then type the entire TAPE statement on the console.

ALTMODE
specifies whether the tape is in alternate mode or normal mode. If the tape is in normal mode, specify ALTMODE=NO, or omit the operand. If the tape is in alternate mode, specify ALTMODE=YES.

A tape in normal mode, even parity, contains a 1 in bit 1 of the EBCDIC character; a tape in normal mode, odd parity, contains a 0 in bit 1:

n1nnnnnn - Even parity
n0nnnnnn - Odd parity

When tapes are in alternate mode, the emulator does not make a distinction between odd and even parity. Data in alternate mode is the same as data in normal mode, even parity.

TYPEFLE=
INOUT
indicates that an input tape may later be used as an output tape. If TYPEFLE is not specified, INOUT is assumed.

INPUT
the emulator opens the file for input. (You need not reply to the OS file protect warning message.)

OUTPUT
opens the output data set when the emulator might open it incorrectly. This operand should be specified when the tape is an output tape, but the first I/O operation is a read operation to check for a 1400 label. When the file is opened, a 20-byte record is written on the tape, and the tape is then repositioned to the beginning. This is done for all data formats.

DILCNT=nnn
indicates the number of 1400 instructions that must be executed before a 1400 I/O interruption occurs for an overlapped instruction. nnn is a decimal value from 0 through 126. If this operand is omitted, a value of 25 is assumed. This operand makes it possible to execute most time-dependent programs. (For more information, see "Time-Dependent Programs" in the section "Emulator - 1400 Differences.")

UR - 1400 Unit Record Information

Use the UR statement to identify 1400 unit record devices to be emulated. One UR statement must be coded for each unit record device used by the 1400 program. Multiple operands may be specified if they are separated by commas, but each operand may be specified only once.

UNIT
identifies the 1400 unit record device being emulated. This operand is required. CNSL specifies the console, PNCH the card punch, PTR the printer, and RDR the card reader.

NAME=ddname
identifies the DD statement that describes the System/370 device that is to emulate the 1400 device. This operand is used to connect emulator control information in the UR statement with system control information in JCL statements. This operand should not be used if the unit is a console. When STACKER=YES, 1, 2 or 3 is coded, the ddname must not be SYSEMCTL.

STACKER

indicates that the Selective Stacker feature is to be emulated. (If, however, STACKER was not in the IODVTYP parameter of the EM1401 macro instruction, the Select Stacker operation is not performed.) When the operand YES is coded, the emulator stacks the cards read or punched in the pocket selected by the 1400 program; when the operand 1, 2, or 3 is coded, the emulator stacks all cards in the appropriate pocket and ignores the 1400 program stacker request. When STACKER=YES, 1, 2, or 3 is coded, the ddname of the NAME parameter must not be SYSEMCTL.

DILCNT=nnn

indicates the number of 1400 instructions that must be executed before a 1400 I/O interruption occurs for an overlapped instruction. nnn is a decimal value from 0 through 126. If omitted, 25 is assumed. This operand is not needed if the unit is a console. This operand makes it possible to execute most time-dependent programs. For more information, see "Time-Dependent Programs" in the section "Emulator - 1400 Differences."

USER - Include Your Routine

This statement gives information about a routine you have written that you want loaded with the emulator. When the emulator is initialized, the operation code table is modified so that it points to your routine rather than an emulator routine. The USER statement and its operands are discussed in Appendix F.

)LC - End of Data

Use the)LC statement to show the end of data for the 1400 program or, if there is no data, to show the end of the 1400 program.

)LC

indicates that the last card of BCD data or, if there is no data, of the 1400 program follows. If there is more than one data or program card immediately after the)LC statement, the first one is used and the rest are ignored.

If the emulator runs out of data cards before finding the)LC statement, it continues to issue read operations until a JCL card is encountered, whereupon the emulator ends the 1400 job, issuing the message:

IIQ072D CARD INPUT EXHAUSTED

)RC - Emulating Several 1400 Programs

Use the)RC statement when the emulator is to execute several 1400 programs in the same OS job step.

)RC

indicates that a 1400 program follows immediately. This card must be used only when you are executing several 1400 programs in the same OS job step; it must be inserted before every set of emulator control cards except the first.

EXAMPLES OF JCL AND EMULATOR CONTROL STATEMENTS

EXAMPLE 1: Figure 17 shows an emulator job. This example assumes a 1401 program that requires:

- One card reader
- One card punch (PUNCH1 DD statement)
- One printer (PRINT1 DD statement)
- Two tape files in spanned format using the same nine-track tape unit (TAPE1 and TAPE1A DD statements)
- One tape file in 1400 format on a seven-track tape (TAPE2 DD statement)
- One 1311 file on one 2314 disk pack (DISK1 DD statement)
- The operator console

The DD statement named TAPE1A is not referred to in the emulator control statements. After the tape file defined by the TAPE1 DD statement has been removed, the tape file defined by the TAPE1A DD statement can be mounted. Control information for TAPE1A must be typed on the console by the operator, as follows:

```
TAPE  UNIT=12,NAME=TAPE1A
```

The DISK1 DD statement defines the data set created by the disk formatting program for the 1311 Disk Storage Drive defined by the DISK statement. The data set name (HIJ) must be the same as the one created by the disk formatting program. Since LOAD CARD is specified, the emulator uses the ddname in the UR statement with UNIT=RDR to determine the device on which the 1400 program is located.

Note: Performance of the emulator job is reduced when the data sets described by the PRINT1 and SYSEMCTL DD statements are on the same disk drive.

```
//HOLDJOB    JOB    MSGLEVEL=(1,1),REGION=100K,CLASS=A,TYPRUN=HOLD
//STEP       EXEC   PGM=EMUL14
//SYSUDUMP   DD     SYSOUT=A
//SYSPRINT   DD     SYSOUT=A
//SYSEMOUT   DD     SYSOUT=A
//READER     DD     UNIT=00C,DCB=(RECFM=F,BLKSIZE=80)
//PUNCH      DD     UNIT=00D,DCB=(RECFM=FA,BLKSIZE=80)
//PRINTER    DD     UNIT=00E,DCB=(RECFM=FA,BLKSIZE=133)
//DISK0      DD     UNIT=2311,VOL=SER=RPG401,DCB=(RECFM=FT),               X
//                  DISP=OLD,DSN=MSFILE
//SYSEMCTL   DD     DATA
             DISK   UNIT=10,NAME=DISK0,FROMDEV=1311,BUFID=10,MODE=MIXED
             UR     UNIT=PNCH,NAME=PUNCH
             UR     UNIT=RDR,NAME=READER
             UR     UNIT=PTR,NAME=PRINTER
             UR     UNIT=CNSL
             LOAD   CARD
/*
```

Figure 21. JCL and Emulator Control Statements - Example 2

EXAMPLE 2: Figure 21 shows an emulator job that uses online unit record devices. Only the cards shown in Figure 21 are placed in the card reader. The operating system reads the cards into the hold queue because TYPRUN=HOLD was coded in the JOB statement.

After the reader has been closed, a 1400 program and its data should be placed in the card reader and the job released. The printer, punch, and reader must not be allocated to another job nor to a system task.

EXAMPLE 3: An emulator job that uses two tapes but only one tape drive is shown in Figure 22. To use only the number of tape drives needed to execute the 1400 program, the unit affinity parameter (UNIT=AFF=TAPE1) is used. By having separate DD statements, a DCB (data control block) is created for each tape file. The first file is a 1400 tape-to-print program (TAPE1 DD statement). The second file is the data to be printed (TAPE1A DD statement).

After the program has been read, the operator reassigns the tape drive by typing:

REPLY id,'TAPE UNIT=11,NAME=TAPE1A'

The emulator then closes the data set that contains the 1400 program and rewinds and unloads the tape. The operator then mounts the next tape and continues by typing:

REPLY id,'START'

Note: A specific value has been requested for the space parameter in the PRINTER DD statements.

EXAMPLE 4: The emulator job in Figure 23 executes a 1400 program that needs four tape units, but only three are used at a

```
//TAPEOPT    JOB    MSGLEVEL=(1,1),REGION=100K
//STEP       EXEC   PGM=EMUL14
//SYSUDUMP   DD     SYSOUT=A
//SYSPRINT   DD     SYSOUT=A
//SYSEMOUT   DD     SYSOUT=A
//PRINTER    DD     SYSOUT=A,SPACE=(CYL,(10,10))
//TAPE1      DD     UNIT=2400-2,LABEL=(,NL),DISP=(OLD,KEEP),DCB=(RECFM=VBS,      X
//                  DEN=2,BLKSIZE=3200,TRTCH=C),VOL=SER=TP1
//TAPE1A     DD     UNIT=AFF=TAPE1,LABEL=(,NL),DISP=(OLD,KEEP),DCB=(RECFM=U,     X
//                  DEN=1,BLKSIZE=5000,TRTCH=ET),VOL=SER=TP1A
//SYSEMCTL   DD     *
             EMCTL  OPSERV,SENSE=B
             TAPE   UNIT=11,NAME=TAPE1
             UR     UNIT=CNSL
             UR     UNIT=PTR,NAME=PRINTER
             LOAD   TAPE
```

Figure 22. JCL and Emulator Control Statements - Example 3

time. Only three tape units are defined in JCL statements; the characteristics of the fourth are the same as those of the first. When the emulator cannot open the dummy file, it issues message IIQ054I TAPEJOB FILE=DUMMY COULD NOT BE OPENED. The operator should respond:

```
REPLY id,'TAPE UNIT=11,NAME=DUMMY'
REPLY id,'TAPE UNIT=15,NAME=TAPE1'
REPLY id,'START'
```

The first two responses swap tape files, making the active file the dummy and the dummy file active. Before typing the START command, the operator must mount the fourth tape file on the same unit used for the first.

EXAMPLE 5: Figure 24 shows an emulator job with a buffer shared by two 1311 disk units. The buffer identification is 01 (BUFID=01).

EXAMPLE 6: Figure 25 shows an emulator job of two 1400 programs in the same OS job step. The first 1400 job stops at a halt with AAR=500 and gives control to the initialization routine to start the second 1400 job. The Selective Stacker feature is emulated in the second job.

LOADING THE 1400 PROGRAM

The emulator loads the 1400 program from the device specified in the LOAD emulator control statement:

- If LOAD CARD is specified, the emulator finds the System/370 device that emulates the 1400 card reader and it emulates the Load key of the 1400 reader. This reads the first record of the card file into emulated 1400 storage, and gives it control.

- If LOAD TAPE is specified, the emulator finds the System/370 device that emulates the tape unit connected to the 1400 processor unit at channel 1 unit position 1 and emulates the Load Tape key of the 1400 processor unit. This reads the first record of the tape file into emulated 1400 storage, and gives it control.

EXECUTING THE 1400 PROGRAM

The emulator executes the 1400 program by fetching, interpreting, and executing each 1400 instruction. It uses the data management macro instructions of the operating system to execute 1400 I/O operations. The emulator continues to execute 1400 instructions until:

- The emulator must wait for some action by the operator. (Once the action is completed the emulator continues.)

- An end-of-1400-job condition is recognized.

- An unrecoverable error occurs.

End-of-Job Conditions

The end-of-job conditions for the 1400 program are:

- A 1400 Halt or Halt and Branch instruction is encountered and the end-of-job condition established by the EMCTL emulator control statement is met.

- The EOJ emulator command is typed by the operator.

- An unrecoverable error occurs in the 1400 program; for example, there is invalid data in 1400 storage.

If a)RC control statement follows the 1400 program in which the end-of-job condition occurred, the emulator proceeds to execute the next 1400 program. The operator can end the entire OS job step by entering the EOJ ALL or EOJ A command.

Error Handling

The emulator recognizes the following error conditions of the 1400 processor unit:

- Invalid instruction format

- Invalid operation code

- Invalid I/O instruction

- Invalid address

- Wrong-length record

- Data check

When one of these errors occurs, a message is printed on the operator console to explain the error, or error indicators are set to pass the error to the 1400 program. Operator commands ALTER, SET, TN, and TF may be used to change information in the 1400 program (see "Communicating With the Emulator").

I/O errors for all devices are processed by OS error recovery procedures. Those that are corrected are not normally passed to the 1400 program. Uncorrectable I/O errors on tape input and on disk, such as permanent I/O errors, data checks, and wrong-length length records, are recorded by the emulator, and control is returned to

```
//TAPEJOB     JOB    MSGLEVEL=(1,1),REGION=112K,TIME=60
//STEP        EXEC   PGM=EMUL14
//SYSUDUMP    DD     SYSOUT=A
//SYSPRINT    DD     SYSOUT=A
//SYSEMOUT    DD     SYSOUT=A
//TAPE1       DD     UNIT=2400,LABEL=(,NL),DCB=(RECFM=U,BLKSIZE=4800),            X
//                   DISP=(OLD,KEEP),VOL=SER=TP1
//TAPE2       DD     UNIT=2400,LABEL=(,NL),DCB=(RECFM=VBS,BLKSIZE=3000),          X
//                   DISP=(NEW,KEEP),VOL=SER=TP2
//TAPE3       DD     UNIT=2400-2,VOL=SER=TP3,DISP=(OLD,KEEP),LABEL=(,NL),         X
//                   DCB=(RECFM=U,DEN=1,BLKSIZE=16000,TRTCH=ET)
//PRINT       DD     SYSOUT=A
//SYSEMCTL    DD     *
              EMCTL  OPSERV
              TAPE   UNIT=11,NAME=TAPE1
              TAPE   UNIT=12,NAME=TAPE2
              TAPE   UNIT=13,NAME=TAPE3
              TAPE   UNIT=15,NAME=DUMMY
              UR     UNIT=CNSL
              UR     UNIT=PTR,NAME=PRINT
              UR     UNIT=RDR,NAME=SYSEMCTL
              LOAD   CARD
The 1400 program (except for the last card) goes here. There is no data.
)LC
The last card goes here.
```

Figure 23. JCL and Emulator Control Statements - Example 4

```
//BUFSHR      JOB    MSGLEVEL=(1,1),TIME=90,REGION=100K
//STEP        EXEC   PGM=IIQE14
//SYSUDUMP    DD     SYSOUT=A
//SYSEMOUT    DD     SYSOUT=A
//SYSPRINT    DD     SYSOUT=A
//DISK0       DD     DSNAME=D13110,UNIT=2311,DISP=(OLD,KEEP),VOL=SER=0025EU
//DISK1       DD     DSNAME=D1301,UNIT=2314,DISP=(OLD,KEEP),VOL=SER=0030EU
//DISK2       DD     DSNAME=D13114,UNIT=2311,DISP=(OLD,KEEP),VOL=SER=0026EU
//SYSEMCTL    DD     *
              DISK   UNIT=10,NAME=DISK0,FROMDEV=1311,MODE=MIXED,BUFID=01
              DISK   UNIT=10,NAME=DISK1,FROMDEV=1301,MODE=MIXED,BUFID=02
              DISK   UNIT=12,NAME=DISK2,FROMDEV=1311,MODE=MIXED,BUFID=01
              UR     UNIT=CNSL
              UR     UNIT=RDR,NAME=SYSEMCTL
              LOAD   CARD
The 1400 program goes here.
The 1400 data goes here (except for the last data card).
)LC
The last data card goes here.
/*
```

Figure 24. JCL and Emulator Control Statements - Example 5

```
//RESTART   JOB   MSGLEVEL=(1,1),TIME=80,REGION=100K
//STEP      EXEC  PGM=IIQE14
//SYSUDUMP  DD    SYSOUT=A
//SYSEMOUT  DD    SYSOUT=A
//SYSPRINT  DD    SYSOUT=A
//READER    DD    UNIT=00C,DCB=(RECFM=F,BLKSIZE=80)
//PUNCH     DD    UNIT=00D,DCB=(RECFM=FA,BLKSIZE=80)
//SYSPNCH   DD    SYSOUT=B
//DISK0     DD    DSNAME=D13110,UNIT=2311,DISP=(OLD,KEEPe,                  X
//                VOL=SER=666666
//TAPE1     DD    UNIT=2400,LABEL=(,NL),DCB=(RECFM=U,BLKSIZE=4800),         X
//                DISP=OLD,VOL=SER=TP1
//TAPE2     DD    UNIT=2400,LABEL=(,NL),DCB=(RECFM=VBS,BLKSIZE=3000),       X
//                DISP=OLD,VOL=SER=TP2
//TAPE3     DD    UNIT=2314,VOL=SER=6666,DISP=NEW,SPACE=(CYL,(50,10)),      X
//                DCB=(RECFM=VBS,BLKSIZE=3500)
//TAPE01    DD    UNIT=2400,LABEL=(,NL),DCB=(RECFM=U,BLKSIZE=4800),         X
//                DISP=OLD,VOL=SER=TP01
//SYSEMCTL  DD    *
            EMCTL OPSERV,SENSE=B,EOJAADDR=500
            DISK  UNIT=10,NAME=DISK0,FROMDEV=1311
            TAPE  UNIT=11,NAME=TAPE1
            TAPE  UNIT=12,NAME=TAPE2,TYPEFLE=OUTPUT
            UR    UNIT=PNCH,NAME=SYSPNCH
            UR    UNIT=CNSL
            UR    UNIT=PTR,NAME=SYSPRINT
            UR    UNIT=RDR,NAME=SYSEMCTL
            LOAD  CARD
The  1400 program goes here.
The  1400 data (except for the last card) goes here.
)LC
The  last card goes here.
)RC
            EMCTL OPSERV,EOJBADDR=111
            TAPE  UNIT=11,NAME=TAPE01
            TAPE  UNIT=12,NAME=TAPE2,TYPEFLE=OUTPUT
            TAPE  UNIT=13,NAME=TAPE3,TYPEFLE=OUTPUT
            DISK  UNIT=10,NAME=DISK0,FROMDEV=1311
            UR    UNIT=RDR,NAME=READER,STACKER=YES
            UR    UNIT=PTR,NAME=SYSPRINT
            UR    UNIT=PNCH,NAME=PUNCH
            LOAD  CARD
The 1400 program goes here.
The 1400 data (except for the last card) goes here.
)LC
The last card goes here.
```

Figure 25. JCL and Emulator Control Statements - Example 6

the 1400 program. The 1400 program can either ignore the error or test for the various types of errors using 1400 instructions. The emulator abnormally ends the 1400 program when it encounters an uncorrectable error on tape output or unit record operations.

JCL statements are scanned by the operating system for syntax errors. If an error is found, the job is ended but the remainder of the JCL is scanned; emulator control statements are ignored. Emulator control statements are analyzed by the emulator during initialization. If a control statement error is found, the 1400 job is ended, but the remainder of the statements are scanned and error messages are printed on the system output device.

Operating system errors, such as inability to allocate a device, and hardware errors, are handled by the operating system. The operator is informed so that he can take appropriate action.

The emulator uses the dynamic device reconfiguration (DDR) facilities of the operating system for all devices except tape devices that have 1400 format tapes. Additional information about DDR can be found in Operator's Reference, GC28-6691.

CATALOGUING

A 1400 object deck can be catalogued on a direct access device. For example, using the IEBUPDTE utility program, each object deck can be catalogued as a member of a partitioned data set (a library of 1400 programs) or a sequential data set.

Probably the easiest way to execute a 1400 program is to catalog the program, the emulator control statements, and the data in the same data set. If they are catalogued together, components must be in the same order as they appear in the input stream: control statements, 1400 program, and BCD data. If your 1400 program cannot be catalogued with control statements and data, each component should be catalogued separately.

If the program to be executed is in a 1400 program library, the ddname of the DD statement that describes the library must be specified in the NAME operand of the emulator control statement that describes the 1400 device used for program loading. For example, if the 1400 program library is on a 2311 with a volume serial number of 145000, the device originally used for program loading was the card reader, and the 1400 program to be executed has the data set name P14KKA, the backward reference would be:

```
            .
            .
            .
//PROGLIB   DD DSNAME=P14KKA,UNIT=2311,    X
//             VOL=SER=145000,DISP=OLD
            .
            .
            .
//SYSEMCTL  DD *
            UR UNIT=RDR,NAME=PROGLIB
            .
            .
            .
```

Emulator control statements, which are normally in the input stream, may be on tape or disk. If the emulator control statements are in the input stream, the SYSEMCTL DD statement should have an asterisk or DATA in the operand field and should be followed by emulator control statements, as shown in Figure 24.

Emulator control statements must have been catalogued by a previous job step or utility program in order to be called from tape or disk by the emulator. Once catalogued, the data set name and other information are specified in the SYSEMCTL DD statement. For example, if the emulator control statements and the 1400 program are catalogued on the same 2311 as the 1400 program library used in the example above, and the data for the 1400 program is in the input stream, the SYSEMCTL DD statement would be modified as follows:

```
            .
            .
            .
//SYSEMCTL  DD DSNAME=P14ECS,UNIT=2311,    X
//             VOL=SER=145000,DISP=OLD
//PROGLIB   DD DSNAME=P14KKA,UNIT=2311,    X
//             VOL=SER=145000,DISP=OLD
//          DD *
The BCD data (except for the last card)
should be entered here.
)LC
The last card of BCD data should be entered
here.
/*
```

In this example, the SYSEMCTL data set on the program library contains either:

```
            .
            .
            .
            UR    UNIT=RDR,NAME=PROGLIB
            LOAD  CARD
or
            .
            .
            .
            TAPE  UNIT=11,NAME=PROGLIB
            LOAD  TAPE
```

When the 1400 program is catalogued, all of the following conditions must be met:

- The type of device from which the program is loaded must be the same type of device as the OS CPP (concurrent periphical processing) data set. The CPP data set is used for intermediate storage of input stream data.

- The program has the same DCB attributes as the CPP data set. (For example, the block sizes must be equal.)

- The 1400 program and data are concatenated, as the DD statements in the example show.

Additional information on the CPP data set is in System Programmer's Guide, GC28-6550.

COMMUNICATING WITH THE EMULATOR

The operator communicates with the emulator by responding to messages it issues. He can respond to any WTOR (Write To Operator with Reply) message issued by the emulator. He must respond using the operating system REPLY command, and within the reply, he types the emulator command. The format is:

```
REPLY id,'command  keyword=operand'
```

The emulator command with its associated keywords and operands becomes the text of the REPLY command. The emulator command and its keyword and operands must be enclosed in single quotes. Using the commands, the operator can get displays and dumps, and can modify the 1400 program. Most emulator commands have several keywords, and many keywords have several operands. The operating system uses the id to identify the REPLY command as a response to the emulator message.

The programmer who codes the emulator control statements determines whether the operator can interrupt the 1400 program.

If the programmer includes the OPSERV parameter in the EMCTL emulator control statement, the operator can interrupt the 1400 program. The OPSERV parameter causes the emulator to issue a WTOR message just before the program begins. The operator should not reply until he needs to type an emulator command. When the operator replies, the emulator issues another WTOR message, so that another command can be entered. When a command is entered, the emulator is interrupted and the command is executed. If, however, there are several 1400 programs being executed in a single OS job step, replying to the WTOR message during the execution of a 1400 program for which OPSERV is not specified has no effect.

If the OPSERV parameter is not coded, the operator cannot interrupt the 1400 program. He can, however, type most emulator commands in response to other messages issued by the emulator. Any emulator command can be typed when the emulator encounters a Halt or Halt and Branch instruction in the 1400 program.

Note: If your emmulator has an automatic reply routine, you can enter a message only if the OPERATOR SERVICES AVAILABLE message is printed on the console.

EMULATOR COMMANDS

All emulator commands are defined in Figure 31. This figure, located just before the appendixes, can be folded out and read with the rest of this section. When you become familiar with the definitions of each command and operand, Figure 31 can be used as a reference when typing emulator commands.

The rules that the emulator follows when replying to commands are the same ones that must be followed when typing commands. They are:

- BCD data is typed or printed in load mode. Only the ALTER command can be used to enter BCD data in move mode.

- A BCD character with a wordmark is typed or printed with an underscore preceding the character.

- A word separator is printed as an underscore. A word separator preceding a BCD character with a wordmark is displayed as two underscores followed by the character.

- A word separator cannot be typed at the console in load mode; it would be interpreted as a wordmark.

- Two commas must be typed to get one comma in 1400 storage. Single commas are presumed to be delimiters. For example, if you need a string of three commas in storage, type six. By typing seven, you get three in storage and the emulator uses the seventh as a delimiter.

- Some BCD characters cannot be typed or printed on System/370 consoles. Figure 26 shows graphic differences and should be used to determine which BCD character a System/370 character represents.

All emulator control statements except CHBUF and the)LC and)RC cards can be typed at the console. You can type them to change the emulator during execution of the 1400 program. Additional information is given in this section under "Typing Emulator Control Statements at the Console" and in the section "Executing the Emulator" under the definition of each control statement.

Syntax and coding conventions are defined in Appendix B.

1407 Console Character	1447 Console Character	Corresponding System/370 Character
⌑	⌑	<
(	[	(
<	<	+
⧻	⧻	|
)	]	)
△	△	¬
=	ᵐ	_ (underscore)
'	\	>
"	⧺	?
¢	ƀ	:
:	:	'
>	>	=
√	✓	"
?	?	g
!	!	p
ǂ	ǂ	x
b	blank	space

Figure 26. Differences in Console Graphics

Abbreviations for emulator commands are shown in Figure 31. Only the first four characters of a command need be typed to make the command valid. For example, the START command can be typed as either STAR or S.

Emulator commands can be used to emulate 1400 console operations, and as debugging and maintenance aids. Commands that have operands starting with X, such as ALTER XADDR=nnnnnn,XDATA=string, are useful for emulator maintenance. Operands beginning with X allow EBCDIC characters to be entered anywhere in the emulator partition.

CAUTION: Hexadecimal addresses specified in the XADDR operand must be within the limits of the emulator partition. If they are not, the emulator receives either a program check (if the address is outside main storage) or a protection check (if the address is outside the emulator partition and in a protected area of main storage). Either one causes the operating system to end the emulator job. Accordingly, hexadecimal addresses should be used only by a system programmer during program debugging.

CODING COMMONLY USED OPERANDS

The ALTER, CONVERT, DISPLAY, DUMP, SET, and START commands have operands in which you must specify the decimal address of a location in emulated 1400 storage. The first position of 1400 storage is location 0. The largest address that can be specified is limited by the size of the 1400 system being emulated. If the address is higher than the highest address in 1400 storage, the message IIQ024D jobname ADDRESS=addr IS INVALID IN COMMAND is issued, and the command is ignored.

The ALTER and DISPLAY commands handle data. The data is typed in EBCDIC by using the XDATA operand, in BCD by using the DATA operand, or in emulator internal code by using the XADDR operand to address 1400 storage and the XDATA operand to enter the internal code.

HOW TO TYPE EMULATOR COMMANDS

You must use the REPLY command of the operating system to reply to all emulator messages containing a reply identification. The REPLY command is described in Operator's Reference, GC28-6691. The text of the REPLY command must be an emulator command. For example, to display the I/O configuration for a 1400 program, you type:

```
REPLY id,'DISPLAY CONFIG'
```

Emulator commands longer than one line can be continued by typing a hyphen in the position following the last character in the line. The emulator recognizes the hyphen as a continuation character and does not include it as part of the command. It then issues the message IIQ034A jobname CONTINUE INPUT OF COMMAND STRING, so that the next line can be typed. Commands or command strings as long as 485 console characters can be typed.

An embedded hyphen, that is, one between two other characters, is not a continuation character and is considered as part of the command. When two or more hyphens are typed at the end of the line, all but the last one are considered part of the command, and the last one is a continuation character.

If the last character of the command must be a hyphen, enter an extra one for the continuation character. When the continuation message is received, type:

```
REPLY id,''.
```

When the emulator processes your command, it prints the message IIQ061D EMULATOR WAITING. You can then type other commands; to continue executing the 1400 program , type REPLY id,'START'. The TN INQUIRY command is an exception to this rule. If you type it, the emulator automatically continues at the next sequential instruction of the 1400 program.

ALTER - Modify Storage

Use the ALTER command to change System/370 main storage within the emulator partition, or to change emulated 1400 storage. The ALTER command is useful during debugging and maintenance of a 1400 program. A separate ALTER command must be used for each contiguous string of data you type.

ADDR=nnnnn
: is the emulated 1400 storage address at which the data is to be stored. nnnnn must be a decimal value.

DATA=data
: is a string of BCD data that is stored at the 1400 storage location nnnnn. The string of data can be one or more characters long. Valid characters are listed in the 1400 graphic column of Appendix A. Figure 26 shows graphic differences, and should be used to determine which BCD character a System/370 character represents. If the last character in the data string is a blank, invert the order of the ADDR and DATA operands as shown below:

```
ALTER  DATA=_J00138 ,ADDR=1573
```

<u>Note</u>: If you type an invalid BCD character, it is written as an asterisk in 1400 storage.

MODE
: is the mode in which the data is entered. If MODE=M, data is entered in move mode. If MODE=L, or if the MODE operand is omitted, data is entered in load mode.

XADDR=nnnnnn
: is the main storage address (in hexadecimal) at which the data is to be stored.

XDATA=data
: is a string of EBCDIC data that is to be stored at main storage location nnnnnn. The string of data can be two or more characters long, with each two characters being the hexadecimal equivalent of one byte of storage. This operand is invalid if you type an odd number of characters. If the data is to be stored in emulated 1400 storage, the characters must be typed using the hexadecimal representation of internal code (see Appendix A).

<u>EXAMPLE 1</u>: Alter location 650 of emulated 1400 storage to contain character A.

```
REPLY id,'ALTER ADDR=650,DATA=A'
```

<u>EXAMPLE 2</u>: Alter the Punch Card and Branch instruction at location 680 to branch to location 1758. Location 1758 is X58 in 1400 machine language.

```
REPLY id,'ALTER ADDR=680,DATA=_4X58'
                       or
REPLY id,'ALTER ADDR=681,DATA=X58'
```

<u>EXAMPLE 3</u>: Alter the word of data at main storage location 66AC0 to read "P 10" when printed on the printer. Location 66AC0 is in the emulator partition, but not in emulated 1400 storage.

```
REPLY id,'ALTER XADDR=66AC0,XDATA=D740F1F0'
```

CLEAR - Clear 1400 Storage

Use the CLEAR command to set all of 1400 storage to a given character. The character can be typed with or without a wordmark.

CORE
: The emulator program is to clear emulated 1400 storage to the BCD character indicated by n, or the BCD character with wordmark indicated by _n. Valid characters are listed in the 1400 graphic column of Appendix A. Figure 26 shows graphic differences.

  If no operand is specified, 1400 storage is cleared to blanks. If you want to clear storage to blanks with wordmarks, the command must be coded REPLY id,'CLEAR CORE=_ ', with a blank following the underscore that represents the wordmark.

CONVERT - Convert to Hexadecimal Address

Use the CONVERT command to convert a 1400 storage address to its corresponding System/370 main storage address. Since the emulator is relocatable, the location of 1400 storage will not be the same from one run to the next. To find the System/370 address of a known displacement in 1400 storage, type the CONVERT command.

ADDR=nnnnn
: is the 1400 address to be converted. nnnnn must be a decimal value.

DISPLAY - Display on Console

Use the DISPLAY command to display information about the emulator and the 1400 job. This command should not be confused with the OS display command. The emulator DISPLAY command must be in the text field of the OS REPLY command; for example, it must be entered REPLY id,'DISPLAY REG'.

At least one operand must be coded with the DISPLAY command. Multiple operands may be entered if they are separated by commas, and operands can be entered more than once.

SENSE
: displays the status of sense switches A through G for the 1400 system. Each switch is either on or off; only the switches that are on are displayed.

INQUIRY
: displays the status of the 1400 inquiry status latch. The latch is either on or off, but the status is displayed only if it is on.

REG
: displays the contents of the I-address, A-address, and B-address registers. The contents are displayed as decimal values.

STATUS
: displays the SENSE, INQUIRY, and REG operands.

TAPE[=cu]
: displays both the name of the DD statement that defines the System/370 device used to emulate the tape unit and the System/370 device address. cu identifies the device by its channel (c) and unit (u) number. c must be 1. u is a value from 1 through 6. All tape unit assignments are displayed if only TAPE is specified.

If there is no tape unit at the channel and unit addresses specified, message IIQ065I jobname 1400 DEVICE cccc NOT ASSIGNED is displayed. If there is a tape unit at the addresses specified, but there is no System/370 device to emulate it, the message IIQ066I jobname TPcu=blanks=blanks is displayed.

DISK[=cm]
: displays the name of the DD statement that defines the System/370 device used to emulate the disk unit. cm identifies a 1311 disk unit by its channel (c) and module (m) number. c must be 1. m must be 0, 2, 4, 6, or 8. All disk unit assignments are displayed if only DISK is specified.

If there is no 1311 disk unit at the addresses specified, message IIQ065I jobname 1400 DEVICE cccc NOT ASSIGNED is displayed. If there is a disk unit, but there is no System/370 device to emulate it, the message IIQ066I jobname cm=blanks is displayed.

UR
: displays the names of the DD statements that define the System/370 devices used to emulate the unit record devices assigned to the 1400 job and displays the addresses of the System/370 devices. If there are no unit record devices specified, message IIQ065I jobname 1400 DEVICE UR NOT ASSIGNED is displayed. If there is a unit record device specified, but there is no System/370 device to emulate it, the message IIQ066I jobname unit=blanks=blanks is displayed.

CONFIG
: displays the TAPE, DISK, and UR operands.

ADDR=nnnnn
: displays 40 positions of emulated 1400 storage starting at address nnnnn. nnnnn must be a decimal value. The 40 positions of 1400 storage are translated from internal code to EBCDIC. Differences in graphics between the 1407 or 1447 console and System/370 operator consoles are shown in Figure 26.

<u>Note</u>: The number of 1400 storage positions displayed is shortened by one position for each wordmark.

XADDR=nnnnnn
: displays 20 bytes of emulator partition starting at address nnnnnn. nnnnnn must be a hexadecimal address that permits all 20 bytes displayed to be within the emulator partition. Two console characters represent one byte of main storage. The data is not translated as with the ADDR=nnnnn operand. The data is displayed in EBCDIC.

DUMP - Print Main Storage

Use the DUMP command to record the conditions under which an error occurred. You can dump as little as 100 bytes, or you can dump the entire partition. You cannot dump the nucleus or any area of main storage outside your partition that is in a storage protected area.

The dump is printed on the data set defined by the SYSEMOUT DD statement. If the SYSEMOUT DD statement is omitted or is incorrectly specified, no dump is printed. At the beginning of each dump the emulator prints several lines of status information. This information summarizes the CPU and I/O status at the time the dump was requested. Figure 27 shows all the status conditions that can be printed. Abbreviations used in Figure 27 are:

CPU Status

I=nnnnn	Contents of the I-address register.
A=nnnnn	Contents of the A-address register.
B=nnnnn	Contents of the B-address register.
HI	Compare was high.
LO	Compare was low.
EQ	Compare was equal.
ARITH OVFLO	Arithmetic overflow.
INQU REQ	Inquiry request.
INQU CLR	Inquiry clear.
SSW=ABCDEFG	Sense switches specified are on.

I/O Status

cccccccc	Last I/O instruction attempted.
UR-	Unit record.
RDR ERR	Card reader error.
PTR ERR	Printer error.
PNCH ERR	Card punch error.
TP-	Tape.
ERR	Tape Error.
EOF/R	End of file or end of reel.
DK-	Disk.
ERR(x)	Disk error (x=d-modifier).
WLR(x)	Wrong-length record (x=d-modifier).
UNEQ ADDR(x)	Unequal address compare (x=d- modifier).
ACC INOP(x)	Disk unit inoperable (x=d-modifier).

You can dump emulated 1400 storage in BCD on the printer. The format of the 1400-storage dump is shown in Figure 28. A wordmark is represented by a 1 under the character, a groupmark by a 2 under the character, a groupmark wordmark by a 3 under the character. Any position that contains a BCD character that is not on the print chain is left blank. Lines that contain 100 blanks are not printed. (In Figure 27, note the absence of lines 00500, 00600, and 00700, and lines 01800 onward.) Lines that contain unprintable characters are printed. Line 01600 is an example, with the character at location 01610 having a wordmark. To determine what characters are in line 01600, you must use a hexadecimal dump and translate it to BCD. Remember that the characters are in internal code in 1400 storage.

Addresses in 1400 instructions are in machine language. To read the dump, Figure 29 must be used to translate from machine language to decimal numbers. For example, the 1400 instruction at location 001100 in Figure 28 is B/22917. When the addresses are translated using Figure 29, /22 becomes 1122 and 917 remains the same.

```
 JOBNAME    1400 STATUS
  I=nnnnn A=nnnnn B=nnnnn INDICATORS ON - HI,LO,EQ,ARITH OVFLO,INQU REQ,INQU CLR  SSW=ABCDEFG
 LAST I/O INST=cccccccc UR - RDR ERR,PTR ERR,PNCH ERR,LAST CARD,TP-ERR,EOF/R
 DK-ERR(V),WLR(W),UNEQ ADDR(X),ACC INOP(N)
```

Figure 27. Status Information in Emulator Dump

```
jobname        1400 STORAGE PRINT

00000   0120JB420                                                               01V0060061/073B099902 110B1
00099   1  1 1                                                                   11       1   1       1  1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

00100                                                         END OF TEST                        220 B125S220B1
00199                                                                                             1   1   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

00200                                                         END OF TEST
00299

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

00300                                                           /073B368902 1393B377AB385M%G1001RM%G1001R,001B00
00399                                                    3      1   1       1   1    1   1       1       1   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

00400  1001,B416       B352N476I96 B/00
00499       1   1      1   1      21   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

00800            M849199L285185B8449021M%G1101GB9894989N                       V0
00899            1      1      1       1       1   1   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

00900    1                                                                       M+88201B8179884B+31902 B+7
00999                                                                            1      1       1       1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01000  09021B+44M%T0201WB+89*B+17<B+44M%Y1201WB+89+B+56201*BT57N+65EB+52.+5212+39 /201B981CB+52*.9810 4W01C
01099       1   1       1    1    1   1       1    1       1   1    1   1   11   11   1    1   11   1 1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01100  B/22917 Y917/23D917/23U1/344/,V22LV22345MX01/70MX01+56BT669036BT669031BS06MV21225M+99206MX04+55B981M
01199  1       1      1      1 1   11   1      1      1      1       1       1   1      1      1      1   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01200  X05/70/344/,200YS15V45YS22V42Y/95V29MV88344MV88280MV88216MX08V25BU67GMX11+55B981AS80V25BT13V231M3442
01299        1   11   1      1      1      1      1      1      1      1    1      1   1      1       1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01300  00M343344BS64B/54D/344/BT579031BT579036MV04261MX14+55B981.999999NNMW06344MW06MW06MW06MW06MW06MW06MW0
01399    1      1   1    1   11       1       1      1      1   1      111      1   1   1   1   1   1   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01400  6BS57BU529036BU529031MX15344M344MX18+55B981BU32GBT13MX19344M344BU32BU05X00 ,X00MX00344M344BU32END OF
01499   1   1       1       1      1   1      1   1    1   1      1   1   1       1   1      1   1   1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01500   TESTRIPPLE PRINTPUNCH 100 .<   &$*   -/,%    #@   &ABCDEFGHIJKLMNOPQR STUVWXYZ01234567891234567890
01599       1                31  1    2                                                         1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01600
01699            1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........

01700   N/99B000S80T57HU438
01799  111  11  1  1  11  1

       0.........1.........2.........3.........4.........5.........6.........7.........8.........9.........
END OF 1400 STORAGE PRINT
```

Figure 28. Format of the 1400 Storage Dump

	ADDRESSES 0000-3999		ADDRESSES 4000-7999 A-Bit (0-Zone) over Units Position		ADDRESSES 8000-11999 B-Bit (11-Zone) over Units Position		ADDRESSES 12000-15999 AB-Bits (12-Zone) over Units Position	
	Addresses	Codes	Addresses	Codes	Addresses	Codes	Addresses	Codes
	0000-0099	000-099	4000-4099	00ǂ-09Z	8000-8099	00!-09R	12000-12099	00?-09I
	0100-0199	100-199	4100-4199	10ǂ-19Z	8100-8199	10!-19R	12100-12199	10?-19I
	0200-0299	200-299	4200-4299	20ǂ-29Z	8200-8299	20!-29R	12200-12299	20?-29I
	0300-0399	300-399	4300-4399	30ǂ-39Z	8300-8399	30!-39R	12300-12399	30?-39I
	0400-0499	400-499	4400-4499	40ǂ-49Z	8400-8499	40!-49R	12400-12499	40?-49I
	0500-0599	500-599	4500-4599	50ǂ-59Z	8500-8599	50!-59R	12500-12599	50?-59I
	0600-0699	600-699	4600-4699	60ǂ-69Z	8600-8699	60!-69R	12600-12699	60?-69I
	0700-0799	700-799	4700-4799	70ǂ-79Z	8700-8799	70!-79R	12700-12799	70?-79I
	0800-0899	800-899	4800-4899	80ǂ-89Z	8800-8899	80!-89R	12800-12899	80?-89I
	0900-0999	900-999	4900-4999	90ǂ-99Z	8900-8999	90!-99R	12900-12999	90?-99I
A-Bit (0-Zone) over Hundreds Position	1000-1099	ǂ00-ǂ99	5000-5099	ǂ0ǂ-ǂ9Z	9000-9099	ǂ0!-ǂ9R	13000-13099	ǂ0?-ǂ9I
	1100-1199	/00-/99	5100-5199	/0ǂ-/9Z	9100-9199	/0!-/9R	13100-13199	/0?-/9I
	1200-1299	S00-S99	5200-5299	S0ǂ-S9Z	9200-9299	S0!-S9R	13200-13299	S0?-S9I
	1300-1399	T00-T99	5300-5399	T0ǂ-T9Z	9300-9399	T0!-T9R	13300-13399	T0?-T9I
	1400-1499	U00-U99	5400-5499	U0ǂ-U9Z	9400-9499	U0!-U9R	13400-13499	U0?-U9I
	1500-1599	V00-V99	5500-5599	V0ǂ-V9Z	9500-9599	V0!-V9R	13500-13599	V0?-V9I
	1600-1699	W00-W99	5600-5699	W0ǂ-W9Z	9600-9699	W0!-W9R	13600-13699	W0?-W9I
	1700-1799	X00-X99	5700-5799	X0ǂ-X9Z	9700-9799	X0!-X9R	13700-13799	X0?-X9I
	1800-1899	Y00-Y99	5800-5899	Y0ǂ-Y9Z	9800-9899	Y0!-Y9R	13800-13899	Y0?-Y9I
	1900-1999	Z00-Z99	5900-5999	Z0ǂ-Z9Z	9900-9999	Z0!-Z9R	13900-13999	Z0?-Z9I
B-Bit (11-Zone) over Hundreds Position	2000-2099	!00-!99	6000-6099	!0ǂ-!9Z	10000-10099	!0!-!9R	14000-14099	!0?-!9I
	2100-2199	J00-J99	6100-6199	J0ǂ-J9Z	10100-10199	J0!-J9R	14100-14199	J0?-J9I
	2200-2299	K00-K99	6200-6299	K0ǂ-K9Z	10200-10299	K0!-K9R	14200-14299	K0?-K9I
	2300-2399	L00-L99	6300-6399	L0ǂ-L9Z	10300-10399	L0!-L9R	14300-14399	L0?-L9I
	2400-2499	M00-M99	6400-6499	M0ǂ-M9Z	10400-10499	M0!-M9R	14400-14499	M0?-M9I
	2500-2599	N00-N99	6500-6599	N0ǂ-N9Z	10500-10599	N0!-N9R	14500-14599	N0?-N9I
	2600-2699	O00-O99	6600-6699	O0ǂ-O9Z	10600-10699	O0!-O9R	14600-14699	O0?-O9I
	2700-2799	P00-P99	6700-6799	P0ǂ-P9Z	10700-10799	P0!-P9R	14700-14799	P0?-P9I
	2800-2899	Q00-Q99	6800-6899	Q0ǂ-Q9Z	10800-10899	Q0!-Q9R	14800-14899	Q0?-Q9I
	2900-2999	R00-R99	6900-6999	R0ǂ-R9Z	10900-10999	R0!-R9R	14900-14999	R0?-R9I
AB-Bits (12-Zone) over Hundreds Position	3000-3099	?00-?99	7000-7099	?0ǂ-?9Z	11000-11099	?0!-?9R	15000-15099	?0?-?9I
	3100-3199	A00-A99	7100-7199	A0ǂ-A9Z	11100-11199	A0!-A9R	15100-15199	A0?-A9I
	3200-3299	B00-B99	7200-7299	B0ǂ-B9Z	11200-11299	B0!-B9R	15200-15299	B0?-B9I
	3300-3399	C00-C99	7300-7399	C0ǂ-C9Z	11300-11399	C0!-C9R	15300-15399	C0?-C9I
	3400-3499	D00-D99	7400-7499	D0ǂ-D9Z	11400-11499	D0!-D9R	15400-15499	D0?-D9I
	3500-3599	E00-E99	7500-7599	E0ǂ-E9Z	11500-11599	E0!-E9R	15500-15599	E0?-E9I
	3600-3699	F00-F99	7600-7699	F0ǂ-F9Z	11600-11699	F0!-F9R	15600-15699	F0?-F9I
	3700-3799	G00-G99	7700-7799	G0ǂ-G9Z	11700-11799	G0!-G9R	15700-15799	G0?-G9I
	3800-3899	H00-H99	7800-7899	H0ǂ-H9Z	11800-11899	H0!-H9R	15800-15899	H0?-H9I
	3900-3999	I00-I99	7900-7999	I0ǂ-I9Z	11900-11999	I0!-I9R	15900-15999	I0?-I9I

Units Position:		Units Position:		Units Position:	
Address Digit	Code	Address Digit	Code	Address Digit	Code
0	ǂ	0	!	0	?
1	/	1	J	1	A
2	S	2	K	2	B
3	T	3	L	3	C
4	U	4	M	4	D
5	V	5	N	5	E
6	W	6	O	6	F
7	X	7	P	7	G
8	Y	8	Q	8	H
9	Z	9	R	9	I

Figure 29. 1400 Storage Addresses in 1400 Machine Language

The dump is printed on the unit specified in the SYSEMOUT DD statement of the JCL for the emulator job. If there is no SYSEMOUT DD statement, the DUMP command is ignored.

DUMP (without operands)
: dumps the entire emulator partition. The dump contains an EBCDIC representation of the entire emulator partition and a BCD representation of emulated 1400 storage. The format of the emulator partition dump is shown in Programmer's Guide to Debugging, GC28-6670. The format of the 1400 storage dump is shown in Figure 28.

FROM=nnnnn
: dumps emulated 1400 storage in BCD, starting at address nnnnn. nnnnn can be any decimal address in 1400 storage, but it is rounded down to an even hundred by the emulator. For example, 12370 is rounded to 12300.

  If the FROM operand is the only operand specified, storage is dumped from the specified address to the end of 1400 storage.

TO=nnnnn
: indicates where the dump of 1400 storage is to end. nnnnn must be a decimal address in 1400 storage equal to or greater than the address in the FROM=nnnnn operand. nnnnn is rounded up to an even hundred by the emulator. If, after rounding, nnnnn is less than the address in the FROM=nnnnn operand, 100 bytes are dumped starting at the FROM address.

XFROM=nnnnnn
: dumps the emulator partition in EBCDIC, starting at address nnnnnn. nnnnnn can be any hexadecimal value, but it is rounded down to an even hundred by the emulator.

XTO=nnnnnn
: indicates where the dump of the emulator partition is to end. nnnnnn must be a hexadecimal value equal to or greater than the address in the XFROM=nnnnnn operand. nnnnnn is rounded up to an even hundred by the emulator.

EOJ - End the 1400 Program

Use the EOJ command to end the 1400 program. The emulator stops executing the 1400 program, and does not test whether the 1400 program is ended normally or abnormally. If an error has occurred, you may wish to type DUMP followed by EOJ, after which no further commands will be accepted.

When several 1400 programs are being executed in a single OS job step, entering the EOJ command causes the emulator to start execution of the next 1400 program. To end the entire OS job step, you must specify the operand ALL (or A). If you do so, the job step being executed is ended and, in a multiple-step job, control is given to the next OS job step.

SET - Set 1400 Register

Use the SET command to load the IAR, AAR, or BAR with a 1400 storage address.

At least one operand must be coded with the SET command. Multiple operands may be entered if they are separated by commas, but each operand can be entered only once.

IAR=nnnnn
: sets the IAR to the decimal address nnnnn.

AAR=nnnnn
: sets the AAR to the decimal address nnnnn.

BAR=nnnnn
: sets the BAR to the decimal address nnnnn.

START - Start 1400 Program

Use the START command to start a 1400 program. The program is started at the next sequential instruction or at a specified address. If no operand is specified, the program begins at the next sequential instruction.

Use the START command to emulate the START-RESET and START keys of the 1400 console.

ADDR=nnnnn
: sets the IAR to the 1400 decimal address nnnnn, and execution begins at that address.

RESET
: emulates the START-RESET and START keys. All 1400 I/O device and processor unit indicators are reset and the program starts at the next sequential instruction.

TF - Turn Off Switches

Use the TF command to turn off emulated sense switches or the emulated inquiry status latch.

You turn sense switches on using the TN command and turn them off using the TF command. Additional information on how the sense switches interact with the 1400 program can be found in Special Features Instructions: IBM 1401 Data Processing System, IBM 1460 Data Processing System, A24-3071.

At least one operand must be coded with the TF command. Multiple operands can be entered if they are separated by commas, but each operand can be entered only once.

SENSE[=abcdefg]
turns off the specified sense switches. If SENSE is specified without any sub-operands, all switches are turned off.

To turn off one or more switches, enter the letter that identifies the switch. For example, enter SENSE=BF to turn off switches B and F.

INQUIRY
turns off the inquiry status latch. Some 1400 programs test the inquiry status latch before issuing a Read from Console instruction. If the latch is off, the 1400 program does not issue the instruction. If the 1400 program issues the instruction without checking the latch, the instruction is emulated and the message IIQ002A jobname I is issued. The TF INQUIRY command does not delete the message if you have no data to enter, nor can it be a response to a message.

TN - Turn On Switches

Use the TN command to turn on emulated sense switches or the emulated inquiry status latch.

At least one operand must be coded with the TN command. Multiple operands may be entered if they are separated by commas, but each operand can be entered only once.

SENSE[=abcdefg]
turns on the specified sense switches. One or more switches may be specified. If SENSE is specified without any suboperands, all switches are turned on.

To turn on one or more switches, enter the letter that identifies the switch. For example, enter SENSE=AB to turn on switches A and B.

INQUIRY
turns on the inquiry status latch. Some 1400 programs test the inquiry status latch before issuing a Read from Console instruction. If the latch is on, the 1400 program issues the instruction.

TYPING EMULATOR CONTROL STATEMENTS AT THE CONSOLE

Although emulator control statements are generally entered on cards, they can be typed at an operator console. To enter a control statement, the operator uses the REPLY command in the same way he uses it to enter emulator commands, specifying the control statement in the text field:

```
REPLY id'UR UNIT=PTR,NAME=PNTR01'
```

The control statement is typed using the format for punched cards, but the first character of the text field does not have to be blank. To continue a statement from one line to the next, a hyphen is typed after the last character. The hyphen can be typed after any character of the operand string. The message IIQ034A jobname CONTINUE INPUT OF COMMAND STRING is displayed after each continued line. The next character in the operand must immediately follow the single quote that defines the text field. There can be no embedded blanks.

You reply differently to errors in the UNIT operand of the DISK statement and in the UNIT operand of the TAPE statement. If you specify a module or unit number and no device is at that location, you receive a message advising you that the UNIT operand is wrong. The job is not ended.

You can enter or reassign the STACKER parameter during a 1400 job by reassigning the 1400 card reader or punch with the STACKER parameter omitted or changed.

Note: The)LC and)RC cards cannot be typed at the console.

REASSIGNING THE CARD READER: When more than one 1400 data file is to be entered from the card reader, the operator must assign the next file to be processed by:

1. Typing the UR control statement after the first file has been read, specifying the UNIT and NAME operands.

2. Typing START with the RESET operand.

This closes the first data file and readies the next one so that it can be opened when the 1400 program issues the next Read a Card instruction.

REASSIGNING OTHER I/O DEVICES: When more than one 1400 data file is to be entered from a tape or disk unit, or written on a tape or disk unit, a card punch, or a printer, the operator must assign the next file to be processed by:

1. Typing the appropriate control statement (TAPE, DISK, or UR) after the first file has been processed, specifying the UNIT and NAME operands.

2. Typing the CCTL control statement if the device is the printer and the carriage control tape image must be changed.

3. Typing START.

The emulator uses the original ddname each time it reopens a tape file after performing a Rewind and Unload operation, unless the operator reassigns the tape and changes the ddname.

CHAINING COMMANDS AND CONTROL STATEMENTS

Whenever you must type more than one command or control statement for a task, you can save time by chaining commands. Chaining commands means typing more than one command or control statement in a single response.

The plus sign is used to indicate chaining. The plus sign is entered between each command in the command chain. There can be no blanks before or after the plus sign; a blank ends the command chain. For example, to display the sense switches and then continue the 1400 program, type:

```
REPLY id,'DISPLAY SENSE+START'
```

To add the display of all tape assignments to the above, type:

```
REPLY id,'DISPLAY SENSE,TAPE+START'
```

Commands and control statements are executed in the sequence listed, except for the START command. The START command can be typed at any time, but it is not executed until everything else is done. If you type the EOJ command in the middle of a chain, the emulator ends the 1400 program and ignores the rest of the chain.

If an error is found in the command chain, all commands preceding the command in error were executed. Portions of the command in error may not be executed, depending on the command and the error. The interrelationship between the type of error and the command is discussed in Appendix G under messages IIQ012D and IIQ013D.

Control	Operand	Description
		CCTL defines the carriage-control tape of the 1400 printer.
CCTL	CHcc=nnn[/nnn]...	The channel number (cc) and the column numbers (nnn) of the carriage-tape punches to be emulated.
	[LENGTH={nnn 132}]	The line length of the tape.
		If there is no CCTL statement, a standard image is assumed: • A channel-1 carriage-tape punch in columns 4 and 70 • A channel-12 carriage-tape punch in columns 63 and 129 • A 132-character carriage tape
CHBUF	CH1=nnnnn	A channel buffer is to be set up and used for all tape units whose block size (BLKSIZE) is equal to or smaller than the size of the channel buffer.
COMMENT	comments	Information to be displayed on the operator console when the job is begun.
		DISK defines a 1400 disk device.
{DISK DK}	UNIT=cm	The channel number (c) and module number (m) of the 1400 disk.

	FROMDEV={1301
	1311
	1405}
		The unit type of the 1400 disk.

	[NAME=ddname]	The name of the DD statement that defines the System/370 device used to emulate the 1400 disk.
	[BUFID=nn]	Two-digit buffer identification. If two or more DISK statements have the same buffer identification, the buffer is shared by those devices.
	[MODE={MIXED SECTOR}]	The track format used on the 1400 disk, where MIXED indicates that track-record mode or both track-record and sector modes are used. Note: The BUFID and MODE operands are ignored when typed at the console.
		EMCTL specifies control information for the emulator.
EMCTL	[EOJAADDR=nnnnn] [EOJBADDR=nnnnn] [EOJIADDR=nnnnn]	The contents of the AAR, BAR, and IAR when the 1400 program reaches normal end-of-job.
	[SENSE[=abcdefg]]	The sense switches are to be turned on. If a switch is not specified, it is turned off.
	[OPSERV]	The operator can enter commands at any time during the emulator job. Note: The OPSERV operand is ignored when typed at the console.
LOAD	{CARD TAPE}	Type of device, either a card reader (CARD) or a tape unit (TAPE), used to load the 1400 program.
		TAPE defines a 1400 tape device.
{TAPE TP}	UNIT=cu	The channel number (c) and unit number (u) of the 1400 tape.
	[NAME=ddname]	The name of the DD statement that defines the System/370 device used to emulate the 1400 tape device.
	[ALTMODE={YES NO}]	Specifies whether the tape is in alternate mode (YES) or normal mode (NO).
	[TYPEFLE={OUTPUT INOUT INPUT}]	Specifies if the tape is an input tape (INPUT), an output tape (OUTPUT), or an input tape that is later to be used as an output tape (INOUT).

	DILCNT={nnn
	25}
		The number of 1400 instructions to be executed during the overlap interval.

		UR defines a 1400 unit record device.

UR	UNIT={CNSL
	PNCH
	PTR
	RDR}
		The unit to be emulated is a console (CNSL), a card punch (PNCH), a printer (PTR), or a card reader (RDR).

	[NAME=ddname]	The name of the DD statement that defines the System/370 device used to emulate the 1400 device. Do not specify this operand if UNIT=CNSL.

	STACKER={YES
	1
	2
	3}
		Specifies if the 1400 program stacks the cards read or punched (YES), or if the cards are to be placed in stacker 1, 2, or 3.

	[DILCNT={nnn 25}]	The number of 1400 instructions to be executed during the overlap interval.
USER	NAME=name	The load module name of a routine you have written.

	OPCODE={c
	IO
	AR}
		The operation code to be emulated. c specifies a BCD operation code. IO must be coded for all I/O operation codes. AR specifies automatic reply.

	[CONTROL={nn 00}]	The control byte to be used by the compatibility feature.
)LC		End of 1400 data. This card is inserted just before the last card of 1400 data in the input stream.
)RC		When several 1400 programs are being executed in a single OS job step, this card is inserted before every set of emulator control cards except the first.

Figure 30. Format of Emulator Control Statements

Command	Operand	Description
{ALTER|A}	{ADDR|A} =nnnnn, {DATA|D} =data[,MODE={M|L}]	Stores BCD characters (DATA) in move (M) or load (L) mode at the 1400 storage address (ADDR).
	XADDR=nnnnnn,XDATA=data	Stores hexadecimal data (XDATA) at the hexadecimal address (XADDR).
{CLEAR|CL}	CORE={n|_n}	Sets all of 1400 storage to an alphameric character (n) or an alphameric character with a wordmark (_n).
CONVERT	{ADDR|A} =nnnnn	Converts a 1400 storage address to its corresponding System/370 hexadecimal address.
{DISPLAY|D}	SENSE	Displays the status of the sense switches.
	{INQUIRY|I}	Displays the status of the Inquiry Status latch.
	{REG|R}	Displays the contents of the IAR, AAR, and BAR.
	{STATUS|S}	Displays all of the above operands.
	{TAPE|TP}[=cu]	Displays both the ddname of the System/370 device assigned to emulate the 1400 device and the System/370 device address; if only TAPE is entered, all tape assignments are displayed.
	{DISK|DK}[=cm]	Displays the ddname of the System/370 device assigned to emulate the 1400 device; if only DISK is entered, all disk assignments are displayed.
	UR	Displays the assignments of all of the 1400 unit record devices.
	CONFIG	Displays the assignments of all devices used by the 1400 program.
	{ADDR|A} =nnnnn	Displays 40 positions of emulated 1400 storage starting at the address specified.
	XADDR=nnnnnn	Displays 20 bytes of emulator partition starting at the address specified.
DUMP	no operands	Dumps the entire emulator partition (in EBCDIC) and 1400 storage (in BCD).
	FROM=nnnnn[,TO=nnnnn]	Dumps 1400 storage from the FROM address to the TO address (in BCD), or if the TO operand is omitted, dumps from the FROM address to the end of emulated 1400 storage.
	XFROM=nnnnnn,XTO=nnnnnn	Dumps the emulator partition from the XFROM address to the XTO address (in EBCDIC).
EOJ	no operands	Ends the 1400 job.
	{ALL|A}	Ends the OS job step when several 1400 programs are being executed in a single job step.

{SET|T}	IAR=nnnnn
	AAR=nnnnn
	BAR=nnnnn
		Loads the specified address into the IAR, AAR, or BAR.

{START|S}	no operands	Starts a 1400 program at the next sequential instruction.
	{ADDR|A} =nnnnn	Starts at the address specified.
	{RESET|E}	Emulates the Start-Reset and Start keys.
{TN|TF}	SENSE[=abcdefg]	Turns on (TN) or off (TF) the sense switches specified, or all switches if only SENSE is specified.
	{INQUIRY|I}	Turns on or off the Inquiry Status latch.

Figure 31. Format of Emulator Commands

Because 1400 programs and data are in BCD, the emulator accepts BCD characters, but it translates them into codes that are easier to use with System/370. The sequence of BCD characters before translation is CWBA8421, where C represents the check bit and W the wordmark bit. After being translated into internal code, sequence is 8421BAWC, where bit C is always zero. Figure 32 shows this translation as well as translations to other codes used by the emulator; all are for BCD characters without wordmarks. For BCD characters with wordmarks, bit six of the internal code character is 1. For example, an A with a wordmark is 1E in internal code.

Collating Sequence	1400 Card Code	BCD Code	EBCDIC or Alternate Mode or Even-parity Normal Mode (Bit 1=1)	Odd-parity Normal Mode (Bit 1=0)	Internal Code (Bit 7=0)	1400 Graphic	1400 Character
00		C	40	00	00		Blank
01	12-3-8	BA821	4B	0B	BC	.	Period
02	12-4-8	CBA84	4C	0C	CC	⌑)	Lozenge or Right Parenthesis
03	12-5-8	BA841	4D	0D	DC	[	Left Bracket
04	12-6-8	BA842	4E	0E	EC	<	Less Than
05	12-7-8	CBA8421	4F	0F	FC	≢	Groupmark
06	12	CBA	50	10	0C	& +	Ampersand or Plus
07	11-3-8	CB821	5B	1B	B8	$	Dollar Sign
08	11-4-8	B84	5C	1C	C8	*	Asterisk
09	11-5-8	CB841	5D	1D	D8	]	Right Bracket
10	11-6-8	CB842	5E	1E	E8	;	Semicolon
11	11-7-8	B8421	5F	1F	F8	Δ	Delta
12	11	B	60	20	08	-	Minus
13	0-1	CA1	61	21	14	/	Slash
14	0-3-8	CA821	6B	2B	B4	,	Comma
15	0-4-8	A84	6C	2C	C4	% (	Percent or Left Parenthesis
16	0-5-8	CA841	6D	2D	D4	⌒	Word Separator
17	0-6-8	CA842	6E	2E	E4	\	Backslash
18	0-7-8	A8421	6F	2F	F4	⧻	Segment Mark
19	2-8	A	7A	3A	04	ƀ	Substitute Blank
20	3-8	821	7B	3B	B0	# =	Number or Equal Sign
21	4-8	C84	7C	3C	C0	@ '	At Sign or Single Quote
22	5-8	841	7D	3D	D0	:	Colon
23	6-8	842	7E	3E	E0	>	Greater Than
24	7-8	C8421	7F	3F	F0	√	Tapemark
25	12-0	CBA82	C0	80	AC	?	Plus Zero
26	12-1	BA1	C1	81	1C	A	A
27	12-2	BA2	C2	82	2C	B	B
28	12-3	CBA21	C3	83	3C	C	C
29	12-4	BA4	C4	84	4C	D	D
30	12-5	CBA41	C5	85	5C	E	E
31	12-6	CBA42	C6	86	6C	F	F

Figure 32. Character Code Correspondence (Part 1 of 3)

Collating Sequence	1400 Card Code	BCD Code	EBCDIC or Alternate Mode or Even-parity Normal Mode (Bit 1=1)	Odd-parity Normal Mode (Bit 1=0)	Internal Code (Bit 7=0)	1400 Graphic	1400 Character
32	12-7	BA421	C7	87	7C	G	G
33	12-8	BA8	C8	88	8C	H	H
34	12-9	CBA81	C9	89	9C	I	I
35	11-0	B82	D0	90	A8	!	Minus Zero
36	11-1	CB1	D1	91	18	J	J
37	11-2	CB2	D2	92	28	K	K
38	11-3	B21	D3	93	38	L	L
39	11-4	CB4	D4	94	48	M	M
40	11-5	B41	D5	95	58	N	N
41	11-6	B42	D6	96	68	O	O
42	11-7	BC421	D7	97	78	P	P
43	11-8	CB8	D8	98	88	Q	Q
44	11-9	B81	D9	99	98	R	R
45	0-2-8	A82	E0	A0	A4	‡	Record Mark
46	0-2	CA2	E2	A2	24	S	S
47	0-3	A21	E3	A3	34	T	T
48	0-4	CA4	E4	A4	44	U	U
49	0-5	A41	E5	A5	54	V	V
50	0-6	A42	E6	A6	64	W	W
51	0-7	CA421	E7	A7	74	X	X
52	0-8	CA8	E8	A8	84	Y	Y
53	0-9	A81	E9	A9	94	Z	Z
54	0	C82	F0	B0	A0	0	0
55	1	1	F1	B1	10	1	1
56	2	2	F2	B2	20	2	2
57	3	C21	F3	B3	30	3	3
58	4	4	F4	B4	40	4	4
59	5	C41	F5	B5	50	5	5
60	6	C42	F6	B6	60	6	6
61	7	421	F7	B7	70	7	7
62	8	8	F8	B8	80	8	8
63	9	C81	F9	B9	90	9	9

Figure 32. Character Code Correspondence (Part 2 of 3)

The following MLP (Multiple Line Punch) codes are accepted when reading from cards only. Cards are read in even-parity normal mode, but bit 1 is 0. To make the MLP code valid, the 8-9 punch is stripped off the 1400 card code and the remainder is translated to internal code. The 1400 graphic corresponds to internal code, not to 1400 card code.

Note: MLP codes 12-8-9, 11-8-9, 0-8-9, and 8-9 produce 00 in internal code.

Collating Sequence	1400 Card Code	BCD Code	EBCDIC or Alternate Mode or Even-parity Normal Mode (Bit 1=1)	Odd-parity Normal Mode (Bit 1=0)	Internal Code (Bit 7=0)	1400 Graphic	1400 Character
00	12-8-9		08		00		
01	12-1-8-9		09		1C	A	
02	12-2-8-9		0A		2C	B	
03	12-3-8-9		0B		3C	C	
04	12-4-8-9		0C		4C	D	
05	12-5-8-9		0D		5C	E	
06	12-6-8-9		0E		6C	F	
07	12-7-8-9		0F		7C	G	
08	11-8-9		18		00		
09	11-1-8-9		19		18	J	
10	11-2-8-9		1A		28	K	
11	11-3-8-9		1B		38	L	
12	11-4-8-9		1C		48	M	
13	11-5-8-9		1D		58	N	
14	11-6-8-9		1E		68	O	
15	11-7-8-9		1F		78	P	
16	0-8-9		28		00		
17	0-1-8-9		29		14	/	
18	0-2-8-9		2A		24	S	
19	0-3-8-9		2B		34	T	
20	0-4-8-9		2C		44	U	
21	0-5-8-9		2D		54	V	
22	0-6-8-9		2E		64	W	
23	0-7-8-9		2F		74	X	
24	8-9		38		00		
25	1-8-9		39		10	1	
26	2-8-9		3A		20	2	
27	3-8-9		3B		30	3	
28	4-8-9		3C		40	4	
29	5-8-9		3D		50	5	
30	6-8-9		3E		60	6	
31	7-8-9		3F		70	7	

Figure 32. Character Code Correspondence (Part 3 of 3)

APPENDIX B. CONVENTIONS AND SYNTAX USED TO DESCRIBE EMULATOR CONTROL CARDS AND COMMANDS

This appendix presents the conventions used to describe emulator control cards and operator commands. It also gives the rules of syntax that must be followed when preparing emulator control cards or entering commands.

CONVENTIONS

Lower case letters in the description of a control card or operator command represent variables for which specified information is substituted in the actual statement. For numeric operands, the number of letters equals the maximum number of digits that may be written (except for the DUMP and ALTER commands).

Upper case letters, numbers, and the symbols = / ' _ and , represent themselves.

The characters shown below never appear in control cards and commands.

- Braces { } indicate alternative items.
- Brackets [ ] indicate optional items.
- Elipsis ... indicates that the preceding item may be repeated.

SYNTAX

The identification field in an emulator control card or command is a string from one to eight alphanumeric characters long.

The keyword is a string from one to eight alphanumeric characters long.

The operand is either a character, an alphanumeric character string, a 1400 decimal address, a System/370 hexadecimal address, 1400 data, or hexadecimal data.

The following syntactic rules must be observed:

- Control card information may begin anywhere in a card; commands cannot begin with a blank.
- The order in which a sequence of keywords and operands is specified does not matter. Each keyword plus operand must be separated from the following one by a comma.
- Leading zeros in both 1400 and hexadecimal addresses are ignored and need not be entered.
- You can use lower or upper case letters for commands, except in 1400 data, which must be entered according to its meaning.
- Numbers and the symbols = / ' _ and , must be coded as they appear.
- The first blank after the first keyword ends the command or control card statement, except when the blank is part of 1400 data, or when a continuation character indicates the statement is not complete, or when no keyword is specified.
- You can use only one identification field per card or line.
- Hexadecimal addresses and data are defined and recognized by the keywords XADDR and XDATA, while 1400 decimal addresses and 1400 data are defined and recognized by the keywords ADDR and DATA.
- You can punch an emulator control statement on several cards. To do so, place any character in column 72, and continue the control statement on the next card starting anywhere.
- You can type an emulator command on several lines. To do so, type the character hyphen (-) as continuation character and continue the command on the next line.
- A keyword or operand can span two cards or two lines.
- The maximum length of a command is 485 characters.

APPENDIX C: SAMPLE PROGRAM

A sample program is provided so that the emulator can be tested after it has been generated. The sample program is a 1400 object program, in the EMUL.EMSAMP data set. After Stage II of emulator generation but before the Emulator Distribution Library has been removed, the sample program should be either executed to see if any errors can be detected or punched on cards to be executed later. The program tests emulation of the 1400 card reader, card punch, and printer.

PUNCHING THE SAMPLE PROGRAM

The JCL statements for punching the sample program are shown in Figure 33. Underlined values represent variables; the others must be coded as shown. The EMUL.EMSAMP data set must be on the unit specified in the UNIT parameter of the SYSUT1 DD statement. The JCL statements should be modified when:

- The program name of the emulator to be tested is other than the default name IIQE14.
- The emulator needs an MVT region larger than 61,440 bytes (60K), or an MFT partition other than that assigned to the default class.

EXECUTING THE SAMPLE PROGRAM

The output of the IEBPTPCH utility program contains a MEMBER card and the JCL statements needed to execute the sample program, as shown in Figure 34. The MEMBER card must be removed before the job is put into the input stream.

If the job is executed using the MFT control program, the job is assigned to the default class. The EXEC statement instructs the operating system to execute program IIQE14. This is the default name

```
//PNCHSAMP  JOB    MSGLEVEL=(1,1)
//PNCH      EXEC   PGM=IEBPTPCH
//SYSPRINT  DD     SYSOUT=A
//SYSUT1    DD     DSNAME=EMUL.EMSAMP,UNIT=2311,                       X
//                 VOL=SER=EMDLIB,DISP=OLD
//SYSUT2    DD     SYSOUT=B
//SYSIN     DD     *
            PUNCH  TYPORG=PO
/*
```

Figure 33. JCL Statements for Punching the Sample Program

```
MEMBER NAME SPGM1
//EMUL01H   JOB    MSGLEVEL=(1,1),REGION=60K
//GO        EXEC   PGM=IIQE14
//SYSUDUMP  DD     SYSOUT=A
//SYSEMOUT  DD     SYSOUT=A
//PRINTER1  DD     SYSOUT=A
//PUNCH01   DD     SYSOUT=B
//SYSEMCTL  DD     *
            UR     UNIT=RDR,NAME=SYSEMCTL
            UR     UNIT=PTR,NAME=PRINTER1
            UR     UNIT=PNCH,NAME=PUNCH01
            ALTER  ADDR=902,DATA=1
            EMCTL  EOJAADDR=999
            LOAD   CARD
The 1400 program goes here.  There are no data cards.
```

Note: Underlined values represent variables; the remaining values must be coded as shown. The ALTER command is required to emulate 1401 and 1460 systems. Remove the ALTER command if the system is a 1440.

Figure 34. JCL Statements for Executing the Sample Program

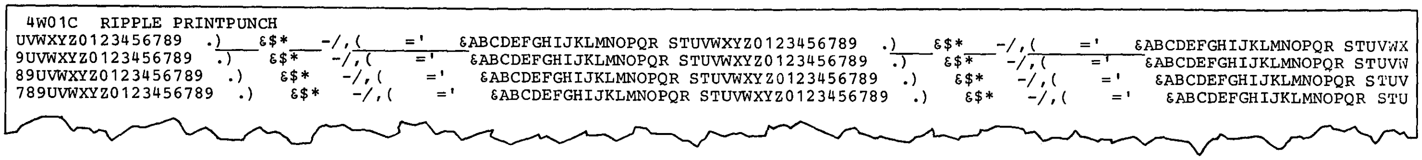
assigned to the emulator at emulator generation, and it must be changed if you have selected another name.

COMPARING THE RESULTS

Figure 35 shows the first five lines of printer output from the sample program. Each line contains 132 BCD characters, printed using an HN print chain. Underlined characters in the first line are substitutes for the BCD characters. (Figure 8 in the section "Emulator-1400 Differences" shows the differences in printer graphics.)

The sample program tests the card punching routines by punching the first 80 characters in each line of printer output. Comparing the output of the printer and card punch can help you check for errors.

Figure 36 shows the punches in the first card of punched output. The character at the top of each column is the BCD character that the punch represents.



```
 4W01C  RIPPLE PRINTPUNCH
UVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUVWX
9UVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUVW
89UVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUV
789UVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STUVWXYZ0123456789  .)    &$*    -/,(     ='    &ABCDEFGHIJKLMNOPQR STU
```

Figure 35. Printed Output of the Sample Program

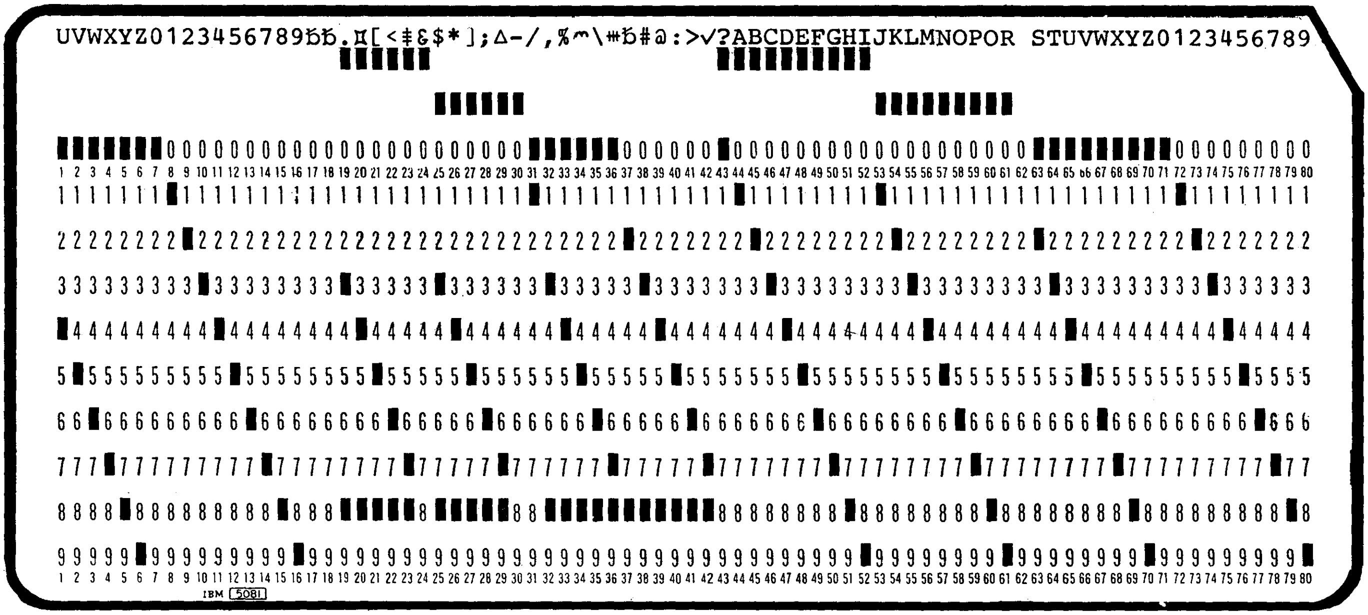

UVWXYZ0123456789bb.¤[<‡&$*];Δ-/,%^\⧣b#@:>√?ABCDEFGHIJKLMNOPOR STUVWXYZ0123456789
IBM 5081

Figure 36. Punched Output of the Sample Program (First Card Only)

APPENDIX D: TAPE FORMATTING PROGRAMS

Because of differences in record formats and data representation of 1400 tapes, it is often advantageous to reformat data on tape before executing a 1400 program on System/370. The tape preprocessor program and the tape postprocessor program run as problem programs under control of OS MFT and MVT. The programs are started using JCL statements.

The preprocessor program converts data files in 1400 format on seven-track or nine-track tape to data sets in spanned format. Data sets in spanned format can be on seven-track or nine-track tape or on direct access storage devices. The preprocessor reads physical records from 1 to 200,000 bytes long and writes physical records up to 32,755 bytes lcng.

The postprocessor program converts data in spanned format written by the emulator to data files in 1400 format on seven-track or nine-track tape. The postprocessor reads physical records up to 32,755 bytes long and writes physical records from 1 to 200,000 bytes long.

The programs process tapes with densities of 200, 556, 800, and 1600 bits per inch (bpi) and with mixed densities. The programs also process tapes with data in even, odd, and mixed parity. Tapes with data in even parity and odd parity are processed by the programs themselves, while tapes with data in mixed parity are processed partly by an error recovery procedure (ERP) of the operating system.

MINIMUM SYSTEM REQUIRED

You need:

- 4,096 bytes of main storage for the tape preprocessor program and 5,226 bytes for the tape postprocessor program (does not include buffers)
- One input and one output buffer large enough to hold the longest record to be formatted
- One system input device for JCL and control statements
- One system output device for a summary listing and error messages
- One system console
- At least two I/O devices to read and write data

Seven-track and nine-track tapes may be used as input or output for preprocessing and postprocessing jobs. The Data Converter must be installed on the control unit of a seven-track tape in order to use seven-track tape for preprocessor output or postprocessor input. Output from preprocessor and input to the postprocessor can be on direct access devices.

BUFFER STORAGE

The input buffer for a preprocessor job and the output buffer for a postprocessor job must be large enough to hold the longest 1400 record in the file. If the size of the longest record is unknown, select a buffer size you think sufficient and run the preprocessor. The output data set contains an end-of-volume summary for every tape volume processed. The summary gives the size of the longest record formatted, and indicates whether there were any records too long for the buffer. You may use this information to establish the correct buffer size for subsequent formatting.

Because output from a preprocessing job and input to a postprocessing job are in spanned format, each physical record includes some control information (see "Processing 1400 Labels" in this appendix and "Spanned Format" in the section "Processing Data"). You must include space for this control information when computing the buffer size. Buffer space is increased by 4+4n bytes, where n is equal to the number of input records or segments of input records in an output block. For example, if the input block size is 80 bytes per record and you want to block your data at five records per block, the output block size should be 424 or (80•5)+(4+(4•5)). When specifying buffer space:

- The emulator input block size should equal the preprocessor output block size.
- The postprocessor input block size should be equal the emulator output block size.

TAPE CORRESPONDENCE

In order to keep the original 1400 labels when converting tapes to spanned format, tapes are converted as one-reel reel files. If you cannot convert one input reel to one output reel:

- Use a larger output tape.
- Specify a higher output density.
- Specify a larger buffer for spanned records (preprocessor).
- Re-create the input tape, using a smaller buffer for spanned records (postprocessor).
- Replace a worn output tape with one that is in better condition.
- Create a shorter input file.

If none of the above are appropriate, two output tapes may be used, but the emulator cannot backspace from the second tape back to the first.

Note: Tapes in 1400 format having records longer than 32,755 bytes must be formatted before being used by the emulator.

PROCESSING 1400 TAPEMARKS

OS cannot distinguish System/370 tapemarks from 1400 tapemarks. Consequently, both types of tapemarks cannot be written on the same tape. On 1400-format tapes, any tapemark encountered is assumed to be a 1400 tapemark. On System/370 tapes (spanned format), any tapemark encountered is assumed to be written by the operating system. To save 1400 tapemarks, the preprocessor rewrites them as "tapemark records." These tapemark records define the relative position of the 1400 tapemark.

The format of the tapemark record is shown in Figure 37. After the interrecord gap (IRG), there is a block descriptor word (BDW) of 4 bytes and a segment descriptor word (SDW) of 4 bytes. These are followed by 12 bytes of data describing the tapemark. The identification field of the tapemark record will always contain hexadecimal FF000000. The tapemark trace field contains the relative record number (block number) of the preceding tapemark record on a tape volume. If this is the first tapemark record, the field is set to 1. If the data set is being written on a direct access device, the tapemark trace field contains the relative track and record number of the preceding tapemark record.

PROCESSING 1400 LABELS

To avoid having more than one label on a tape, 1400 header and trailer labels are written as by the preprocessor data records; that is, each header or trailer label will have its own BDW and SDW in spanned format. This permits the operating system to recognize System/370 labels, but only the emulator and the tape formatting programs can recognize the data record as a label. Figure 38 shows how the 1400 tape is modified by the preprocessor. The postprocessor removes the BDW and SDW when returning the tape to its original format.

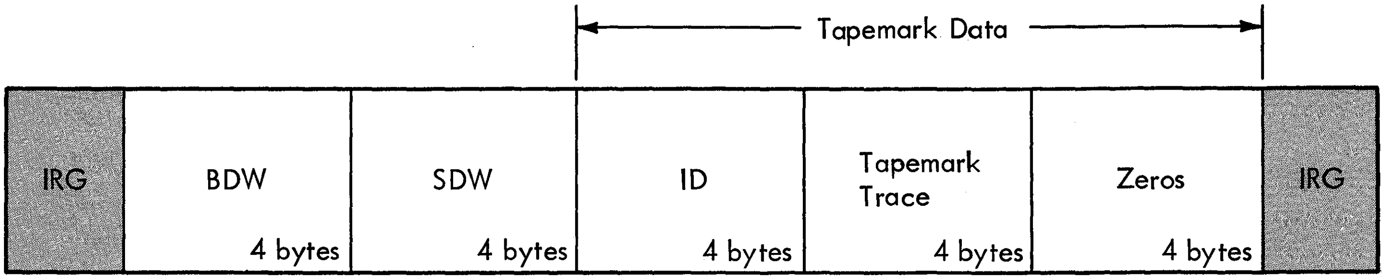


Figure 37. Format of the Tapemark Record

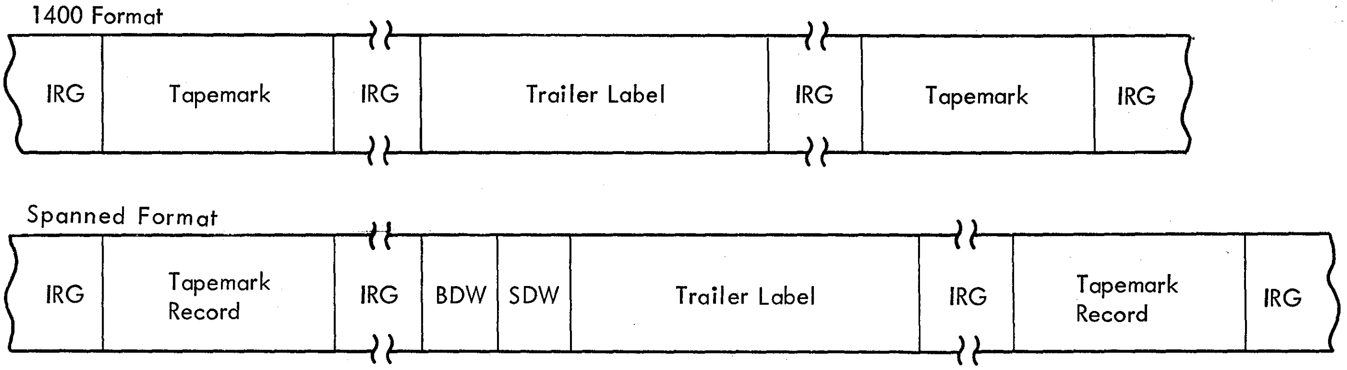


Figure 38. Modifying 1400 Header and Trailer Labels

CHARACTER CODE CORRESPONDENCE

Figures 39 and 40 show the correspondence between input and output character codes for both tape formatting programs. Even-parity normal mode and alternate mode characters in spanned format are the same as FBCDIC characters. Seven-track tapes are accepted and produced at 200, 556, or 800 bpi and nine-track tapes are accepted and produced at 800 and 1600 bpi.

TAPE PREPROCESSOR PROGRAM

The tape preprocessor program converts data sets in 1400 format on seven-track or nine-track tape to data sets in spanned format. It reads and reformats any 1400 record up to 200,000 bytes long, so that the 1400 file can be read by the emulator.

You should know the exact number of tapemarks on the reel of tape to be formatted. Although tapes can be formatted without specifying the number of tapemarks, the preprocessor operates more efficiently when it is known. The tapemark information cannot be supplied using JCL statements. The console operator will supply it when responding to the WTOR message IIQ301D.

After receiving the operator response, the preprocessor formats one reel of tape, and stops after having read the number of tapemarks specified by the operator. If the operator has specified ALL, the preprocessor formats all readable data on the tape, and stops when the tape comes off the reel or when an I/O error is encountered. After the data has been formatted, the preprocessor issues an end-of-volume summary (message IIQ305I), closes both input and output data sets, and issues message IIQ301D again. If there are other files to be formatted, the operator should mount the appropriate files and respond to message IIQ301D by specifying the number of tapemarks on the next file.

INPUT (1400 Format)		OUTPUT (Spanned Format)
7-Track Tapes (Data Translator On - BCD)	9-Track Tapes	Tape Data Sets on Disk, or 9-Track Tapes, or 7-Track Tapes (Data Converter On - EBCDIC)
Even Parity	Normal Mode, Even Parity	Normal Mode, Even Parity
Odd Parity	Normal Mode, Odd Parity	Normal Mode, Odd Parity
Mixed Parity	Normal Mode, Even and Odd Parities Interspersed	Normal Mode, Even and Odd Parities Interspersed
	Alternate Mode	Alternate Mode (Appears to be even parity)

Figure 39. Data Formats Used with the Preprocessor

JCL STATEMENTS

JCL statements are used to start the tape preprocessor program and to define the input and output data sets. The statements for executing the program will vary depending on where the program is catalogued, what devices are used for the input and output data sets, and the input and output data sets themselves.

The JCL statements used with the program are shown in Figure 41. (See *Job Control Language Reference*, GC28-6704 for more information on JCL statements, and *Tape Labels*, GC28-6680 for detailed information on the LABEL parameter of the DD statement.)

Defining Input Data

Parameters that define input data are shown in Figure 42. The block size of the input data is the only attribute not defined in the SYSUT1 DD statement. The length of the longest input record (block size) for the preprocessor should be specified in the PARM field of the EXEC statement. More than one input volume can be formatted. After the first volume is formatted, the operator is asked to remove the formatted volume and mount an unformatted one.

INPUT (Spanned Format)	OUTPUT (1400 Format)	
Tape Data Sets on Disk, or 9-Track Tapes, or 7-Track Tapes (Data Converter On - EBCDIC)	9-Track Tapes, or 7-Track Tapes (Data Converter On - EBCDIC)	7-Track Tapes (Data Translator On - BCD)
Normal Mode, Even Parity	Normal Mode, Even Parity	Even Parity
Normal Mode, Odd Parity	Normal Mode, Odd Parity	Odd Parity
Normal Mode, Even and Odd Parities Interspersed	Normal Mode, Even and Odd Parities Interspersed	Mixed Parity
Alternate Mode	Alternate Mode	Even Parity

Figure 40. Data Formats Used With the Postprocessor

Statement	Use
JOB	Starts the job.
EXEC	Specifies the program name (PGM=IIQPRE)and the block size of the input data set (PARM=blocksize).
SYSPRINT DD	Defines a sequential message data set containing system messages, error messages, and volume summaries.
SYSUT1 DD	Defines a sequential data set on a magnetic tape to be used as input to the preprocessor.
SYSUT2 DD	Defines a sequential data set on a magnetic tape or a direct access device to receive the output from the preprocessor.

Figure 41. JCL Statements for the Preprocessor

Data Attribute	7-Track Tape	9-Track Tape
Status/Disposition	Not Required	Not Required
Device	UNIT=2400-2	UNIT=2400
Label	LABEL=(,NL)	LABEL=(,NL)
Volume Identification	VOL=SER=number	VOL=SER=number
Density[1]	$(DEN=\left\{\begin{matrix}0\\1\\2\end{matrix}\right\})$[2]	Default to 800 bpi
Block Size[3]	PARM=blocksize	PARM=blocksize
Parity	Not Required[4]	Not Required

[1] This attribute is a subparameter of the DCB parameter of the SYSUT1 DD statement.
[2] 0=200 bpi, 1=556 bpi, 2=800 bpi.
[3] The maximum input block size is specified by the PARM keyword of the EXEC statement. The other parameters are specified in the SYSUT1 DD statement. The default value for PARM is 10,000 bytes; the maximum is 200,000 bytes. The PARM keyword is used so that values greater than 32,765 bytes can be specified.
[4] Tapes can be read in even, odd, or mixed parity.

Figure 42. Defining Input Data for the Preprocessor

Defining Output Data

All attributes of the output data are defined in the SYSUT2 DD statement. At least one output volume should be allocated to each input volume. If the data from one input volume might overflow a single output volume, two output volumes should be specified. Figure 43 shows the parameters of the SYSUT2 DD statement that define the output data set for the preprocessor.

The data can be sent to any tape or disk device accepted by BSAM. If using a disk device, the Record Overflow feature should not be used because the emulator cannot backspace over a record that is split between two cylinders.

TAPE POSTPROCESSOR PROGRAM

The tape postprocessor program converts data sets in spanned format to data files in 1400 format. The data sets in spanned format can be on seven-track or nine-track tape or on a direct access device. The data files in 1400 format must be on a seven-track or nine-track tape.

The postprocessor writes records from 1 to 200,000 bytes long. When formatted, the tape may be used on a 1400 system, by a stand-alone emulator, or by Compatibility Support/30 or Compatibility Support/40 programs. Input to the postprocessor must be in spanned format. If you want 1400 tape marks on the output tape, there must be a tapemark record wherever a 1400 tapemark is desired. All 1400 labels must be data records; that is, they must have a BDW and an SDW. All data sets produced by the emulator in spanned format and all data sets produced by the preprocessor have this format.

You should know when an input volume will need more than one output volume so that additional information can be provided for the postprocessor. The information is provided by program control statements submitted with the JCL statements. Program control statements define end-of-tape (tapemarks and trailer label) on the first output volume and beginning-of-tape (tapemarks and header label) for the second output volume. If the postprocessor cannot write all the data on one volume and you do not provide the appropriate program control statements, it writes two tapemarks on the first volume and starts the second volume without writing a tapemark or label. (For more information on multiple output volumes, see "Defining Output Data.")

You should know whether mixed-density output is needed. The postprocessor can write tapes in mixed densities when you include the DENSITY program control statement with the JCL statements for the postprocessor job.

Program control statements are discussed further under "Defining Output Data" later in this section.

Data Attribute	7-Track Tape	9-Track Tape	Disk
Status/Disposition	DISP=(NEW,KEEP)[1]	DISP=(NEW,KEEP)[1]	DISP=(NEW,KEEP)[1]
Device	UNIT=2400-2	UNIT={2400 2400-3 2400-4}[2]	UNIT={2311 2314 3330}[3]
Label	LABEL=(,{SL NL})	LABEL=(,{SL NL})	Not Applicable
Volume Identification	VOL=SER=(number,...)[4]	VOL=SER=(number,...)[4]	VOL=SER=(number,...)
Density[5]	(DEN={0 1 2})[6]	(DEN=3) or Default to 800 bpi[7]	Not Applicable[8]
Block Size[5]	(BLKSIZE={blocksize 3500})	(BLKSIZE={blocksize 3500})	(BLKSIZE={blocksize 3500})
Parity[5]	(TRTCH=C)	Default to Odd	Default to Odd

Space on DASD	Not Applicable	Not Applicable	SPACE=dasd space
			SPLIT=dasd space
			SUBALLOC=dasd space

Data Set Name	DSNAME=dsname	DSNAME=dsname	DSNAME=dsname

Record Format	RECFM={VS
	VBS}
		RECFM={VS
		VBS}
			RECFM={VS
			VBS}

[1] Specify when LABEL=(,SL).

[2] 2400 specifies a nine-track, 800 bpi tape drive; 2400-3 specifies a nine-track, 1600 bpi tape drive; 2400-4 specifies a nine-track, 800 or 1600 bpi tape drive. Any nine-track tape unit accepted by BSAM may be specified.

[3] May be any disk unit accepted by BDAM.

[4] If an input volume requires more than one output volume, two output volume serial numbers must be specified.

[5] This attribute is a subparameter of the DCB parameter of the SYSUT2 DD statement.

[6] 0=200 bpi, 1=556 bpi, 2=800 bpi.

[7] If output density is 1600 bpi, specify the parameter DEN=3; otherwise let DEN default to 800 bpi.

[8] Density is not specified for disk, but SPACE must be indicated: SPACE=(units,(quantities,increments))

Figure 43. Defining Output Data for the Preprocessor

JCL STATEMENTS

JCL statements are used to start the tape postprocessor program and to define the input and output data sets. The statements for executing the program vary, depending on where the program is catalogued, what devices are used for the input and the output data sets, and the input and output data sets themselves.

The JCL statements used with the postprocessor are shown in Figure 44. (See *Job Control Language Reference*, GC28-6704 for more information on JCL statements and *Tape Labels*, GC28-6680 for detailed information on the LABEL parameter of the DD statement.)

Defining Input Data

All attributes of postprocessor input data are defined in the SYSUT1 DD statement. Input to the postprocessor is in spanned format. Figure 45 shows the parameters of the SYSUT1 DD statement that define the input data set for the postprocessor.

Statement	Usage
JOB	Starts the job.
EXEC	Specifies the program name (PGM=IIQPOS) and the block size for the output data set (PARM=blocksize).
SYSPRINT DD	Defines a sequential message data set containing system messages, error messages, and volume summaries.
SYSUT1 DD	Defines a sequential data set on a magnetic tape or a direct access device to be used as input to the postprocessor.
SYSUT2 DD	Defines a sequential data set on a magnetic tape to receive the output from the postprocessor.
SYSIN DD	Provides control data for the postprocessor job. The control data set is in the input stream and contains the program control statements needed to create header and trailer labels when a single input volume (SYSUT1) needs more than one output volume (SYSUT2). Note: Postprocessor jobs with no program control statements must specify SYSIN DD DUMMY, with the DCB parameter specifying a blocksize.

Figure 44. JCL Statements for the Postprocessor

Defining Output Data

Except under specific conditions, the block size of the output data is the only attribute not defined in the SYSUT2 DD statement. These conditions are explained below. The physical record length (block size) of the output data file for a postprocessor job should be specified in the PARM keyword parameter of the EXEC statement. The PARM keyword is used so that values greater than 32K can be specified. The JCL parameters that define output data for the postprocessor are shown in Figure 46.

SINGLE-VOLUME INPUT WITH MULTIPLE-VOLUME OUTPUT: If a full or nearly full emulator output volume is being postprocessed, the postprocessor output may need two tape volumes. In this case, the postprocessor creates a trailer label for the first output volume and a header label for the second output volume. The TLABEL and HLABEL program control statements specify the labels and tapemarks to be created by the postprocessor.

If you do not provide these control statements, two tapemarks are written when the end-of-volume reflective strip is sensed on the first output volume, and the output file is continued on the next volume without a header label.

The TLABEL program control statement defines the series of records or tapemarks, or both, to be written when the end-of-volume reflective strip is sensed on the first of the two output volumes. The HLABEL program control statement defines the series of records or tapemarks, or both, to be written when a second output volume is mounted while postprocessing one input volume. The formats of the HLABEL and TLABEL statements are described under "Program Control Statements."

MIXED-DENSITY TAPES: The operating system does not write tapes in more than one density. If a mixed-density tape must be prepared for a 1400 system, the postprocessor writes the first part of the tape in one density and then modifies the DCB and writes the rest of the tape in another density. The DENSITY program control statement defines how many records are to be written in the first density. The format of the DENSITY statement is described under "Program Control Statements."

PROGRAM CONTROL STATEMENTS

Normally, JCL statements provide the required information for output data. Additional information must be supplied to the postprocessor when:

- A postprocessor input tape will, when reformatted, use two output tapes.

Data Attribute	7-Track Tape	9-Track Tape	Disk
Status/Disposition	DISP=(OLD,KEEP)[1]	DISP=(OLD,KEEP)[1]	DISP=(OLD,KEEP)[1]
Device	UNIT=2400-2	UNIT=$\left\{\begin{matrix}\text{2400}\\ \text{2400-3}\\ \text{2400-4}\end{matrix}\right\}$[2]	UNIT=$\left\{\begin{matrix}\text{2311}\\ \text{2314}\\ \text{3330}\end{matrix}\right\}$[3]
Label	LABEL=(,$\left\{\begin{matrix}\underline{\text{SL}}\\ \text{NL}\end{matrix}\right\}$)	LABEL=(,$\left\{\begin{matrix}\underline{\text{SL}}\\ \text{NL}\end{matrix}\right\}$)	Not Applicable
Volume Identification	VOL=SER=(number,...)	VOL=SER=(number,...)	VOL=SER=(number,...)
Density[4]	(DEN=$\left\{\begin{matrix}0\\ 1\\ 2\end{matrix}\right\}$)[5]	(DEN=3) or Default to 800 bpi[6]	Not Applicable
Block Size[4]	(BLKSIZE=blocksize)	(BLKSIZE=blocksize)	(BLKSIZE=blocksize)
Parity[4]	(TRTCH=C)	Default to Odd	Default to Odd
Data Set Name	DSNAME=dsname	DSNAME=dsname	DSNAME=dsname

[1] Specify when LABEL=(,SL).

[2] 2400 specifies a nine-track, 800 bpi tape drive; 2400-3 specifies a nine-track, 1600 bpi tape drive; 2400-4 specifies a nine-track, 800 or 1600 bpi tape drive. Any nine-track tape unit accepted by BSAM may be specified.

[3] May be any disk unit accepted by BDAM.

[4] This attribute is defined as a subparameter of the DCB parameter of the SYSUT1 DD statement.

[5] 0=200 bpi, 1=556 bpi, 2=800 bpi.

[6] If input density is 1600 bpi, specify the parameter DEN=3; otherwise let DEN default to 800 bpi.

Figure 45. Defining Input Data for the Postprocessor

Data Attribute	7-Track Tape	9-Track Tape
Status/Disposition	Not Required	Not Required
Device	UNIT=2400-2	UNIT=2400
Label	LABEL=(,NL)	LABEL=(,NL)
Volume Identification	VOL=SER=number	VOL=SER=number
Density[1]	(DEN=$\left\{\begin{matrix}0\\ 1\\ 2\end{matrix}\right\}$)[2]	Default to 800 bpi
Block Size[3]	PARM=blocksize	PARM=blocksize
Parity	Not Required[4]	Not Required

[1] This attribute is a subparameter of the DCB parameter of the SYSUT2 DD statement.

[2] 0=200 bpi, 1=556 bpi, 2=800 bpi. If the DENSITY program control statement is used, this parameter is not needed.

[3] The output block size is specified by the PARM keyword parameter of the EXEC statement. The other parameters are specified in the SYSUT2 DD statement. The default value for PARM is 10,000 bytes; the maximum is 200,000 bytes.

[4] Tapes can be written in even, odd, or mixed parity.

Figure 46. Defining Output Data for the Postprocessor

- Mixed-density postprocessor output is needed.

Program control statements provide this information.

There are two types of program control statements: function statements and data statements. There are three function statements: HLABEL, TLABEL, and DENSITY. A function statement must have a period in column 1, a slash in column 2, and a blank in column 3. The function and its parameters can appear anywhere between column 4 and column 71. Only one function can be coded on a card.

A data statement contains the image, or a segment of the image, of a header or trailer label. The data statement has no keyword. It is identified by its following an HLABEL or TLABEL function statement.

Note: No comments are allowed on function or data statements.

DENSITY Program Control Statement

Using the DENSITY statement, the postprocessor can create a mixed-density 1400 tape. The first part of the tape, usually the header label and tapemarks, is in one density, and the rest of the tape is in another density. One DENSITY statement handles all tapes formatted in one postprocessor job. If two tapes need different densities, two postprocessor jobs must be run to format the tapes. The DENSITY statement overrides the DEN parameter in the SYSUT2 DD statement; if the DENSITY statement is omitted, the records are written in the density specified in the DEN parameter.

Operation	Operand
./ DENSITY	[FIRST(n)=den1][,REST=den2]

FIRST(n)=den1
    specifies that the first n records of the tape (including the header label and tapemarks) are to be written in the density specified by den1. A header label is one record and each tapemark is one record. n is a number from 1 through 9. den1 must be 2, 5, or 8, indicating 200, 556, and 800 bpi. If FIRST(n)=den1 is omitted, one record at 200 bpi is written. If FIRST=den1 is specified, one record is written in the density specified by den1.

REST=den2
    specifies that all records after those indicated by the FIRST parameter are to be written in the density specified by den2. den2 must be 2, 5, or 8, indicating 200, 556, and 800 bpi. The default value of den2 is 800 bpi.

Note: Do not code the DENSITY statement for 1400 tapes that have only one density.

Example: The postprocessor output is to be on a mixed-density tape. The input tape has one 1400 header label, which is preceded by a tapemark record and followed by two tapemark records. The 1400 header label and tapemarks are to be written at a density of 556 bpi; the rest of the tape is to be written at 200 bpi. The DENSITY statement for this tape is:

```
./ DENSITY FIRST(4)=5,REST=2
```

TLABEL Program Control Statement

The TLABEL statement defines the trailer label to be written when the postprocessor senses the end-of-volume reflective strip on the output tape. The trailer label is normally a series of tapemarks and a record that describes the tape.

In general, tapes produced by the emulator can be returned to 1400 format on one output tape. If the input tape is full or nearly full, or if the input tape is at a higher density than the output tape, two output tapes may be needed; in this case, the TLABEL statement should be specified.

The TLABEL statement is ignored if provided but not needed. Nine TLABEL statements can be coded for each postprocessing job. If no statement is coded and the end-of-volume reflective strip is sensed, two tapemarks are written on the tape, and the output file is continued on the next volume.

Operation	Operand
./ TLABEL	DATA=$\left\{\begin{matrix}\text{TM}\\ \text{nnn}\end{matrix}\right\}$

DATA=TM
    specifies that one tapemark is to be written. This operand can be specified once in each TLABEL statement; one TLABEL statement must be coded for each tape mark wanted. specifies that you want a trailer

label and that one or more data statements follow this TLABEL statement to define that label. (For a description of the data statement, see "Data Program Control Statement" later in this section.) nnn is a value frcm 1 through 999 that specifies the size in bytes of the trailer label. A discrepancy between the value specified in this operand and the number of characters in the data statement is processed as follows:

- If the data statement is omitted, the record is filled with blanks.
- If the value in DATA=nnn is greater than the number of characters in the data statement, the record is padded with blanks.
- If the value in DATA=nnn is smaller than the number of characters in the data statement, the remaining characters in the data statement are ignored.

HLABEL Program Control Statement

The HLABEL statement defines the header label to be written by the postprocessor when data must be written on the second of two output tapes. The header label is normally a series of tapemarks and a record of data that describes the contents of the tape. This statement should be specified to identify the second volume.

The HLABEL statement is ignored if provided but not needed. Nine HLABEL statements can be coded for each postprocessing job. If no statement is coded and a second output volume is needed, the postprocessor program does not write a header label or tapemark on the second volume.

Operation	Operand
./ HLABEL	$\text{DATA=}\left\{\begin{array}{l}\text{TM}\\\text{nnn}\end{array}\right\}$

DATA=TM
specifies that one tapemark is to be written. One HLABEL statement must be coded for each tapemark wanted.

DATA=nnn
specifies that you want a header label and that one or more data statements follow this HLABEL statement to define that label. nnn is a value from 1 through 999 that specifies the size in bytes of the header label. A discrepancy between the value specified in this operand and the number of characters in the data statement is processed as follows:

- If the data statement is omitted, the record is filled with blanks.
- If the value in DATA=nnn is greater than the number of characters in the data statement, the record is padded with blanks.
- If the value in DATA=nnn is smaller than the number of characters in the data statement, the remaining characters in the data statement are ignored.

Data Program Control Statement

The data statement specifies the contents of the trailer and header labels defined by the TLABEL and HLABEL statements. There is no keyword for a data statement; the statement must follow the TLABEL or HLABEL statement it refers to (when the DATA=nnn operand is coded).

variable string

variable string
specifies the characters in the header or trailer label. Code the exact number of characters specified for the label in the DATA=nnn operand of the TLABEL or HLABEL statement. One card is used for each 80-character segment of data until the required number of characters is reached. If this statement is omitted, the label specified by the DATA=nnn parameter is filled with blanks.

Example: Input to the postprocessor is a single volume in spanned format. The output will need two volumes. The end-of-volume indicator for the first output volume will be one tapemark followed by an 80-character trailer label followed by two more tapemarks. The first four characters of the trailer label are 1EOR; the rest of the label is blank. The header label of the second output volume is preceded by two tapemarks and followed by a single tapemark. The header label is 84 bytes long and contains 1HDR in the first four bytes and 1212 in bytes 81 through 84. The rest of the label is blank.

The program control statements needed to run this job are:

```
./  TLABFL DATA=TM
./  TLABFL DATA=80
1EOP
./  TLABEL DATA=TM
./  TLABFL DATA=TM
./  HLABEL DATA=TM
./  HLABEL DATA=TM
./  HLABFL DATA=84
1HDR
1212
./  HLABEL DATA=TM
```

FXECUTING THE TAPE FORMATTING PROGRAMS

The operating system reads the JCL statements that define the tape formatting program, allocates devices needed for the program, and then loads the program and gives it control. To show how the tape formatting programs and the emulator are related, the JCL examples provided below show 1400 files formatted by the preprocessor, and the same files formatted by the postprocessor after emulation.

The operator is asked to enter the number of tapemarks on the tape when the job is started. The program formats the data until that number of tapemarks is read. If the operator replies that he wants all tapemarks read, or if he gives a number greater than the number of tapemarks on the tape, the program formats the data until the tape runs off the reel or until a permanent I/O error is encountered.

When the preprocessor finishes formatting an input volume, it prints a summary for that volume. The postprocessor prints a summary for each output volume. The summary is printed on the device defined by the SYSPRINT DD statement. After a summary is printed, the program asks the operator if there are any more volumes to be formatted. The program continues to format volumes until there are no more volumes. Each volume must have the same characteristics as those defined in the JCL statements.

JCL EXAMPLES FOR THE TAPE PREPROCESSOR

PRFPROCESSOR EXAMPLE 1: A seven-track tape at 200 bpi is to be formatted. The tape has two files and eight tapemarks. The records are 80 characters long and are to be blocked into 800-byte blocks. The output will be on nine-track tape at 800 bpi. Although the tape is not labeled, an input volume serial number of 1400001 is used to help the operator identify the tape. The output number is 1400A1 and the data set name is PREOUT. The input reel is only half full and will not overflow the output volume. The JCL for this job is shown in Figure 47.

When the preprocessor issues message IIQ301D, the operator should respond REPLY id,'8'. The program then processes volume 140001, formats the data, writes the new data set on volume 1400A1, and records the tapemark and record distribution for the end-of-volume summary. The preprocessor stops at the eighth tapemark on the input volume. Both files are preprocessed.

The preprocessor prints the end-of-volume summary and issues message IIQ301D. This time the operator should respond REPLY id,'EOJ'. The preprocessor then returns control to the operating system, which ends the job.

PREPROCESSOR EXAMPLE 2: Two nine-track, 800 bpi tapes are to be formatted. The longest input record is 33,000 characters; output records are to be 4,000 characters long. The first volume has eight tapemarks; the second has sixteen.

The input serial numbers 000001 and 000002 help the operator identify the tape. The first input volume is almost full and, when formatted, might overflow onto a second volume. Thus, two volumes may be needed for the output of volume 000001. Output volumes for volume 000001 are numbered 0000A1 and 0000A2. The second input volume (000002) must also use the output volume serial number 0000A1, because 0000A1 is automatically requested by the operating system when starting the second input volume. The data set name is PREOUT for both output data sets. The control

```
//SAMPLE1    JOB    MSGLEVEL=(1,1)
//STEP1      EXEC   PGM=IIQPRE,PARM=80
//SYSPRINT   DD     SYSOUT=A
//SYSUT1     DD     UNIT=2400-2,DCB=(DEN=0),                            X
//                  LABEL=(,NL),VOL=SER=140001
//SYSUT2     DD     DISP=(NEW,KEEP),UNIT=2400,DSNAME=PREOUT,            X
//                  LABEL=(,SL),VOL=SER=1400A1,DCB=(BLKSIZE=800)
```

Figure 47. JCL Statements for Preprocessor - Example 1

statements for this job are shown in Figure 48.

The preprocessor issues message IIQ301D three times during the job. The responses should be:

REPLY id,'8' to format volume 000001
REPLY id,'16' to format volume 000002
REPLY id,'EOJ' to end the job

Two end-of-volume summaries are issued; one summary for volume 000001; the other for volume 000002.

PREPROCESSOR EXAMPLE 3: One seven-track, 556 bpi tape is to be formatted and the preprocessed data written on disk. The number of tapemarks on this volume is unknown. Input records are 80 characters long and are to be blocked into 800-byte blocks.

The input serial number is 1400A6, and the output serial number is 444444. The data set name for the output is PREOUT, and SYSUT2 describes a disk. The control statements for this job are shown in Figure 49.

Several operating system and preprocessor messages are issued. To the first IIQ301D message issued by the preprocessor, the operator should respond REPLY id,'ALL' to start formatting volume 1400A6. The preprocessor formats the volume until the input tape runs off the reel. The operating system detects this and issues message IEA000A. The preprocessor also issues message IIQ304D PREPROCESSOR INTERVENTION REQUESTED--REPLY EOV OR SKIP (jobname). The operator must respond to message IEA000A by readying the unit and to message IIQ304D by entering REPLY id,'EOV'. The preprocessor closes the input and output volumes and writes the end-of-volume summary before reissuing message IIQ301D. This time the operator should respond REPLY id,'EOJ' to end the preprocessing job.

JCL EXAMPLES FOR THE TAPE POSTPROCESSOR

POSTPROCESSOR EXAMPLE 1: The tape prepared in preprocessor example 1 has been processed by the emulator. The output is in spanned format and is to be postprocessed, During emulation the data set was expanded; the postprocessed output will probably need two volumes.

If the output overflows onto the second output volume, the trailer label for the first volume is to be 80 characters long, preceded and followed by a tapemark. The first four characters of the trailer label are to be 1EOR; the rest of the label is to be blank. The header label for the second volume is to be 80 characters long, preceded and followed by one tapemark. The first four characters of this label are to be 1HDR; the rest of the label is to be blank. The control statements for this job are listed in Figure 50.

The postprocessor scans and verifies the program control statements. It then issues message IIQ307D POSTPROCESSOR READY - REPLY ALL OR EOJ (jobname). The operator should respond REPLY id,'ALL'.

The postprocessor then formats the input volume. When the end of the first output volume is reached, the postprocessor writes

```
//SAMPLE2    JOB    MSGLEVEL=(1,1)
//STEP1      EXEC   PGM=IIQPRE,PARM=33000
//SYSPRINT   DD     SYSOUT=A
//SYSUT1     DD     UNIT=2400,LABEL=(,NL),VOL=SER=000001
//SYSUT2     DD     DISP=(NEW,KEEP),UNIT=(2400,2),LABEL=(,SL),               X
//                  DSNAME=PREOUT,DCB=(BLKSIZE=4000),                        X
//                  VOL=SER=(0000A1,0000A2)
```

Figure 48. JCL Statements for Preprocessor - Example 2

```
//SAMPLE3    JOB    MSGLEVEL=(1,1)
//STEP1      EXEC   PGM=IIQPRE,PARM=80
//SYSPRINT   DD     SYSOUT=A
//SYSUT1     DD     UNIT=2400-2,DCB=(DEN=1),                                 X
//                  LABEL=(,NL),VOL=SER=1400A6
//SYSUT2     DD     DISP=(NEW,KEEP),UNIT=2311,DCB=(BLKSIZE=844),             X
//                  VOL=SER=004444,SPACE=(CYL,(10,1)),DSNAME=PREOUT
```

Figure 49. JCL Statements for Preprocessor - Example 3

the trailer label and tapemarks specified in the TLABEL and data statements. The operating system then requests that the second volume be mounted using the same volume serial number as for the first volume.

After the second volume has been mounted, the postprocessor writes the header label and tapemarks specified in the HLABEL and data statements and continues formatting the input tape. When end-of-volume for the input volume is reached, the postprocessor closes both input and output files, prints an end-of-volume summary for that input volume, and reissues message IIQ307D. The operator should respond REPLY id,'EOJ'. The postprocessor then returns control to the operating system.

POSTPROCESSOR EXAMPLE 2: The tapes formatted in preprocessor example 2 have been processed by the emulator. The output is in spanned format and is to be postprocessed. The control statements for this job are listed in Figure 51. No program control statements are needed.

The postprocessor issues message IIQ307D three times during the job. The operator's responses should be:

REPLY id,'ALL' to format volumes 0000B1 and 0000B2 and to print the end-of-volume summary

REPLY id,'ALL' to format the third input volume (0000B1) and to print the end-of-volume summary

REPLY id,'EOJ' to end the job

POSTPROCESSOR EXAMPLE 3: The emulator output data is in tape format on a direct access device with volume number 400000. The output volume for the postprocessor is a seven-track tape with volume serial number 40000A. The input data set name is EMOUT. Input records are 3,500 bytes long; output records are 80 bytes long.

```
//SAMPLE1P  JOB   MSGLEVEL=(1,1)
//STEP1P    EXEC  PGM=IIQPOS,PARM=80
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    DISP=(OLD,KEEP),UNIT=2400,DCB=(BLKSIZE=800),          X
//                LABEL=(,SL),VOL=SER=1400A2
//SYSUT2    DD    UNIT=2400-2,DCB=(DEN=0),                              X
//                LABEL=(,NL),VOL=SER=140001
//SYSIN     DD    *
./                TLABEL DATA=TM
./                TLABEL DATA=80
1EOR
./                TLABEL DATA=TM
./                HLABEL DATA=TM
./                HLABEL DATA=80
1HDR
./                HLABEL DATA=TM
```

Figure 50. JCL Statements for Postprocessor - Example 1

```
//SAMPLE2P  JOB   MSGLEVEL=(1,1)
//STEP1P    EXEC  PGM=IIQPOS,PARM=33000
//SYSPRINT  DD    SYSOUT=A
//SYSUT1    DD    DISP=(OLD,KEEP),UNIT=(2400,2),                        X
//                LABEL=(,SL),DSNAME=PREOUT,                            X
//                VOL=SER=(0000B1,0000B2),DCB=(BLKSIZE=4000)
//SYSUT2    DD    UNIT=2400,LABEL=(,NL),VOL=SER=000001
//SYSIN     DD    DUMMY
/*
```

Figure 51. JCL Statements for Postprocessor - Example 2

```
//SAMPLE3P   JOB    MSGLEVEL=(1,1)
//STEP1P     EXEC   PGM=IIQPOS,PARM=80
//SYSPRINT   DD     SYSOUT=A
//SYSUT1     DD     DISP=(OLD,KEEP),UNIT=2311,DCB=(BLKSIZE=3500),        X
//                  VOL=SER=400000,DSNAME=EMOUT
//SYSUT2     DD     UNIT=2400-2,LABEL=(,NL),VOL=SER=40000A
//SYSIN      DD     *
./ DENSITY          FIRST(3)=2,REST=5
```

Figure 52. JCL Statements for Postprocessor - Example 3

The header label and its two accompanying tapemarks are to be written at 200 bpi and the rest of the tape at 556 bpi. Density is specified in the DENSITY program control statement rather than in the SYSUT2 DD statement. The control statements for this job are shown in Figure 52.

MESSAGES

Messages issued by the tape formatting programs are listed below. Explanations of the messages are in Appendix G.

IIQ301D PREPROCESSOR READY-REPLY nnn (NO. OF TM),ALL OR EOJ (jobname)

IIQ302D PREPROCESSOR INPUT I/O ERROR-REPLY EOV OR SKIP (jobname)

IIQ303D PREPROCESSOR INPUT RECORD TOO LONG-REPLY EOV OR SKIP (jobname)

IIQ304D PREPROCESSOR INTERVENTION REQUESTED-REPLY EOV OR SKIP (jobname)

IIQ305I PREPROCESSOR EOV, REEL nnn SUMMARY
tapemark and record distribution
NUMBER OF TAPEMARKS SPECIFIED- nnn
NUMBER OF TAPEMARKS FOUND- nnn
NUMBER OF INPUT RECORDS- nnnnn
SIZE OF LARGEST INPUT RECORD- nnnnnn BYTES
MAXIMUM OUTPUT BLKSIZE- nnnnn
INPUT RECORDS TOO LONG- nnn
INPUT I/O ERRORS- nnn

IIQ306I INVALID PARM FIELD ON EXEC CARD

IIQ307D POSTPROCESSOR READY-REPLY ALL OR EOJ-(jobname)

IIQ308D POSTPROCESSOR OUTPUT RECORD TOO LONG-REPLY EOV OR SKIP (jobname)

IIQ309I POSTPROCESSOR EOV, REEL nnn
SUMMARY
NUMBER OF TAPEMARKS WRITTEN- nnn
NUMBER OF OUTPUT RECORDS- nnn
SIZE OF LARGEST OUTPUT RECORD- nnnnnn BYTES
OUTPUT RECORDS TOO LONG- nnn
OUTPUT REELS CREATED FROM THIS INPUT- nnn

IIQ310I THE FOLLOWING POSTPROCESSOR CONTROL STATEMENTS ARE INVALID statements

IIQ311I POSTPROCESSOR OUTPUT I/O ERROR

IIQ312I POSTPROCESSOR INPUT BLOCKSIZE TOO SMALL

IIQ313I INVALID REPLY-MESSAGE WILL BE REPEATED-(jobname)

APPENDIX E: DISK FORMATTING PROGRAM

The disk formatting program is a problem program supplied with the emulator. It builds a System/370 data set in which data from a 1400 disk can be written. The data set contains blank records that are large enough for an entire track of data from the 1400 disk.

The disk formatting program can be executed on any System/360 or System/370 CPU, but the data set must be written on the disk pack to be used during emulation. A JCL parameter specifies the type of 1400 disk being formatted. The program creates a data set based on the size and number of 1400 records.

Note: Executing the disk formatting program is the second step of a three-step process to prepare a System/370 direct access device to be used to emulate a 1400 disk device. Preparing the device is described under "Converting Disk Files" in the section "Processing Data."

When the disk formatting program uses the Record Overflow feature, you need fewer System/370 cylinders to store the data from the 1400 device. Figures 53 and 54 may be used to determine the space needed for the System/370 data sets.

Emulated Device	1400 Tracks Per System/370 Track			Number of System/370 Cylinders Required		
	2311	2314	3330	2311	2314	3330
1301 Disk: Sector Mode	1.5	3	5.7	634	161	92
1301 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2.5	4.8	744	188	10
1311 Disk: Sector Mode	1.5	3	5.7	64	17	9
1311 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2	4.2	87	22	12
1405 Disk, Model 1	3	6	11	316	82	47
1405 Disk, Model 2	3	6	11	632	163	93

Figure 53. Track and Cylinder Correspondence Using the Record Overflow Feature

Emulated Device	1400 Tracks Per System/370 Track			Number of System/370 Cylinders Required		
	2311	2314	3330	2311	2314	3330
1301 Disk: Sector Mode	1	3	5	1000	167	105
1301 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2	4	1000	250	132
1311 Disk: Sector Mode	1	3	5	100	17	11
1311 Disk: Track-Record Mode, or Both Track-Record and Sector Modes	1	2	4	100	25	14
1405 Disk, Model 1	3	6	11	334	84	48
1405 Disk, Model 2	3	6	11	667	167	96

Figure 54. Track and Cylinder Correspondence Without the Record Overflow Feature

MINIMUM SYSTEM REQUIRED

The disk formatting program needs a System/360 or System/370 with:

- 2,364 bytes of main storage and a buffer large enough to hold the formatted record (maximum buffer size is 2,992 bytes)
- One system input device for JCL statements
- One system output device for error messages
- One system console
- Direct access storage for the output file

BUFFER STORAGE

The size of the buffer used by the program is determined by the type of 1400 device to be emulated. Figure 55 shows the buffer (data set) needed to hold the formatted record. Buffer storage must be added to the program storage when determining main storage needed for the program. The program obtains its own buffer during execution.

Figure 55 shows the System/370 data set needed to emulate the 1400 device. The data set is larger than the track because 12 bytes of control information precede each record; if in sector mode, 8 bytes of control information precede each sector but the first, which is preceded by 12 bytes.

JCL STATEMENTS

The control information needed to run this program is given in JCL statements. The JOB and EXEC statement parameters are:

```
//jobname    JOB  MSGLEVEL=(1,1)
//stepname   EXEC  PGM=IIQDFU,
//                PARM='ddddd-n[,ddddd-n]'
```

You supply the job name and step name. The PARM parameter identifies the type and number of 1400 disk devices to be formatted.

ddddd

specifies the type of 1400 disk to be formatted. The ddddd field must be replaced by one of the five-character PARM values listed in Figure 55.

-n

specifies number of devices of type ddddd to be formatted. The n field must be replaced with a value from 1 through 9. Other values are invalid and cause the program to stop. If this field is omitted, 1 is assumed.

Examples of the PARM parameter are:

PARM=1301S

A data set is created to emulate a 1301 in sector mode.

PARM='1311M-4'

Four data sets are created to emulate four 1311s in both track-record and sector modes.

1400 Device	PARM Value	System/370 Data Set Size	Actual 1400 Track Size
1301 Sector Mode	1301S	10,000 2164-Byte Records	20 Sectors at 100 Characters per Sector
1301 Track, or Both Track and Sector Modes	1301M	10,000 2555-Byte Records	2543 Characters
1311 Sector Mode	1311S	1,000 2164-Byte Records	20 Sectors at 100 Characters per Sector
1311 Track, or Both Track and Sector Modes	1311M	1,000 2992-Byte Records	2980 Characters
1405, Model 1	1405A	10,000 1044-Byte Records	5 Sectors at 200 Characters per Sector
1405, Model 2	1405B	20,000 1044-Byte Records	5 Sectors at 200 Characters per Sector

Figure 55. Disk Formatting Program PARM Values and Data Set Sizes

```
//DISKFP     JOB    CLASS=I,MSGLEVEL=(1,1),RD=RNC,REGION=4K
//STEP1      EXEC   PGM=IIQDFU,PARM=1405A
//SYSUDUMP   DD     SYSOUT=A
//SYSPRINT   DD     SYSOUT=A
//SYSUT1     DD     DSNAME=MSFILE,UNIT=(2311,2),DISP=(NEW,KEEP,DELETE), X
//                  VOL=SER=(145000,145001),DCB=(RECFM=F),              X
//                  SPACE=(CYL,(198,136)),LABEL=(,,,EXPDT=73001)
/*
```

Underscored values represent variables. All other values must be coded as shown.

Figure 56. JCL Statements for the Disk Formatting Program - Example

PARM='1301S,1311M-4'

All of the above are emulated in one job.

Three DD statements are required: a SYSUDUMP DD statement to be used when the operating system abnormally ends the program, a SYSPRINT DD statement to define normal printer output, and a SYSUT1 DD statement to define the formatted data set. These DD statements are shown in Figure 56.

The SYSUT1 DD statement must have the six operands defined below and can have other operands as needed. Additional information on the DD statement can be found in Job Control Language Reference, GC28-6704.

DSNAME=name
specifies a data set name for each data set created by the disk formatting program.

UNIT=(type[,number])
specifies the type and number of units needed by the program.

DISP=(NEW,KEEP,DELETE)
specifies what to do with the data set.

VOL=SER=(number[,number]...)
specifies the volume serial number of each volume.

DCB=(RECFM=F) or DCB=(RECFM=FT)
specifies a data set with fixed-length records. If the Record Overflow feature is used, code DCB=(RECFM=FT).

SPACE=
specifies the secondary storage needed for the data. The number of cylinders for each device is shown in Figure 53 and 54. You should not request more than 198 cylinders on a 2311 or 2314 (generally the maximum number of cylinders available after subtracting the VTOC). You should not request more than 402 cylinders on a 3330. If the number of cylinders exceeds the maximum, the excess must be requested using the secondary quantity request field. For example, a 1311 disk with data in sector mode requires 100 cylinders on a 2311 without the Record Overflow feature. The SPACE parameter should be coded:

SPACE=(CYL,100)

The 1405 Disk Model 1, requires 334 cylinders. The SPACE parameter should be coded:

SPACE=(CYL,(198,136))

If data sets are being prepared for more than one 1400 disk device, additional DD statements are needed. These data sets must be defined by DD statements with the names SYSUT2, SYSUT3, SYSUT4, and so on. Each DD statement must contain the same required operands as the SYSUT1 DD statement, and each may have optional operands. The disk formatting program takes each DD statement in the order listed in the JCL and uses it to create the data set for the 1400 disk that is next in the PARM field. The program continues to create data sets for all the devices in the PARM field. If there are not enough DD statements, or if a DD statement is missing, the program is abnormally ended by the operating system.

Using the PARM parameter in the example above, PARM='1301S,1311M-4', five DD statements are required. The SYSUT1 DD statement defines the 1301, and the SYSUT2 through SYSUT5 DD statements define the four 1311s.

You can use the UNIT and SPACE parameters to:

- Indicate the device used for the data set created by the program. (However, OS actually assigns the data set to the device.)
- Specify the number of disk units used to emulate a 1400 disk device.

The example below shows a data set in which the first 2311 is used only for that data set, and the second for the rest of the data set. If there were a need to divide the data set evenly between the two devices, the SPACE parameter could be coded SPACE=(CYL,(167,167),RLSE); or you could create a dummy data set to limit the available space on the device, and then run the disk formatting program. The operating system would allocate the remaining space to the disk formatting program.

Example: The disk formatting program is to be run so that the emulator can use two 2311s to emulate a 1405 Disk Storage, Model 1. There is no Record Overflow feature on the control unit of the 2311s. All of one 2311 is used for the data set, and the second for the rest of the data set. The CLASS parameter is optional, but is used because the program is I/O bound. Replace the "I" with the class name that you use to indicate high I/O activity. The JCL statements are shown in Figure 56. The LABEL parameter is used to keep the data set from being deleted accidentally by another program. (If you need to delete such a data set, it can be done using the IEHPROGM utility program. The IEHPROGM utility is described in Utilities, GC28-6586.)

MESSAGES

Messages issued by the disk formatting program are listed below. Explanations of the messages are in Appendix G.

IIQ401I ddname ASSIGNED PARM VALUE NOT FOUND

IIQ402I ddname UNABLE TO OPEN DATA SET

IIQ403I ddname IMPROPER RECORD FORMAT SPECIFIED

IIQ404I ddname BUFFER WAS NOT AVAILABLE

IIQ405I I/O ERROR, jobname, stepname, unit addr, device type, ddname, operation attempted, error description, last seek addr, access method

APPENDIX F: ADDING ROUTINES YOU HAVE WRITTEN

You may write your own routines to perform functions not provided by the emulator. This section is written for the systems analyst who must decide whether a special routine is required, and for the systems programmer who must design and code it.

Your routines must be assembled and catalogued before you execute the emulator job in which they are used. When the job is loaded, your routines are loaded with the emulator.

You are modifying Type I coding when you add a routine, and you may be billed for a program maintenance call if it contains or causes an error.

USER STATEMENT

The USER emulator control statement tells the emulator which routines you want loaded with the emulator and what 1400 operations they are emulating. This statement is coded and placed in the SYSEMCTL data set with the other emulator control statements. The format of the USER statement is shown in Figure 30, a fold-out page preceding Appendix A.

You can write routines to emulate 1400 operations not included in the emulator. For example, you may write routines to read and process column binary cards and to process data on paper tape.

In addition, you can write a routine to process 1400 program messages directly, without having them printed on the console. This is called "automatic reply."

EMULATING 1400 OPERATIONS

One USER statement must be coded for each 1400 operation code to be emulated, except for I/O operation codes. Only one USER statement is needed for all I/O operation codes.

The format of the USER statement for emulating 1400 operation codes is:

```
USER NAME=name,OPCODE={IO},CONTROL=nn
                      {c }
```

where:

NAME=name
identifies the load module in which your routine is located. Whenever your operation code is encountered, control is passed to the first instruction in the load module.

OPCODE
specifies the operation code your routine inspects or emulates. IO must be coded for all I/O operation codes: M, L, U, K, F, and 1 through 9. For other operation codes, replace c with the BCD operation code. There are three operation codes that must be multiple-punched to be entered. The codes and the punch combinations are:

Operation Code		Punch Combination
!	(Zero and Subtract)	11-0
?	(Zero and Add)	12-0
⌑	(Clear Word Marks)	12-4-8

CONTROL
specifies the control byte used to decode the 1400 instruction. Before branching to your routine, the compatibility feature uses the control byte for decoding and error checking. If the CONTROL operand is not specified, 00 is assumed. If OPCODE=IO, the CONTROL operand is ignored. The values of the control byte are shown in Figure 57.

Your routines must follow the conventions used by the emulator. These conventions include:

- Use of registers
- Complete handling of I/O operations for an emulated device
- Proper posting of the status of the operation
- Return of control

Bit No.	Bit Setting	Description
0 and 1	00nnnnnn	The operation code is not M, L, Q, or U (non-I/O operation).
	01nnnnnn	The operation code is Q.
	10nnnnnn	The operation code is M or L.
	11nnnnnn	The operation code is U.
2	nn0nnnnn	The compatibility feature analyzes the instruction and loads the A-address of the instruction into the AAR and the B-address into the BAR.
	nn1nnnnn	This is a double address instruction. The compatibility feature loads the A-address of the instruction into both the AAR and the BAR.
3 and 6	nnn0nn0n	Branch to your routine.
	nnn1nn0n	Invalid.
	nnn0nn1n	This bit configuration should not be coded. The operation code is for a branch instruction that is executed by the compatibility feature (operation codes V and W).
	nnn1nn1n	The operation code is B. The instruction is executed by the compatibility feature except for Branch on Indicator, where a branch is made to your routine. If you set bits 3 and 6 to 1, you must emulate the Branch on Indicator instruction in your routine. Note: If your routine emulates a V or W opcode, bits 3 and 6 must be 0 for that operation code.
4	nnnn0nnn	The compatibility feature should search for a wordmark to stop decoding this instruction. The wordmark is on the operation code of the next instruction.
	nnnn1nnn	The operation code is Clear Storage (/) or Set Wordmark (,). The compatibility feature does not search for a wordmark to stop decoding the instruction.
5	nnnnn0nn	The compatibility feature should immediately branch to the routine that emulates the instruction. Your routine must update the IAR. Note: If this early branch is required, the results may be unpredictable unless the control byte is zero (CONTROL=00).
	nnnnn1nn	The compatibility feature should update the IAR before branching to your routine.
7	nnnnnnn1	This bit must be set to 1.

Figure 57. Bit Settings for the USER Control Byte

GETTING CONTROL AND USING REGISTERS

Information about the 1400 instruction to be emulated is in the general purpose registers:

- Register 2 is used by the compatibility feature and must not be modified by your routine.
- Register 3 contains the address of the communication region, and should be used as a base register for the communication region DSECT (DIIQCR).

It must not be modified by your routine.

- Registers 4 and 5 contain the main storage addresses of the B-address and A-address fields of the 1400 instruction. These addresses are set according to conventions of the 1400 system, but are maintained in registers 4 and 5 as System/370 binary addresses. They must be updated by your routine as a normal part of emulating a 1400 instruction. The high-order byte is a control byte for the compatibility feature. (For more information, see "BIFLAG-Branch if Flag" under "Compatibility Feature Macro Instructions.")

- Register 6 contains the DILCNT value in the first 7 bits of the high-order byte and the emulated 1400 IAR in the three low-order bytes. The contents of the three rightmost bytes, a System/370 binary address, depend on the setting of bit 5 of the control byte. If OPCODE=IO, the IAR always contains the address of the next sequential 1400 instruction. For operation codes other than I/O, the IAR is set according to bit 5 of the control byte, which is explained in Figure 57. You must not alter the leftmost byte of this register.

- Register 7 contains the entry point address of your routine, and can be used as the base register for the routine except when OPCODE=IO. If OPCODE=IO, register 7 contains the entry point address of the routine that called your routine. Do not modify register 7 when OPCODE=IO.

- Register 13 contains the address of a save area in the communication region. This save area can be used by your routine, and the locations in the save area that contain the AAR, BAR, and IAR can be modified.

- Register 14 is variable unless OPCODE=IO. If OPCODE=IO, register 14 contains the emulator return address when the I/O instruction in your routine is not emulated.

- Register 15 is variable unless OPCODE=IO. If OPCODE=IO, register 15 contains the entry point address of your routine. You may use register 15 for your base register for all I/O instructions.

The contents of registers should be saved when entering your routine and returned to the emulator exactly as they were, except for registers 4, 5, 6, and 7, which may be updated by your routine.

The first word of the communication region contains additional information about the 1400 instruction, and other control information. This information is presented in Figure 58.

Your routine is brought into storage using a LOAD macro instruction and, as such, cannot contain V-type address constants or EXTERN statements. If your routine is not in the SYS1.LINKLIB data set, the JCL for the emulator job must contain a JOBLIB or STEPLIB DD statement defining the library the routine is in.

EMULATING 1400 INSTRUCTIONS

A 1400 instruction is emulated using the System/370 universal instruction set and seven macro instructions that execute microprogrammed routines in the compatibility feature. These macro instructions simplify the access to and translation of the 1400 program and data, and are described under "Compatibility Feature Macro Instructions."

Your routine should be assembled with the DSECT DIIQCR (the emulator communication region). The DSECT is located in the Emulator Distribution Library, and you may use the same SYSLIB card for your routine as you used when assembling Stage I of emulator generation. The USING statement for the DSECT should be coded USING EMCOMRG,COMREG. COMREG is register 3.

<u>EMULATING ENTIRE NON I/O OPERATION CODES</u>: To emulate an entire operation code, the control byte specified in the USER statement should be X'00'. After getting control and saving registers, you should emulate the operation code, restore registers, update the IAR, and exit using either the DIL or BDIL macro instruction. (For information on DIL and BDIL, see "Compatibility Feature Macro Instructions.")

You may code a control byte other than X'00' to update the IAR automatically or to provide other functions that would otherwise have to be in your routine. The control byte is described in Figure 57.

<u>EMULATING PORTIONS OF NON-I/O OPERATION CODES</u>: To handle only a few instructions, (such as one or more branch d-modifiers), and let the emulator handle the remaining instructions in that operation code, you

must find the control byte used to emulate the operation code and specify it in the CONTROL operand of the USER statement. The control byte is the leftmost byte of each word in the CROPCODE table (Communication Region Operation Code table). The CROPCODE table is in the CSECT IIQCR01, which is assembled during emulator generation.

The address of the routine that emulates the operation code is also in the full word that contains the control byte. When your routine finds that it does not emulate an instruction it must return control to this emulator routine. The emulator routine address is destroyed, however, when the address of your routine is placed in the CROPCODE table.

To obtain the address of the emulator routine:

<table>
<tr><th colspan="2">Displacement and Name</th><th>Description</th></tr>
<tr><td>0</td><td>CRDMOD</td><td>A 1-byte field set by the compatibility feature. If there is a d-modifier associated with the instruction, it is stored in CRDMOD in internal code. (See Appendix A for internal code information.) CRDMOD contains a d-modifier only when bit 7 of CRLENGTH is 1. If bit 7 is 0, or if CRIAREG contains an address, the value is a d-modifier from a previous instruction.</td></tr>
<tr><td>1</td><td>CRIAREG</td><td>A 3-byte field set by the compatibility feature. The field contains the absolute main storage address of the instruction being emulated if the instruction has an M, L, or U operation code, and an x-control field. The field is not set if these conditions are not met. You have an address in this field when the first two bytes are other than hexadecimal '0000'.</td></tr>
<tr><td>3</td><td>CRLENGTH
(or CRTRUTH)</td><td>A 1-byte field set by the compatibility feature when there is no address in the CRIAREG field. This field is set to zero if the control byte for the instruction is zero. For all other conditions, bit settings are determined by the length of the 1400 instruction. Bits 0 through 3 show the instruction length in bytes (in binary). Bits 4 through 7 are indicators and if set to 1 mean:

Bit 4 - Reserved.
Bit 5 - The instruction has a complete A-address.
Bit 6 - The instruction has a complete B-address.
Bit 7 - The instruction has a d-modifier.

Bit settings for CRLENGTH are:

<table>
<tr><th colspan="4">Instruction Length</th><th colspan="4">Indicators</th></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

Figure 58. Information from the Communication Region

- Locate the entry point of the emulator routine (named in the source code of the CROPCODE table of CSECT IIQCR01) in the link edit map for the emulator load module (default name IIQE141).

- Define, in your routine, a constant equal to the displacement for the entry point that you found in the link edit map.

- Add the constant to the contents of CRLOADAD (which contains the load address of the emulator load module). CRLOADAD is defined in the DSECT DIIQCR.

After getting control and saving registers, emulate the instruction and return control using the DIL or BDIL macro instruction. If your routine does not emulate the instruction, restore the registers, load register 7 with the address of the routine that emulates the instruction, and exit using the address in register 7.

EMULATING I/O INSTRUCTIONS: When the USER emulator control statement specifies OPCODE=IO, the compatibility feature branches to a routine that decodes the instruction. Then, for every I/O operation, it branches to your routine. Your routine must examine the instruction to see whether it emulates it. If it does not, it must return control to the emulator program. The return address is in register 14.

If you emulate one I/O instruction for a device, you must emulate all I/O operations for that device, including opening and closing the data sets. When your routine emulates I/O instructions for a device, it must emulate both single-operation instructions (Read, Punch, Print, etc.) and the appropriate parts of combined-operation instructions (Read and Punch, Read, Punch, and Print, etc.) for that device.

EMULATING PORTIONS OF COMBINED OPERATIONS: Your routine can emulate portions of a combined operation by modifying the operation code and the IAR. For example, if your routine emulates the read portion of the Read, Punch, and Print instruction (operation code 7), it must emulate the read portion of the instruction, change the operation code of the 1400 instruction in emulated 1400 storage to 6, reset the IAR to point to the 6, and then issue a DIL macro instruction. Since 6 is also an I/O operation code (the Punch and Print instruction), your routine gets control again after the emulator has saved the Punch and Print instruction. Your routine must now reset the operation code back to 7 so that the 1400 program will be executed properly if this section of coding is entered again. Your routine must then return control to the emulator at the return address in register 14. The emulator executes the Punch and Print instruction and fetches the 1400 instruction.

RETURNING CONTROL TO THE EMULATOR

You must branch on register 14 when you have examined a 1400 I/O instruction and found it is not one that your routine emulates. Register 14 contains the address of the emulator routine that emulates the instruction or the address of an error routine if the instruction cannot be emulated.

You must load register 7 with a return address and branch on register 7 when you have examined a non-I/O instruction and found that it is not one that your routine emulates. Choosing the proper return address is discussed under "Emulating 1400 Instructions."

When your routine has emulated the 1400 instruction, you must use the DIL or BDIL macro instruction to fetch the next sequential instruction.

COMPATIBILITY FEATURE MACRO INSTRUCTIONS

Use compatibility feature macro instructions whenever you need access to the 1400 program or its data or when you wish to return control to the compatibility feature. The 1400 program and its data are located in emulated 1400 storage in internal code. Some compatibility feature macro instructions work with internal code and others translate it to EBCDIC so that it can be used by System/370 instructions. All parameters of the macro instructions are positional and must be replaced with commas when not coded. Whenever your routine contains a compatibility feature macro instruction that uses a location in the DSECT DIIQCR, that DSECT must be assembled with the routine.

ANUM - Add Numeric

The ANUM macro instruction is the equivalent of the 1400 instructions:

- Add

- Subtract

- Zero and Add

- Zero and Subtract

- Modify Address

All arithmetic operations are performed in internal code under algebraic sign control. The operations is executed starting in the units position. The overflow indicator is set, and the reg1 and reg2 registers are decremented by the number of bytes processed. All operands must be coded.

Name	Operation	Operand
[label]	ANUM	reg1,reg2,addr

reg1
    is a general register containing the address of the destination field (B-field).

reg2
    is a general register containing the address of the source field (A-field).

addr
    is the address of the control byte. The control byte identifies the 1400 instruction to be emulated. Only the following hexadecimal values may be coded in the control byte:

01	-	Add
03	-	Subtract
05	-	Zero and Add
07	-	Zero and Subtract
09	-	Modify Address

The overflow indicator is in the DSECT DIIQCR at location CRCPUIND.

Macro Example: ANUM BAR,AAR,FP14AD where FP14AD is hexadecimal '01'. This example adds the A-field to the B-field.

BDIL - Branch and Do Interpretive Loop

The BDIL macro instruction places the address from the IAR (register 6) into the BAR (register 4) and the address from the AAR (register 5) into the IAR. It then returns control to the compatibility feature, which emulates 1400 I-fetch. This macro instruction is issued when your routine has emulated a 1400 instruction that requires a branch. The compatibility feature fetches the instruction in the 1400 program pointed to by the new IAR, decodes it, and branches to an emulation routine. The IAR must point to a valid 1400 instruction in internal code, but the instruction does not have to be in 1400 storage. Both operands must be coded.

Name	Operation	Operand
[label]	BDIL	CROPCODE,EMCOMRG

CROPCODE
    is a location in DSECT DIIQCR that points to the operation code branch table in the communication region. This operand must appear exactly as shown.

EMCOMRG
    is the name of the communication region in the DSECT DIIQCR. This operand must appear exactly as shown.

BIFLAG - Branch If Flag

The BIFLAG macro instruction tests for error flags that may have been generated when the compatibility feature translated and loaded the A- and B-fields of the 1400 instruction into the AAR and BAR. If the AAR or BAR contains an error, byte 0 of that register is not zero, and this macro instruction branches to an error routine. If byte 0 is 0, processing continues at the next sequential instruction. All operands must be coded.

Name	Operation	Operand
[label]	BIFLAG	reg1,reg2,EMCOMRG

reg1
    is a general register that contains the emulated AAR or BAR.

reg2
    is a general register that contains the emulated AAR or BAR.

EMCOMRG
    is the name of the communication region in the DSECT DIIQCR. This operand must appear exactly as shown.

Macro Example: BIFLAG AAR,AAR,EMCOMRG Only the address in the AAR is checked for validity.

COMP - Compare

The COMP macro instruction is the equivalent of the 1400 Compare instruction. This macro instruction compares the 1400 data field from right to left. (This is from high to low in System/370 main storage.) The arguments do not have to be in 1400 storage, but they must be in internal code. The first wordmark encountered in either A-field or B-field stops the compare, with the PSW condition code set as follows:

01 - A wordmark was found in the B-field
10 - A wordmark was found in the A-field
11 - A wordmark was found in both fields

The registers identified by reg1 and reg2 are decremented by the number of bytes compared. All operands must be coded.

Bits 1, 2, and 3 of location CRCPUIND in the DSECT DIIQCR can be tested to see whether the ccmpare was low, high, or equal:

n100nnnn - B is less than A (low)
n010nnnn - B is equal to A
n001nnnn - B is greater than A (high)

Name	Operation	Operand
[label]	COMP	reg1,reg2,CRBCDEBC

reg1
    is a general register containing the address of the A-field.

reg2
    is a general register containing the address of the B-field.

CRBCDEBC
    is the name of the collating sequence table (translation table) in the DSECT DIIQCR. It must be coded exactly as shown.

<u>DIL - Do Interpretive Loop</u>

The DIL macro instruction returns control to the compatibility feature, which emulates 1400 I-fetch. This macro instruction is issued when your routine has emulated the current instruction. The compatibility feature fetches the next sequential instruction, decodes it, and branches to an emulation routine. The IAR (register 6) must point to a valid 1400 instruction in internal code, but the instruction does not have to be in 1400 storage. Both operands must be coded.

Name	Operation	Operand
[label]	DIL	CROPCODE,EMCOMRG

CROPCODE
    is a location in DSECT DIIQCR that points to the operation code table in the communication region. This operand must appear exactly as shown.

EMCOMRG
    is the name of the communication region in the DSECT DIIQCR. This operand must appear exactly as shown.

<u>MCPU - Move Data in CPU</u>

The MCPU macro instruction is the equivalent of the 1400 move instructions. Data in internal code is moved character by character according to information in the control byte. The control byte provides the same information as the d-modifier. All operands must be coded.

Name	Operation	Operand
[label]	MCPU	reg1,reg2,addr

reg1
    is a general register that contains the address of the destination field (B-field).

reg2
    is a general register that contains the address of the source field (A-field).

addr
    is the address of the control byte. The following bit settings are permitted:

    00010001 - Move Numeric
    00100001 - Move Zone
    00111101 - Move
    10111101 - Move Record
    01110101 - Load
    10001001 - Scan Right to Groupmark Wordmark
    00001001 - Scan Left to Wordmark in B-field

<u>Macro Example</u>: MCPU BAR,AAR,FPPF
FPPF is defined as hexadecimal '75'.

<u>MIO - Move Data for Input/Output</u>

The MIO macro instruction is used to translate data in EBCDIC into internal code and vice versa. Data is in EBCDIC in System/370 data management buffers (or in a modified form of EBCDIC, where bit 1 of each byte indicates mode or parity). Data is in internal code in emulated 1400 storage. Data is moved character by character according to information in the control byte until the count register reaches zero. All operands must be coded.

<u>Control Byte</u>: Figure 59 shows the bit settings permitted in the control byte. (Note that bit setting nn01nnnn results in a specification exception.)

<u>Count Register</u>: The number of the count register is one lower than that of the register identified in the reg1 operand. This register must contain the size of the System/370 buffer in bytes. Bytes 2 and 3 are used for the count; bytes 0 and 1 are ignored.

The count register is decremented by one after each character is moved. After the register is decremented, it is tested. When the count reaches zero, data movement stops. The two conditions that can stop data movement before the count reaches zero are:

- Encountering a groupmark wordmark when bit 6 of the control byte is 1
- Reaching the end of emulated 1400 storage

Upon completing the MIO instruction, byte 0 of the count register contains one of three binary values:

00000000 - The count reached zero.
00010000 - Data movement stopped at end of emulated 1400 storage.
00100000 - Data movement stopped at a groupmark wordmark. The groupmark wordmark was moved.

Name	Operation	Operand
[label]	MIO	reg1,reg2,addr

Bit No.	Bit Setting	Description
0 and 1	00nnnnnn	Always set to zero.
2	nn0nnnnn	Move mode is to be used to move the data.
	nn1nnnnn	Load mode is to be used to move the data.
3 and 4	nnn00nnn	The System/370 buffer is for nine-track tapes with data in odd-parity normal mode. On data moves from the System/370 buffer, bit 1 of each byte is tested to make sure it is 0. If it is not, there is a parity error, and an asterisk is written in 1400 storage and the invalid data condition code is set. On data moves from 1400 storage, bit 1 is set to 0 before the data is moved so that the correct parity is maintained.
	nnn01nnn	The System/370 buffer is for unit record devices, seven-track tapes, or nine-track tapes with data in EBCDIC or even-parity normal mode. On data moves from the System/370 buffer, bit 1 of each byte is tested to make sure it is 1. If it is not, there is a parity error, and an asterisk is written in 1400 storage and the invalid data condition code is set. On data moves from 1400 storage, the data is translated and bit 1 is set to 1.
	nnn10nnn	The System/370 buffer is for disk units. The data is moved, but is not translated (output only).
	nnn11nnn	The System/370 buffer is for disk units. Data is translated as it is moved.
5	nnnnn0nn	Data is to be moved from 1400 storage to the System/370 buffer.
	nnnnn1nn	Data is to be moved from the System/370 buffer to 1400 storage.
6	nnnnnn0n	The data move is not to be stopped if a groupmark wordmark is encountered.
	nnnnnn1n	The operation is to be stopped if a groupmark wordmark is encountered.
7	nnnnnnn1	Bit 7 is always set to 1.

Figure 59. Bit Settings for the MIO Control Byte

reg1
: is a general register containing the address of a System/370 buffer. This register cannot be register 2, 3, or 4. The count register must contain the number of bytes in the buffer.

reg2
: is a general register containing the address of a data field in 1400 storage.

addr
: is the address of a control word. This word must contain a control byte in byte 0 (see Figure 59) and the address of a translation table in bytes 1 through 3. There are two translation tables in DSECT DIIQCR. When moving data from 1400 storage to the System/370 buffer (output), use the table at location CRBCDEBC. When moving data from the System/370 buffer to 1400 storage (input), use the table at location CREBCDIC.

  Special considerations must be made for printer operations because of printer graphic differences. Interleaved in the translation table at location CRBCDEBC is a table that resolves differences in printer graphics. The interleaved table differs from the basic table in that unprintable characters are replaced by EBCDIC blanks, thereby reducing search time for characters on the print chain. The basic table should be used for all output operations to card-punch, tape, disk, and console devices. The translation table that should be used for output operations to the printer is at location CRBCDEBC+1 byte.

<u>Notes</u>:

1. Console graphic differences are handled by the emulator.

2. You may use your own translation tables. Additional information may be found in <u>1401/1440/1460 OS Emulator on Models 145/155</u>, GY33-7011.

<u>Macro Example</u>: MIO DKCFIVE,DKCSIX,DKCTLBYT where DKCFIVE and DKCSIX are registers 5 and 6. Register 4 is the count register. DKCTLBYT has been defined as:

```
DKCTLBYT  DS   0F
          DC   X'01'
          DC   AL3(CRBCDEBC+1)
```

WRITING DEBUGGING AIDS

You can write a routine for address stops, instruction step, or snap dumps during 1400 programs. An invalid operation code should be selected and a routine written to take control when the invalid code is encountered. By inserting the invalid operation code at selected locations in the 1400 program, control is passed to the routine when these locations are reached. The 1400 program can be modified by inserting the invalid operation code in the object deck before the program is put in the reader, by using the ALTER command, or by including the modification in the routine. The operation code G is an example of an invalid operation code.

AUTOMATIC REPLY

You can write a routine that suppresses the printing of 1400 program messages on the console and handles them directly. If you specify that certain messages are to be replied to automatically, your routine receives control each time a message is to be printed on the console and checks whether this is one of the messages for which you have specified an automatic reply. If it is, your routine replies and the message is not sent to the console; if it is not, your routine sends the message to the console.

The format of the USER statement for automatic reply is:

```
USER NAME=name,OPCODE=AR
```

where

NAME=name
: identifies the load module in which your routine is located. Whenever a message that is to be printed on the operator console is encountered, control is passed to the first instruction in the load module.

OPCODE=AR
: specifies automatic reply.

Using automatic reply, you can:

- Change the routing code before sending a message to the console.

- Turn on the inquiry latch immediately for 1400 programs that have console wait loops (which cannot be detected by the emulator).

REGISTER USAGE

Your routine must save and restore the contents of registers. The parameters passed to your routine are placed in register 0 (X'00' if WTO; X'04' if WTOR) and register 1 (the address of the parameter list). The formats of the WTO and WTOR parameter lists are shown in Figures 60 and 61.

Your routine saves the address of the ECB (event control block) and the reply address of the outstanding WTOR. Register 15 contains a return code that determines if a message is to be sent to the console. If it contains X'00', the message is not sent to the console; otherwise, it is sent. The address of the descriptor field is found by adding the contents of the message length field to the address of the message length field. (For a description of event control blocks, see System Control Blocks, GC28-6628; for a description of reply address and descriptor codes, see Supervisor and Data Management Macro Instructions, GC28-6647.)

Note: Your routine is always called by the BALR 14,15 instruction.

SAMPLE ROUTINE USING AUTOMATIC REPLY

Figure 62 is a sample routine that illustrates some of the techniques used in coding a routine for automatic reply. This routine replies to halt messages detected at IAR=01234, IAR=05602, and AAR=111, and sends all other messages to the console.

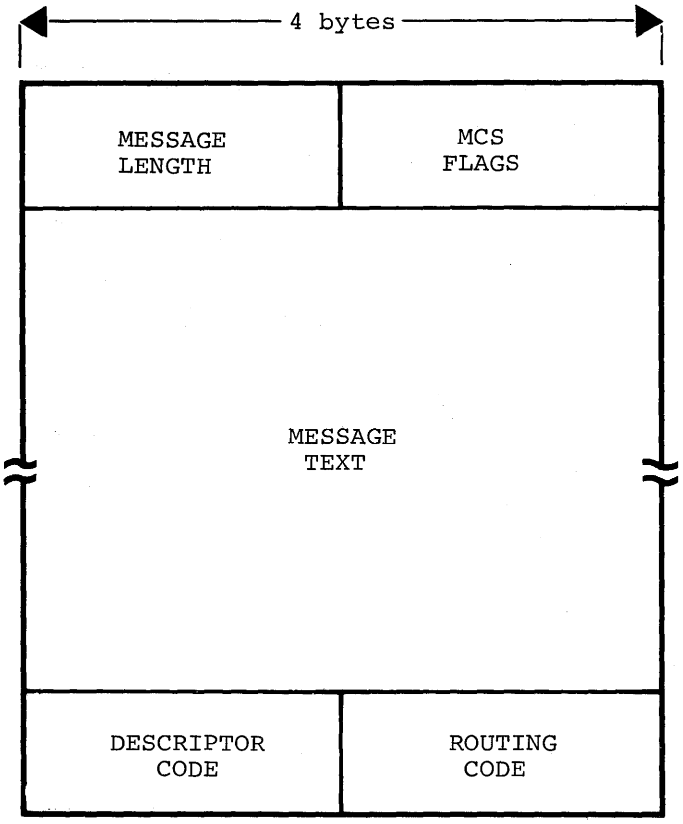


Figure 60. WTO Parameter List

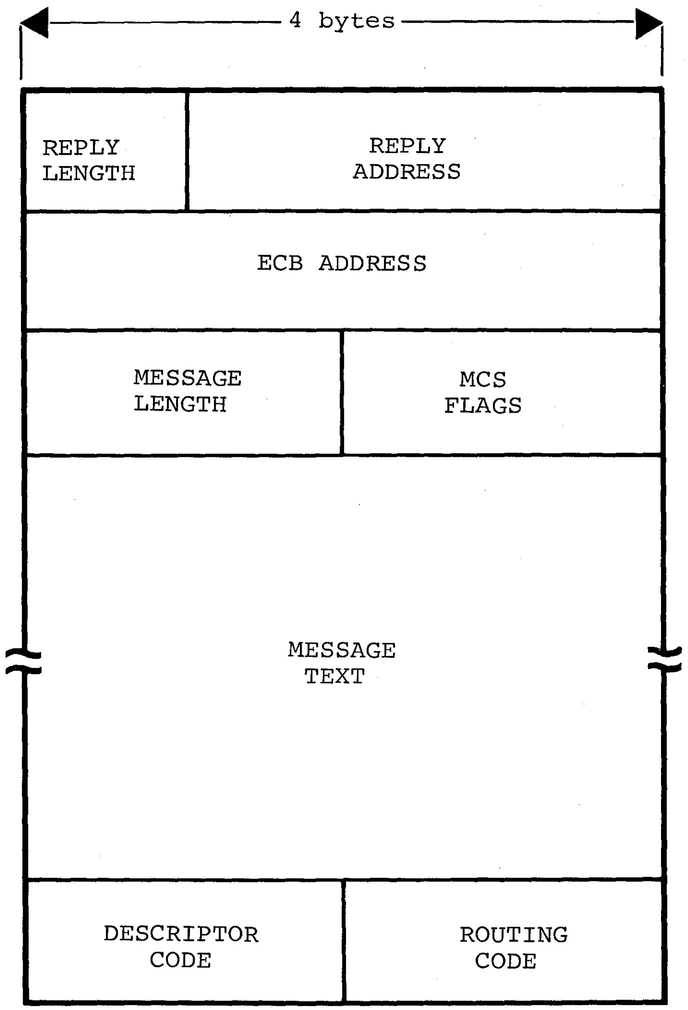


Figure 61. WTOR Parameter List

```
AR       CSECT
         USING *,12
         SAVE  (14,12)
         LR    12,15
         LTR   0,0                      IS IT A WTOR MESSAGE
         BZ    NOTFOUND                 NO, BRANCH
         CLC   16(2,1),AR21             IS IT A HALT MESSAGE
         BNE   WTOR2                    NO, BRANCH
         CLC   38(5,1),AR1234           IS IT A HALT IAR=01234 MESSAGE
         BE    STARTJ                   YES, BRANCH
         CLC   38(5,1),AR5602           IS IT A HALT IAR=05602 MESSAGE
         BE    SENSE                    YES, BRANCH
         CLC   50(5,1),AR111            IS IT A HALT AAR=111 MESSAGE
         BNE   NOTFOUND                 NO, BRANCH
         MVC   0(3,5),AREOJ             REPLY IS 'EOJ'
         B     FIND
STARTJ   MVC   0(5,5),ARSTART           REPLY IS 'START'
         B     FIND
SENSE    MVC   0(16,5),ARSENSE          REPLY IS 'TN SENSE=E+START'
FIND     L     5,4(1)                   MESSAGE REPLIED TO
         POST  (5)
         B     MESFIND
WTOR2    CLC   16(2,1),AR01             IS IT OPERATOR SERVICES AVAILABLE MESSAGE
         BNE   NOTFOUND                 NO, BRANCH
         LM    4,5,0(1)
         STM   4,5,ARREP01
MESFIND  SR    15,15                    RETURN CODE=0
         B     RETURN                   BRANCH
NOTFOUND LA    15,4                     RETURN CODE=4
RETURN   RETURN (14,12),RC=(15)         RETURN
AR21     DC    C'21'
ARREP01  DC    2F'0'
AR1234   DC    C'01234'
AR5602   DC    C'05602'
AREOJ    DC    C'EOJ'
ARSTART  DC    C'START'
ARSENSE  DC    C'TN SENSE=E+START'
AR01     DC    C'01'
AR111    DC    C'00111'
         END
```

Figure 62. Sample Routine Using Automatic Reply

This appendix lists, explains, and gives appropriate responses to messages printed by the emulator, the tape formatting programs, and the disk formatting program. The messages are presented in alphanumeric order:

IIQ000x - IIQ299x	Emulator
IIQ300x - IIQ399x	Tape Formatting Programs
IIQ400x - IIQ499x	Disk Formatting Program

Messages in this appendix are not in Messages and Codes, GC28-6631. This appendix may be removed and placed in Messages and Codes.

When there are several operator or programmer actions, preceded by bullets, you should select the most appropriate one. When actions are numbered, you should respond in numeric order, stopping after the action that gives the desired result. When actions are lettered, you should determine which explanation caused the message to be issued, then select the lettered action that matches the explanation. For example, action C is for explanation C.

EMULATOR MESSAGES

IIQ000I jobname MESSAGE NUMBER nnn NOT FOUND

Explanation: The emulator module IIQMW cannot find message nnn in the message text dictionary (IIQMT). nnn are the fourth, fifth, and sixth characters of the message number.

System Response: The 1400 program is ended. The emulator produces a System/370 dump if a SYSEMOUT DD statement was included with the job.

Operator Action: None.

Programmer Action: Ensure that the 1400 program or an operator command has not incorrectly altered main storage. If the message number (nnn) is not one of those listed in this appendix or if the error persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ001I jobname OPERATOR SERVICES AVAILABLE

Explanation: This message is printed (1) when the 1400 program begins and (2) each time the operator types an emulator command.

System Response: The operating system continues processing until the operator types a reply. The reply is then processed, and the message is issued again. If the operator does not respond, the emulator continues.

Operator Action: To communicate with the emulator, type any valid emulator command. If the command is longer than one line, type a hyphen at the end of the line. Message IIQ034A will be printed; continue the command string by responding to message IIQ034A. Up to 485 characters can be typed in the command string.

Note: If there are other outstanding messages, respond to them before you respond to this one. When there are several 1400 programs being executed in a single OS job step, replying to a WTOR message during the execution of a 1400 program for which OPSERV is not specified has no effect.

IIQ002A jobname I

Explanation: A Read from Console instruction is being executed in the 1400 program and inquiry from the console can be processed.

System Response: Control is given to the operating system, and the emulator task remains in the wait state until the operator acts.

Operator Action: Type the data or the 1400 command that the 1400 program is expecting. No emulator commands can be typed in response to this message. If the data or 1400 command exceeds one line, or if you want to split the input into two or more lines, end each line you wish to continue with a hyphen.

If the data or 1400 command is continued, the emulator will reissue this message. If you have no data to enter, give a null response (REPLY id,'').

IIQ003I jobname P variable string

Explanation: This message is the data or text that is written when a Write on Console instruction is executed in the 1400 program. If the data or text exceeds 50 characters, the message is repeated as often as needed to print the entire string. A hyphen in position 51 indicates that the data or text exceeds 50 characters.

System Response: The emulator prints the data or text from 1400 storage until a groupmark wordmark is encountered. Processing then continues at the next sequential 1400 instruction.

Operator Action:

- None.

- To stop printing text, type any emulator command as a reply to message IIQ001I jobname OPERATOR SERVICES AVAILABLE, then reset the IAR or alter 1400 storage so that the Write on Console instruction encounters a groupmark wordmark. Any portion of the data or text that was queued to the output queue of the console before you replied to message IIQ001I is printed.

IIQ005I jobname CONTROL STATEMENT REQUESTED UNSUPPORTED FEATURE

Explanation:

A. The block size specified for a 1400 format tape was greater than 32,755 bytes.

B. The mode for a 1400 format tape was not "set density, data converter off, and translator on." The mode is set with the DCB parameters DEN and TRTCH.

C. The disk was formatted with a record size larger than the DISK control statement requested; that is, the disk was formatted for mixed mode (track-record and sector), but the DISK control statement specified MODE=SECTOR.

D. A DD statement was encountered for a tape in spanned format and the emulator was generated for tapes with records in 1400 format only, or vice versa.

System Response: The emulator requests a System/370 dump if there is a SYSEMOUT DD statement in the JCL. The 1400 program stops.

Programmer Action:

A. Correct the block size in the DD statement.

B. Correct the DCB parameter in the DD statement.

C. Correct the MODE parameter in the DISK emulator control statement so that it specifies the mode in which the disk is formatted.

D. Correct the RECFM parameter in the DD statement, or use an emulator that emulates the record format.

E. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ011D jobname INPUT EXCEEDS BUFFER SPACE, nnn BYTES, REENTER

Explanation: The 485-byte input buffer is not large enough for the line of the command string just typed. nnn bytes is the number of characters that can be typed in the line.

System Response: The emulator ignores the command string and gives control to OS; the emulator task is placed in the wait state until the operator acts.

Operator Action:

A. Retype the entire command string.

B. Type any valid emulator command. START causes the emulator to resume processing at the next sequential 1400 instruction.

C. Request a dump of the emulator partition when the command string has been typed correctly but the problem persists.

Programmer Action: Analyze and fix the error. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump before calling IBM for programming help.

IIQ012D jobname ccccccccc IS INVALID COMMAND OR KEYWORD PARAMETER

Explanation: The command or keyword identified by ccccccccc:

- Is misspelled.
- Is not valid.
- Is a command that requires a keyword.
- Is a keyword that requires an operand.
- Has an invalid delimiter in the command string in or near the characters ccccccccc.

System Response: When a command or control statement has independent keywords, all keywords prior to the error (ccccccccc) have been processed. When a command or control statement has a keyword that depends on another, no keywords have been processed. In either case, control is given to OS and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- To ignore the erroneous command or keyword, type any emulator command. If you type START, the emulator resumes processing at the next sequential 1400 instruction.
- To correct the command or keyword, retype the command string or control statement. The following commands and control statements must be typed in their entirety:

Commands	Control Statements
ALTER	CCTL
CLEAR	DISK
CONVERT	TAPE
DUMP	UR
EOJ	
START	

The following commands may be completed by typing the erroneous keyword and all subsequent keywords:

DISPLAY
SET
TF
TN

Only control statements that can be typed at the console can be corrected therefrom.

- Request a dump of the emulator partition using the OS CANCEL command when the command string has been typed correctly but the problem persists. Do not use the emulator DUMP command; it does not dump the routine in error.

Programmer Action: Analyze and fix the error. If the problem persists, ensure that a SYSUDUMP DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, the listings of Stages I and II of emulator generation, and a dump of the emulator partition before calling IBM for programming help.

IIQ013D jobname INVALID OPERAND=operand

Explanation: The operand field of the emulator command just typed is not an accepted operand. 'operand' identifies the unaccepted operand.

System Response: When a command has independent keywords, all keywords prior to the error have been processed. When a command has a keyword that depends on another, no keywords have been processed. In either case, control is given to OS and the emulator task is placed in the wait state until the operator acts. The rest of the command is not executed.

Operator Action:

- To ignore the erroneous command or keyword, type any emulator command. If you type START, the rest of the command is ignored and the emulator resumes processing at the next sequential 1400 instruction.

- To correct the command or keyword operand, retype the command or control statement. The following commands and control statements must be typed in their entirety:

Commands	Control Statements
ALTER	CCTL
CLEAR	DISK
CONVERT	TAPE
DUMP	UR
EOJ	
START	

Errors in keyword operands (on the right side of the equal sign) are treated differently from errors in the keywords themselves. If a command was typed incorrectly and a default value was overridden, you will not change the value when correcting the command if you assume the default value: for example, if you typed:

```
TAPE  UNIT=12,DILCNT=40,
TYPEFLE=OUTIN
```

where OUTIN is coded instead of OUTPUT. As you are retyping the correct command, if you omit the DILCNT parameter, the value will remain unchanged at 40. You must type the parameter to return the value to 25:

```
TAPE  UNIT=12,DILCNT=25,
TYPEFLE=OUTPUT
```

The following commands may be completed by correctly typing the keyword operand in error and typing all subsequent keywords:

DISPLAY
SET
TF
TN

Only control statements that can be typed at the console can be corrected from it.

- Request a dump of the emulator partition using the OS CANCEL command when the command string has been typed correctly but the problem persists. Do not use the emulator DUMP command; it does not dump the routine in error.

Programmer Action: Analyze and fix the error. If the problem persists, ensure that a SYSUDUMP DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ014D jobname KEYWORD PARAMETERS INCOMPLETE

Explanation: One or more keywords are missing from an ALTER or DUMP command.

System Response: Control is given to OS, and the emulator task is placed in the wait state until the operator acts. The command is not executed.

Operator Action:

- To ignore the erroneous command, type any emulator command. If you type START, the emulator resumes processing at the next sequential 1400 instruction.

- To correct the command, retype it completely.

- Request a dump of the emulator partition using the OS CANCEL command when the command string has been typed correctly but the problem persists. Do not use the emulator DUMP command; it does not dump the routine in error.

Programmer Action: Analyze and fix the error. If the problem persists, ensure that a SYSUDUMP DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ021D jobname HALT IAR=addr AAR=addr BAR=addr

Explanation: A Halt instruction has been encountered in the 1400 program, but the 1400 program is not at end of job. The contents of the IAR, AAR, and BAR are displayed in the addr fields. If the address in the AAR or BAR is invalid or meaningless, the respective address field of this message is filled with blanks.

System Response: Control is given to OS, and the emulator task is placed in the wait state until the operator acts.

Operator Action: Type the information or perform the action for a 1400 halt. Any emulator command can be typed; if you type START, the emulator resumes processing at the next sequential 1400 instruction.

IIQ022D jobname HALT/BRANCH IAR=addr AAR=addr BAR=addr

Explanation: A Halt and Branch instruction has been encountered in the 1400 program, but the 1400 program is not at end of job. The contents of the IAR, AAR, and BAR are displayed in the addr fields. If the address in the AAR or BAR is invalid or meaningless, the respective address field of this message is filled with blanks.

System Responce: Control is given to OS, and the emulator task is placed in the wait state until the operator acts.

Operator Action: Type the information or perform the action for 1400 halt and branch. Any emulator command can be typed; if you type START, the emulator resumes processing at the branch address. START RESET causes the emulator to reset all indicators and resume processing at the next sequential 1400 instruction.

IIQ023I jobname EOJ HALT IAR=addr AAR=addr BAR=addr

Explanation: A Halt instruction or a Halt and Branch instruction has been encountered and it satisfies the end-of-job condition. The contents of the IAR, AAR, and BAR are displayed in the addr fields.

System Response: The 1400 program stops.

Operator Action: None.

IIQ024D jobname ADDRESS=addr IS INVALID IN COMMAND

Explanation: There is an invalid address as the operand of an ADDR= or XADDR= keyword in an emulator command.

System Response: Control is given to OS, and the emulator task is placed in the wait state until the operator acts.

Operator Action: Retype the command with the correct address, or type START to begin processing at the next sequential instruction. If the address has been typed correctly and the problem persists, dump the emulator partition using the OS CANCEL command. Do not use the emulator DUMP command; it does not dump the routine in error.

Programmer Action: Analyze and fix the error. If the problem persists, ensure that a SYSUDUMP DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump before calling IBM for programming help.

IIQ031I jobname BUFFER TOO SMALL FOR PHYSICAL RECORD FILE=name

Explanation:

A. A record in 1400 format is larger than the System/370 buffer specified in the BLKSIZE parameter of the DD statement.

B. A tape file with records in 1400 format is not positioned correctly, or there is a bad record on the tape, or both.

C. The first I/O instruction for an output data set on tape is a Read Tapemark instruction and no tapemark was found.

D. The records are in 1400 format, but the RECFM parameter of the DD statement is not RECFM=U.

E. The BDW indicates a record length greater than that specified in the BLKSIZE parameter of the DD statement.

FILF=name identifies the 1400 device by its channel and unit number.

System Response: For explanations D and E, the emulator requests a System/370 dump if there is a SYSEMOUT DD statement in the JCL. The emulator stops. For explanations A through C, the emulator turns on the wrong-length record indicator and issues message IIQ061D.

Operator Action: Reply to message IIQ061D. If the record is a checkpoint record, type START; otherwise, type EOJ. The 1400 program may handle the error that caused this message (IIQ031I) to be issued, and the programmer who submitted the job may provide operator responses for specific situations.

Programmer Action:

A. Ensure that the BLKSIZE parameter of the DCB operand specifies a block size large enough to hold the record, or preprocess the tape to reduce the physical record size.

B. Ensure that the records on the tape are good and then rerun the job.

C. Change the TAPE emulator control statement to include TYPEFLE=OUTPUT. The emulator will write a tapemark on the tape before the 1400 program uses it for output.

D. Correct the RECFM parameter to RECFM=U.

E. Ensure that the BLKSIZE parameter of the DCB operand specifies a block size large enough to hold the record.

F. If the problem persists, ensure that the SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump before calling IBM for programming help.

IIQ032I jobname nnn ERROR OCCURRED DURING EXECUTION OF IIQEIOCS

Explanation: An emulator error occurred. The first three characters, nnn, indicate the cause of the error.

nnn Explanation

001 - The error code passed to IIQBMOUT or the input code passed to IIQEIOCS was invalid.

002 - EMPTR or EMBSR in IIQEIOCS could not backspace over a record just written.

003 - EMPUT in IIQEIOCS detected that the length of the last record in the segment descriptor word (SDW) is smaller than 4.

004 - A record could not be written in the requested parity.

005 - While executing EMREAD, EMPUT, or EMBSR in IIQEIOCS, the record being read or written is larger than the buffer.

006 - A Block Descriptor Word (BDW) was greater than the block size on the output file during execution of EMFORCE in IIQEIOCS.

007 - EMBSR in IIQEIOCS backspaced records to the beginning of the file, but it did not find the beginning of the record (spanned format).

008 - EMBSF in IIQEIOCS backspaced to what should have been but was not the last tapemark record (spanned format).

009 - The error code passed to the IIQBMOUT routine was invalid.

010 - An error occurred in the console portion of the NDURGET routine. The emulator could not execute a Read from Console instruction.

011 - EMREAD in IIQEIOCS got a data check while reading a tapemark record.

System Response: The emulator produces a System/370 dump if a SYSEMOUT DD statement was included with the job. The 1400 program stops.

Operator Action: None.

Programmer Action:

- If nnn is 002, 003, 004, 005, 006, 007, 008, or 011, the tape may have been preprocessed incorrectly. Preprocess the tape again if using preprocessed tape. Ensure that the 1400 program does not issue an invalid Backspace Record instruction, and ensure that the tape or disk drives are working properly, then rerun the job.
- If nnn is 001, 009, or 010, ensure that the operator has not improperly altered main storage, and rerun the job.
- If the problem persists, make sure that MSGLEVEL=(1,1) was specified in the JOB statement, that a SYSEMOUT DD statement was included for the job step, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ033I jobname stepname, unit addr, device type, ddname, operation attempted, error description, track addr or relative block number, access method

Explanation: An I/O error occurred while processing an emulator data set. Error analysis information displayed is:

stepname - the step name of the emulator task in which the error occurred.

unit addr - the unit address of the device on which the error occurred.

device type - the type of device on which the data set is located.

ddname - the name of the DD statement that defines the device.

operation attempted - the type of I/O operation being attempted.

error description - the SYNAD 14-character description of the error.

track addr - if the unit is a disk device, the track address at which the error occurred.

relative block number - if the unit is a tape unit, the relative number of the block in which the error occurred.

access method - the access method used on the data set.

System Response: The emulator requests a System/370 dump if there is a SYSEMOUT DD statement in the JCL. If the DD statement is for a tape or unit record device, the emulator stops. If the DD statement is for a disk device, this message is printed, appropriate disk error indicators are set, and the next 1400 instruction is fetched and emulated.

Programmer Action: Analyze and correct the error, then resubmit the job. For disk errors, the job may not have to be rerun when the 1400 program handles the error. If the problem persists, make sure that MSGLEVEL=(1,1) was specified in the JOB statement, that a SYSEMOUT DD statement was included for the job step, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ034A jobname CONTINUE INPUT OF COMMAND STRING

Explanation: A previous operator command requested that the command be continued on another line. This message is issued when a hyphen is found at the end of a reply.

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action: Type the remaining command-string input. If no input remains, type a null reply (REPLY id,'').

IIQ042D jobname INVALID OP-CODE

Explanation: The emulator encountered an operation code in the 1400 program that is not emulated.

System Response: Control is given to OS , and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Type DUMP to dump the contents of the emulator partition and EOJ to end the 1400 job.

- Use the DISPLAY and ALTER commands to correct the error from the console. The IAR value is unpredictable, and it should be set to the desired value before typing START to continue program execution.

- Type START to begin processing at the next sequential 1400 instruction if the invalid instruction can be ignored. The IAR value is unpredictable, and it should be set to the desired value before typing START.

Programmer Action: Correct the error and submit the job again.

If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log , the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ043D jobname INVALID INSTRUCTION FORMAT

Explanation: The character pointed to by the IAR does not have a wordmark, and it may be a valid operation code.

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Use the DUMP command to dump the contents of the emulator partition and the EOJ command to end the 1400 job.

- Use the DISPLAY and ALTER commands to correct the error from the console. The IAR value is unpredictable, and it should be set to the desired value before typing START to continue program execution.

- Use the START command to begin processing at the next sequential 1400 instruction if the invalid instruction can be ignored. The IAR value is unpredictable, and it should be set to the desired value before typing START.

Programmer Action: Correct the error and submit the job again.

If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ045D jobname INVALID I/O INSTRUCTION

Explanation: The emulator encountered a 1400 I/O instruction that was invalid because of:

- An invalid A-field

- An invalid d-modifier

- A non-emulated I/O device

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Use the DISPLAY and ALTER commands to correct the error from the console.

  Note: The IAR points to the next 1400 instruction to be fetched.

- When you cannot correct the error from the console, use the DUMP command to dump the contents of the emulator partition and the EOJ command to end the 1400 job.

- If the invalid instruction can be ignored, use the START command to begin processing at the next sequential 1400 instruction.

Programmer Action: Correct the error and submit the job again.

If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ046D jobname INVALID ADDRESS

Explanation: The emulator encountered an invalid address in the I-address, A-address, or B-address register, which was caused by:

- An address wraparound

- An invalid character in 1400 storage

System Response: Control is given to OS, and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Use the DISPLAY and ALTER commands to correct the error from the console. The contents of the registers are unpredictable, and they should be set to the desired values before typing START to continue execution of the 1400 program.

- Use the DUMP command to dump the contents of the emulator partition and the EOJ command to end the 1400 job.

- If the invalid address can be ignored, use the START command to continue processing at the next sequential 1400 instruction. If the invalid address is in the IAR, set the IAR to the desired value before typing START.

Programmer Action: Correct the error and submit the job again.

If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ052D jobname FILENAME NOT GIVEN FOR cccc

Explanation: A ddname was not found for the 1400 device indicated by cccc, where cccc can be any of the following:

- RDR - the card reader

- PTR - the printer

- PNCH - the card punch

- TPcu - tape unit u on channel c

- DISK - a 1405 or 1301 disk

- DKcm - module m of a 1311 on channel c

If message IIQ054I precedes this message, a ddname was given in the emulator control statement, but the ddname may have been incorrect.

System Response:

- If the operator types START, the emulator determines whether a ddname has been entered. If no ddname was entered, this message is issued again. If a valid ddname is found, the 1400 program continues at the point of interruption.

- If the IAR is set before typing START, or if the reply is START ADDR=nnnnn, where nnnnn is the starting address, the emulator will continue the 1400 program at nnnnn.

Operator Action:

- Type DISPLAY CONFIG to display the I/O devices and their corresponding ddnames.

- Enter an acceptable ddname using the NAME operand of the TAPE, DISK, or UR emulator control statement.

  If you enter the TAPE or UR statement, you must specify all operands for which a default is not desired, since the command you enter overlays the statement read with the JCL.

- Type START if I/O operations on the device is not required. The 1400 program begins at the next sequential instruction.

- If you cannot identify an acceptable ddname, use the DUMP command to dump the emulator partition and the EOJ command to end the 1400 job.

Programmer Action: Determine whether the NAME operand of the emulator control statement should be specified. If it cannot be, give the operator an acceptable ddname to be typed when this message is received.

If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ053I jobname CORE UNAVAILABLE FOR DCB
IIQ053I jobname CORE UNAVAILABLE FOR BUFFER. FILE=ddname

Explanation:

A. The emulator tried to get main storage for a DCB and there was not enough.

B. The emulator tried to get main storage for buffers and there was not enough. The ddname identifies the DD statement that describes the data set.

C. The emulator encountered a DD statement for a reader, printer, or punch that requested a logical record length (LRECL) greater than 256 bytes.

System Response: The emulator program requests a System/370 dump if there is a SYSEMOUT DD statement in the JCL. The 1400 program stops.

Programmer Action:

A. If storage was not available for the DCB, increase the amount of main storage requested in the REGION parameter of the JOB or EXEC statement (MVT), or obtain a larger MFT partition.

B. If storage was not available for a buffer, first check the value specified in the BLKSIZE parameter of the DCB operand (for tape and disk only). If the BLKSIZE value is too large, correct it. If it is not, increase the amount of main storage requested in the REGION parameter of the JOB or EXEC statement (MVT), or obtain a larger MFT partition.

C. If the LRECL parameter of the DD statement is wrong, correct it and resubmit the job.

D. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ054I jobname FILE=ddname COULD NOT BE OPENED

Explanation:

- The emulator could not open the file identified by ddname because there was no DD statement.

- A ddname in the TAPE, DISK, or UR emulator control statement was misspelled or otherwise typed incorrectly.

System Response: The emulator issues message IIQ052D.

Operator Action: Reply to message IIQ052D by typing the correct ddname in a TAPE, DISK, or UR emulator control statement.

Programmer Action: Determine whether the NAME operand of the emulator control statement should be specified. If it cannot be, give the operator an acceptable ddname to be typed when this message is received with message IIQ052D.

IIQ055I jobname SYSEMOUT DCB NOT OPEN

Explanation: An operator command or response on the SYSEMOUT data set was unsuccessfully recorded because the data set DCB was not opened or was opened incorrectly.

System Response: The emulator issues message IIQ061D.

Operator Action: Reply to message IIQ061D.

Programmer Action: If the SYSEMOUT DD statement was not included in the JCL, include one and submit the job again. If the statement is included, check it. If the problem persists, ensure that a SYSABEND DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

Request the dump using the OS CANCEL command. Do not use the emulator DUMP command; it does not dump the routine in error.

IIQ061D jobname EMULATOR WAITING

Explanation: The emulator is waiting for a response from the operator. There are many conditions during emulation that can cause this message to be issued. It is issued whenever the emulator cannot proceed without some operator action.

System Response: The emulator waits for a response to this and possibly other WTOR messages. Control is given to OS and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Type any command that requests an emulator service, such as DISPLAY.
- Type any command that satisfies the conditions the emulator is waiting for. EOJ may have to be typed if the emulator is waiting because of an unrecoverable error. (You should dump the emulator partition before typing EOJ.)
- If you can find no reason for an error in the emulator job, dump the contents of the emulator partition and cancel the job.

Type START if no further emulator services are needed and if all conditions the emulator was waiting for are satisfied. This causes processing to begin at the next sequential 1400 instruction.

IIQ062I jobname HEX ADDRESS=nnnnnn

Explanation: This message is issued in response to the emulator command CONVERT ADDR=nnnnn. HEX ADDRESS=nnnnnn is a 1- to 6-digit hexadecimal address that is the System/370 main-storage location of the 1400 storage address specified in the CONVERT command.

System Response: The emulator issues message IIQ061D.

Operator Action: Reply to message IIQ061D.

IIQ063I jobname NEXT 1400 ADDRESS=nnnnn

Explanation: This message enables the operator to display contiguous areas of main storage. It is issued after message IIQ066I when the operator typed DISPLAY ADDR.

The number of characters displayed in message IIQ066I varies, since two EBCDIC characters indicate a 1400 character with a wordmark. nnnnn indicates the next 1400 storage address if more storage needs to be displayed.

System Response: The emulator issues message IIQ061D.

Operator Action: Reply to message IIQ061D.

IIQ065I jobname 1400 DEVICE cccc NOT ASSIGNED

Explanation: A device specified in an operator request was unassigned. An example of an operator request is REPLY id,'DISPLAY TAPE=12'. cccc is TAPE, DISK, or UR.

System Response: The emulator issues message IIQ061D.

Operator Action: Reply to message IIQ061D.

IIQ066I jobname text

Explanation: This message displays the COMMENT emulator control statement and the response to the DISPLAY emulator command. It consists of variable text and uses the following abbreviations for 1400 devices:

- RDR - the card reader
- PTR - the printer
- PNCH - the card punch
- CNSL - the console
- TPcu - the channel (c) and unit (u) addresses of a tape unit
- DKcm - the channel (c) and module (m) addresses of a disk drive

If a DISPLAY REG command is typed and the value in one of the registers is invalid or meaningless, that portion of the text field is filled with blanks.

If a DISPLAY ADDR or DISPLAY XADDR command was typed, the address is repeated and then the storage is displayed between single quotes. The data is in load mode; both word separators and wordmarks appear as underscores.

System Response: The emulator issues message IIQ061D.

If this message is in response to the DISPLAY ADDR command, message IIQ063I is issued before message IIQ061D.

Operator Action: Reply to message IIQ061D.

IIQ071D jobname END OF 1400 STORAGE REACHED

Explanation: The emulator encountered the end of 1400 storage during:

- A disk or tape I/O operation
- A unit record output operation

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Use the DISPLAY and ALTER commands to correct the error from the console.
- Use the DUMP command to dump the contents of the emulator partition and the EOJ command to end the 1400 job.
- If the invalid instruction can be ignored, use the START command to start processing at the next sequential 1400 instruction.
- Type another emulator command.

Programmer Action: Correct the I/O instruction and submit the job again. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ072D jobname CARD INPUT EXHAUSTED

Explanation: The emulator has encountered a read operation in the 1400 program, but there are no more cards to be read. The)LC card may not have been placed before the last data card, or a blank card was not placed after the)LC card of a 1440 program.

System Response: The emulator turns on the "last card" indicator. Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Type START to begin processing at the next sequential 1400 instruction. The 1400 job should end at normal end of job.
- If you are emulating a 1440 program and there is a)LC card, type START to clear the reader station. Most 1440 jobs will end at normal end of job. If you receive this message again, type START a second time. If the 1440

program still does not end at normal end of job, dump the emulator partition and type EOJ.

- Type another emulator command.

Programmer Action: Ensure that the)LC card is placed before the last data card and submit the job again.

Some 1440 programs issue one or more read commands to eject the last data card while closing the 1442 file. Before you run the job again, insert one blank card after the)LC card if the program does not read ahead, and two blank card after the)LC card if it does.

If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ073D jobname INVALID 1400 DATA

Explanation: The emulator detected invalid BCD data on a read operation from the card reader or the console.

System Response: The emulator reads the card or line, but changes all invalid characters to asterisks. It then gives control to the operating system. The emulator task is placed in a wait state until the operator acts.

Operator Action:

- Ensure that valid data was entered from the console. To replace an asterisk with a valid BCD character, use the DISPLAY and ALTER commands. The data is located in the read buffer of 1400 storage (locations 1 to 80 for 1401 and 1460) and has been translated to internal code.

- To ignore the error, type START to begin processing at the next sequential 1400 instruction.

- Type another emulator command.

Programmer Action: If the invalid character was on a data card and the operator ended your job, correct the invalid characters and resubmit the job. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ074D jobname CARRIAGE CONTROL TAPE IN ERROR

Explanation: The emulator encountered a carriage control instruction in the 1400 program that specified a skip to a channel number not in the carriage control tape image defined in the CCTL emulator control statement.

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action: If the carriage control tape image was not typed correctly, retype the CCTL emulator control statement. Other actions are:

- End the emulator job.

- Type another emulator command.

- Type START, which causes the carriage control instruction to be skipped and processing to begin with the next sequential 1400 instruction.

Programmer Action: Ensure that the necessary carriage control information is in the carriage control statement and resubmit the job. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ075I jobname INVALID DATA IN 1400 STORAGE

Explanation: A character string ends with a wordmark or the emulator detected an invalid character (bit 7 is 1) in 1400 storage. The invalid character may be the result of a machine operator error.

System Response: If this message results from an emulator DUMP command, the dump is ended after printing the line preceding the one containing the invalid character, then message IIQ061D is issued. Otherwise, the emulator requests a System/370 dump if there is a SYSEMOUT DD statement in the JCL. The 1400 program stops.

Programmer Action: Eliminate any causes of invalid characters and submit the job again. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ081I jobname EOM-OPERATION TERMINATED

Explanation:

A. The emulator ended an operator command because the end of 1400 storage was reached while executing the command.

B. The end of 1400 storage was reached while the emulator was executing a 1400 Read a Card instruction (1440 only).

System Response:

A. The emulator issues message IIQ061D.

B. When the end of 1400 storage is sensed, the emulator moves the remainder of the data into the beginning of 1400 storage (wraparound). After all data is moved, processing continues at the next sequential 1400 instruction.

Operator Action:

A. Reply to message IIQ061D.

B. None.

IIQ083D jobname 1400 DEVICE NOT ASSIGNED FOR cccc

Explanation: A 1400 I/O instruction has been detected for a 1400 device not assigned by an emulator control statement. cccc indicates the 1400 device not assigned:

- PTR - printer
- RDR - reader
- PNCH - punch
- CNSL - console
- TPcu - tape where 'cu' is the channel and unit number
- DISK - 1405 or 1301 disk
- DKcm - 1311 disk where 'cm' is the channel and module number

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Programmer Action: Include the correct control statement and resubmit the job. If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming help.

Operator Action: Type the missing control statement and the START command.

IIQ084I jobname cccc DEVICES NOT SUPPORTED

Explanation: The operator used one or more I/O devices which are not in the emulator. cccc indicates the 1400 device not generated.

System Response: The emulator issues message IIQ061D.

Operator Action: Reply to message IIQ061D.

Programmer Action: Ensure that the generated emulator includes all necessary devices. If the problem persists, ensure that a SYSUDUMP DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, a dump of the emulator partition, and the Stage I and II listings of the emulator generation before calling IBM for program help.

IIQ086I jobname ERROR IN OPERATOR SERVICES

Explanation: The emulator services control routine (module IIQOA) received an invalid input parameter from another emulator routine. (Control byte CMOSCCTL was improperly set.)

System Response: The emulator produces a System/370 dump if a SYSEMOUT DD statement was included with the job. The 1400 program stops.

Operator Action: Attach a copy of the console listings to the dump.

Programmer Action: Ensure that the operator has not improperly altered main storage. If the problem persists, ensure that a SYSEMOUT DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ092D jobname INVALID DEVICE REASSIGNMENT

Explanation: A device assignment statement specified one of the following:

- An invalid unit number
- An channel or device that is not accepted by the emulator
- A device that was not assigned at initialization

System Response: Control is given to the operating system, and the emulator task is placed in the wait state until the operator acts.

Operator Action:

- Retype the entire emulator control statement with the correct device assignment.
- Request a dump of the emulator partition when the control statement has been typed correctly, but the problem persists.
- Type any valid emulator command. START will cause the emulator to resume processing at the next sequential 1400 instruction.

Programmer Action: Ensure that all devices required for the job are assigned by emulator control statements. If a device is to be reassigned during program execution, be sure to give the operator the correct information. If the problem persists, ensure that a SYSUDUMP DD statement was included for the job step, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, the listings of Stages I and II emulator generation, and a dump of the emulator partition before calling IBM for programming help.

IIQ093I jobname EMULATOR CONTROL STATEMENT ERROR

Explanation: If the message is not followed by a control statement printed on the SYSEMOUT data set, the error was caused by:

- SYSEMCTL not used as the input data set ddname
- An I/O error while reading the input data set
- Not enough space to open the SYSEMCTL DCB successfully

If the message is followed by a control statement or command printed on the SYSEMOUT data set, the error was caused by:

- An invalid parameter in the control statement.
- The control statement requested a 1400 device that is not emulated. For example, there is no 1400 device at the channel and unit position requested.
- The device assigned by the control statement has already been assigned by a previous control statement.
- An emulator command that cannot be typed in the input stream.
- An end-of-file condition was detected in the SYSEMCTL data set before the LOAD control statement was found.
- A missing LOAD control statement caused the 1400 program or data to be flagged as an invalid emulator control statement.

- Insufficient buffer space was available for the control statement.

- NAME=SYSEMCTL is coded with STACKER=parameter.

All emulator control statements and commands in error are printed on the SYSEMOUT data set, but only the message and the first statement or command in error is printed on the console.

System Response: The emulator task is ended after all control statements have been read and analyzed.

Programmer Action: Correct the control statement error and resubmit the job. Ensure that the control statement does not exceed 485 bytes. If the problem persists, punch a DUMP emulator command on a card as if it were an emulator control statement (first column blank), insert it after the control statement in error, and run the program again. Ensure that a SYSEMOUT DD statement was included, that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the Stage I and II emulator generation listings, the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

IIQ094I jobname INVALID RECFM - DDNAME=name

Explanation:

- The record format specified was RECFM=VS, but the emulator found that the data was VBS. The name of the DD statement in error is identified by DDNAME=name.

- A tape file is being emulated and the RECFM parameter specified a record format other than U, VS, or VBS. The name of the DD statement in error is identified by DDNAME=name.

System Response: The emulator produces a System/370 dump if a SYSEMOUT DD statement was included with the job. The 1400 program stops.

Programmer Action: The operator may have mounted the wrong tape or the DD statement may have the RECFM parameter coded incorrectly. Analyze and correct the error, then resubmit the job. If the problem persists, make sure that MSGLEVEL=(1,1) was specified in the JOB statement, that a SYSEMOUT DD statement was included for the job step, and that you have the master console log, the printer listing, and a dump of the emulator partition before calling IBM for programming help.

TAPE FORMATTING PROGRAM MESSAGES

IIQ301D PREPROCESSOR READY-REPLY nnn (NO. OF TM),ALL OR EOJ (jobname)

Explanation: The tape preprocessor program is ready to format the data on a 1400 tape.

System Response: The preprocessor is placed in the halt state until the operator acts.

Operator Action:

- If you know the number of tapemarks on the tape to be formatted, type that number in the reply to this message. The preprocessor formats the tape until that number of tapemarks has been read.
- If you do not know the number of tapemarks on the tape to be formatted, type ALL. The preprocessor reads all readable data on the tape and stops when the tape runs off the reel, when a permanent I/O error occurs, or when there is a record that is too long for the buffer.
- If there are no tapes to format, type EOJ.

IIQ302D PREPROCESSOR INPUT I/O ERROR-REPLY EOV OR SKIP (jobname)

Explanation: A permanent I/O error was encountered by the tape preprocessor program. The operating system could not successfully read the next record in the input file. All error recovery procedures were performed, the I/O operation was ignored, control was returned to the preprocessor, and this message was issued. This may not be an abnormal condition (see message IIQ301A).

System Response: The preprocessor is placed in the wait state until the operator acts.

Operator Action: If you entered a specific number of tapemarks when message IIQ301D was issued, a permanent I/O error occurred in the input file:

- Type EOV when you are sure that all records have been correctly formatted. The input and output files are closed and an end-of-volume summary is written.
- Type SKIP to ignore the record with the I/O error. Processing will continue with the next record in the input file.

If the reply to message IIQ301A was ALL, there may be a permanent error in one of the records in the input file, or the entire file has been processed and the preprocessor is reading garbage on the end of the tape:

- Type EOV if the entire file has been processed. The program will close the input and output files and issue an end-of-volume summary.
- Type SKIP to bypass the record. The program will read the next record. If a permanent I/O error is encountered while reading a subsequent record, message IIQ302D is issued again.

Programmer Action: Analyze the end-of-volume summary (message IIQ305I) and determine whether all records have been formatted. If they have not, ask the operator to clean tape drives, check for bad tapes, etc., and run the job again. If the error persists, make sure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log, the printer listing, and the tape files before calling IBM for programming help.

IIQ303D PREPROCESSOR INPUT RECORD TOO LONG-REPLY EOV OR SKIP (jobname)

Explanation: An input record was read that was longer than the input block size specified in the PARM parameter of the EXEC statement (or the PARM default value of 10,000 bytes). All error recovery procedures were performed. The I/O operation was not completed. The name on the JOB statement is in (jobname). In some cases, this message is issued when the preprocessor reads past the end of the input file.

System Response: The preprocessor is placed in the wait state until the operator acts.

Operator Action: You have entered a specific number of tapemarks in response to message IIQ301D. An input record was too long.

- Type EOV if the record must be read.

- Type SKIP to ignore the record with the I/O error. Processing continues with the next record in the input file.

Programmer Action:

- Increase the value in the PARM field of the EXEC statement so that it equals or exceeds the size of the input record that was too long, and resubmit the job.

- If you are certain that none of the records are longer than the buffer, treat the condition as an I/O error.

All data formatted on the output file prior to the error can be used by the emulator. If the end-of-volume summary (message IIQ305I) shows that all records on the file have been formatted, record the number of tapemarks and give the number to the operator the next time this job is run. If the error persists, make sure that MSGLEVEL=(1,1) was specified in the JOB statement and that you have the master console log, the printer listing, and the tape files before calling IBM for programming help.

IIQ304D PREPROCESSOR INTERVENTION REQUESTED-REPLY EOV OR SKIP (jobname)

Explanation: A condition was discovered by the operating system that must be corrected by the operator. Two conditions can cause this message to be issued:

- The end of the tape was detected on the input reel. If the tape has run off the reel, the operating system issues a system message and returns control to the preprocessor. The preprocessor issues this message.

- The operator pressed the RESET and START buttons on the input tape drive while an input volume was being processed. (This allows the operator to stop formatting an input volume.)

System Response: The preprocessor is placed in the wait state until the operator acts.

Operator Action:

- Respond to the system message by readying the unit. If the unit cannot be made ready, cancel the job. If the tape was run off the input reel, thread and rewind the tape.

- Respond to this message by typing:

  A. EOV, if the tape has been run off the reel or if you caused the message by pressing RESET and START on the input tape drive.

  B. SKIP, if there is more data to process on the tape.

When you type EOV, the preprocessor closes the input and output files and prints the end-of-volume summary. When you type SKIP, the preprocessor continues reading the tape; no data will be lost.

If you had to cancel the preprocessor job in response to the system message, you need not reply to this message.

Programmer Action: If the job was ended, make the necessary changes as indicated by other messages and resubmit the job. If the tape was run off the reel, get the number of tapemarks from the end-of-volume summary and give them to the operator the next time the job is submitted.

IIQ305I PREPROCESSOR EOV, REEL nnn SUMMARY
tapemark and record distribution
NUMBER OF TAPEMARKS SPECIFIED- nnn
NUMBER OF TAPEMARKS FOUND- nnn
NUMBER OF INPUT RECORDS- nnnnn
SIZE OF LARGEST INPUT RECORD- nnnnnn BYTES
MAXIMUM OUTPUT BLKSIZE- nnnnn
ccc FORMAT
INPUT RECORDS TOO LONG- nnn
INPUT I/O ERRORS- nnn

Explanation: The tape preprocessor program has finished formatting an input file; this message is the end-of-volume summary.

The section of the summary identified as the "tapemark and record distribution" varies depending upon the data on the tape. In general, TM is printed for each tapemark encountered and nnnn RECORD(S) PROCESSED is printed to indicate the number of records encountered between two tapemarks or between a tapemark and the end of the volume. For example, a volume has the following tapemark and record distribution:

TM	TM	10 Records	TM	556 Records	TM

The end-of-volume summary would indicate that distribution as:

```
0       RECORD(S) PROCESSED
TM
0       RECORD(S) PROCESSED
TM
10      RECORD(S) PROCESSED
TM
556     RECORD(S) PROCESSED
TM
```

NUMBER OF TAPEMARKS SPECIFIED-nnn is the number of tapemarks specified by the operator in response to message IIQ301A, where nnn is either a decimal number or ALL.

NUMBER OF TAPEMARKS FOUND-nnn is the number of tapemarks found by the preprocessor.

NUMBER OF INPUT RECORDS-nnnnn is the number of input records including tapemarks, header label records and trailer label records that were successfully formatted.

SIZE OF LARGEST INPUT RECORD-nnnnnn BYTES is the size of the largest input record that was successfully read and formatted.

MAXIMUM OUTPUT BLOCKSIZE-nnnnn ccc FORMAT is the size in bytes (nnnnn) of the physical records written into the output data set (equals the value specified in the BLKSIZE parameter in the SYSUT2 DD statement), and the record format (ccc) of the data.

INPUT RECORDS TOO LONG- nnn is the number of records longer than the length specified in the PARM field of the EXEC statement for the preprocessor job. For every input record that is too long, the operator will have replied SKIP to message IIQ303D PREPROCESSOR INPUT RECORD TOO LONG-REPLY EOV or SKIP (jobname).

INPUT I/O ERRORS-nnnn is the number of I/O errors encountered when reading the input data set.

System Response: The preprocessor issues message IIQ301D.

Programmer Action:

- If the number of tapemarks specified does not equal the number of tapemarks found:

  A. Ensure that the input tape given to the operator was the correct tape.

  B. Ensure that the operator specifies the correct number the next time this tape file is formatted.

- If one or more input records was too long (other than checkpoint records), increase the value specified in the PARM field of the EXEC statement and submit the job again. There is no way of determining the exact length of the input record that was too long. The end-of-volume summary will give you the size of the largest input record processed. For subsequent runs, the PARM field can be reduced to the appropriate size.

- If this summary indicates that there were I/O errors, normal tape maintenance such as cleaning tape drives and checking tapes should be performed. If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming help.

IIQ306I INVALID PARM FIELD ON EXEC CARD

Explanation: The PARM field in the EXEC statement of the tape preprocessor program or the tape postprocessor program was not a decimal number from 1 through 200000.

System Response: The job is ended.

Programmer Action: Correct the value in the PARM field and resubmit the job.

If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming help.

IIQ307D POSTPROCESSOR READY-REPLY ALL OR EOJ-(jobname)

Explanation: The tape postprocessor program is ready to process a tape.

System Response: The postprocessor is placed in a wait state until the operator responds to this message.

Operator Action:

- If there is a tape to format, type ALL. The postprocessor formats the entire tape, closes the input and output files, and prints an end-of-volume summary.
- If there are no more tapes to format, type EOJ.

IIQ308D POSTPROCESSOR OUTPUT RECORD TOO LONG-REPLY EOV OR SKIP-(jobname)

Explanation: A record is to be written that exceeds the output block size specified in the PARM parameter of the EXEC statement (or the PARM default value of 10,000 bytes). The name of the JOB statement is in (jobname).

System Response: The postprocessor is placed in the wait state until the operator acts.

Operator Action:

- Type SKIP and the record is bypassed, but its length is recorded and printed in the end-of-volume summary at the end of the job.
- Type EOV and the job is ended. The size of the record is printed in the end-of-volume summary.

Programmer Action: Increase the value specified in the PARM field of the EXEC statement so that it equals or exceeds the size of the output record that was too long, then resubmit the job. If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming help.

IIQ309I POSTPROCESSOR EOV, REEL nnn SUMMARY
NUMBER OF TAPEMARKS WRITTEN- nnn
NUMBER OF OUTPUT RECORDS- nnn
SIZE OF LARGEST OUTPUT RECORD- nnnnnn BYTES
OUTPUT RECORDS TOO LONG- nnn
OUTPUT REELS CREATED FROM THIS INPUT- nnn

Explanation: The tape postprocessor program has finished formatting an input file; this message is the end-of-volume summary.

NUMBER OF TAPEMARKS WRITTEN- nnn is the number of tapemarks written on the output file.

NUMBER OF OUTPUT RECORDS- nnn is the number of records written on the output file.

SIZE OF LARGEST OUTPUT RECORDS- nnnnnn BYTES is the size of the largest record that is encountered.

OUTPUT RECORDS TOO LONG- nnn is the number of records that exceed the length specified in the PARM field of the EXEC statement for the postprocessor job. For every output record that is too long, the operator replied SKIP or EOV to message IIQ308D POSTPROCESSOR OUTPUT RECORD TOO LONG-REPLY EOV OR SKIP-(jobname).

OUTPUT REELS CREATED FROM THIS INPUT- nnn is the number of reels of output data created from the input file.

System Response: The postprocessor issues message IIQ307D.

Programmer Action: If one or more output records was too long, increase the value specified in the PARM field of the EXEC statement to a value equal to or greater than the value given in "SIZE OF LARGEST OUTPUT RECORDS", and submit the job again.

If this summary indicates that there were I/O errors, normal tape maintenance such as cleaning tape drives and checking tapes should be performed. If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling the IBM for programming help.

IIQ310I THE FOLLOWING POSTPROCESSOR CONTROL STATEMENTS APE INVALID (statements)

Explanation: The program control statements listed under this message are invalid.

System Response: Invalid statements are listed, and the postprocessor job is ended.

Programmer Action: Correct the errors. Common errors are:

- A function statement does not have a ./ in columns 1 and 2 or a blank in column 3.
- A function statement has not been limited to columns 4 through 71.
- A comment has been placed on a function statement.
- More than one DENSITY statement has been included.
- More than nine TLABEL or nine HLABEL statements were included.
- The DATA operand was not included on a TLABEL or HLABEL statement.

If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming help.

IIQ311I POSTPROCESSOR OUTPUT I/O EPROR

Explanation: A permanent I/O error occurred during a write operation.

System Response: The postprocessor job is ended.

Programmer Action: The next time this program is run, use a different tape or tape drive. If the problem persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming help.

IIQ312I POSTPROCESSOR INPUT BLOCKSIZE TOO SMALL

Explanation: An input record was encountered that was larger than the size specified in the BLKSIZE specification in the DCB parameter of the SYSUT1 DD statement.

System Response: The postprocessor job is ended.

Programmer Action: Increase the BLKSIZE specification to a value that equals the block size of the input data. If the problem recurs, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and the printer listing before calling IBM for programming assistance.

IIQ313I INVALID REPLY-MESSAGE WILL BE REPEATED - (jobname)

Explanation: The operator gave an invalid response to the previous WTOR message.

System Response: The preprocessor or postprocessor reissues the message that received the invalid response.

Operator Action: No response is required for this message, but you must type a valid response to the message that is reissued.

Programmer Action: If a valid response is typed and this message is repeated, make sure that MSGLEVEL(1,1) was specified in the JOB statement, and that you have the master console log and printer listing before calling IBM for programming help.

DISK FORMATTING PROGRAM MESSAGES

IIQ401I ddname ASSIGNED PARM VALUE NOT FOUND

Explanation: The PARM value does not correspond to any of the accepted values. ddname identifies the name of the DD statement that was to be used. The disk units defined by previous DD statements have been formatted and need not be repeated.

System Response: The 1400 program stops.

Programmer Action: Correct the PARM value that corresponds to the DD statement.

Pefore resubmitting the job, modify the JCL statements by:

- Removing all DD statements for the disk units that were successfully formatted.
- Pecoding the PARM field to identify only the devices that remain to be formatted.
- Recoding the ddnames of the DD statements to SYSUT1, SYSUT2, etc.

If the error persists, include the MSGLEVEL=(1,1) operand in the JOB statement so that error messages are printed. Before calling IBM for programming help, make sure that you have the master console log and the printer listing.

IIQ402I ddname UNABLE TO OPEN DATA SET

Explanation: The program could not open the data set to be created. ddname identifies the DD statement that defines the data set. The disk units defined by previous DD statements have been formatted and need not be repeated.

System Response: The job is ended.

Programmer Action: Check for invalid parameters in the DD statement for the data set being created. Ensure that a DD statement describes the data set to be created, and that the DD statement has an acceptable ddname as defined in Appendix E. Include the MSGLEVEL=(1,1) operand in the JOB statement so that error messages will be printed. Before resubmitting the job, modify the JCL statements by:

- Removing all DD statements for the disk units that were successfully formatted.
- Recoding the PARM field to identify only the devices that remain to be formatted.
- Recoding the ddnames of the DD statements to SYSUT1, SYSUT2, etc.

If the error persists, ensure that you have the master console log and the printer listing before calling IBM for programming help.

IIQ403I ddname IMPROPER RECORD FORMAT SPECIFIED

Explanation: The record format was specified incorrectly in the DD statement. ddname identifies the DD statement. The disk units defined by previous DD statements have been formatted and need not be repeated.

System Response: The 1400 program stops.

Programmer Action: Ensure that the record format subparameter of the DCB parameter specifies either RECFM=F (for fixed-length record format) or RECFM=FT (for fixed-length record format with record overflow).

Before resubmitting the job, modify the JCL statements by:

- Removing all DD statements for the disk units that were successfully formatted.
- Recoding the PARM field to identify only the devices that remain to be formatted.
- Recoding the ddnames of the DD statements to SYSUT1, SYSUT2, etc.

If the error persists, ensure that MSGLEVEL=(1,1) was specified in the JOB statement, and that you have the master console log and a printer listing before calling IBM for programming help.

IIQ404I ddname BUFFER WAS NOT AVAILABLE

Explanation: The main storage requested by the program was not available. ddname identifies the DD statement for the disk unit being formatted. The disk units defined by previous DD statements have been formatted and need not be repeated.

System Response: The 1400 program stops.

Programmer Action: Ensure that the MSGLEVEL=(1,1) parameter is coded in the JOB statement. Before resubmitting the job, modify the JCL statements by:

- Removing all DD statements for the disk units that were successfully formatted.

- Recoding the PARM field to identify only the devices that remain to be formatted.

- Recoding the ddnames of the DD statements to SYSUT1, SYSUT2, etc.

If the error persists, make sure that you have the master console log and the printer listing before calling IBM for programming help.

IIQ405I I/O ERROR, jobname, stepname, unit addr, device type, ddname, operation attempted, error description, last seek addr, access method

Explanation: An I/O error occurred while processing a disk formatting program data set. Disk units defined by previous DD statements have been formatted and need not be repeated. Error analysis information is displayed which consists of:

jobname - is the job name of the disk formatting program run in which the error occurred.

stepname - is the step name of the job step.

unit addr - is the unit address of the device on which the error occurred.

device type - is the type of device on which the error occurred.

ddname - is the name of the DD statement that defines the device.

operation attempted - is the type of I/O operation being attempted.

error description - is the SYNAD, 14-character description of the error.

last seek addr - is the track and record address used for the last seek operation (which may be the address at which the error occurred).

access method - is the access method used on the data set.

System Response: The 1400 program stops.

Programmer Action: Correct the error. Ensure that the MSGLEVEL=(1,1) parameter is coded in the JOB statement. Before resubmitting the job, modify the JCL statements by:

- Removing all DD statements for the disk units that were successfully formatted.

- Recoding the PARM field to identify only the devices that remain to be formatted.

- Recoding the ddnames of the DD statements to SYSUT1, SYSUT2, etc.

If the error persists, make sure that you have the master console log and the printer listing before calling IBM for programming help.

Descriptor Codes							Operator Messages	Routing Codes												
1	2	3	4	5	6	7		*	1	2	3	4	5	6	7	8	9	10	11	12
						x	IIQ000I jobname MESSAGE NUMBER nnn NOT FOUND													x
						x	IIQ001I jobname OPERATOR SERVICES AVAILABLE													x
						x	IIQ002A jobname I													x
						x	IIQ003I jobname R text													x
						x	IIQ005I jobname CONTROL STATEMENT REQUESTED UNSUPPORTED FEATURE													x
						x	IIQ011D jobname INPUT EXCEEDS BUFFER SPACE, nnn BYTES, REENTER													x
						x	IIQ012D jobname cccccccc IS INVALID COMMAND OR KEYWORD PARAMETER													x
						x	IIQ013D jobname INVALID OPERAND=operand													x
						x	IIQ014D jobname KEYWORD PARAMETERS INCOMPLETE													x
						x	IIQ021D jobname HALT IAR=addr AAR=addr BAR=addr													x
						x	IIQ022D jobname HALT/BRANCH IAR=addr AAR=addr BAR=addr													x
						x	IIQ023I jobname EOJ HALT IAR=addr AAR=addr BAR=addr													x
						x	IIQ024D jobname ADDRESS=addr IS INVALID IN COMMAND													x
						x	IIQ031I jobname BUFFER TOO SMALL FOR PHYSICAL RECORD													x
						x	IIQ032I jobname nnn ERROR OCCURRED DURING EXECUTION OF IIQEIOCS													x
						x	IIQ033I jobname stepname, unit addr, device type, ddname, operation attempted, error description, track addr or relative block number, access method													x
						x	IIQ034A jobname CONTINUE INPUT OF COMMAND STRING													x
						x	IIQ042D jobname INVALID OP-CODE													x
						x	IIQ043D jobname INVALID INSTRUCTION FORMAT													x
						x	IIQ045D jobname INVALID I/O INSTRUCTION													x
						x	IIQ046D jobname INVALID ADDRESS													x
						x	IIQ052D jobname FILENAME NOT GIVEN FOR cccc													x
						x	IIQ053I jobname CORE UNAVAILABLE FOR DCB jobname CORE UNAVAILABLE FOR BUFFER. FILE=ddname													x
						x	IIQ054I jobname FILE=ddname COULD NOT BE OPENED													x
						x	IIQ055I jobname SYSEMOUT DCB NOT OPEN													x
						x	IIQ061D jobname EMULATOR WAITING													x
						x	IIQ062I jobname HEX ADDRESS=addr													x
						x	IIQ063I jobname NEXT 1400 ADDRESS=addr													x
						x	IIQ065I jobname 1400 DEVICE cccc NOT ASSIGNED													x
						x	IIQ066I jobname text													x
						x	IIQ071D jobname END OF 1400 STORAGE REACHED													x
						x	IIQ072D jobname CARD INPUT EXHAUSTED													x
						x	IIQ073D jobname INVALID 1400 DATA													x
						x	IIQ074D jobname CARRIAGE CONTROL TAPE IN ERROR													x
						x	IIQ075I jobname INVALID DATA IN 1400 STORAGE													x
						x	IIQ081I jobname EOM-OPERATION TERMINATED													x
						x	IIQ083D jobname 1400 DEVICE NOT ASSIGNED FOR cccc													x
						x	IIQ084I jobname cccc DEVICES NOT SUPPORTED													x
						x	IIQ086I jobname ERROR IN OPERATOR SERVICES													x
						x	IIQ092D jobname INVALID DEVICE REASSIGNMENT													x
						x	IIQ093I jobname EMULATOR CONTROL STATEMENT ERROR													x
						x	IIQ094I jobname INVALID RECFM - DDNAME=name													x
						x	IIQ301D PREPROCESSOR READY-REPLY nnn (NO. OF TM),ALL OR EOJ (jobname)												x	x
						x	IIQ302D PREPROCESSOR INPUT I/O ERROR-REPLY EOV OR SKIP (jobname)												x	x
						x	IIQ303D PREPROCESSOR INPUT RECORD TOO LONG-REPLY EOV OR SKIP (jobname)												x	x
						x	IIQ304D PREPROCESSOR INTERVENTION REQUESTED-REPLY EOV OR SKIP (jobname)												x	x
						x	IIQ305I PREPROCESSOR EOV, REEL nnn SUMMARY tapemark and record distribution NUMBER OF TAPEMARKS SPECIFIED- nnn NUMBER OF TAPEMARKS FOUND- nnn NUMBER OF INPUT RECORDS- nnnnn SIZE OF LARGEST INPUT RECORD- nnnnnn BYTES MAXIMUM OUTPUT BLKSIZE- nnnnn ccc FORMAT INPUT RECORDS TOO LONG- nnn INPUT I/O ERRORS- nnn												x	x
						x	IIQ306I INVALID PARM FIELD ON EXEC CARD												x	x
						x	IIQ307D POSTPROCESSOR READY-REPLY ALL OR EOJ-(jobname)												x	x
						x	IIQ308D POSTPROCESSOR OUTPUT RECORD TOO LONG-REPLY EOV OR SKIP-(jobname)												x	x
						x	IIQ309I POSTPROCESSOR EOV, REEL nnn SUMMARY NUMBER OF TAPEMARKS WRITTEN- nnn NUMBER OF OUTPUT RECORDS- nnn SIZE OF LARGEST OUTPUT RECORD- nnnnnn BYTES OUTPUT RECORDS TOC LONG- nnn OUTPUT REELS CREATED FROM THIS INPUT- nnn												x	x
						x	IIQ310I THE FOLLOWING POSTPROCESSOR CONTROL STATEMENTS ARE INVALID (statements follow)												x	x
						x	IIQ311I POSTPROCESSOR OUTPUT I/O ERROR												x	x
						x	IIQ312I POSTPROCESSOR INPUT BLOCKSIZE TOO SMALL												x	x
						x	IIQ313I INVALID REPLY-MESSAGE WILL BE REPEATED - (jobname)												x	x
						x	IIQ401I ddname ASSIGNED DDNAME NOT FOUND												x	x
						x	IIQ402I ddname UNABLE TO OPEN DATA SET												x	x
						x	IIQ403I ddname IMPROPER RECORD FORMAT SPECIFIED												x	x
						x	IIQ404I ddname BUFFER WAS NOT AVAILABLE												x	x
						x	IIQ405I I/O ERROR, jobname, stepname, unit addr, device type, ddname, operation attempted, error description, last seek addr, access method												x	x

A-address: A three-character field of a 1400 instruction that (1) gives the location of the units position of the A-field, or (2) is used to select a special device or feature, depending on the 1400 instruction.

A-address register (AAR): The register that normally contains the storage address of data in the A-address portion of the instruction.

A-field: The data field specified by the A-address.

alternate mode: A method used to record data on tape; each six-bit BCD character is represented by its equivalent eight-bit EBCDIC character. The 1400 program does not distinguish between even-parity and odd-parity data in alternate mode; odd-parity is equivalent to even-parity normal mode. Bit 1 of each character is 1 in alternate mode.

B-address: A three-character field of a 1400 instruction that gives the location of the B-field. It is usually the address of the units position of the B-field, but in some operations (such as tape read and write) it is the address of the high-order position of a record storage area.

B-address register (BAR): The register that normally contains the storage address of the data in the B-address portion of an instruction.

B-field: The data field specified by the B-address.

BCDIC-8: See binary coded decimal interchange code.

binary coded decimal (BCD): A character code used by 1400 systems where the bit configuration of a character is determined by two zone positions and four numeric positions (BA8421) and a check bit.

binary coded decimal interchange code (BCDIC-8): A character code used in stand-alone emulators that is similar to EBCDIC, except that parity is represented by bit 1 of each byte. Bit 1 is always one in even parity; even parity BCDIC-8 is equivalent to EBCDIC. Bit 1 is always zero for odd parity. Normal mode, which is the equivalent of BCDIC-8, is the term used in this publication.

binary format: A tape format produced by stand-alone emulators. It is equivalent to odd-parity normal mode format.

carriage control: Positioning forms on the printer using an instruction.

channel punches: Holes punched in a carriage control tape and used by carriage control instructions to control the movement of forms.

compressed tape feature: Two additional instructions that enable the 1401 processing unit to read a tape record written with zero elimination by a 7070/7074 System and to expand it in core storage.

DOS: Disk Operating System.

emulation: A combination of programming and hardware that permits one computing system to execute programs written for another.

file: A collection of related records treated as a unit.

I-address: A three-character field of a 1400 instruction that gives the location of the next instruction to be executed after a program branch.

I-address register (IAR): The register that contains the current address of the stored program. It keeps track of the program character by character, and is increased by one instruction after each storage cycle.

I-fetch: The function that obtains the next sequential 1400 instruction and converts it to a usable form (address conversion, register updating, etc).

integrated emulator: An emulator that is executed under control of a system control program using multiprogramming.

inverted print edit feature: A feature that interchanges the comma and the period. Thus, the period designates thousands and the comma designates decimals. When this feature is generated in the emulator, you cannot eliminate comma and decimal-point inversion.

I/O device correspondence: The association of the I/O devices of one system with those used by another system that emulates them.

I/O device independence: The ability to execute I/O operations on different devices. For example, emulating a card punch operation on a card punch, a magnetic tape unit, or a disk unit.

load mode: A method of operation in which 1400 data is transferred with associated wordmarks. See move mode.

MFT: Multiprogramming with a fixed number of tasks.

move mode: A method of operation in which only 1400 data is transferred. Wordmarks are not transferred. See load mode.

MVT: Multiprogramming with a variable number of tasks.

normal mode: A method used to record data on nine-track tapes that allows the emulator to distinguish 1400 data in even parity from 1400 data in odd parity. Each six-bit BCD character is represented by an equivalent eight-bit character. For characters in even parity, bit 1 is always one; for characters in odd parity, bit 1 is always zero.

partition: The term used in this publication to identify either an MFT partition or an MVT region.

postprocessor: A program used to process the tape files produced by the emulator, giving tape output in 1400 format.

preprocessor: A program used to process 1400-format tape files before emulation.

region: See partition.

relocatable: Capable of being anywhere in main storage.

selective stacker: A feature of 1400 systems that stacks cards processed by a card reader or a card punch in pockets (or stackers). Different models have different stacker arrangements. The program uses stacker selection to place a card in a specified stacker.

sense switches: A group of seven switches on the console of 1400 processor units that controls program operations. The switches are toggle switches that can be set before the start of a job. The switches can be tested by the program and used to control operations such as suppressing printing or punching.

simulation: The use of programming techniques only to duplicate the operations of a given computing system on another computing system.

spanned format (spanned variable-length record format): The standard operating system data format used on tapes.

stand-alone emulator: A program that allows programs written for one computing system to be executed on another computing system using a compatibility feature. The stand-alone emulator, however, does not share system resources with other programs, but controls them all. When the stand-alone emulator is used, no other jobs can be run on the computer.

tapemark record: A System/370 data record used to emulate a 1400 tapemark. Only operating system tapemarks appear on tapes created the tape preprocessor program and the emulator. Tapemark records give the relative position of the 1400 tapemarks.

unrecoverable error: An error that the system operator cannot correct.

1400: The term used in this publication when discussing the 1401, 1440, and 1460 Data Processing Systems.

1400 format: The term used in this publication when discussing a data file produced by a 1400 system, a stand-alone emulator, or Compatibility Support/30 or Compatibility Support/40 using DOS.

1401: The term used in this publication when discussing the 1401 Data Processing System.

1440: The term used in this publication when discussing the 1440 Data Processing System.

1460: The term used in this publication when discussing the 1460 Data Processing System.

Where more than on page reference is given, the major reference is first.

READER'S COMMENT FORM

1401 / 1440 / 1460 OS Emulator
on Models 135 / 145 / 155

GC33-2008-0

- How did you use this publication?

  As a reference source .... □
  As a classroom text .... □
  As a self-study text .... □

- Based on your own experience, rate this publication . . .

As a reference source:	.... Very Good	.... Good	.... Fair	.... Poor	.... Very Poor
As a text:	.... Very Good	.... Good	.... Fair	.... Poor	.... Very Poor

- What is your occupation? ....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

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

YOUR COMMENTS PLEASE . . . .

This SRL manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note : Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Cut Along Line

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Department 813 (LGP)

fold fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

GC33-2008-0

1401/1440/1460 OS Emulator on Models 135/145/155 Reference Printed in U.S.A. GC33-2008-0

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]