File No. S360-50
GC20-1685-0

IBM Data Processing Techniques

System/360 Disk Operating System User's Guide: Control Statement Techniques

This publication provides guidance in the use of control statements as related to compilations, linkage editing, utilities, sorts, and user programs. It presents tested examples with supporting explanations as an aid to understanding the input stream requirements for processing in the System/360 Disk Operating System environment. The control statements discussed are job control statements, linkage editor control statements, and operator commands.
A list of related publications appears in the Preface.

PREFACE

The objectives of this publication are:

1. To bring into perspective for the new user the Disk Operating System job stream requirements
2. To furnish typical job stream examples and reference data covering the normal processing needs of the DOS user

Job stream examples are presented in building block fashion. Each new type of statement is discussed when it is first used; only unusual circumstances are pointed out in succeeding examples. To simulate an installation environment, a specific System/360 machine configuration and DOS system generation are assumed. The List of Examples provides quick reference to specific job stream examples and discussions pertinent to a given control statement type. Appendices furnish statement summary data.

No attempt has been made to cover all possibilities, but rather to present sufficient information through a number of examples to enable the user to modify an example for his particular needs or to create his own job stream. The manual does not take the place of formal training, but supplements the data discussed in classes and related SRL publications by bringing it together in functional form.

The examples were tested on system release 9; changes through system release 12 were reviewed to assure validity of the information.

Related publications are:

1. IBM System/360 Disk Operating System: System Control and System Service Programs (C24-5036)
2. IBM System/360 Disk Operating System: Operating Guide (C24-5022)
3. IBM System/360 Disk and Tape Operating Systems: Utility Programs Specifications (C24-3465)
4. IBM System/360 Disk Operating System: Sort/Merge Program Specifications (C24-3444)
5. IBM System/360 Disk and Tape Operating System: Tape Sort/Merge Program Specifications (C24-3438)
6. IBM System/360 Disk Operating System: Autotest Specifications (C24-5062)
7. IBM System/360 Reference Data: TOS/DOS Job Control Statement and Operator Communication (X20-1748)

First Edition (Reprinted December 1969)

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601.

CONTENTS

FIGURES

EXAMPLES

INTRODUCTION

BACKGROUND VS FOREGROUND PROGRAMS

The IBM System/360 Disk Operating System is designed to handle two types of problem programs: background and foreground. Background programs are initiated by job control from the batched job input stream. Foreground programs are initiated by the operator from the printer-keyboard. A foreground initiator (in lieu of job control) interprets the operator foreground initiation commands. Foreground programs do not execute from a stack. When one foreground program is completed, the operator must explicitly initiate the next program.

The system is capable of concurrently operating one background program and one or two foreground programs. Background and foreground programs initiate and terminate asynchronously from each other. Neither is aware of the other's existence. The number of problem program areas (called partitions) and the sizes of the areas are determined by the user initially at system generation time. Each partition is defined as a block of contiguous core storage locations. Although the number and size of the partitions can be redefined subsequently by the operator, the partitions are considered fixed.

At linkage edit time, a program is related to the specific core storage locations in which it is to operate and thus is established as either a foreground or a background program. An exception to this rule is a self-relocating program, which -- because it initializes its own address constants at execution time -- can execute from any core storage location irrespective of the linkage editor assignments. Object programs resulting from COBOL, FORTRAN, PL/I, and RPG are not self-relocating. The utility input/output macros and the Assembler Language, written with special programming considerations, can produce self-relocating programs. The logical IOCS access methods, except BTAM and QTAM, can be self-relocating.

A foreground program cannot require modification of the background communication region or access to system symbolic units (SYSRDR, SYSPCH, etc.). Assembler, RPG, and COBOL object programs are restricted to those language functions which do not need these facilities. FORTRAN and PL/I object programs do not operate as foreground programs.

All IBM-supplied programs operate only as background programs. These include the language translators (Assembler, COBOL, FORTRAN, PL/I, RPG), the system service programs (librarian, linkage editor), the service programs (sorts, utilities, Autotest), and job control. In addition, Autotest requires a dedicated (no foreground partitions) system.

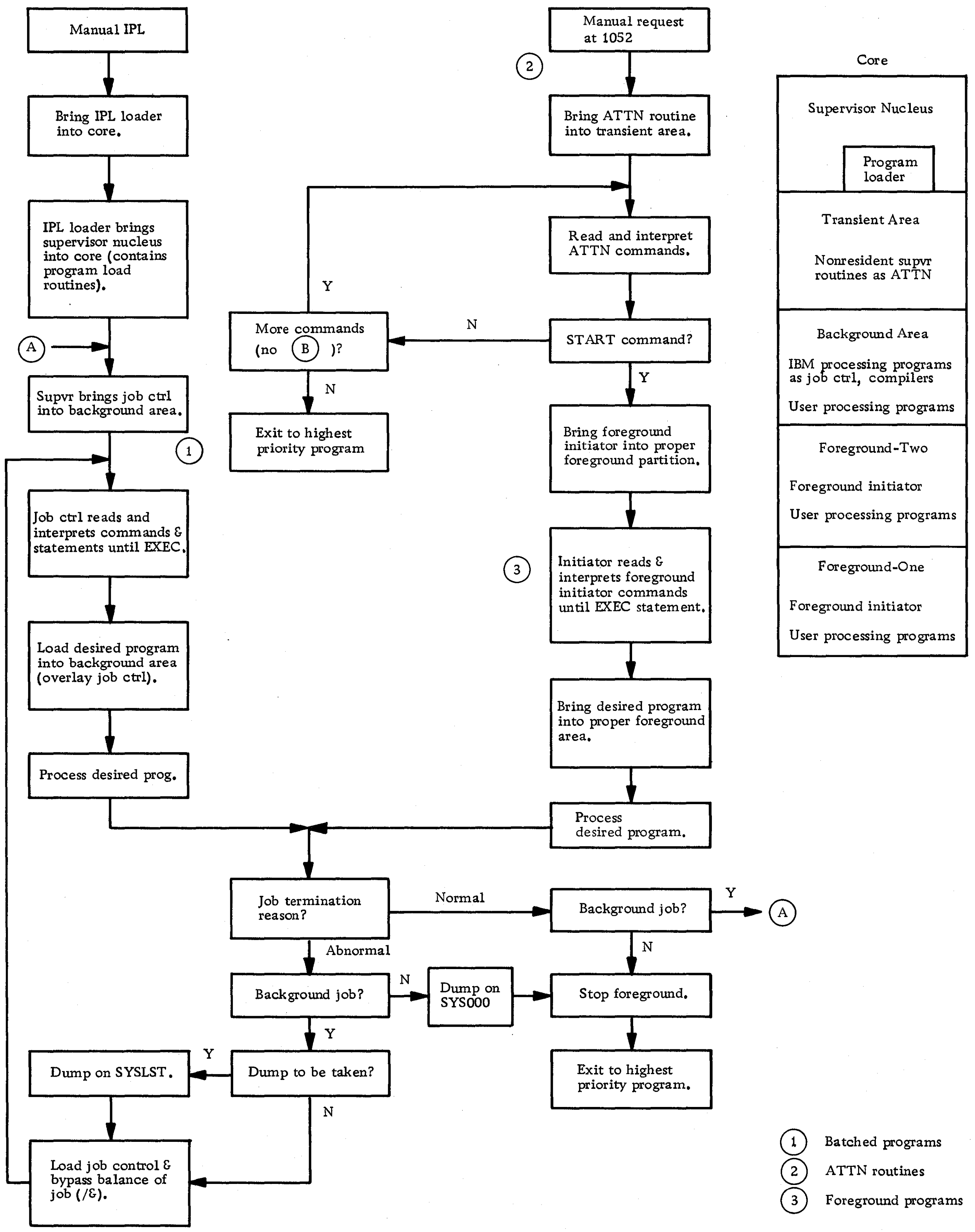
Programs using the utility input/output macros can operate as either background or foreground programs.

SYSTEM FLOW

The system flow schematic is shown in Figure 1. Because the IBM System/360 Disk Operating System is disk-resident, operation of the system is initialized through an initial program load (IPL) procedure from the resident disk. The IPL loader is loaded into main storage from the resident disk simply by selecting the address of the unit in the load-unit switches on the system console and pressing the load key. The loader then reads the nucleus of the supervisor into low main storage from the resident disk. After successfully reading in the supervisor nucleus, the IPL loader performs certain initializing and housekeeping functions, and reads and interprets the IPL control statements. Then control is transferred to the supervisor, which uses its system loader routine to load the job control program. The IPL procedure need be performed only at the beginning of the day unless an unusual condition requires a new IPL.

The job control program is loaded into 10K of main storage in the background program area (following the transient area of the supervisor). Functioning in the stacked-job environment, job control reads and interprets the job control statements and the operator job control commands in the input stream, prepares each job step to be run, and provides job-to-job transition. A job may consist of either the execution of a single program or the execution of more than one program. Each execution is called a job step. Job control performs its functions between job steps and is not present while a background problem program is being executed.

Upon encountering an execute job control statement in the input stream, job control returns control to the supervisor. The supervisor loads the required problem program from the core image library into the background program area, where it overlays job control. Control is transferred to the entry point in the problem program.



Manual IPL
Bring IPL loader into core.
IPL loader brings supervisor nucleus into core (contains program load routines).
A
Supvr brings job ctrl into background area.
1
Job ctrl reads and interprets commands & statements until EXEC.
Load desired program into background area (overlay job ctrl).
Process desired prog.
Manual request at 1052
2
Bring ATTN routine into transient area.
Read and interpret ATTN commands.
START command?
N
Y
More commands (no B)?
Y
N
Exit to highest priority program
Bring foreground initiator into proper foreground partition.
3
Initiator reads & interprets foreground initiator commands until EXEC statement.
Bring desired program into proper foreground area.
Process desired program.
Job termination reason?
Normal
Abnormal
Background job?
Y
A
N
Stop foreground.
Exit to highest priority program.
Background job?
N
Dump on SYS000
Y
Dump to be taken?
Y
Dump on SYSLST.
N
Load job control & bypass balance of job (/&).
Core
Supervisor Nucleus
Program loader
Transient Area
Nonresident supvr routines as ATTN
Background Area
IBM processing programs as job ctrl, compilers
User processing programs
Foreground-Two
Foreground initiator
User processing programs
Foreground-One
Foreground initiator
User processing programs
1 Batched programs
2 ATTN routines
3 Foreground programs

Figure 1. System flow schematic

The processing of the problem program continues until one of three conditions occurs:

1. Normal end of job.

   For a normal end-of-job (EOJ) macro, control is returned to the supervisor, which loads job control. Job control reads and interprets the next job control statements and commands in the input stream until an execute statement occurs. The cycle is then repeated.

2. Abnormal end of job.

   For an abnormal end of job, a dump of main storage occurs, depending on whether the system previously has been given authority to do so. Then control is returned to the supervisor, which loads job control. Because of job cancellation, job control bypasses all job control statements in the input stream until the next job (not just job step) is reached. Job-to-job transition occurs, and normal interpretation of the job control statements and commands begins.

3. Problem program calling another problem program.

   If a new program is to be fetched as part of the same job step, control returns to the supervisor, which loads the required program and transfers control to the entry point of the new program. The problem program can request the supervisor to load rather than fetch. In that case, the data is loaded, but control returns to the problem program that requested the load.

Because job control loads into the background program area, and because background and foreground programs initiate and terminate asychronously, foreground programs obviously cannot use job control for fear of disturbing a problem program in process in the background area. Therefore, the system flow beyond the IPL procedure, as discussed earlier, is applicable only to background programs.

The supervisor contains subroutines that are required only occasionally. These are not included in the nucleus supervisor, which is core-resident at all times, but instead are brought into the transient area of main storage when needed. Among these transient routines is the message (ATTN) routine. When a foreground program is to be initiated, the operator presses the request key on the printer-keyboard. As soon as the transient area is not being used by another program, the supervisor loads the ATTN routine into the transient area and transfers control to it. The ATTN routine reads commands from the operator via the printer-keyboard. By initiating the START command (which designates foreground partition one or two) the operator returns control to the message routine, which determines whether the area specified is allocated and does not contain a program. If so, the 2K foreground initiator routine is loaded into the requested foreground partition; otherwise, the operator is notified that he has given an invalid command.

The foreground initiator reads subsequent commands to prepare the job step to be run. Upon encountering the execute command, the initiator directs the supervisor to provide loading information about the program to be executed. When the initiator receives the loading information, it determines the proper load address, depending on whether the program is self-relocating. The supervisor loads the program from the core image library into the proper foreground area.

A foreground program is terminated normally through an end-of-job (EOJ) macro or abnormally by operator action, a DUMP/CANCEL macro in the program, program error, or certain I/O error conditions. Following program closing operations, the operator may initiate another program in this area.

Although the system is capable of handling one to three partitions concurrently, the central processing unit (CPU) can do only one thing at a time. Therefore the programs compete for CPU processing time according to a fixed sequence of descending priority: supervisor (highest), the error recovery routines (QUIESCE), the ATTN routine, foreground-one, foreground-two, and background (lowest). The supervisor selects which program is to be given control. When an interruption occurs, the supervisor gains control, processes the interruption, and gives control to the highest priority program that is in the ready state. Control is taken away from a high-priority program when that program encounters a condition that prevents continuation of processing until a specified event, such as a write I/O operation, has occurred. Control is taken away from a low-priority program when an event for which a higher priority program was waiting, such as completion of a write I/O operation, has occurred. Although it is easier to view the system flow for the partitioned (multiprogramming) environment as strictly sequential, it must be realized that, due to the competition for CPU processing time, the operations are interleaved across the various partitions.

I/O DEVICE ASSIGNMENT

Programs do not reference I/O devices by their actual physical addresses but rather by symbolic names. This ability to reference an I/O device by a symbolic name instead of a physical address permits a program to be written that is dependent only on the device type and not on the actual device address. The programmer selects the symbolic name of a device from a fixed set of symbolic names. At execution time the symbolic name is associated with an actual physical device through tables stored in the supervisor. The assignment entries in these tables are normally made at system generation time. However, job control statements and operator commands can alter the standard device assignment before program execution.

The symbolic names are divided into two classes: system logical units and programmer logical units. System logical units (SYSIPT, SYSLNK, SYSLOG, SYSLST, SYSPCH, SYSRES, and SYSRDR) are used by the control program and by IBM-supplied processing programs. SYSRDR, SYSIPT, SYSLST, SYSPCH, and SYSLOG also can be used by user programs operating in the background area. Foreground programs may not reference any system logical unit except SYSLOG. For the convenience of the user, two additional names, SYSIN and SYSOUT, are defined for background program assignments. These names are valid only to job control and cannot be referenced in a user program. SYSIN can be used when SYSRDR and SYSIPT are assigned to the same device; SYSOUT must be used when SYSLST and SYSPCH are assigned to the same magnetic tape unit.

Foreground and background programs may reference any programmer logical unit (SYS000-SYS244). At system generation time, programmer logical units are defined for each partition -- background, foreground-one and foreground-two. The same SYSnnn can be defined for all three partitions. At execution time there can be no overlap of actual physical devices between partitions, except direct access storage devices (DASD). For example, SYS001 can be assigned to separate physical devices in all three areas, but tape unit X'181' cannot be assigned to SYSnnn in the background and SYSnnn in the foreground even if SYSnnn in the background and SYSnnn in the foreground are different. Standard assignments cannot be made for foreground programmer logical units at system generation time.

Appendices F and G list the logical units, their functions, the devices to which they can be assigned, their usage by IBM-supplied processing programs, and special assignment considerations.

ASSUMED DISK RESIDENT SYSTEM CONFIGURATION

To simulate an installation environment, all of the examples in the manual assume that the Disk Operating System configuration and options shown in Figure 2 were generated at system generation time.

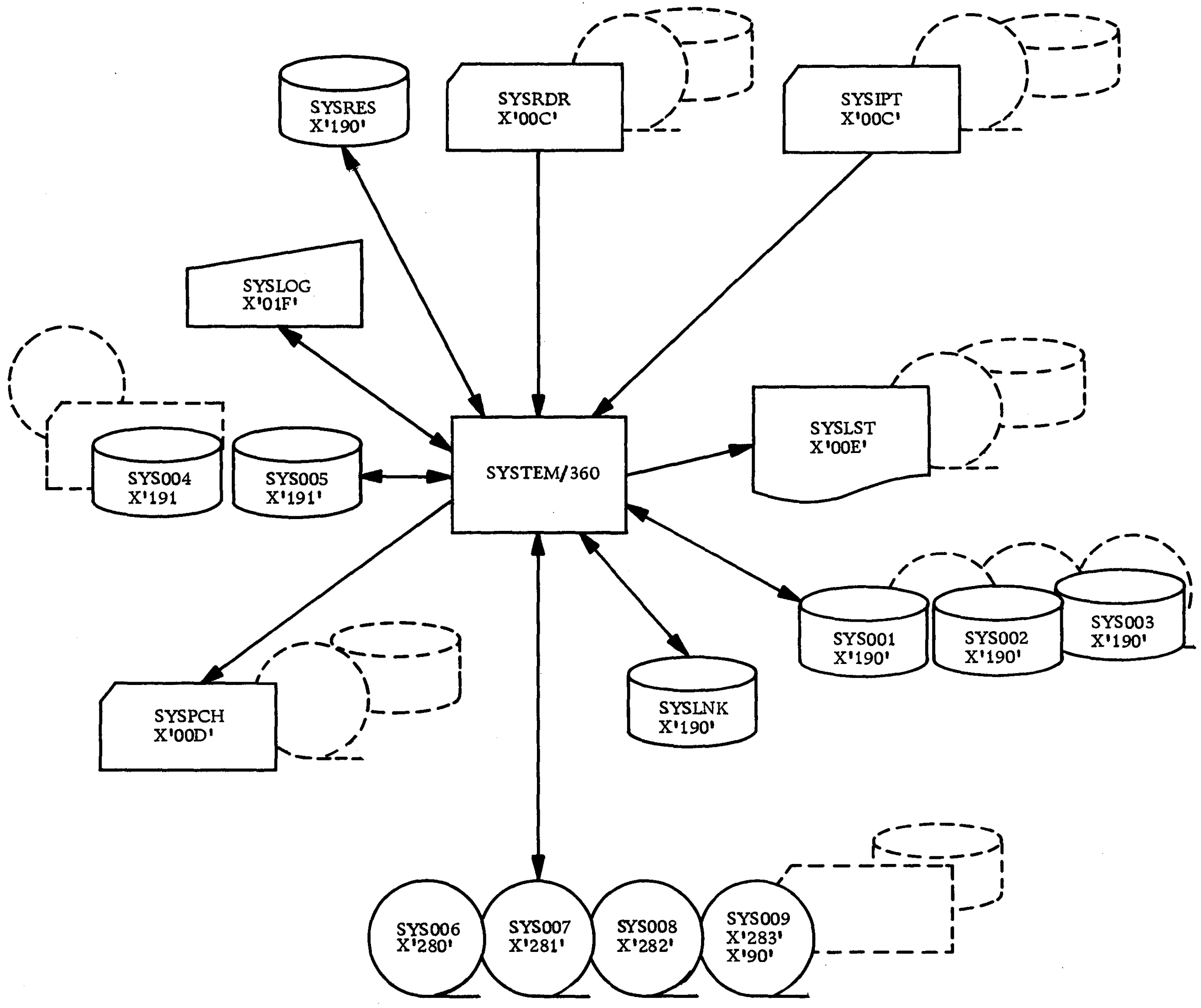
The assumed system units and their physical and logical assignments are:

Unit	Standard Physical Assignment	Standard Logical Assignment
2540 Card Read Punch	X'00C' (reader)	SYSIPT, SYSRDR
	X'00D' (punch)	SYSPCH
1052 Printer-Keyboard	X'01F'	SYSLOG
1403 Printer	X'00E'	SYSLST
2311 Disk Storage Drive	X'190'	SYSLNK, SYSRES SYS001-SYS003
2311 Disk Storage Drive	X'191'	SYS004, SYS005
2402 Magnetic Tape Unit	X'280'	SYS006
2402 Magnetic Tape Unit	X'281'	SYS007
2402 Magnetic Tape Unit	X'282'	SYS008
2402 Magnetic Tape Unit	X'283', X'90'	SYS009

(X'90' = 7-track tape with data convert)

The assumed options are:

DECK	Output object module of language translators on SYSPCH.
LIST	Output listing of source module of language translators on SYSLST; also object module listing and errors (Assembler).
XREF	Output symbolic cross-reference list (Assembler) on SYSLST.
LISTX	Output Procedure Division Map (COBOL) on SYSLST; output object module (PL/I) on SYSLST.
ERRS	Output summary of errors in source program (COBOL/FORTRAN/PL/I) on SYSLST.
LOG	Log control statements on SYSLST.
SYM	Output symbol table (Assembler) on SYSPCH; output Data Division Map (COBOL) on SYSLST; output symbol table (PL/I) on SYSLST.



Note: Dotted units indicate other types of devices that may be assigned to corresponding system logical units before execution of a program.

Figure 2. Assumed disk resident system configuration

THE INPUT STREAM -- BATCHED PROGRAMS

JOB CONTROL INPUT

The job control program accepts three kinds of control statements: job control statements, job control commands, and linkage editor control statements. Job control statements are also called programmer statements, and job control commands are called operator statements.

Job Control Statements

Job control statements have the following main characteristics:

- They enter the system through the device assigned to SYSRDR.

SYSRDR may be a card reader, magnetic tape unit, or disk unit. If SYSRDR is a tape or disk, the job control statements are written unblocked in card image format by a utility program before submission of the jobs for execution.

- They contain // blank in positions 1—3 of each statement.

There are three exceptions: end of job, which is /& blank; end-of-data input, /* blank; and comments, * blank.

- They are essentially the same for each execution of a given job or job step, and are effective for a job or job step rather than across jobs.

Job Control Commands

Job control commands have the following main characteristics:

- They are distinguished from job control statements by the absence of // blank in positions 1—3 of each command.

- They permit the operator to adjust the system according to today's operating conditions.

This is particularly true in the area of device assignment, where the operator may need to (1) communicate to the system that a device is unavailable, and (2) designate a different device as the standard for a given symbolic logical unit. Therefore these commands normally are not a part of the regular input stream for a given job. They tend to be effective across jobs, whereas job control statements are confined within a job.

- They enter the system through the device assigned to SYSRDR or the printer-keyboard (SYSLOG).

If entered through SYSRDR these commands may be interspersed with job control statements. To enter through the printer-keyboard, the operator must gain control of the system when the job control program is being executed (between jobs or job steps) by one of these methods:

1. A PAUSE job control statement in the input stream (SYSRDR)
2. A PAUSE job control command in the input stream (SYSRDR)
3. A PAUSE attention command issued after pressing the request key

The same rules apply for entering commands at the printer-keyboard as for other operator communications.

1. Either upper- or lowercase letters are acceptable. Lowercase is suggested for simplicity and identification of operator action.
2. Each command must be followed by the end-of-communications indicator Ⓑ (alter code 5) to cause processing to continue.
3. If a mistake in entering is made, the cancel indicator Ⓒ (alter code 0) is keyed, and control returns to the operator.
4. A Ⓑ without any preceding data signifies that the operator has finished entering commands, and control is given to SYSRDR.

Linkage Editor Control Statements

Linkage editor control statements have the following main characteristics:

- They initially enter the system through the device assigned to SYSRDR as part of the input job stream. PHASE and INCLUDE statements may be present on SYSIPT or in the relocatable library.
- They must be blank in position 1 of the statement.
- They are validated for operation (INCLUDE, ACTION, ENTRY, or PHASE) and are copied to SYSLNK to become input when the linkage editor is executed.

DATA INPUT

Input data is often interspersed with job control data in the input stream. This occurs when the same physical device is used for input to job control (SYSRDR) and input to the processing program being called by job control (SYSIPT or SYSnnn).

For example, in the assumed configuration used in this manual, SYSRDR and SYSIPT are both assigned to the same card reader. Job control will look to SYSRDR for input. Upon encountering the execute command in the job stream, control passes to the desired processing program. Assuming that this processing program calls for data on the card reader defined as SYSIPT, such data must follow the execute statement. To signal the end of processing program data, a /* (end-of-file) record follows the last data record.

The assignment of the same physical device to SYSRDR and SYSIPT or SYSnnn usually implies that the data to be submitted to these symbolic logical units has the characteristics of card data -- 80-character, unblocked records. If SYSRDR and SYSIPT or SYSnnn are assigned to the same tape or disk unit, the same interspersed arrangement will be required with all records unblocked in card image format.

ATTN INPUT

The normal processing of the System/360 can be interrupted at any time by pressing the request key on the printer-keyboard. The ATTN routine is loaded into the transient area, and control is brought to the printer-keyboard for entry of ATTN commands.

ATTN commands have the following main characteristics:

- They are entered through the printer-keyboard in the same fashion as job control commands.
- They can be processed at any time, since they do not depend on the job control program.
- They permit operator action relative to both foreground and background programs.

The ATTN routine is executed from the transient area. The commands are useful for such functions as initiation of a foreground program, reallocation of main storage, and cancellation of the execution of background or foreground programs.

COMPILE

GENERAL CONSIDERATIONS

Required Symbolic Logical Unit Assignment

The language translators (Assembler, COBOL, FORTRAN, PL/I, and RPG) require the following symbolic logical unit assignments:

SYSIPT	Source statement input
SYSLOG	Operator communication
SYSLST	Programmer messages, listing
SYSPCH	If DECK or SYM option specified
SYSRDR	Job control statements
SYSLNK	Linkage editor input if LINK or CATAL option specified
SYS001	Work file
SYS002	Work file (not required by FORTRAN)
SYS003	Work file (not required by FORTRAN)
SYS004	Work file (only if COBOL + debug program used)

Program Names (operand of EXEC statement)

Assembler	ASSEMBLY
COBOL	COBOL
FORTRAN	FORTRAN
PL/I	PL/I
RPG	RPG

SINGLE COMPILATION, SYSRDR and SYSIPT COMBINED (EXAMPLE 1)

JOB Statement

```
// JOB jobname [comments]
```

This statement is required to indicate the beginning of control information for a job. Jobname must be one to eight alphameric characters and is stored in the background communication region. The name selected has no significance to the system except in restart, when it must be identical to that used when the checkpoint was taken. A blank jobname is invalid.

The JOB statement always will print in positions 1 through 72 on SYSLST and SYSLOG regardless of the LOG options. If the timer feature is present, the time of day also prints. The JOB statement causes an eject to a new page before printing on SYSLST.

Several initializing operations occur as a result of processing the JOB statement. The following are of significance to the user:

- Restoration of all job control options to standard (as set up at system generation); cancellation of LINK and CATAL options
- Restoration of all programmer logical unit assignments to permanent (as set up at system generation plus job control command permanent assignments); restoration of all system logical unit assignments to permanent unless SYSRDR is temporarily assigned.
- Modification of the communication region
  1. Restoration of date to the system standard as set up at IPL time
  2. Transfer of new jobname
  3. Zeroing of user's program area including UPSI byte
- Deletion of user labels and restoration to user label mode

EXEC Statement

```
// EXEC [progname]
```

An EXEC statement indicates the end of preparation for the execution of a job step. All control statements necessary for execution must be processed before this time.

Progname can be one to eight alphameric characters and must correspond to the phase name used when the program was cataloged to the core image library. If progname is omitted, the system expects that the program to be executed has just been processed by the linkage editor. In this case, the program has been placed behind the last previously cataloged phase in the core image library. Any program to be executed must be in the core image library.

Storage is cleared from the beginning address of the problem program to the end of the background partition, and a fetch is issued for the desired program. Control is given to the fetched program at its entry point.

PAUSE Statement or Command

```
// PAUSE [comments]
   PAUSE [comments]
```

The PAUSE can be used to allow for operator action between job steps. With the first format the PAUSE is effective just before the next input control record for the job stream is read. The second format causes the PAUSE the next time job control is brought into core. Both forms always print on SYSLOG regardless of the LOG option. The comments are for a message to the operator or documentation.

* Comments Statement

* comments (position 2 must be blank)

This statement always prints through position 72 on SYSLOG and can be used for any user comments. If followed by a PAUSE statement, it can be used to request operator action.

/* End of Data File Statement

/* rest of statement is ignored
(position 3 must be blank)

This statement is the last statement for an input data file. It signals the end of the data file by turning on the end-of-file indicator for:

- SYSRDR and SYSIPT regardless of the physical device assignment
- SYS000—SYS244 when a card reader is assigned and logical IOCS is used

The /* is used by a processing program. It is ignored by job control providing position 3 is blank; otherwise it is invalid.

Certain processing programs do not handle the /* in accordance with the general rules given above. Some examples are:

1. The utility programs recognize /* as the end of a card input file only if the rest of the card is blank. This ability is essential to build job input streams on tape or disk. See Example 11. Placement of comments (positions 4—80) in all /* statements as standard practice would permit loading of an input stream to tape or disk without modification of the /* statements.
2. The MPS utility macro INCARD uses the physical device end to signal the end of file. It does not recognize /*. The user can add a routine to test for a signal card such as /*.
3. COBOL permits the use of SYSIPT only with the ACCEPT statement. The ACCEPT routines do not use /* to signal an end of file automatically for SYSIPT. An analysis is made for /*, and the record is bypassed without transfer to the user I/O area. This means the user must recognize his own end of file in some fashion other than /*.
4. Currently, FORTRAN uses the END statement to signal the end of the source deck. No /* is needed. If a /* is placed after these source decks, no problem will arise as long as SYSRDR and SYSIPT are the same device, because when job control takes over after compilation it will read from SYSRDR and ignore the /*. However, if SYSIPT is a different device, the /* will be read as the first data record to the new job step, and the compilation will fail. All other language translators require /* to signal the end of the source deck.

/& End-of-Job Statement

/& rest of statement is ignored
(position 3 must be blank)

This statement must be the last statement for each job (not job step). It signals the end of the input job stream for the job. If SYSRDR and SYSIPT are assigned to different devices, the /& should be present on both streams to permit proper operation in case of an abnormal end of job.

Several operations occur as a result of processing the /& statement. The following are of interest to the user:

- Restoration of all job control options to standard as set up at systems generation, and cancellation of LINK and CATAL options
- Restoration of all symbolic logical unit assignments to permanent as set up at systems generation plus job control command permanent assignments
- Modification of the communication region
  1. Restoration of date to the system standard as set up at IPL time
  2. Transfer of NONAME to jobname
  3. Zeroing of user's program area including UPSI byte
- Display of EOJ message on SYSLST and SYSLOG with the time of day if the interval timer feature is present
- Listing of all nonzero tape error statistics (TEB's)
- Assurance that SYSIPT is at end of file
- Deletion of user labels and restoration to user label mode

The JOB and /& statements work together to define the beginning and end of the input stream for a given job. Only one JOB and one /& are permitted per job. If a /& is omitted, the JOB statement will transfer to the /& routine to simulate the /& statement.

To avoid operational problems due to improper job stream input, the effect of the /& and job termination, both normal and abnormal, on SYSRDR and SYSIPT must be fully understood. Since the assignment of these units is involved, detail on this subject is given in the discussion of the ASSGN statement.

Example 1. Single Compilation, SYSRDR and SYSIPT Combined

```
① // JOB COMPILE PROGA, JOHN DOE                                        } Input from
② *  SINGLE COMPILATION, SYSRDR & SYSIPT COMBINED, PUNCHED OUTPUT       } SYSRDR
③ // EXEC COBOL                                                         }

          Source statements                                             } Input from
④ /*                                                                    } SYSIPT
⑤ // PAUSE REMOVE OBJECT DECK FROM STACKER   (optional)                  } Input from
⑥ /&                                                                    } SYSRDR
```

Discussion

The assumed configuration and options are in effect (see Figure 2). Since SYSLNK and SYS001-003 are on disk, it is further assumed that the required disk labels are already on the standard label track.

① The JOB statement will assure the restoration of options and symbolic logical unit assignments to standard. The jobname, COMPILE, will be placed in the background communication region, and the user area and UPSI byte zeroed. The rest of the information prints for documentation purposes.

② Comments may appear anywhere in the SYSRDR input. However, if placed ahead of the job card, the comments will print on a separate sheet because the JOB statement ejects to a new sheet on SYSLST. Comments are entirely free-form; the only requirement is one blank following the *. For all job control input, the principle of delimiting (separating) each field by at least one blank, or in some instances a comma, is a requirement to permit the job control scan routine to locate the parameters. Here two blanks have been used so that, when logged, the statements will be easier to read because they are lined up.

③ The EXEC statement causes the COBOL language translator to be loaded from the core image library and to begin execution.

④ COBOL calls for source statement input on SYSIPT. Since SYSRDR and SYSIPT are the same physical device, the source statements must follow the EXEC statement. Source statements are read until the /* is encountered. This signals the end of the source statements for this particular COBOL compilation.

The options in effect, the specific language translator, and the logical unit assignments determine the output. Because the DECK option is in force, the resultant relocatable object module is output to SYSPCH, which in this case is a card punch. The Procedure Division Map, error diagnostics, and source program are listed on SYSLST (a printer in this example) because of the LISTX, ERRS, and LIST options respectively. See Example 2 for further option information.

The end of the COBOL compilation causes job control to be loaded.

⑤ Job control calls for SYSRDR, so the PAUSE statement is read. The PAUSE is in the job control statement format; therefore it is effective when the next job control statement is read. It prints on both SYSLST and SYSLOG, and the comments give instruction to the operator. Placement earlier in the job stream (before EXEC) would be too soon, since no cards would have been punched.

A job control command PAUSE (no //) could have been used between JOB and EXEC. The PAUSE action is delayed for this format by setting a switch, which is interrogated whenever job control is initialized, that is, loaded as on completion of a job step. In effect this is when a job or job step is completed. However, placement between the /* and /& causes a pause at the end of the next job step, because job control is already in core storage.

When the PAUSE is effective, control is brought to the printer-keyboard. When the operator has complied with the instructions, he returns control by entering the end-of-communication signal Ⓑ (alter code 5).

⑥ As a result of processing the /& statement, the EOJ message prints on SYSLST and SYSLOG, the options and logical unit assignments are restored to standard, the background communication region is zeroed in the user area and UPSI byte, and the date is restored to the system standard.

MULTIPLE COMPILATIONS, SYSRDR and SYSIPT COMBINED (EXAMPLE 2)

OPTION Statement

```
// OPTION option1 [,option2...]
```

The OPTION statement is used to specify certain functions (options) of the system. Multiple options can appear in the same statement in any sequence, separated by commas. The options are:

LOG	Log control statements on SYSLST
DUMP	Dump registers and main storage on SYSLST if abnormal program end
LINK	Indicate object module is to be linkage edited; write output of language translator on SYSLNK
DECK	Output object module on SYSPCH
LIST	Output listing of source module on SYSLST; also object module listing and errors (Assembler)
LISTX	Output Procedure Division Map (COBOL) on SYSLST; output object module (PL/I) on SYSLST
SYM	Output symbol table (Assembler) on SYSPCH; output Data Division Map (COBOL) on SYSLST; output symbol table (PL/I) on SYSLST
XREF	Output symbolic cross reference list (Assembler) on SYSLST
ERRS	Output summary of errors in source program (COBOL, FORTRAN and PL/I) on SYSLST
48C	48-character set on SYSIPT (PL/I)
60C	60-character set on SYSIPT (PL/I)
CATAL	Catalog program/phase in core image library after linkage editing
STDLABEL	Cause all sequential DASD or tape labels to be written on the standard label track
USRLABEL	Cause all DASD or tape labels to be written on the user label track

Note: To suppress any of the above options, prefix operation with NO -- for example, LOG, NOLOG. In FORTRAN, NODECK is not effective unless LINK is also specified.

The following facts concerning options will bring their use into proper perspective:

- Standard options are established at system generation time except for CATAL, LINK or NOLINK, STDLABEL and USRLABEL.
- USRLABEL is assumed if STDLABEL is not specified.
- LINK or CATAL are required only when the linkage editor program is to be executed.
- The majority of the options pertain only to the language translators.
- Most jobs or job steps will require option statements only if it is desirable to deviate from the system standard.
- An option remains in effect until

  1. A contrary option statement is read, or

  2. A JOB or /& statement is encountered, which causes a reset to system generation standard.

A more detailed discussion of LINK, CATAL, USRLABEL, and STDLABEL is given in the sections entitled "Standard Tape and DASD Labels" and "Linkage Editing".

Example 2. Multiple Compilations, SYSRDR and SYSIPT Combined

```
①// JOB  MULTCOMP
  *  MULTIPLE COMPILATIONS, SYSRDR & SYSIPT COMBINED, PUNCHED OUTPUT
②//  EXEC   FORTRAN                                                    Job step 1

          Source statements

  /*
②//   EXEC     FORTRAN                                                  Job step 2

          Source statements

  /*
③//    OPTION     NOSYM
②//     EXEC       ASSEMBLY                                               Job step 3

          Source statements

  /*
②//      EXEC        RPG                                                  Job step 4

          Source statements

  /*
②//       EXEC         COBOL

          Source statements.                                              Job step 5

  /*
④/&
```

Discussion

① The number of blanks between the parameters has been varied in this example to show the flexibility of the format.

② This job consists of five job steps because there are five EXEC statements before a /& is read. Processing flows from one job step to the next, normally without operator intervention. The job control program is used six times.

Note that each program to be compiled requires its own EXEC statement even though the same compiler may be used.

③ The standard option SYM (see assumed options) is overriden by the option statement NOSYM. The new option will remain in effect until a /& or JOB statement resets all options to the system standard; therefore, the symbol table for the Assembler program and the data division map for the COBOL program are suppressed. If the COBOL compilation were to occur before the option statement, only the Assembler symbol table would be suppressed.

④ Because all five job steps comprise a single job, if anything happens to cause cancellation of a job step, all succeeding job steps are bypassed (input is read but not acted upon) until the /& statement is located. For this reason, care should be taken in constructing job steps to avoid bypassing those job steps that are not truly dependent upon the successful completion of the previous job step. This is more important when SYSRDR and SYSIPT are assigned to tape or disk, because the job stream is not changed as easily as when it is on a card reader.

MULTIPLE COMPILATIONS, SYSRDR and SYSIPT SEPARATE (EXAMPLE 3)

ASSGN Statement or Command

```
                    ⎧X'cuu'⎫ ⎡⎧,X'ss'⎫⎤
// ASSGN  SYSxxx,   ⎨UA    ⎬ ⎢⎨      ⎬⎥
                    ⎩IGN   ⎭ ⎣⎩,ALT  ⎭⎦

                    ⎧X'cuu'⎫ ⎡⎧,X'ss'⎫⎤
   ASSGN  SYSxxx,   ⎨UA    ⎬ ⎢⎨      ⎬⎥ [,TEMP]
                    ⎩IGN   ⎭ ⎣⎩,ALT  ⎭⎦
```

The ASSGN statement or command is used to assign a logical I/O unit to a physical device. The new assignment is effective immediately. The entries in the operand field represent the following:

SYSxxx	The symbolic unit name -- SYSRDR, SYS001, etc.
deviceaddress	Select one of these: 1. X'cuu' -- the channel and unit number is expressed in hexadecimal. c = 0 (multiplexor) or 1—6 (selector); uu = 00 - FE. 2. UA -- the logical unit is unassigned. Reference to the unit causes job cancellation; useful to free devices for foreground allocation. 3. IGN -- the logical unit is unassigned. Reference to the unit is ignored; useful in testing and certain rerun situations. IGN cannot be used if information from the device is required for proper operation. Therefore, use of IGN results in job cancellation at open time if tape or DASD files are processed by logical IOCS.
Magnetic tape specifications	Select one of these (optional): 1. X'ss' -- the mode settings (density, parity, etc.) for 7-track and dual-density 9-track tapes. See Appendix E for specific codes. This parameter is optional because the system assumes 800 BPI, odd parity, translate feature off, and convert feature on for 7-track tape, and 800 single-/1600 dual-density for 9-track tape. 2. ALT -- indicates an alternate tape unit that is used when the capacity of the original is reached. The alternate must have the same mode setting as the original since no x'ss' specification can be given for this unit. Multiple alternates may be assigned to the same symbolic unit. Rotation is in the sequence in which assignment is made.
TEMP	Valid only for command format (no //); indicates the assignment will be reset by the next /& or JOB statement.

To effectively use the ASSGN statement/command, the difference between the two formats must be understood.

Device assignments made by the ASSGN statement (//) are considered temporary. They hold until another ASSGN statement/command for that logical unit, or a RESET statement/command for that logical unit, or the next /& or JOB statement is read -- whichever occurs first. If a RESET, /&, or JOB statement is encountered, assignment reverts to the standard assignments as set up at system generation time plus any modification by an ASSGN command (no //). There is one exception to the rules stated above. If SYSRDR or SYSIN is temporarily assigned between jobs (between a /& and a JOB statement), system unit assignments will not be reset, but programmer unit assignments will be reset to standard.

The ASSGN command (no //) without the TEMP option permanently alters the standard assignment of the logical unit and holds from job to job until superseded permanently by another ASSGN command (no //), or temporarily by an ASSGN statement (//) or an ASSGN command with the TEMP option.

If the TEMP option is used, the ASSGN command (no //) functions in the same way as the ASSGN statement.

If a new IPL is required, assignments are set to the standards established at system generation.

The assignment routines validate the operands for the relationship between the physical device, the symbolic logical unit, the type of assignment (permanent or temporary), etc. The following list summarizes the most pertinent items to remember when making assignments:

1. No physical device except DASD can be assigned simultaneously to a foreground and a background program.

2. All system logical unit assignments to disk must be permanent.

3. SYSIN must be assigned if both SYSRDR and SYSIPT are to be assigned to disk. They must occupy the same extent.

4. SYSOUT cannot be assigned to disk. It must be a permanent assignment on tape.

5. IGN and ALT are invalid for SYSRDR, SYSIPT, or SYSIN.

6. SYSLNK must be assigned before issuing the LINK or CATAL options, or the options will be ignored.

7. If SYSRDR, SYSIPT, SYSLST, or SYSPCH is assigned to tape or disk when the system is generated, it will be unassigned by IPL. Such assignments can be made effective only with the ASSGN statement or command because the ASSGN also opens the file.

8. SYSLOG cannot be assigned after a foreground program is initiated. It must be assigned to a 1052 to initiate a foreground program.

9. A physical device that is unavailable to the system because of a DVCDN (device down) command cannot be assigned until a DVCUP (device up) command has been given.

10. SYSRES can never be assigned by an ASSGN statement or command. An IPL is required to change the SYSRES assignment.

11. If the 1052 is inoperable, SYSLOG and SYSLST must be assigned to the printer; SYSOUT cannot be assigned.

The assignment of SYSRDR and SYSIPT deserves special consideration because the physical positioning in the input job stream for these two logical units is closely tied to job termination. Figure 3 gives the sequence of events that occur when a job closes normally or abnormally. Only pertinent items are shown.

A study of the schematic in Figure 3 reveals the following conclusions:

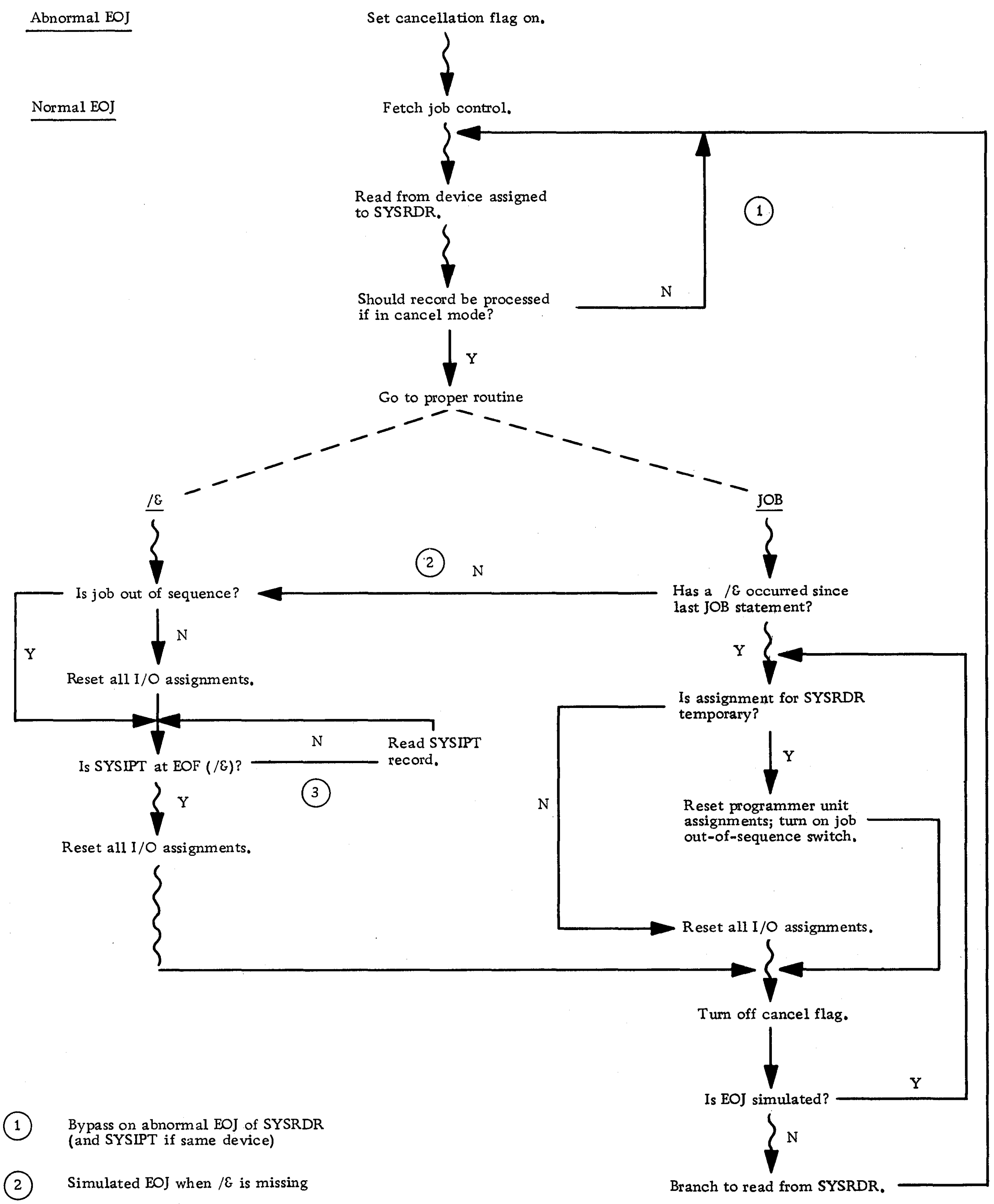
- Abnormal job step termination
  1. The device assigned to SYSRDR will read records (bypass) until a /& or JOB statement is encountered. If SYSIPT is assigned to the same device, input data records are also passed, since they do not meet the criteria.
  2. If SYSIPT is assigned to a different device, the bypass of input data records to the next /& does not occur until after the /& or JOB statement has been read by SYSRDR. The device that represents SYSIPT at this point will be (a) the temporary or permanent device in effect when the job step was canceled, if SYSRDR was temporarily assigned between jobs, or (b) the permanent SYSIPT device, if SYSRDR is permanently assigned.
- Normal job step termination
  1. The device assigned to SYSRDR reads and interprets statements. This may result in the execution of another job step or the termination of a job (/& or JOB). If the job is finished and SYSIPT is assigned to the same device, it, too, will be at the end of data input for the job.
  2. If SYSIPT is assigned to a different device, the /& record has not yet been read. The SYSIPT device that will be advanced is the same as in the case of abnormal end with separate SYSRDR.

The final conclusion is that a temporary assignment for SYSIPT should not be made when SYSRDR is assigned permanently, because the wrong SYSIPT will be advanced to the next /&.

RESET Statement or Command

$$// \text{ RESET } \begin{Bmatrix} \text{SYS} \\ \text{PROG} \\ \text{ALL} \\ \text{SYSxxx} \end{Bmatrix}$$

$$\text{RESET } \begin{Bmatrix} \text{SYS} \\ \text{PROG} \\ \text{ALL} \\ \text{SYSxxx} \end{Bmatrix}$$


Abnormal EOJ
Set cancellation flag on.
Normal EOJ
Fetch job control.
Read from device assigned to SYSRDR.
1
Should record be processed if in cancel mode?
N
Y
Go to proper routine
/&
JOB
2
N
Is job out of sequence?
Has a /& occurred since last JOB statement?
N
Y
Y
Reset all I/O assignments.
Is assignment for SYSRDR temporary?
N
Read SYSIPT record.
Is SYSIPT at EOF (/&)?
Y
3
Y
N
Reset programmer unit assignments; turn on job out-of-sequence switch.
Reset all I/O assignments.
Reset all I/O assignments.
Turn off cancel flag.
Is EOJ simulated?
Y
N
Branch to read from SYSRDR.

① Bypass on abnormal EOJ of SYSRDR (and SYSIPT if same device)

② Simulated EOJ when /& is missing

③ Bypass on SYSIPT if not same device as SYSRDR (standard assignment unless job out of sequence)

Figure 3. Job termination schematic

Both RESET formats function alike. The I/O assignments specified by the operand are reset to the permanent assignments. Permanent assignments are those specified when the system was generated plus those made with the ASSGN command (no //) without the TEMP option. The I/O assignments are reset as follows:

SYS	All system logical units
PROG	All programmer logical units
ALL	All logical units (system and programmer)
SYSxxx	The logical unit specified

Foreground assignments are unaffected by the RESET.

LISTIO Statement or Command

$$// \ \text{LISTIO} \ \left\{\begin{array}{l}\text{SYS}\\\text{PROG}\\\text{F1}\\\text{F2}\\\text{ALL}\\\text{SYSxxx}\\\text{UNITS}\\\text{DOWN}\\\text{UA}\\\text{X'cuu'}\end{array}\right\}$$

LISTIO   Same operands as above

LISTIO is used to cause the system to print a listing of the I/O assignments specified in the operand. The statement format uses SYSLST as output, whereas the command format uses SYSLOG. The listing helps to avoid conflicting assignments, especially in a multiprogramming environment. The listing varies according to the operand used:

SYS	Lists the physical units assigned to all system logical units
PROG	Lists the physical units assigned to all background programmer logical units
F1	Lists the physical units assigned to all foreground-one programmer logical units
F2	Lists the physical units assigned to all foreground-two programmer logical units
ALL	Lists the physical units assigned to all system and programmer logical units
SYSxxx	Lists the physical units assigned to the logical unit specified
UNITS	Lists the logical units assigned to all physical units
DOWN	Lists all physical units specified as inoperative
UA	Lists all physical units not currently assigned to a logical unit
X'cuu'	Lists the logical units assigned to the physical unit specified

MTC Statement or Command

$$// \ \text{MTC opcode, SYSxxx} \ [\text{,nn}]$$

$$\text{MTC opcode,} \left\{\begin{array}{l}\text{SYSxxx}\\\text{X'cuu'}\end{array}\right\} [\text{,nn}]$$

The MTC statement/command controls magnetic tape operations. The opcode parameter determines the specific operation as follows:

BSF	Backspace file
BSR	Backspace record
ERG	Erase gap
FSF	Forward-space file
FSR	Forward-space record
REW	Rewind
RUN	Rewind and unload
WTM	Write tape mark

The second operand designates the unit on which the operation is to be performed. Note that any symbolic logical unit is a valid entry, but that the actual device address can be used only with the command format.

The optional third parameter designates the number of times the specified operation is to be performed. If the parameter is omitted, 1 is assumed.

CLOSE Command

```
                ⎡⎧,X'cuu' [,X'ss']⎫⎤
CLOSE   SYSxxx  ⎢⎨,UA             ⎬⎥
                ⎢⎪,IGN            ⎪⎥
                ⎣⎩,ALT            ⎭⎦
```

The CLOSE command is used to close either a system or a programmer output logical unit.

The operation of the CLOSE command depends upon the type of device assigned to the SYSxxx specified.

For a 2311 device the following rules apply:

1. Only SYSIN, SYSRDR, SYSIPT, SYSLST, and SYSPCH are valid.
2. The CLOSE causes an end-of-file record to be written if the file is an output file.
3. The second operand is mandatory, since the CLOSE always reassigns the unit to the value of this operand. ALT is invalid, since it applies to tape. IGN is invalid for SYSRDR, SYSIPT, or SYSIN.
4. If the new file is a disk or tape unit, it is opened.
5. A CLOSE statement is required for a system logical unit assigned to a 2311. If a CLOSE has not been issued, a subsequent assignment to the same 2311 will be rejected.

For a magnetic tape unit the following rules apply:

1. Output logical units for SYS000-SYS244, SYSPCH, SYSLST, or SYSOUT are valid.
2. The CLOSE causes the writing of a tape mark, the EOV trailer record, and two tape marks. It then causes rewinding and unloading of the tape.
3. The second operand, if present, causes the unit to be reassigned to the value specified. Otherwise, the assignment remains unchanged. The second operand is required for SYSPCH, SYSLST, or SYSOUT. ALT is valid only for SYSPCH, SYSLST, or SYSOUT units that are currently assigned to tape.

Example 3. Multiple Compilations, SYSRDR and SYSIPT Separate

```
①      ASSGN SYSIPT,X'281'
    // JOB A
    *  MULTIPLE COMPILATIONS, SYSRDR & SYSIPT SEPARATE, SYSPCH ON TAPE JOB A
②      ASSGN SYSPCH,X'282',TEMP
③  // EXEC FORTRAN
    // EXEC COBOL
④      CLOSE SYSPCH,X'00D'
⑤  /&
    // JOB B
    // EXEC RPG
    // EXEC ASSEMBLY
    // EXEC FORTRAN
⑤  /&
⑥      MTC RUN,SYSIPT
⑦      ASSGN SYSIPT,X'00C'
```

Discussion

① Since the source statements are to be on tape, the ASSGN job control command changes the standard assignment of SYSIPT from the card reader to a tape unit. This assignment is permanent and will be effective from job to job (A, B, etc.) until reset or overriden by another ASSGN. Note that the assignment has been made outside the limits of a job, not between the JOB and /& statements. Placement between JOB and the first EXEC is equally satisfactory.

Another function of the ASSGN statement/command is to open the file if the unit involved is SYSRDR, SYSIPT, SYSLST, or SYSPCH. For a tape input unit the OPEN will be performed only if the tape is at load point. This provides for interruption of the stream and return. The OPEN accepts a leading tape mark, no labels, or any label set beginning with VOL1. No actual checking of labels is performed, and no label statements need to be submitted for labeled input.

② The ASSGN job control command with the TEMP option assigns SYSPCH to a tape unit instead of to the card punch. This assignment is temporary and resets to the card punch as soon as a /& or JOB statement is encountered. Thus, when job B is executed, the card punch will be SYSPCH. An ASSGN job control statement gives the same results. No other placement is satisfactory for this job.

The ASSGN statement/command opens SYSPCH in the same way as SYSIPT. Since SYSPCH is an output tape, it also makes sure that the expiration date is past if the tape has IBM standard labels.

③ The EXEC statements for FORTRAN and COBOL are consecutive because SYSIPT is no longer the same device as SYSRDR. This time the language translators will look to the tape unit for data, since this is the assignment of SYSIPT. SYSIPT must still have the /* following each source deck (except if FORTRAN) to signal the end-of-data file. See Example 11 for building a SYSIPT file.

④ System logical files are not closed automatically; therefore, the CLOSE command is used to write the proper end-of-file records and to rewind and unload the SYSPCH tape. Assignment is switched to the card punch. In this example the assignment portion of the CLOSE statement, which is a required parameter, is actually redundant, since SYSPCH was assigned temporarily. However, because of the CLOSE statement, a permanent assignment for SYSPCH would also have been satisfactory.

⑤ The /& signals the end of job. The SYSIPT assignment remains. The SYSIPT data requires a /& behind the last data for each job so that, in case of job cancellation, the system can advance SYSIPT to the input for the next job.

⑥ Assuming there is no more input on SYSIPT, this input tape can be rewound and unloaded.

⑦ The assignment for SYSIPT is returned to the system generation standard to avoid conflict with future job streams. CLOSE cannot be used to accomplish steps 6 and 7 because the command is not valid for system logical input tape units, since CLOSE includes the writing of tape marks and trailer labels.

STANDARD TAPE AND DASD LABELS (EXAMPLES 4 TO 10)

Because the job control label statements are used only for files with standard labels, the entire discussion on labels that follows is oriented toward files with standard labels.

PHYSICAL LABELS

DASD Files

The standard labels for direct access storage devices include one volume label for each volume and one or more file labels for each logical file on the volume. The volume label is always the third record on cylinder 0, track 0 of the volume. This label is written by the initialize-disk or initialize-data-cell utility program at the time the volume is prepared for use. Two parameters that are specified to the utility program for placement in the volume label are (1) the volume serial number, a unique six-digit identifier, and (2) the location on the volume of the standard file labels.

All of the standard file labels are grouped together and stored on the volume in a specific area selected by the user at initialization time. This area must be contained within one cylinder. On a system-residence volume it must be outside the residence area; on a non-system-residence volume it may follow the volume label. This group of labels is called the volume table of contents (VTOC) because it represents a directory of the location of all data files on the volume. The VTOC itself is a file of records. The initialization utility program preformats the VTOC and creates a label for the VTOC in the first record location of the VTOC.

The standard file labels are entered into the VTOC as the result of the creation of an output or work file by a processing program. Labels are deleted from the VTOC when work files are closed, or when the file expiration date is past and the space is requested by a new file. The labels written for data files follow three standard formats:

Format 1. This format is used for all logical files. It identifies the file by a 44-byte name and provides file and data record specifications such as creation and expiration date, file type, record format, and block length. It also provides the addresses of up to three separate areas (extents). A format 1 label is always in the VTOC for each volume of a multivolume file.

Format 2. This format is required for any file that is organized by the index sequential file management system. It appears only on the volume of the file that contains the cylinder index, and provides additional specifications unique to this file organization, such as index areas.

Format 3. If a logical file uses more than three extents, this format is used to specify the addresses of as many as 13 additional extents. It appears on all volumes of the logical file.

In addition to these three formats, there are two special labels:

Format 4. This is the label for the VTOC itself (first record of the VTOC).

Format 5. This label is used only by the System/360 Operating System for DASD space management. Although DOS does not use it, the area for such a label is reserved by the initialization utility program (second record of the VTOC).

Refer to Appendices A and B for the volume and format 1 label formats. For the specific content of the other labels, see IBM System/360 Disk Operating System, Data Management Concepts (C24-3427).

Tape Files

The standard tape labels consist of one volume label per reel and two standard file labels for each logical file on the reel -- one header label preceding the data records for the file and one trailer label following the file.

The standard volume label identifies the entire volume (reel). It is always the first record on the reel and is written initially at the time the reel is prepared for use in the system by an IBM-supplied utility program. The format of the volume label is the same as for DASD. Therefore it, too, contains a unique volume serial number.

The standard file header and trailer labels provide such information as the file creation and expiration dates, the volume sequence number (for a multivolume file), and the file sequence number (for a multifile volume). The header and trailer labels have essentially the same format and content so that the tape can be read forwards or backwards. Refer to Appendices C and D for the specific content and location on the volume.

The volume label and a dummy header label are written when the reel of tape is initially prepared for use on the system. The volume label remains unchanged, but the header label is rewritten whenever the reel is used for output, provided the previous expiration date is past. The trailer label is written for output reels when the file is closed or when the end of the volume is reached.

SYSTEM FLOW

In order to understand the control statement requirements for standard label processing, knowledge of the system flow is helpful (see Figure 4).

Label Storage

Job control -- or the foreground initiator -- builds label data into a label storage area of SYSRES from VOL, DLAB, and XTENT statements for DASD files, or from VOL and TPLAB statements for labeled tape files. This is strictly a storage operation for later use by a processing program. No actual label processing occurs at this point.

The label storage area is the entire cylinder that follows the source statement library. The cylinder is allocated as follows:

Track 0	Permanent label data
Tracks 1-3	Temporary background label area
Tracks 4-6	Temporary foreground-one label area
Tracks 7-9	Temporary foreground-two label area

The particular label storage section into which the label data is built is determined as follows:

- With job control (background program)
  1. Permanent label data will be created if the option STDLABEL is in effect when the label statements (VOL, DLAB, etc.) are entered. Only sequential DASD or tape labels can be permanent.
  2. Temporary background label data will be created if the option USRLABEL is in effect when the label statements are entered.
- With foreground initiator

  The label statement data is stored in the foreground-one or foreground-two area depending on which partition is being initiated.

Whenever label statement data is entered, storage starts at the beginning of the first, or only, track of the respective section, destroying all previous labels in that section. Permanent label data remains in effect across job boundaries until new permanent label data is submitted. Background temporary label data is not carried from job to job since it is destroyed by the /& or JOB statement. It is effective across job step boundaries unless overwritten by a succeeding job step. Foreground label data is effective for a single execution. Permanent label data is available to all partitions, whereas temporary label data is available only to the respective partition.

The size of the label storage record varies with the type of file as follows:

Tape	80 bytes
Sequential DASD	104 bytes per extent
Nonsequential DASD	104 bytes per file plus 20 for each additional extent (similar to bytes 85-104)

All records are written with an 8-byte key. Figure 5 shows the format and content of the label storage records.

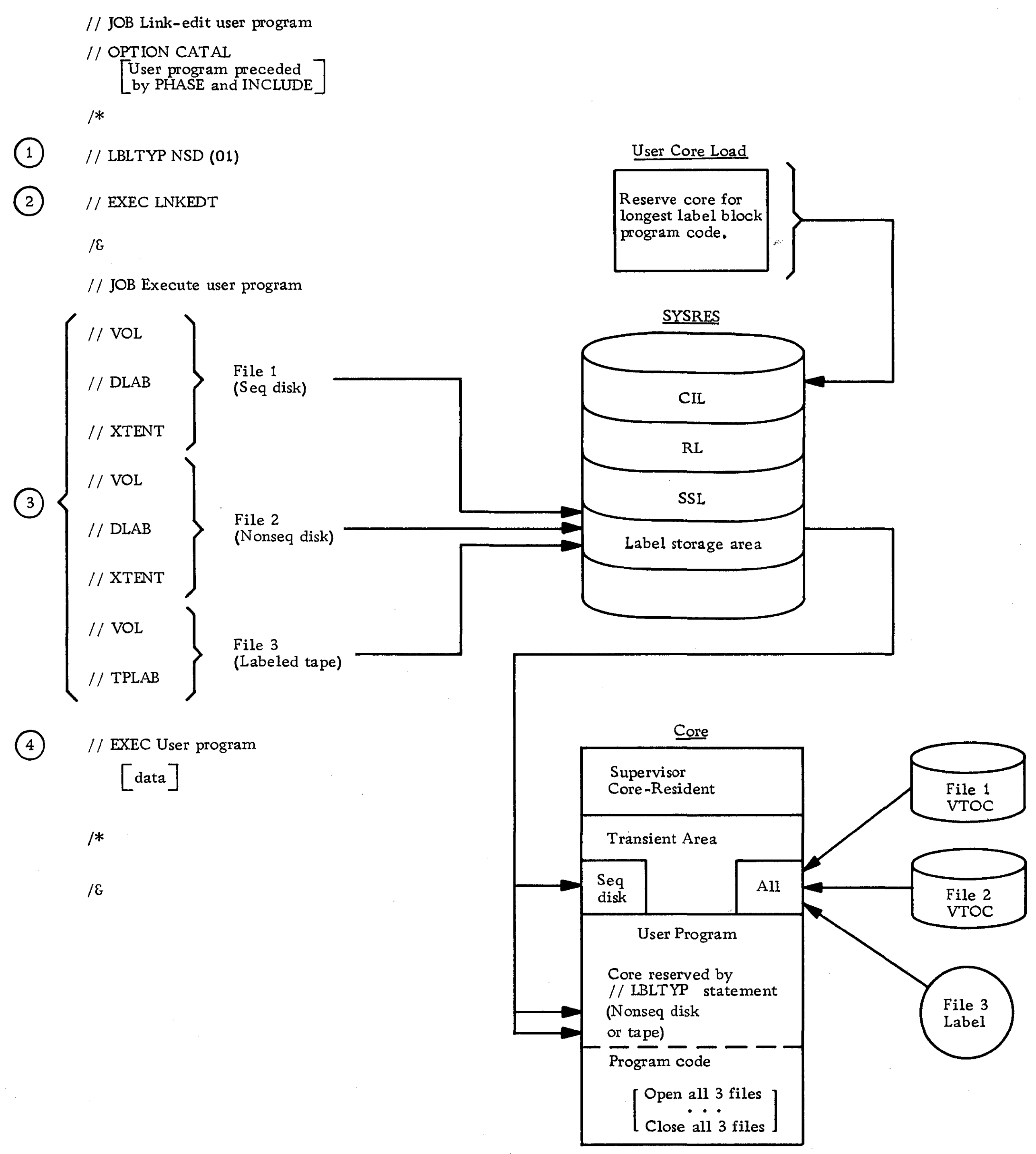
Since multiple label statements are required for each file, checks are made to ensure that the statements are submitted in the proper sequence (VOL, TPLAB, TPLAB continuation; or VOL, DLAB, DLAB continuation, and one or more XTENT's). The actual writing of a label storage record does not occur upon reading the label statements. When a proper set of label statements has been submitted, a switch is set to indicate that a label is in the output area. During the processing of the next VOL or EXEC statement, a label storage record will be written on SYSRES if the switch is on. A record will also be written during the processing of a /& statement while the standard label option is in effect, if the switch is on. Thus a new set of label statements causes the writing of the preceding set; the last or only set is written by the EXEC or /& statement.

For the reasons given above, all label statements normally are placed immediately ahead of their respective EXEC statement. However, in the case of certain disk system logical files (as SYSIPT), the OPEN for the file occurs as part of the processing of the ASSGN statement, and the label data must be in the label storage area at that time. Examples are given later that illustrate these points.

Label Processing

The label data from the label statements is written in the label storage area as a disk record with key. The disk record key is created from the VOL statement parameter "filename", which is the symbolic file name used in the processing program. This symbolic name is also generated as an alphameric constant in the processing program. The OPEN and CLOSE routines use this symbolic name as the search-key-equal argument to locate and read the proper label storage record into main storage. For each file, the temporary label storage area is searched and then, if the record is not found, the permanent storage area.

SAMPLE JOB STREAM (Batched)

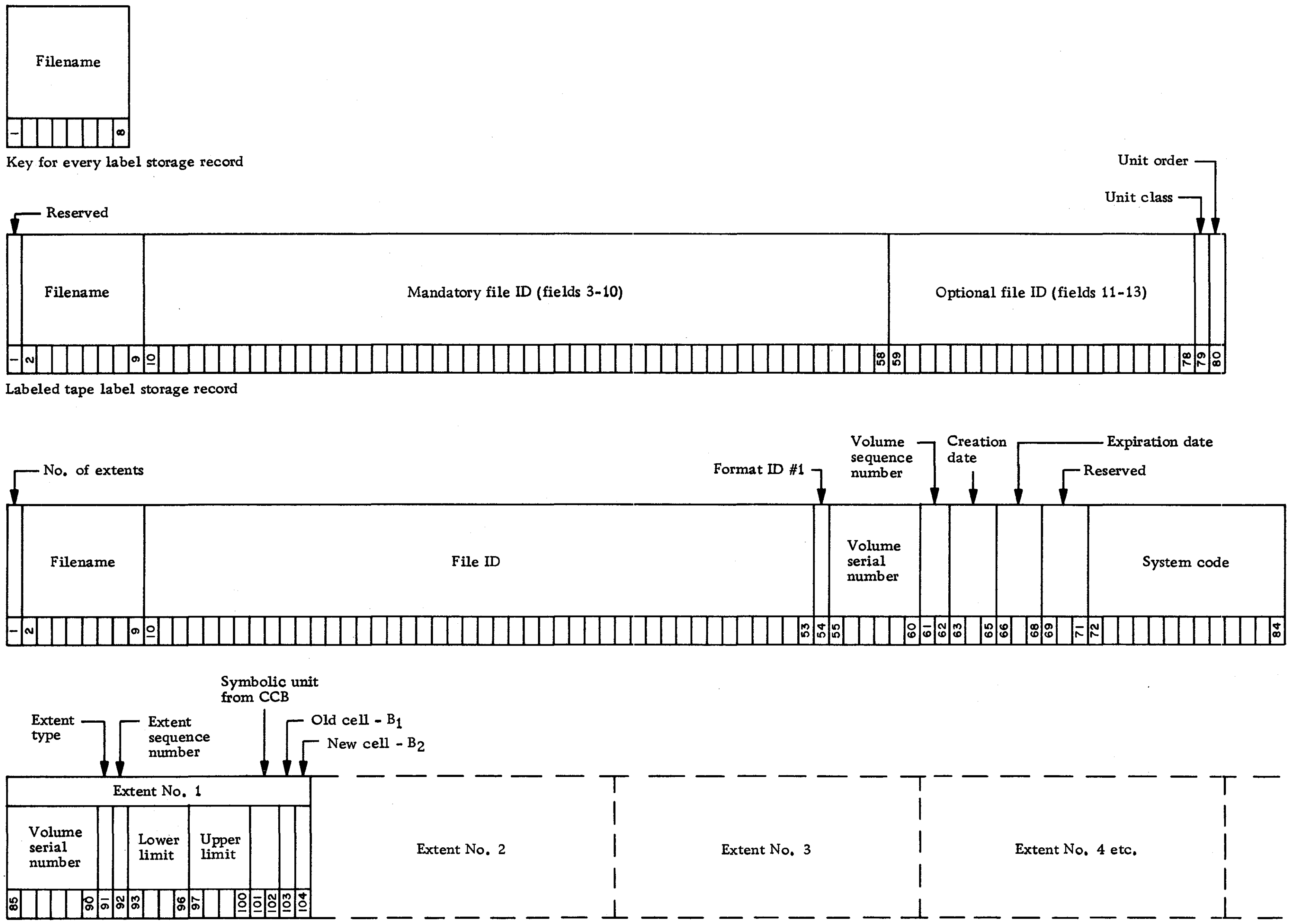


```
        // JOB Link-edit user program
        // OPTION CATAL
           [User program preceded ]
           [by PHASE and INCLUDE  ]
        /*
  (1)   // LBLTYP NSD (01)
  (2)   // EXEC LNKEDT
        /&
        // JOB Execute user program
      { // VOL    }
      { // DLAB   }  File 1
      { // XTENT  }  (Seq disk)
      { // VOL    }
  (3) { // DLAB   }  File 2
      { // XTENT  }  (Nonseq disk)
      { // VOL    }  File 3
      { // TPLAB  }  (Labeled tape)
  (4)   // EXEC User program
           [data]
        /*
        /&
```

Sequence of Events:

1. Reserve object core for label blocks with // LBLTYP.
2. Catalog user program to CIL.
3. Write label information to label storage area.
4. Bring program into core and process OPEN-CLOSE.
   a. Bring label storage data into core.
   b. Bring labels into core.
   c. Compare them.

Figure 4. Tape and DASD label system flow



DASD Label storage record

1. Sequential -- 104 bytes/extent
2. Nonsequential -- 84 bytes/file plus 20 for each extent.

Figure 5. Tape and DASD label storage record formats

The OPEN and CLOSE routines also retrieve the actual file labels. For input disk files the 44 byte alphameric file name is extracted from the label storage record and is used to locate and read the data set format 1 labels from the VTOC.

For input files a comparison is then made between the label storage information and the file label to ensure that the proper file is being accessed. For output files the label storage data is used to create the actual file label, provided that the file area desired is available. For specific details see IBM System/360 Disk Operating System, Supervisor and I/O Macros (C24-5037) and IBM System/360 Disk Operating System, Data Management Concepts (C24-3427).

Main Storage Usage

The actual file labels from the device are brought into the transient area of main storage regardless of the type of files: labeled tape, sequential DASD, or nonsequential DASD.

The label storage data for a sequential DASD file is also brought into the transient area. The label storage data for tape and nonsequential DASD files is read into the low core area of the processing program. Therefore, main storage must be reserved by the user in the problem program if labeled tape files or nonsequential DASD files are to be processed. The LBLTYP statement accomplishes this. For further details see the discussion of the LBLTYP statement in the section entitled "Linkage Editing".

DESCRIPTION OF JOB CONTROL LABEL STATEMENTS

VOL Statement

```
        // VOL  SYSxxx,filename
```

A volume statement is used when checking or writing standard labels for a DASD or tape file. The VOL statement immediately precedes its respective DLAB and XTENT or TPLAB statements. Only one VOL statement is required per file.

SYSxxx represents the symbolic unit name associated with the file. This parameter is used only for tape files, since the XTENT statement supplies the SYSxxx for DASD files. However, the parameter must be present for all files. Any valid SYSxxx is satisfactory for DASD files.

The filename becomes the key for the label storage data set and has a direct relationship to the processing program as follows:

	Symbolic Unit	Filename
User Programs Written in		
Assembler	SYSxxx	DTF filename
COBOL	SYS000-SYS244	SYS000-SYS244
FORTRAN	SYS001-SYS011	IJSYS01-IJSYS11
	SYSIPT	IJSYSIN
	SYSPCH	IJSYSPH
	SYSLST	IJSYSLS
	SYSIN	IJSYSIN
PL/I	SYSxxx	DECLARE filename
RPG	SYSxxx	Filename from file description specification sheet
System Input/Output Units as used by Compilers, etc.		
	SYS001-SYS003	IJSYS01-IJSYS03
	SYS004	FILED (COBOL debug)
	SYSLNK	IJSYSLN
	SYSRES	IJSYSRES
	SYSPCH	IJSYSPH
	SYSLST	IJSYSLS
	SYSRDR	IJSYSIN
	SYSIPT	IJSYSIN
	SYSIN	IJSYSIN
File-to-File Utilities		
Input file	SYS004* (card or tape)	UIN
Output file	SYS005 (printer or tape) SYS006 (card)	UOUT

*DASD input/output may be SYS000-SYSnnn.

	Symbolic Unit	Filename
Disk Sort/Merge		
Sort or merge run		
Output file	SYS001*	FILEO
1st input file	SYS002	FILEA
2nd input file	SYS003	FILEB
3rd input file	SYS004	FILEC
4th input file	SYS005	FILED
Sort run only		
5th input file	SYS006	FILEE
6th input file	SYS007	FILEF
7th input file	SYS008	FILEG
8th input file	SYS009	FILEH
9th input file	SYS010	FILEI
Work area**	SYSxxx	FILEW
Tape Sort/Merge		
Merge run		
1st input file	SYS002	FILEA
2nd input file	SYS003	FILEB
3rd input file	SYS004	FILEC
4th input file	SYS005	FILED
5th input file	SYS006	FILEE
6th input file	SYS007	FILEF
7th input file	SYS008	FILEG
Output file	SYS001	FILEO
Sort run		
1st input file	SYS002	FILEA
2nd input file	SYS002	FILEB
3rd input file	SYS002	FILEC
4th input file	SYS002	FILED
5th input file	SYS002	FILEE
6th input file	SYS002	FILEF
7th input file	SYS002	FILEG
8th input file	SYS002	FILEH
9th input file	SYS002	FILEI
Output file	SYS001	FILEO

Alternates may be assigned to the same SYSnnn for any merge file, the output file, or a multivolume single input file sort. On multifile input to the sort, the switch to an alternate drive does not occur on an end-of-file condition.

TPLAB Statement

```
// TPLAB 'label fields 3-10'
// TPLAB 'label fields 3-13'
```

The TPLAB statement contains file label information for tape label checking and writing. The statement must immediately follow its respective VOL statement. The label fields are an image of the standard tape file header label. Only fields 3—10 are used by the DOS label processing routines, although fields 11—13 will be written on an output file. If fields 11—13 are present, the data immediately follows field 10 extending through position 71 with a continuation code (any nonblank character) in 72. The balance of the data is placed on a continuation statement beginning in position 16. Only one TPLAB statement per file is required regardless of the number of volumes involved.

DLAB Statement

```
                                        Column 72┐
// DLAB 'label fields 1-3',                      C

┌Column 16
↓
xxxx,yyddd,yyddd,'system code' ┌,SD ┐
                               │,DA │
                               │    │
                               │,ISC│
                               └,ISE┘
```

The DLAB statement contains file label information for DASD label checking and creation of the format 1 label. This statement immediately follows its respective VOL statement and is completed on a continuation statement. Only one DLAB statement is required per file.

Label fields 1-3 represent the 44-byte identification of the file, a 1 for format 1 identifier, and the 6-byte volume serial number for the first or only volume of the file. A continuation character (any nonblank character) must be in position 72.

*SYSnnn represents the symbolic logical units that must be assigned for tape input/output. If the file consists of more than one reel, the second reel must be placed on the same unit or on an alternate assigned to the same SYSnnn. There are no restrictions on SYSnnn assignments for disk input/output.

**Work area is a direct access (DA) disk file.

The continuation statement always begins in column 16 and contains the volume sequence number, the file creation date, and the file expiration date. The system code is not used by DOS and may be filled with blanks. The final parameter specifies the file type: SD for sequential, DA for direct access, ISC for index sequential load create, and ISE for index sequential other than load create. When DTFPH is used, SD represents single mounted volumes, and DA represents all volumes mounted.

XTENT Statement

$$// \text{XTENT type, sequence, lower, upper, 'serial no.', SYSxxx} \left[, B_2\right]$$

The extent statement defines each area or extent of a DASD file. One or more extent statements must follow the DLAB statement. The meanings of the parameters are:

type	1 = Data area (no split cylinder) 2 = Overflow area (ISFMS) 4 = Index area (ISFMS) 128 = Data area (split cylinder)
sequence	0-255 to indicate the sequence number of this extent within a multi-extent file. All files but ISFMS begin with 0. In ISFMS, 0 = the master index if present. If there is no master, 1 = the first extent.
lower,upper	The lower and upper limit of the extent (expressed as B_1 C_1 C_1 C_2 C_2 C_2 H_1 H_2 H_2)
'serial no.'	The volume serial number enclosed in apostrophes. This number must agree with the serial number of the volume that contains this extent.
SYSxxx	The symbolic address of the DASD unit on which this volume will be mounted
B_2	Currently assigned cell number (2321). $B_2 = B_1$ if omitted.

TLBL Statement

```
// TLBL filename, ['file-ID'] , [date] ,
             [ file-serial-number ] ,
             [ volume-sequence-number] ,
             [ file-sequence-number ] ,
             [ generation-number], [version-number]
```

The TLBL statement will be available in a future DOS release. Refer to IBM System/360 Disk Operating System, System Control and System Service Programs (C24-5036) and its most recent technical newsletters for information about the availability of this statement.

This single statement can be used to provide information for tape label checking and writing instead of the VOL, TPLAB combination. Only the filename parameter is required; the other parameters are optional. If they are omitted, a default action occurs. These optional parameters are positional. If they are omitted, the comma must still be written so that the scan routine can locate each condition. The comma is not required for omitted terminal parameters.

filename	1 to 7 characters; identical to the symbolic address of the program DTF that identifies the file

The following operands are optional. If an optional operand is omitted for an input file, no checking is done for that parameter. If one is omitted for an output file, the default option is assumed.

'file ID'	1 to 17 alphameric characters contained within apostrophes indicating the name associated with the file
Default:	The DTF file name is used.
date	For input -- creation date in the format yy/ddd, where ddd may be 1 to 3 characters; for output -- a 1 to 4 character retention period in the format dddd, or expiration date in the yy/ddd format. The current date is assumed as the creation date.
Default:	Zero retention period
file serial number	1 to 6 alphameric characters indicating the volume serial number for the first reel of the file. The field is right-justified and zero-padded if less than 6 characters.
Default:	The volume serial number of the first reel of the file is assumed.
volume sequence number	1 to 4 numeric characters in ascending order for each volume of a multiple volume file
Default:	BCD 0001 is assumed.
file sequence number	1 to 4 numeric characters in ascending order for each file of a multiple file volume
Default:	BCD 0001 is assumed.

generation number	1 to 4 numeric characters that modify the file ID
Default:	BCD 0001 is assumed.
version number	1 or 2 numeric characters that modify the generation number
Default:	BCD 01 is assumed.

SUMMARY OF DASD FILE LOCATION CONSIDERATIONS

The following points should be kept in mind for DASD files.

System input/output files (SYSRDR, SYSIPT, SYSLST, SYSPCH, SYSIN):

- Only one XTENT permissible
- Must be sequential (SD)
- Type of extent may be 1 (data area, no split cylinder) or 128 (data area, split cylinder).

System work files (SYSLNK, SYS001-SYS003):

- SYS003 and SYSLNK may occupy the same extent unless compile-and-execute mode of operation is desired.
- SYSLNK requires a single extent.

Any work file:

- An expiration date of 99365 is suggested to protect the file against use by another program in a multiprogramming environment.
- Subsequent programs may use these areas, since a logical IOCS CLOSE erases the reference to the label in the VTOC.
- Type of extent may be 1 (data area, no split cylinder) or 128 (data area, split cylinder).
- Must be sequential (SD)

Sequential files:

- The number of extents per volume is not limited, since a format 1 label can point to a format 3 label, which can point to another format 3 label.
- The number of volumes per file is not limited and may be on the same drive or different drives. Volumes may be mounted as needed.
- The extent location is not restricted and may be either type 1 (data area, no split cylinder) or type 128 (data area, split cylinder).
- The extent sequence number begins with 0, and the extents are made available in the order of the sequence number.

Direct access files:

- The maximum number of extents per volume is 16. A format 1 label can point to a format 3 label, but a format 3 cannot point to another format 3.
- A single file may occupy several volumes. All volumes must be online at execution time and assigned in XTENT statements to a sequential set of symbolic unit numbers, such as SYS006, SYS007, SYS008.
- Only extent type 1 (data area, no split cylinders) is permissible.

Index sequential files:

- If a master index is used, it may be located on the same volume as the logical file records, or on a different volume that will be online whenever the records are processed. The master index must immediately precede the cylinder index on a volume and may occupy successive cylinders, but may not be continued from one volume to another. It is always extent sequence number 0 and extent type 4 (index area).
- The cylinder index may be located on the same volume as the logical file records as long as it is on a separate cylinder, or it may be on a separate volume that will be online whenever this logical file is processed. The index may occupy successive cylinders but may not be continued from one volume to another. If a master index is present, the cylinder index must follow it immediately. This index must be defined as a separate extent and is always extent sequence number 1, extent type 4.
- The track index and cylinder overflow areas are included in the prime data area. The prime data area for a logical file must be contained within one extent on the volume. The extent area must span full cylinders (track 0 through track 9 for 2311 or track 19 for 2321). The extent type is 1 (data area, no split cylinders).
- A single file may occupy several volumes. All volumes must be online at execution time. The prime data area must continue from the last track of one volume to the first track (track 0) of cylinder 1 on the next volume so that the area is considered continuous. Therefore the VTOC for volumes 2 through (n-1) of the file must be on cylinder 0 and on other than cylinder 199 on the first volume.
- If an independent overflow area is used, the extent may be specified either on the same volume as the logical data records or on a different volume that is online. However, it must be contained in

a single volume. The extent type is 2 (overflow area).

- The extent sequence numbers for the prime data and independent overflow areas are 2 through n.
- The XTENT statement for the independent overflow tracks may be placed either before or after all of the XTENT statements for the prime data area. However, all XTENT statements must be submitted in order of consecutive ascending extent sequence numbers.

Files to be sorted:

- ISFMS files are not valid input to the sort program.
- Split cylinder data areas are not valid input to the sort program.

Example 4. Standard Label Storage

```
   // JOB STDLABEL
   *  WRITE STANDARD LABELS ON THE STANDARD LABEL TRACK
①  // OPTION STDLABEL
②  // VOL SYSRES,IJSYSRES
   // DLAB 'DOS SYSTEM RESIDENCE FILE                    1123456',        X
③                 0001,67001,99365,'              ',SD
④  // XTENT 1,0,000000001,000126009,'123456',SYSRES
   // VOL SYSLNK,IJSYSLN
   // DLAB 'SYSTEM WORK FILE NO. 0                       1123456',        X
                  0001,67001,99365,'              ',SD
   // XTENT 1,0,000134000,000148009,'123456',SYSLNK
   // VOL SYS001,IJSYS01
   // DLAB 'SYSTEM WORK FILE NO. 1                       1123456',        X
                  0001,67001,99365,'              ',SD
   // XTENT 128,0,000149000,000198002,'123456',SYS001
   // VOL SYS002,IJSYS02
   // DLAB 'SYSTEM WORK FILE NO. 2                       1123456',        X
                  0001,67001,99365,'              ',SD
   // XTENT 128,0,000149003,000198005,'123456',SYS002
   // VOL SYS003,IJSYS03
   // DLAB 'SYSTEM WORK FILE NO. 3                       1123456',        X
                  0001,67001,99365,'              ',SD
   // XTENT 128,0,000149006,000198009,'123456',SYS003
⑤  // VOL SYSIN,IJSYSIN
   // DLAB 'COMBINED SYSRDR & SYSIPT                     1123456',        X
                  0001,67001,99365,'              '
   // XTENT 1,0,000127000,000129006,'123456',SYSIN
⑥  /&
⑦     ASSGN SYSIN,X'190'
```

Discussion

The purpose of this job is to build in the standard (permanent) label storage area sets of label storage data, which are accessible to any partition and which remain until replaced by another job using option STDLABEL. These particular labels represent the SYSRES volume, the work files required for compilations and linkage editing, and a combined SYSRDR-SYSIPT file on disk. Assume that only the latter file is to be added to those already in the label storage area.

① The STDLABEL option will cause all label data subsequently submitted to be written on the standard label track (0). Although only the label data for SYSIN needs to be added to the information already present, all label data sets must be resubmitted for any file still needed that was previously on the standard track, because writing starts at the beginning of the track.

② The VOL statement precedes the DLAB in every case. Note the use of IJSYSxx as the filename for these files.

③ The DLAB statement immediately follows its respective VOL statement. The following points are of interest concerning the parameters:

- 123456 corresponds to the volume serial number assigned when the volume was initialized.
- 99365 is used as the expiration date on the work files to protect against use of the area by another file.
- SYSLNK and SYS003 have separate extents to permit compile-and-execute mode. Faster compile time would result if two work files, such as SYS001 and SYS002, were placed on a second drive.
- SD could have been omitted, since the parameter is assumed. See label set for SYSIN. Only sequential files are acceptable for standard label storage.

(Continued on next page)

④ The XTENT statement follows its DLAB continuation statement. Note that work files 1 — 3 split cylinders 149 — 198 as follows: file 1 occupies tracks 0-2, file 2 has tracks 3-5, and file 3 has tracks 6-9. The concept of split cylinders takes advantage of two facts: (1) an entire cylinder can be accessed with no loss of time for access mechanism movement, and (2) movement time is related to the number of cylinders involved. Splitting cylinders between two or more files reduces the access arm movement. Only sequential files can occupy split cylinders. Since the beginning of all files is on the same cylinder, one setting will service all files initially. The advantage of split cylinders is greatest when the files are processed at about the same rate. However, even if they are not, the total number of cylinders moved may be less than that required when files are located in entirely separate cylinders. The VTOC for this volume is in cylinder 199.

⑤ This data set represents a combined SYSRDR-SYSIPT file on disk. If the data set were for SYSIPT or SYSRDR, only the SYSxxx entries would change; the IJSYSIN would remain. This indicates that file storage data can exist at one time for only one of the three possibilities: SYSRDR, SYSIPT, or SYSIN. SYSIN is the required name for a combined file.

⑥ Each VOL statement causes the preceding label set to be written into the label storage area. The /& writes the final label set and restores the system to the user label mode.

⑦ If the job input stream is on disk at this time, the ASSGN statement will open the file making use of the label storage data. Execution of the job stream on disk follows, since job control will look immediately to the disk for its next statement.

Example 5. Sequential Disk File Labels, Single Drive

```
   // JOB SAMLABEL
   *  LABELS FOR A MULTI-VOLUME SEQUENTIAL DISK FILE WITH ONE DRIVE
 ①// ASSGN SYS005,X'191'
   // VOL SYS005,FILENAME
 ②// DLAB 'ALPHAMERIC FILE NAME                         1000020',        X
                0001,67125,67155,'            '
   // XTENT 1,0,000132000,000150009,'000020',SYS005
   // XTENT 1,1,000008000,000081009,'000020',SYS005
   // XTENT 1,2,000127005,000133008,'000100',SYS005
   // XTENT 128,3,000050000,000140006,'000006',SYS005
 ③// EXEC PROG100
 ④/&
```

Discussion

PROG100 requires access to a sequential disk file located on three different packs that are to be mounted successively on the same drive. The job stream shown in this example illustrates the label statements required.

① Because of the assumed configuration, SYS005 need not be assigned. The statement is included only to emphasize the relationship between assignment and the label statements. The XTENT statements indicate the SYSxxx required.

② Although this file consists of four extents on three packs, only one VOL and one DLAB statement are required. The volume serial number in the DLAB statement corresponds to the volume serial number for the first extent. One XTENT statement is submitted for each of the four extents, and the statements must be placed into the job stream in extent number sequence. One of the extents occupies a split cylinder area, to illustrate that this is acceptable for sequential files. All of the label statements will be written on the background temporary track, since USRLABEL mode is in effect. This will be the only data in that area, since writing starts at the beginning of track 1.

③ Logical IOCS in PROG100 opens the first extent by using the filename in the VOL statement, the alphameric file name in the DLAB statement, and the volume serial number and SYSxxx unit supplied in the first XTENT statement (0 sequence number) to locate the actual label in the VTOC. When PROG100 has processed all of the data for the first extent, logical IOCS opens the second extent, based on the extent sequence number. Note that the first two extents are on the same volume.

When the open routine has checked the volume serial number (000100) in the third extent statement against the volume serial number (000020) in the volume label of the disk volume currently mounted on SYS005, it notifies the operator of the discrepancy. The first volume (000020) can be removed, and the new volume (000100) mounted on SYS005. The operator's reply causes the open routine to process the new volume.

④ The background label storage area is cleared when the /& statement is encountered. Therefore, if the next job requires the same file, the label statements must be resubmitted. See Example 10 for the relationship of label statements to multiple job steps within a job.

Example 6. Sequential Disk File Labels, Multiple Drives

```
    // JOB SAMLABEL
    *  LABELS FOR A MULTI-VOLUME SEQUENTIAL DISK FILE WITH MULTIPLE DRIVES
    // ASSGN SYS005,X'191'
①   // ASSGN SYS006,X'192'
    // ASSGN SYS007,X'193'
    // VOL SYS005,FILENAME
    // DLAB 'ALPHAMERIC FILE NAME                       1000020',      X
②                  0001,67125,67155,'         '
    // XTENT 1,0,000132000,000150009,'000020',SYS005
    // XTENT 1,1,000008000,000081009,'000020',SYS005
    // XTENT 1,2,000127005,000133008,'000100',SYS007
    // XTENT 128,3,000050000,000140006,'000006',SYS006
③   // EXEC PROG100
    /&
```

Discussion

This example has the same requirements as Example 5 except that the three volumes are mounted on three different drives.

① To make this example realistic, two more disk drives have been added to the assumed configuration, and assignment statements are submitted. The drives involved are in no way restricted. The SYSnnn and the actual device address need not be consecutive. However, SYSnnn must correspond to the SYSnnn represented in the XTENT statements.

② All label statements submitted are identical to Example 5 except for SYSnnn in the XTENT statements.

③ Logical IOCS opens each extent in the same way as described in Example 5, except that processing does not stop for removal and mounting of packs, since enough devices are online to contain the file. A combination of Examples 5 and 6 could be used to reduce handling time without excessively increasing the total drive requirements.

Example 7. Direct Access File Labels

```
    // JOB DALABEL
    *  LABELS FOR A MULTI-VOLUME DIRECT ACCESS FILE
    // ASSGN SYS005,X'191'
(1) // ASSGN SYS006,X'193'
    // ASSGN SYS007,X'192'
    // VOL SYS005,FILENAME
    // DLAB 'ALPHAMERIC FILENAME                             1000065',         X
(2)                 0001,67125,67200,'                 ',DA
    // XTENT 1,0,000132000,000150009,'000065',SYS005
    // XTENT 1,1,000008000,000081009,'000065',SYS005
    // XTENT 1,3,000050000,000140006,'000025',SYS006
    // XTENT 1,2,000127005,000133008,'00010?',SYS007
(3) // EXEC PROG101
    /&
```

Discussion

The purpose of this example is to illustrate the label statements as they apply to a direct access file.

(1) The addition of two more disk drives to the configuration is assumed, and assignments are made.

(2) Only one VOL and one DLAB statement are required. These statements follow the same pattern as for sequential files, except that the DA specification is required to identify the file as direct access. One XTENT statement is submitted for each of the extents, and the statements are placed in the job stream in consecutive ascending SYSnnn sequence. No skips in the SYSnnn numbers are allowed, although the sequence may begin with any SYSnnn. Only type 1 extents (data area, no split cylinders) are permitted, although up to 16 extents per volume are acceptable. All label storage records are written in the temporary background area, since USRLABEL mode is assumed in this example.

(3) PROG101 opens all extents in all volumes. When all of the volumes have been opened, the file is ready for processing. Consequently, all volumes must be online and ready at the same time.

Example 8. Index Sequential File Labels, Load Create

```
    // JOB ISLABEL
    *  LABELS FOR A MULTI-VOLUME INDEXED SEQUENTIAL FILE
    *  NO MASTER INDEX, CYLINDER INDEX ON SAME VOLUME AS DATA RECORDS
    *  NO INDEPENDENT OVERFLOW AREA, LOAD CREATE
    // ASSGN SYS005,X'191'
 ① // ASSGN SYS006,X'192'
    // ASSGN SYS007,X'193'
    // VOL SYS007,FILENAME
    // DLAB 'ALPHAMERIC FILENAME                          1000010',        X
 ②                0001,67125,67225,'            ',ISC
    // XTENT 4,1,000120005,000130009,'000010',SYS007
    // XTENT 1,2,000008000,000199009,'000031',SYS005
    // XTENT 1,3,000001000,000199009,'000028',SYS006
    // XTENT 1,4,000001000,000100009,'000C10',SYS007
 ③ // EXEC PROG102
    /&
```

Discussion

In this example PROG102 contains routines to create an indexed sequential file that is to be located on multiple volumes with no independent overflow area and no master index. The job stream reflects the label statements required for this operation.

① Two additional disk drives are added to the assumed configuration for illustrative purposes.

② The VOL and DLAB statements follow the usual pattern except that ISC is used to define the file operation as indexed sequential load create. The following points are relative to the XTENT statements:

- The cylinder index uses the option of being on the same volume as data records. However, it cannot occupy the same cylinder as data records nor extend into another volume. It can occupy successive cylinders. The extent type is 4, and the sequence number is 1 (0 is reserved for a master index).

- The prime data area follows the rules to provide a single continuous extent, although three XTENT statements are required. These rules are:
  a. Extent type 1
  b. Utilization of entire cylinders (tracks 0-9)
  c. When multiple volumes are needed, location through cylinder 199 of a volume and continuation with cylinder 1 of the next volume.

- The XTENT statements are submitted in extent number sequence.

③ The logical IOCS routines for PROG102 require that all volumes be online and ready at the same time. After all volumes and extents have been opened, the file is ready for processing.

Example 9. Index Sequential File Labels, Other than Load Create

```
    // JOB ISLABEL
    *  LABELS FOR A MULTI-VOLUME INDEXED SEQUENTIAL FILE
    *  MASTER AND CYLINDER INDEX ON SEPARATE VOLUME FROM DATA RECORDS
    *  INDEPENDENT OVERFLOW AREA, NOT LOAD CREATE
    // ASSGN SYS005,X'191'
    // ASSGN SYS006,X'192'
 ①  // ASSGN SYS007,X'193'
    // ASSGN SYS008,X'190'
    // VOL SYS008,FILENAME
    // DLAB 'ALPHAMERIC FILENAME                              1123456',       X
                   0001,67125,67185,'             ',ISE
    // XTENT 4,0,000130005,000131001,'123456',SYS008
    // XTENT 4,1,000131002,000133009,'123456',SYS008
    // XTENT 1,2,000008000,000199009,'000031',SYS005
 ②  // XTENT 1,3,000001000,000199009,'000028',SYS006
    // XTENT 1,4,000001000,000100009,'000010',SYS007
    // XTENT 2,5,000120005,000130009,'000010',SYS007
    // EXEC PROG1C3
 ③  /&
```

Discussion

This example illustrates the label statements required for a previously created index sequential file that has both a master index and an independent overflow area and requires multiple volumes for the data records.

① Appropriate assignments are made.

② The VOL and DLAB statements are the same as for any DASD file except for the ISE parameter, which indicates the file type as indexed sequential other than load create. The XTENT statements follow the rules outlined in Example 8 plus the following:

- The master index uses the option of being on a separate volume from the data records. As in the case of the cylinder index, it can occupy successive cylinders but cannot extend into another volume. The extent type is 4, and the sequence number is 0.

- Once the master index location is selected, the cylinder index must follow it immediately.

- The independent overflow area is identified by extent type 2. In this example it is located on the same volume as the data records but, like the master and cylinder indices, it can be on a separate volume. It can occupy successive cylinders but cannot extend into another volume. The XTENT statement could be placed in the job stream ahead of the prime data XTENT statements. This would require a change in the sequence number.

③ All volumes must be online at the same time. The logical IOCS of PROG103 opens all volumes and extents before making the file available for processing.

Example 10. Standard Tape Labels, Multivolume File with Alternate Assignments

```
    // JOB TAPELBL
    *  STANDARD TAPE LABELS, MULTI-VOLUME FILE WITH ALTERNATE ASSIGNMENTS
    *  LABEL STATEMENTS FOR SUCCESSIVE JOB STEPS
    // ASSGN SYS006,X'280'
 ①  // ASSGN SYS006,X'282',ALT
    // ASSGN SYS006,X'281',ALT
    // ASSGN SYS009,X'283'
 ②  // VOL SYS006,FILEA
    // TPLAB 'FILEA NAME        12345600010001000101 67125 67155'
 ③  // VOL SYS009,FILEB
    // TPLAB 'FILEB NAME        00012300010001000101 67125 671500      DOS SX
                     YSTEM
    // EXEC PROG200
    // EXEC PROG201
 ④  // EXEC PROG202
    /&
```

Discussion

This example shows the label statement requirements for tape files with standard labels. It also demonstrates the submission of label statement data in an earlier job step for use by a later job step, and discusses the effect of placing label statements in the job stream on later job steps. Assume that PROG200 requires a multiple volume output file named FILEA, that PROG201 used FILEA for input, and that PROG202 used FILEA for input and creates FILEB. These three programs are executed as successive job steps of a JOB named TAPELBL.

① Alternate assignments have been made for FILEA. Because alternation is in the sequence in which the ASSGN statements are submitted, volume 1 of the file must be mounted on drive 280, volume 2 on 282, volume 3 on 281, volume 4 on 280, etc.

② Although FILEA consists of multiple volumes, only one pair of VOL and TPLAB statements needs to be submitted. The SYS006 in the VOL statement is the same as the unit referenced by the respective program. FILEB illustrates the optional format of the TPLAB statement. These label storage records are available for use by any of the three programs. Remember that the fields in the TPLAB statement are surrounded by apostrophes and represent the following:

Field No. and Name	No. of Bytes	Remarks
3. File name	17	
4. File (volume) serial no.	6	
5. Volume sequence no.	4	Automatically incremented by IOCS for multivolume files
6. File Sequence no.	4	For multifile volumes; user must provide
7. Generation no.	4	Not used by DOS, but space must be accounted for
8. Version no.	2	
9. Creation date	6	Blank yyddd
10. Expiration date	6	Blank yyddd
11. File security	1	Not used by DOS; reserve space
12. Block count	6	Used in trailer label only
13. System code	13	Not used by DOS; reserve space

Fields 3-10 are required; fields 11-13 are optional.

③ The VOL and TPLAB statements for FILEB are submitted at this time although not required until PROG202. If they were placed between the EXEC for PROG201 and PROG202, another pair of label statements would have to be submitted to PROG202, since the entire background temporary label storage area would be overlaid and the FILEA label storage data lost.

THE UTILITY PROGRAMS

GENERAL CONSIDERATIONS

Logical Unit Assignments

Appendix G, parts 4 and 5, lists the symbolic units required for IBM-supplied utility programs.

The utility programs require four system symbolic unit assignments:

SYSIPT	Utility control statement input
SYSLOG	Operator messages
SYSLST	Programmer messages
SYSRDR	Job control statements

Normally, ASSGN statements are required for the input and output files, since the utility programs expect the file to be on specific SYSnnn units. These requirements can be remembered easily as follows:

SYS004	Input file
SYS005	Output file (except card)
SYS006	Card output file

DASD units are not restricted to the use of SYS004 and SYS005, since any logical SYSnnn may be assigned.

Program Names

The available utility programs and their names are easy to remember if the following rules are kept in mind.

- Standard abbreviations for the five device types are:

CD	Card
TP	Tape
DK	Disk
DC	Data cell
PR	Printer

- Two abbreviations are combined to make up the program name. The first represents the input device, the second, the output device. Thus, CDTP means card-to-tape.
- All combinations are provided except certain card operations. Not provided are:

CDPR CDCD	CDPP is substituted (card to printer and/or punch)
CDDC DCCD	Card and data cell transfers

- Three special utilities are:

TPCP	Tape Compare
CLDC	Clear Data Cell
CLRDSK	Clear Disk

Labels

Standard labels are required for DASD files but are optional for tape. UIN is the input file name, and UOUT is the output file name. For standard label checking, the utility programs require only the submission of the proper job control label statements.

If nonstandard, user, or no label checking is to be performed for tape files, a user program switch indicator (UPSI) job control statement must be submitted.

UPSI Statement

```
// UPSI  nnnnnnnn
```

The UPSI (user program switch indicator) statement allows the user to set program switches that can be tested at execution time. The operand consists of one to eight characters that represent, from left to right, bits 0 through 7 in the UPSI byte in the communication region. Either a /& or a JOB statement resets the UPSI byte to zero.

The UPSI statement permits the user to set values in the UPSI byte according to the following rules:

1. Positions containing 0 will be set to 0.
2. Positions containing 1 will be set to 1.
3. Positions containing X will be unchanged.
4. Unspecified rightmost positions are assumed to be X.

The specific meaning attached to each bit in the UPSI byte depends upon the problem program design. Any combination of the eight bits may be tested by the problem program.

The UPSI byte has the following meaning to the utility programs:

	Bit	Value 0	Value 1
Input tape	0	Standard input label checking	Nonstandard or no input label checking
	1	No user input label checking	User input label checking
Output tape	2	Standard output label checking	Nonstandard or no output label checking
	3	No user output label checking	User output label checking
Output TM handling (nonstandard or no label checking)	4	Leading TM or TM to separate label and data	No leading TM or no TM to separate label and data

Therefore, no UPSI statement is required for standard tape label checking. For no tape labels and a leading TM, the statement will be:

```
// UPSI 101
```

Note: For the convenience of the reader, this page and others in this publication have been left blank to permit printing the longer examples on facing pages.

PREPARATION OF JOB STREAM INPUT ON TAPE OR DISK (EXAMPLES 11 AND 12)

The assignment of both SYSRDR and SYSIPT to the same card reader is the most common system configuration. However, the Disk Operating System permits either or both of these files to be assigned to magnetic tape units or disk extents. The IBM-supplied utility macros or file-to-file utility programs can be used to prepare magnetic tapes or disk extents to be used as SYSRDR, SYSIPT, and/or SYSIN. The job stream data is written in card image format (80-character records, unblocked) and, if on disk, without keys. Standard labels are required for disk but are optional for tape.

Example 11. Creation of SYSIPT on Tape

```
    // JOB BUILDIPT                                                     ┐
    *  BUILD SYSIPT ON TAPE WITH STANDARD LABELS                        │
 ①  // ASSGN SYS004,X'00C'                                              │
    // ASSGN SYS005,X'282'                                              ├ Input to SYSRDR
 ②  // VOL SYS005,UOUT                                                  │
    // TPLAB 'SYSIPT TAPE      12345600010001000101 67001 67002'        │
 ③  // EXEC CDTP                                                        ┘
 ④  // UCT TC,FF,A=(80,80),B=(80,80),OR                                 ┐ Input to SYSIPT
    // END                                                              ┘
         FORTRAN source statements      ┐                               ┐
         COBOL source statements        │                               │
                                        ├ Job A input                   │
    /* SOME PUNCH                       │                               │
 ⑤  /& SOME PUNCH                       ┘                               │
                                                                        │
         RPG source statements          ┐                               ├ Input to SYS004
                                        │                               │
    /* SOME PUNCH                       │                               │
         Assembly source statements     ├ Job B input                   │
    /* SOME PUNCH                       │                               │
         FORTRAN source statements      │                               │
    /& SOME PUNCH                       ┘                               ┘
 ⑥  /*                                                                  ┐ Input to SYSRDR
    /&                                                                  ┘
```

Discussion

This example creates on tape a SYSIPT file with standard labels by use of the card-to-tape utility program. The file contains the data that is used in Example 3 for SYSIPT.

① Temporary assignments are made for the input and output files in accordance with the utility program specifications. The system units are assigned already because of the assumed configuration.

② VOL and TPLAB statements are submitted to provide label data. This presupposes that the tape used for output has a standard label set with a past expiration date. Otherwise, messages will occur. Although the labels will be written (thus preserving labels on labeled tape), job control will not actually check the label when the tape is used later as SYSIPT. Thus it will be the user's responsibility to submit the proper tape. See Example 3.

③ The program name for the card-to-tape utility is CDTP.

④ The // UCT and // END statements represent the control statement input, which specifies the particular file conditions to the generalized utility program. Details on the control statement requirements are given in IBM System/360 Disk and Tape Operating Systems, Utility Programs Specifications (C24-3465).

The statements must be read from the device assigned to SYSIPT. The // END statement is required because it signals the end of the utility control statements.

(Continued on next page)

⑤ Following the END statement, the utility program looks to SYS004 for the input file data. Since this is the card reader, the data to create the SYSIPT file follows immediately. These points are of interest:

- No /* follows FORTRAN source statements (see earlier discussion under /* statement).
- /* statements do follow other language translator source statements.
- /& statements define the end of input for each job to handle abnormal end-of-job situations. All /* and /& statements must have some punch in columns 4-80; otherwise, the utility program will not accept them as data. Position 3 must be blank to permit proper action when the data serves as SYSIPT.

⑥ The /* without any punch in positions 3-80, signals the end of card data for the utility program. The /& performs the usual end-of-job functions.

If no labels were required, this example would be altered as follows:

1. Remove the VOL and TPLAB statements.
2. Insert a // UPSI 001 statement. Note: A 101 parameter is also acceptable, since no input tape file is involved.

By following the principles outlined in this example, a separate SYSRDR or combined SYSRDR/SYSIPT file can be created.

Example 12. Creation of SYSIN on Disk

```
    // JOB BUILDIN
    *  BUILD COMBINED SYSRDR & SYSIPT ON DISK
 ①  // ASSGN SYS004,X'OCC'
    // ASSGN SYS005,X'190'
    // VOL SYS005,UOUT
    // DLAB 'COMBINED SYSRDR & SYSIPT                          1123456',         X
 ②                 0001,67001,99365,'              '
    // XTENT 1,0,000127000,000129006,'123456',SYS005
 ③  // EXEC CDDK
 ④  // UCD TC,FF,A=(80,80),B=(80,80),OY
    // END
    // JOB A                                                       ⎫
         .                                                         ⎪
         .       Other job control statements as required          ⎪
         .                                                         ⎪
    // EXEC PROGA1                                                 ⎪
         .                                                         ⎪
         .       Data for A1 if required                           ⎪
         .                                                         ⎪
    /* SOME PUNCH                                                  ⎪
         .                                                         ⎪
         .       Job control statements as required                ⎪
         .                                                         ⎪
 ⑤  // EXEC PROGA2                                                 ⎪
         .                                                         ⎪
         .       Data for A2 if required                           ⎬ Input to SYS004
         .                                                         ⎪
    /* SOME PUNCH                                                  ⎪
    /& SOME PUNCH                                                  ⎪
    // JOB B                                                       ⎪
         .                                                         ⎪
         .       Other job control statements as required          ⎪
         .                                                         ⎪
    // EXEC PROGB                                                  ⎪
         .                                                         ⎪
         .       Data for B if required                            ⎪
         .                                                         ⎪
    /* SOME PUNCH                                                  ⎪
    /& SOME PUNCH                                                  ⎪
 ⑥     CLOSE SYSIN,X'00C'                                          ⎭
    /*
 ⑦  /&
```

Discussion

The card-to-disk utility is used in this example to create a combined SYSRDR and SYSIPT file on disk. The resultant file can be used in conjunction with Example 4.

① Temporary assignments are made for the input and output files in accordance with the utility program specifications. Since disks can be assigned to any SYSnnn, the disk could have been assigned to any SYSnnn other than SYS004, which is reserved for the input unit. The system units are assigned already because of the assumed configuration.

② VOL, DLAB and XTENT statements are submitted to provide label data. The system file is sequential. It must occupy a single extent, but the extent type may be 1 (no split cylinder) or 128 (split cylinder).

③ The program name for the card-to-disk utility is CDDK.

④ The // UCD and // END statements represent the control statement input, which specifies the particular file conditions to the generalized utility

(Continued on next page)

program. Details on the control statement requirements are given in IBM System/360 Disk and Tape Operating Systems, Utility Program Specifications (C24-3465).

⑤ Following the END statement, the utility program looks to SYS004 for the input file data. Since this is the card reader, the data to create SYSIN follows immediately. See ⑤ in Example 11 for further details.

⑥ A CLOSE statement is included at the end of the file to perform the requirement of closing the system file and returning the assignment to the card reader.

⑦ The /*, without any punch in positions 3-80, signals the end of card data for the utility program. The /& performs the usual end-of-job functions.

Operating Considerations

When SYSRDR, SYSIPT, or SYSIN are on tape or disk, certain operating conditions should be considered:

- After a SYSIN or SYSRDR job stream has been prepared on tape or disk, it may be necessary to interrupt the normal schedule to execute a special rush job. By inserting a PAUSE statement (//) between jobs -- that is, between the /& and the // JOB statement -- at the time the job stream is created, control will come to the operator immediately, before the // JOB statement is read. At this point a temporary assignment for SYSIN can be made to the card reader. The temporary assignment will not be reset by the // JOB statement in the card reader but will revert to the permanent tape or disk assignment when the /& statement is read in the card reader. The original stream will restart at the point of interruption. Because this method anticipates interruption, the pause may have to be nullified by entering the end-of-communication code, Ⓑ .
- Care must be taken in the use of the ATTN PAUSE or PAUSE command for the purpose of changing assignments, since they bring control to the operator the next time job control is in core. This will occur at the end of a job step and not necessarily at the end of a job.
- Remember that when the system opens a SYSRDR, SYSIPT, or combined SYSRDR/SYSIPT file on disk, it always refers to IJSYSIN. Therefore, although several such files may be present on disk simultaneously, each must have a unique file name, and the label storage area may reference only one of the files at a time.

LINKAGE EDITING (EXAMPLES 13 TO 18)

Linkage editing provides the user with the capability of combining separately assembled or compiled program sections or subprograms. In order to make this possible, the output of the language translators is not yet in executable form. Rather, the separate program sections are in relocatable form; that is, the address constants can be modified to allow later assignment of origin point. The linkage editor is the processing program that performs the function of linking and relocating separate program sections into a single module that can be loaded by the control program and then executed.

Every relocatable program must be processed by the linkage editor before it can be executed. Once a program has been edited, it can be executed immediately, it can be cataloged as a permanent entry in the core image library, or it can be both cataloged and executed immediately. When a program has been cataloged in the core image library, the linkage editor is no longer required for that program, since it can be loaded directly from the resident pack by the system loader of the control program. On the other hand, if a program is edited and executed immediately without cataloging, the linkage editor is required again the next time the program is to be executed. Cataloging is a system design decision based on such factors as frequency of use and space available in the core image library.

SYSTEM FLOW

The system flow schematic for the linkage editor program is shown in Figure 6.

Before the linkage editor program is executed, job control must perform certain functions. It must:

- Process the OPTION statement.

OPTION LINK or CATAL turns on control program switches. Unless these switches are on, the linkage editor control statements are invalid.

- Copy linkage editor control statements to SYSLNK.

The linkage editor control statements are ACTION, PHASE, INCLUDE, and ENTRY. The ACTION, PHASE, and ENTRY statements are copied. There are two forms of the INCLUDE statement. INCLUDE's with no operand are not copied but cause the data (object module) on SYSIPT to be written until the end-of-data (/*) occurs. If the object module to be link-edited has been cataloged in the relocatable library, the INCLUDE statement must have the name of the module as an operand. This format of the INCLUDE statement is copied, but not the module.

- Write an ENTRY statement with a blank operand if the job stream does not already contain one.

When the EXEC LNKEDT statement is encountered, an ENTRY statement is created to ensure termination of the linkage edit input.

- Inform the system loader to load the linkage editor program.

The linkage editor program uses the data on SYSLNK as input. It handles the relocatable modules as directed by the PHASE and INCLUDE statements to develop executable program phases. Some of the linkage editor program functions are:

- Extraction from the relocatable library of modules named in INCLUDE statements.

If, in extracting a module, another INCLUDE statement occurs, this module is also retrieved. The nesting of modules is possible up to a depth of five (a level of six).

- Construction of composite dictionaries for ESD and RLD data, to resolve all linkages between different control sections

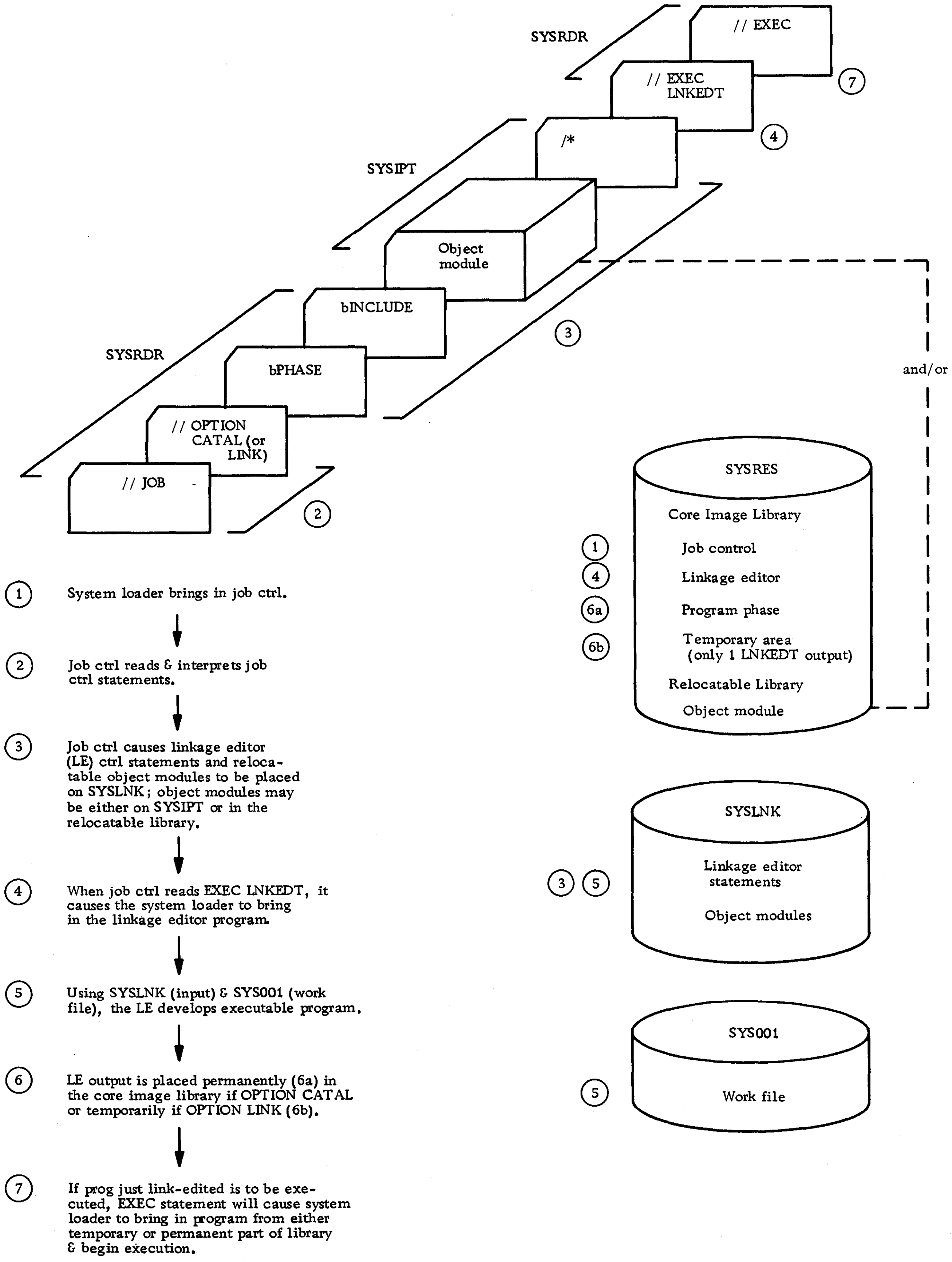
- Relocation of each control section as necessary within a phase

- Assignment of the entire phase to a contiguous area of main storage

- Modification of all relocatable address constants to contain the relocated value of their symbols

- Search of the relocatable library for a cataloged object module with the same name as each unresolved external reference.

The automatic library lookup feature (AUTOLINK) is particularly useful for retrieving IOCS modules. It may be suppressed.



1. System loader brings in job ctrl.

2. Job ctrl reads & interprets job ctrl statements.

3. Job ctrl causes linkage editor (LE) ctrl statements and relocatable object modules to be placed on SYSLNK; object modules may be either on SYSIPT or in the relocatable library.

4. When job ctrl reads EXEC LNKEDT, it causes the system loader to bring in the linkage editor program.

5. Using SYSLNK (input) & SYS001 (work file), the LE develops executable program.

6. LE output is placed permanently (6a) in the core image library if OPTION CATAL or temporarily if OPTION LINK (6b).

7. If prog just link-edited is to be executed, EXEC statement will cause system loader to bring in program from either temporary or permanent part of library & begin execution.

Figure 6. Link-edit system flow schematic

• Building of core image directory phase headers and cataloging to the core image library, if CATAL is specified.

If a phase by the same name has been cataloged previously, the old phase is deleted and the new one is cataloged. Deletion removes the item from the directory but does not release the space in the library until a condense function occurs.

SYMBOLIC UNITS REQUIRED AND PROGRAM NAME

The symbolic units required by the linkage editor are basically a subset of those needed by the language translators:

SYSIPT	Program input, unless already in the relocatable library
SYSLST	If MAP action is desired
SYSLOG	Operator messages
SYSRDR	Job control statements
SYSLNK	Input
SYS001	Work file

The program name is LNKEDT.

LINKAGE EDITOR CONTROL STATEMENTS

Position 1 must be blank on linkage edit control statements. Otherwise, they follow the same format as job control commands.

ACTION Statement

$$\text{ACTION} \left\{ \begin{array}{l} \text{CLEAR} \\ \text{MAP} \\ \text{NOMAP} \\ \text{NOAUTO} \\ \text{CANCEL} \end{array} \right\}$$

This statement specifies linkage editor options. It is not required, but if used must appear as the first linkage editor statement in the input stream. Multiple statements are required if multiple operands are necessary. The ACTION statement is effective only for the next linkage editor execution. The parameters have the following meanings:

CLEAR	Set to binary zero the area of the core image library used by this link-edit operation. Since this is time-consuming, it should be used only if the areas defined by DS statements must be filled with zeros.
MAP	Output main storage map and error diagnostics on SYSLST. Whenever SYSLST is assigned, MAP is automatic unless NOMAP is specified.
NOMAP	Nullify MAP action.
NOAUTO	Suppress AUTOLINK function for the entire program, not just one phase.
CANCEL	Cancel the job if the content of the linkage editor input is in error. See messages 2100I to 2170I listed in IBM System/360 Disk Operating System, Operating Guide (C24-5022).

PHASE Statement

PHASE name, origin [, NOAUTO]

A program phase is the section of a program that is loaded by the system loader as a single overlay with a single FETCH or LOAD. The input for building a single phase consists of the text from one or more complete control sections. Therefore, programs may consist of many phases, or a phase may consist of many subprograms or control sections.

The PHASE statement provides the linkage editor with the phase name and an origin point for the phase. The phase name is used to catalog the phase in the core image library and to retrieve it for execution. Job control uses the phase name to construct a single-track phase directory before each job step is executed. The entries to this directory are taken from the core image directory for any phase where the first four characters of the name are identical to those in the name specified in the EXEC statement. The directory entry contains such information as loading address, entry point, and starting disk address in the core image library. The separate phase directory permits faster retrieval of the phases.

The entries in the operand field represent the following:

name	Symbolic name of the phase, consisting of 1 to 8 alphameric characters. The first 4 characters of a multiphase program should be the same to achieve maximum retrieval efficiency.
origin	Specification of the load address of the phase. The load address can be one of six forms: 1. symbol [(phase)] [±relocation] This format is used to specify an origin point for a phase at the same point as a previously defined symbol (for overlays).

2. * [±relocation]
   This is the most frequently used format and specifies an origin point for a phase at the next available core location.
3. S [+ relocation]
   The origin point will be at the end of the supervisor, the label block area, if any, and the area assigned to the COMMON pool, if any.
4. ROOT
   This phase is designated as the root phase, which will remain in core throughout execution. Its location is the same as with format S.
5. + displacement
   The origin point is set at a specified location; + 0 must be used for any self-relocating program.
6. F+ address
   This is the format for non-self-relocating foreground programs.

A detailed explanation of the origin parameter is given in <u>IBM System/360 Disk Operating System: System Control and System Service Programs</u> (C24-5036). Rather than repeat the information here, the use of this parameter is demonstrated by the linkage edit examples at the end of this section.

NOAUTO	Suppress the AUTOLINK function for this phase only.

<u>INCLUDE Statement</u>

INCLUDE [modulename] [, (namelist)]

This statement specifies that an object module is to be included for editing by the linkage editor. The system assumes the location of the module as follows:

1. Both operands missing -- the object module is on SYSIPT; it is copied to SYSLNK.
2. Modulename given -- the object module is cataloged in the relocatable library under the same name.
3. Second operand only given -- the object module is in the input stream on SYSLNK.

The parameters represent the following:

modulename	This parameter is used only if the module has been cataloged to the relocatable library. It consists of 1 to 8 alphameric characters and must be the same as the name used when the module was cataloged.
(namelist)	This parameter provides the ability to select only particular control sections from a given module; it is expressed as (csnamel, csname2, ...).

<u>ENTRY Statement</u>

ENTRY [entrypoint]

The ENTRY statement signals the end of program input to the linkage editor. The entrypoint operand indicates the transfer address for the first phase as follows:

1. If it is omitted, the first significant address in an END record is used; or, if no such address is found, the load address of the first phase is used.
2. If given, it must be the name of a CSECT or a label definition defined in the first phase.

The ENTRY statement can be completely omitted if action 1 is desired, since job control automatically writes an ENTRY statement with a blank operand when it encounters the EXEC LNKEDT statement.

<u>LBLTYP Job Control Statement</u>

$$// \text{ LBLTYP } \left\{\begin{matrix}\text{TAPE (nn)}\\ \text{NSD (nn)}\end{matrix}\right\}$$

The label storage records for standard labeled tape files and nonsequential DASD files (direct access, indexed sequential, or DTFPH with all packs mounted) are brought into the low main storage area of the processing program involved (refer to the section on tape and DASD labels). Therefore, main storage must be reserved by the user whenever such files are to be processed. Since this area is used during OPEN for one file at a time, the total area needed is that required by the largest file.

This main storage reservation is accomplished by the LBLTYP statement. The amount of main storage reserved is governed by the operand TAPE or NSD as follows:

- TAPE will reserve 80 bytes of main storage.

This format should be used when standard labeled tape files and <u>no</u> nonsequential DASD files are to be processed. nn is ignored by job control. This same 80-byte area is used by all labeled tape files.

- NSD will reserve 84 bytes plus 20 bytes per extent.

The number of extents is specified in the nn parameter for the nonsequential file that has the largest number of extents. This format is used when nonsequential DASD files are processed, regardless of whether labeled tapes are to be processed also. This same area is used by nonsequential DASD files with fewer extents, and by labeled tape files.

The LBLTYP statement is not required if only unlabeled tape files and/or sequential DASD files are being processed. Only one LBLTYP statement is submitted. The placement of the statement in the job stream varies as shown:

- Non-self-relocating (background/foreground) -- immediately preceding the EXEC LNKEDT statement at link-edit time
- Self-relocating (background) -- ahead of the EXEC for the program
- Self-relocating (foreground) -- LBLTYP statement not used

Example 13. Link Edit and Execute

```
   // JOB LINKEXEC
   *  LINK EDIT AND EXECUTE, SINGLE PHASE, SINGLE OBJECT MODULE
   *  RELOCATABLE MODULE NOT CATALOGED, BACKGROUND PROGRAM
   *  NON-SEQUENTIAL DASD & LABELED TAPE FILES TO BE PROCESSED
  ①// ASSGN SYSLNK,X'190'
  ②// OPTION LINK
  ③   PHASE PROGA,*
      INCLUDE

  ④      Relocatable object deck

   /*
  ⑤// LBLTYP NSD(2)
  ⑥// EXEC LNKEDT

  ⑦      Any job statements required for execution as ASSGN or label statements

  ⑧// EXEC

         Data input as required

   /*
   /&
   *  1. TO CATALOG & EXECUTE CHANGE ② TO // OPTION CATAL.
   *  2. TO CATALOG ONLY CHANGE ② TO // OPTION CATAL & REMOVE ALL
   *     STATEMENTS FOLLOWING LNKEDT EXCEPT /&.
   *  3. TO USE MODULE FROM RELOCATABLE LIBRARY CHANGE ③ TO INCLUDE
   *     MODULEA & REMOVE ALL STATEMENTS UP TO // LBLTYP.
```

Discussion

This example illustrates the basic concept of link editing and executing by using the simplest program form--a single phase to be constructed from a single relocatable object deck contained in punched cards. The program will be executed in the background partition. Labeled tape and nonsequential DASD files are to be processed when the phase is executed. No more than two extents are used by any DASD file.

① No assignments are necessary, since the system units required for link editing are in the assumed configuration. However, an ASSGN for SYSLNK is included to illustrate its position relative to the OPTION statement in case assignment is required.

② The OPTION LINK statement sets switches to indicate that a link edit operation is to be performed. If SYSLNK has not been assigned, the statement is ignored. Linkage editor control statements will not be accepted unless the OPTION statement has been processed. Only link editing will be performed; no cataloging to the core image library will occur.

(Continued on next page)

③ The PHASE statement is copied to SYSLNK, since position 1 is blank and the link switch is on. The operands are not examined until SYSLNK becomes input to the linkage editor program.

When the PHASE statement is processed by the linkage editor, only one phase will be constructed, since only one PHASE statement has been submitted for the entire LNKEDT. The name for this phase is PROGA, as specified in the first operand. The second operand indicates the origin point for the phase. Since an * has been used, the phase will begin in the next main storage location available, with forced doubleword alignment. Because this is the first and only phase, it will be at the end of the supervisor, the label storage block area (reserved by LBLTYP), and any area assigned to the COMMON pool, (as designated by a CM entry in the relocatable module).

A relocation factor, either plus or minus, can be used with the *, such as *+1024. This causes the origin point of the phase to be set relative to the * by the amount of the relocation term. This term can be expressed as:

X'hhhhhh'	-- 1 to 6 hexadecimal digits
dddddddd	-- 1 to 8 decimal digits
nK	-- Where K = 1024

*+1024 uses the second format and adds 1024 bytes to the origin location. +1K or +X'400' gives the same result as +1024.

④ The INCLUDE statement has no operands, so the system will read the records from SYSIPT and write them on SYSLNK until SYSIPT has an end-of-data (/*) record. The data on SYSIPT is expected to be the object module in card image format, which is to be used in this link edit operation. If the output of the language translator (SYSPCH) is placed on tape or disk instead of cards, it cannot be used directly as SYSIPT in a link edit operation because the records contain a stacker select code in position 1. SYSPCH must be converted to an 80-position card image record.

⑤ The LBLTYP statement causes a computation of the number of bytes that will be required for label storage data in the program to be link edited. In this example, 124 bytes will be reserved (84 + (2x20)). The calculation is saved and passed on first to the linkage editor and later to LIOCS.

⑥ The EXEC LNKEDT writes an ENTRY statement with no operand on SYSLNK and causes the system loader to bring in the linkage editor program.

Using the data just placed on SYSLNK as input, the linkage editor develops executable code. The output is placed in the next available space of the core image library (immediately after the last cataloged phase). This is true regardless of whether the program is to be cataloged or not. Cataloging causes the updating of the directory to reflect a new ending point for the library. If cataloging does not occur, the next program that is link edited will overlay it. For this reason, a link edited program that is not cataloged is said to be placed in the temporary area of the core image library. For this reason also, a program that is link edited without cataloging must be link edited whenever it is used. No ACTION options are specified. Therefore, in resolving the external references, the system will make use of the AUTOLINK feature. Error diagnostics and a main storage map will be output to SYSLST, since SYSLST is assigned.

⑦ Since the program is not cataloged, it must be executed immediately. Any pertinent job control statements are entered at this point.

⑧ An EXEC statement with no operand indicates that the phase to be executed was just link edited. Therefore no search of the core image directory is required, and the system loader brings the program into core from the temporary area and transfers it to the entry point. In this example, the entry point is either the address specified in the END record, or the phase load address if the END address is omitted, because the automatic ENTRY statement is in effect.

(Continued on next page)

Example 13 (continued). Link Edit and Execute

This example can be modified to illustrate the following:

1. Catalog and execute. To cause this phase to be cataloged rather than merely link edited, change the OPTION (②) from LINK to CATAL. The name used in the PHASE statement may be added to the EXEC statement (// EXEC PROGA).

2. Catalog only. To catalog only, change the OPTION statement (②) from LINK to CATAL and remove all data following the EXEC LNKEDT statement (⑥) up to the /& statement.

3. Object module in relocatable library. The name used to catalog the object module into the relocatable library must be added to the INCLUDE statement. If the name is RELOCA, the statement becomes INCLUDE RELOCA. Of course the relocatable object deck and /* statement are removed. This form of the INCLUDE statement is written on SYSLNK when it is read by job control. The linkage editor retrieves the object module when it encounters the INCLUDE statement, since it uses SYSLNK for input.

Example 14. Catalog to Core Image Library

```
   // JOB CATALCIL
   *  LINK EDIT AND CATALOG TO CORE IMAGE LIBRARY
   *  SINGLE PHASE, MULTIPLE OBJECT MODULES, FOREGROUND PROGRAM
   *  MIXTURE OF CATALOGED & UNCATALOGED RELOCATABLE MODULES
   *  LABELED TAPE FILES & SEQUENTIAL DASD FILES TO BE PROCESSED
①  // ASSGN SYSLNK,X'190'
②  // OPTION CATAL
③     PHASE PROGB,F+32768
④     INCLUDE

        Relocatable object deck

   /*
      INCLUDE SUBRX
      INCLUDE SUBRY
      INCLUDE

        Relocatable object deck

   /*
   // LBLTYP TAPE
   // EXEC LNKEDT
   /&
```

Discussion

The cataloging of a single phase composed of multiple relocatable object modules is illustrated by this example. These modules are located in the input stream and in the relocatable library. Labeled tape files and sequential DASD files are to be processed when the phase is executed. The program will be executed in a foreground partition. Assume that the foreground partition begins at location 32768.

① The SYSLNK assignment is shown to indicate the relationship to the OPTION statement, although it is not required because of the assumed configuration.

② The OPTION CATAL statement sets the link switches as well as a catalog switch. If SYSLNK is not assigned, the statement is ignored. The linkage

(Continued on next page)

editor control statements will not be accepted unless the OPTION statement has been processed. Link editing and cataloging to the core image library will occur.

③ Only one PHASE will be constructed. It will be cataloged to the core image library and retrieved by the name PROGB. Since this is to be a foreground phase, F plus the location address in the foreground partition must be specified.

A program may be link edited to any address that falls within a foreground partition. The load address does not have to coincide with the partition address. However, the program must be of such a size that it can reside in the available core defined from the load address to the end of the partition.

The address may be expressed in one of three forms:

X'hhhhhh' -- A hexadecimal number of 4 to 6 digits

dddddddd -- A decimal number of 5 to 8 digits

nnnnK -- Where K = 1024 and n is 2 to 4 digits

+X'8000', +32768, and +32K are equivalent. The actual origin point of the phase is adjusted up from the address specification to allow for the program name, a register save area, and the label information (LBLTYP statement reservation).

④ Four modules make up this phase. The first and last are not cataloged in the relocatable library; therefore the object decks must be on SYSIPT, and each must be followed by the end-of-data record (/*). SUBRX and SUBRY have been cataloged previously to the relocatable library by those names. Job control will put the noncataloged modules on SYSLNK in place of their INCLUDE statements. Job control will copy the INCLUDE statements for the cataloged modules.

⑤ The LBLTYP statement has the operand TAPE rather than NSD because labeled tapes and no nonsequential DASD files will be processed when the phase is executed. Eighty bytes are reserved ahead of the actual phase for label information. LBLTYP NSD is also satisfactory since it generates a minimum of 104 bytes and tapes require only 80.

⑥ The EXEC LNKEDT statement causes the system loader to bring in the linkage editor program. SYSLNK now becomes input to the linkage editor. It contains the following:

```
PHASE PROGB, F+32768
First uncataloged relocatable deck
INCLUDE SUBRX
INCLUDE SUBRY
Second uncataloged relocatable deck
ENTRY
```

The modules are link edited so that they occupy contiguous areas in main storage in the sequence in which they appear in the input stream. When the link editing is completed, cataloging to the core image library will occur because of the CATAL option. The core image directory is checked to make sure the new phase entries will fit. If not, the job is canceled. The directory is scanned for any match to a phase being cataloged. A match is deleted from the directory. The system directory is updated to reflect the changes. Job control is brought into main storage.

⑦ Because CATAL was specified, a special routine is executed when the /& control statement is read by job control. This routine updates the transient, library-routine, and foreground-program directories. A system status report is printed to reflect the usage and available space in each of the libraries and directories. These operations do not occur in a LINK situation. The /& resets the CATAL option, that is, it turns off the LINK and CATAL switches.

The example can be modified to illustrate a catalog-and-execute operation by inserting the following data between the EXEC LNKEDT and /& statements:

1. Any job control statements required for execution of PROGB
2. A // EXEC PROGB or // EXEC statement
3. Any card reader input for PROGB

Note that the actual update of the directories and the system status report are delayed until completion of the execution of PROGB, when /& is read. From a system design standpoint this is not desirable because of possible operational problems. Making the execution of PROGB a separate job will avoid any difficulties.

Example 15. Compile and Execute

```
     // JOB COMPEXEC
     *  COMPILE OR ASSEMBLE, LINK EDIT AND EXECUTE
     *  SINGLE PHASE, MULTIPLE OBJECT MODULES, BACKGROUND PROGRAM
     *  SEQUENTIAL DASD FILES TO BE PROCESSED
     *  INPUT TO LINKAGE EDITOR FROM LANGUAGE TRANSLATOR, RELOCATABLE LIBRARY
     *    AND SYSIPT
 ①   // ASSGN SYSLNK,X'190'
 ②   // OPTION LINK
 ③      PHASE PROGA,S
 ④   // EXEC COBOL

                    COBOL source statements

     /*
        INCLUDE SUBRX
 ⑤      INCLUDE

                    Relocatable object module

     /*
 ⑥      ENTRY BEGIN1
     // EXEC LNKEDT

 ⑦                  Any job control statements required for PROGA execution

     // EXEC

                    Any input data required for PROGA execution

     /*
     /&
```

Discussion

The language translators provide the option of placing their output on SYSLNK rather than SYSPCH. Since the linkage editor uses SYSLNK for input, a program can be assembled or compiled, then link edited and executed. This operation, known as assemble/compile and execute, is illustrated by this example.

All three sources of object module input to the linkage editor are used: SYSIPT, the relocatable library, and the output from a language translator. It is assumed that the phase will be executed in the background partition, and that only sequential DASD files or unlabeled tape files will be processed.

① The SYSLNK assignment is given to illustrate the relationship to the OPTION statement, although it is not required because of the assumed configuration.

② Since SYSLNK is assigned, the OPTION LINK statement sets the link indicator switches.

③ The PHASE statement must always precede the relocatable modules to which it applies; therefore, it is written onto SYSLNK first for later use by the linkage editor. S is used as the origin point; that is, the phase will originate with the first doubleword at the end of the supervisor, the label block area (as defined by LBLTYP), and the area assigned to the COMMON pool (if any). This gives the same effect as * gives for a single phase or the first phase. As with the *, the S may be used with a relocation factor, for example, S+1024. See ③ , Example 13, for further detail on relocation factors. The factor must always be positive, since a negative factor could cause the origin point to overlay the supervisor.

④ The appropriate language translator, COBOL in this case, is called. The normal rules for compiling are followed: the source deck must be on the unit assigned to SYSIPT; the /* defines the end of the source data. Because the link switches are set, the output of the language translator is written on SYSLNK. Except for PL/I, the DECK option is ignored when SYSLNK is used.

(Continued on next page)

⑤ The INCLUDE SUBRX statement is written on SYSLNK. The linkage editor will retrieve the named module from the relocatable library. Because the operand is blank, the next INCLUDE statement signifies that the relocatable module is on SYSIPT. The data on SYSIPT is copied to SYSLNK until the /* statement.

⑥ The ENTRY statement is written on SYSLNK as the last linkage control statement. The symbol BEGIN1 must be the name of a CSECT or a label definition defined in the first phase. The address of BEGIN1 becomes the transfer address for the first phase of the program. The ENTRY statement is used because the user wishes to provide a specific entry point rather than to use the point specified in the END record or the load address of the phase. The ENTRY statement affects the first, or only, phase.

⑦ No LBLTYP statement is required, since only sequential DASD files are to be processed. The rest of the statements follow the same pattern as that discussed in Example 13. The input from SYSLNK to the linkage editor is:

```
PHASE PROGA, S
Relocatable module produced by COBOL
  compilation
INCLUDE SUBRX
Relocatable module from SYSIPT
ENTRY BEGIN1
```

If a serious or disastrous error were detected during compilation or assembly of a source program, the link option would be suppressed. Under these circumstances the EXEC LNKEDT and EXEC statements would be ignored in this example. This link option suppression should be kept in mind if a series of programs is to be compiled and cataloged as a single job. Failure of one job step would cause failure of all succeeding steps. Remember that an OPTION LINK cannot be given if OPTION CATAL is in effect, because message 1S1nD (STATEMENT OUT OF SEQUENCE) will result. This is an error in instruction to the system because CATAL has unfinished business that is not performed until the next /& statement (see ⑦ Example 14). Therefore, the CATAL switch must remain on, and linkage editing only cannot be performed.

Example 16. Catalog for Phase Overlay

```
    // JOB MULTPHAS
    *  LINK EDIT AND CATALOG TO CORE IMAGE LIBRARY
    *  MULTIPLE PHASES, MULTIPLE OBJECT MODULES, BACKGROUND PROGRAM
    *  NO LABELED TAPE OR NON-SEQUENTIAL DASD FILES TO BE PROCESSED
    // ASSGN SYSLNK,X'190'
 ① // OPTION CATAL
       PHASE PHASEA,ROOT
 ②     INCLUDE MOD1
       PHASE PHASEB,*
 ③     INCLUDE MOD2
       PHASE PHASEC,PHASEB
 ④     INCLUDE MOD3
 ⑤ // EXEC LNKEDT
    /&
```

Discussion

Sometimes it is not possible to bring an entire program into main storage because it requires more core than is available. To solve this problem the program can be broken into separate phases that can be brought into core as required, overlaying all or part of another phase as desired. The link editing of three cataloged relocatable modules into phases for overlay is illustrated by this example.

① The OPTION CATAL sets the switches so that the phases can be link edited and cataloged into the core image library.

② PHASEA is considered the ROOT phase; that is, it will always be resident in core during program execution. The origin point is the first doubleword address after the supervisor, the label storage area (if any), and the area assigned to the COMMON pool (if any). Only the first phase statement is permitted to specify ROOT.

③ The * in the PHASE card for PHASEB will cause MOD2 to be link edited at the end of PHASEA.

④ Because PHASEB is specified as the load address for PHASEC, it will be link edited into the same address as PHASEB. The symbol used to designate the origin point may be a previously defined phase name as in this example, a previously defined control section, or a previously defined external label. A plus or minus relocation factor, for example, PHASE2+100, may be used.

⑤ The EXEC LNKEDT causes all three phases to be link edited and cataloged. When the phases are executed, the ROOT phase normally is loaded by the system loader. The other phases would probably be brought into core by means of the FETCH or LOAD macro issued by the calling phase.

(blank page)

Example 17. Submodular Structure

```
    // JOB SUBMOD
    *  LINK EDIT AND CATALOG TO CORE IMAGE LIBRARY
    *  MULTIPLE PHASES, ONE OBJECT MODULE (SUBMODULAR STRUCTURE)
    *   BACKGROUND PROGRAM, NO LABEL STORAGE RESERVATION
    // ASSGN SYSLNK,X'190'
    // OPTION CATAL
 ①     PHASE PHASE1,*
       INCLUDE ,(CSECT1,CSECT3)
 ②     PHASE PHASE2,*
       INCLUDE ,(CSECT2,CSECT5)
 ③     PHASE PHASE3,PHASE2
       INCLUDE ,(CSECT4,CSECT6)
 ④     INCLUDE

          Relocatable object deck

    /*
    // EXEC LNKEDT
    /&
```

Discussion

In Example 16, several relocatable modules are structured into several phases. In this example, a single object module is broken into several phases. The object module is composed of CSECT1-CSECT6. It will be structured into three phases with overlay. The module is not cataloged in the relocatable library. Only the PHASE and INCLUDE statements are discussed.

① The INCLUDE statement tells the linkage editor to place CSECT1 and CSECT3 into PHASE1. The sequence in which the CSECT's are link edited is determined by the sequence in the input module rather than the sequence in the INCLUDE statement. (CSECT3, CSECT1) would give the same result as (CSECT1, CSECT3). The sequence can be controlled by issuing separate INCLUDE statements. For example, INCLUDE, (CSECT3) followed by INCLUDE, (CSECT1) would cause CSECT3 to be link edited before CSECT1 regardless of the sequence in the object module.

Note that the first operand is missing in the INCLUDE statement, as indicated by the leading comma. This format of the INCLUDE statement will search the next succeeding object module on SYSLNK to locate the named CSECT's. See point ④ .

PHASE1 will be located at the end of the supervisor, the label storage area, and the COMMON area.

② The INCLUDE statement causes CSECT2 and CSECT5 to be used for PHASE2. This phase will be located at the end of PHASE1.

③ PHASE3 is made up of CSECT4 and CSECT6 and will overlay PHASE2 since its origin point is at the same address as PHASE2.

④ This INCLUDE statement with a blank operand is required to write the object module that follows in the card reader onto SYSLNK, in order to satisfy the INCLUDE statements with a blank first operand.

With the sequence of statements shown in the example, the PHASE and INCLUDE statements are read from SYSRDR. However, it is permissible to read PHASE and INCLUDE statements from SYSIPT. To do this, statement ④ (INCLUDE blank) would be placed ahead of statement ① . The INCLUDE with the blank operand would direct job control to read the following data (which includes the PHASE, INCLUDE, and then the object module) onto SYSLNK from SYSIPT to the /* statement. If SYSRDR and SYSIPT are separate devices, care must be taken to place the PHASE and INCLUDE statements on the correct device.

(Continued on next page)

Example 17 (continued). Submodular Structure

PHASE and INCLUDE statements can also be in the relocatable library. If the object module were in the relocatable library under the name MOD1, the following changes would be made:

1. Remove statements ① through ③ , and add module name to statement ④ .

```
// JOB SUBMOD
// OPTION CATAL
// INCLUDE MOD1
// EXEC LNKEDT
```

2. When the relocatable module is cataloged to the library, precede it with the following statements:

```
PHASE PHASE1,*
INCLUDE MOD1, (CSECT1, CSECT3)
PHASE PHASE2,*
INCLUDE MOD1, (CSECT2, CSECT5)
PHASE PHASE3, PHASE2
INCLUDE MOD1, (CSECT4, CSECT6)
Relocatable object deck
```

This form of the INCLUDE statement will cause the linkage editor to search the module that follows the last INCLUDE statement in the library for the required control sections.

Example 18. Self-Relocating and Multiple Link Edits

```
    // JOB MULTCATL
    *  SEVERAL LINK EDITS AS A SINGLE JOB
    // OPTION CATAL
①      PHASE PROG1,+0
       INCLUDE PROG1
    // EXEC LNKEDT
②      PHASE PROG2,*
       INCLUDE
            Relocatable object module
    /*
    // EXEC LNKEDT
       PHASE PROG3,*
       INCLUDE PROG3
    // EXEC LNKEDT
    /&
    // JOB LINKGO
③   // OPTION LINK
       PHASE PROG4,*
       INCLUDE
            Relocatable object module
    /*
    // EXEC LNKEDT
            Any job control statements required for PROG4 execution
    // EXEC
            Any input data required for PROG4
    /*
       PHASE PROG5,*
       INCLUDE PROG5
    // EXEC LNKEDT
            Any job control statements required for PROG5 execution
    // EXEC
            Any input data required for PROG5
    /*
    /&
```

Discussion

The link editing requirements for a self-relocating program and the combining of several cataloging or link-and-execute job steps into a single job are illustrated in this example. Discretion should be used in deciding how many steps should be combined before a /&, since a failure in one step can cause successive steps to be bypassed unnecessarily.

① Use of the +0 displacement as the origin point for PROG1 designates this program as self-relocating. Once this program has been cataloged to the core image library, it may be executed in any partition. It can be initiated into the background partition by job control, provided the supervisor was generated with multiprogramming capabilities. A non-multiprogramming supervisor will give message 0P771 (CANCELLED DUE TO INVALID ADDRESS). However, the program can be loaded by using the LOAD macro in a calling phase.

② PROG2 and PROG3 are link edited and cataloged as job steps within the job MULTCATL. Note that OPTION CATAL holds for these steps.

③ A new job is initiated because the succeeding job steps are to be linked and executed without cataloging. (OPTION LINK cannot be issued when OPTION CATAL is in effect.) Note that OPTION LINK need not be reissued before the next job step.

SUMMARY OF CONSIDERATIONS FOR LINK AND CATAL OPTIONS

1. SYSLNK must be assigned, or LINK and CATAL options are ignored (switches are not set).
2. Unless the switches are set by the LINK or CATAL option, the linkage editor control statements are ignored.
3. The CATAL option sets the LINK as well as CATAL switches.
4. When the LINK switches are set, the output of the language translators is placed on SYSLNK.
5. LINK and CATAL switches are turned off by:
   - /& or JOB statement
   - A serious or disastrous error during compilation or assembly
6. Completion of cataloging (update of transient, library-routine, and foreground-program directories, and system status report) occurs when the /& statement is read by job control.
7. If a successful link edit has not occurred, cataloging will not take place.
8. If multiple link-edit job steps are set up as one job, keep these points in mind:
   - It is not possible to CATAL into the core image library with // OPTION CATAL and then have another link-edit job step with // OPTION LINK in the same job. Operator message 1S1nD (STATEMENT OUT OF SEQUENCE) will result.
   - If a compilation or assembly is being performed, the link switches may be turned off by a serious or disastrous error. When also cataloging to the core image library, therefore, it is advisable to handle multiple job steps as separate jobs, (each with /&) to be sure that the cataloging operation is finished on the /&.

LOADING THE UNIVERSAL CHARACTER SET BUFFER (EXAMPLE 19)

UCS JOB CONTROL COMMAND

```
UCS  SYSxxx,phasename [,FOLD] [,BLOCK]
         [,NULMSG]
```

The Universal Character Set (UCS) special feature for the 1403 Printer permits the use of a variety of print chains or trains. To use this feature, the UCS buffer in the 2821 Control Unit must be loaded with the 240-character image of the particular chain or train mounted on the printer. Before this loading can be accomplished, the 240-character image is cataloged into the core image library under a selected phase name. Then, whenever job control is in core, the UCS job control command can be issued to cause the image specified by the phase name to be loaded into the 2821 UCS buffer storage.

The command parameters have the following meaning:

SYSxxx	The name of the logical unit assigned to the 1403 UCS printer to be loaded, for example, SYSLST
phasename	The symbolic name of the core image library phase containing the 240 EBCDIC characters to be loaded, followed by an 80-character verification message. For example, a phase cataloged for use with the HN train arrangement may be called UCSHN.
FOLD	Signifies the use of the folding option, which in effect allows lowercase letters to print as uppercase letters.
BLOCK	Signifies that the 2821 latch is to be set to inhibit data checks generated by the 1403 UCS printer due to print-line character mismatches with the UCS buffer.
NULMSG	Signifies that the 80-character verification message is not to be printed on the 1403 after the buffer is loaded. If this parameter is not specified, the program will skip to channel 1 after the UCS buffer has been loaded, issue a print of the last 80 characters in the phase specified by phasename, and again skip to channel 1. This message is provided as a means of verifying that the mounted train is compatible with the UCS buffer contents.

Example 19. Catalog UCS Buffer Images and Load Buffer

```
     // JOB UCSLOAD
     *  ASSEMBLE AND CATALOG TO CORE IMAGE LIBRARY PHASES FOR HN AND RN
     *    ARRANGEMENTS FOR USE WITH UNIVERSAL CHARACTER SET BUFFER LOAD
     // OPTION CATAL,NOSYM,NOXREF
 ①      ACTION CANCEL
        PHASE UCSHN,*,NOAUTO
     // EXEC ASSEMBLY
                START 0
                DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
                DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
 ②              DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
                DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
                DC        C'1234567890=''/STUVWXYZ&&,(JKLMNOPQR-$*ABCDEFGHI+.)'
                DC      CL40'HN ARRANGEMENT TYPE IF EQUAL SIGN ='
                DC      40C' '
                END
     /*
 ③   // EXEC LNKEDT
        ACTION CANCEL
        PHASE UCSRN,*,NOAUTO
 ④   // EXEC ASSEMBLY
                START 0
                DC        C'1234567890XY/STUVW''@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
                DC        C'1234567890XY/STUVW%@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
                DC        C'1234567890XY/STUVW#@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
                DC        C'1234567890XY/STUVW<@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
                DC        C'1234567890XY/STUVW&&@$*,=JKLMNOPQR-Z(ABCDEFGHI+.)'
                DC      CL40'RN ARRANGEMENT TYPE IF POUND SIGN #'
                DC      40C' '
                END
     /*
     // EXEC LNKEDT
     /&
 ⑤   // PAUSE MOUNT HN TRAIN
        UCS SYSLST,UCSHN,FOLD,BLOCK
```

Discussion

This example illustrates the assembly, linkage edit, and catalog of UCS buffer phases to the core image library. The buffer is then loaded.

① The usual rules for setting up a catalog operation using the output of a language translator are followed. See OPTION and PHASE statements. ACTION CANCEL is used, since no cataloging should occur if any statement errors exist.

② The Assembler prepares a relocatable module for the printer image data that follows on SYSIPT. Output is written on SYSLNK. Note the use of two ampersands and two apostrophes to cause acceptance of those characters in the string. The characters are submitted in the same sequence in which they occur on the train/chain. The standard arrangements are given in the manual IBM 2821 Control Unit (A24-3312). An 80-character message follows the 240-character printer image. The /* signifies the end of the data.

③ The EXEC LNKEDT statement link edits and catalogs the phase to the core image library.

④ Assuming no errors in the previous assembly, OPTION CATAL is still in effect. The linkage edit control statements that pertain to the next buffer arrangement are read, and the cycle is repeated.

⑤ Now that the buffer images are in the core image library, they can be loaded by means of the UCS job control command. A PAUSE statement

(Continued on next page)

stops the system so that the proper chain/train can be mounted. In this example the HN chain/train arrangement represented by UCSHN will be loaded into the UCS buffer for the printer represented by SYSLST, which is X'00E' in the assumed configuration. Of course, the printer must be equipped with the UCS feature.

The fold option is specified so that any lowercase letters will print as uppercase letters. This is necessary if lowercase codes are expected, because the HN arrangement has only uppercase letters. Otherwise, these lowercase letters can find no matching code in the buffer and cannot print. The BLOCK parameter permits any code not represented in the UCS buffer to print as a blank without causing a data-check stop. If BLOCK is not specified, any unmatched code causes a data check. Remember that the blocking of data checks is also controlled by the programmer through logical IOCS. Hence the effect of this parameter may be transient, depending on the processing programs coming up in the job stream. A verification message will print, since NULMSG is not specified.

FOREGROUND INITIATION (EXAMPLE 20)

GENERAL CONSIDERATIONS

The supervisor of the DOS system can be generated with or without multiprogramming capabilities. Either type can handle any of the batched job operations discussed so far, but only the multiprogramming supervisor (MPS) can provide foreground functions. The foreground system flow given in the introduction and shown in the flowchart in Figure 1 should be reviewed as background for the information in this section.

Allocation of Partitions

The size of the foreground partitions is specified at system generation time. The IPL routine determines the size of main storage and allocates the specified foreground areas downward from high main storage. The operator can reallocate main storage by issuing the ALLOC command as either a job control command or as an ATTN command.

ALLOC Command

$$\text{ALLOC}\ \left\{\begin{matrix}\text{F1=nK}\,[\text{,F2=nK}]\\ \text{F2=nK}\,[\text{,F1=nK}]\end{matrix}\right\}$$

The value n must be an even integer, since the size of the foreground area is specified in 2K increments. Specifying 0K will delete a foreground area. If only one foreground area is referenced, the amount of storage allocated to the other is assumed to remain the same. Storage is not reallocated if it will reduce the background area, relocate an active foreground program, or cause a decrease in the storage allocated to an active foreground program.

Linkage Editing

Since all relocatable object modules must be processed by the linkage editor before execution, and since the linkage editor program functions only in the background, obviously a phase to be executed in a foreground partition must be cataloged to the core image library before execution. The link-and-go or compile-link-and-go operations are not possible. Non-self-relocatable foreground programs can be link edited into any portion of a foreground partition -- not necessarily the beginning of the partition -- as long as they do not extend beyond the limits of the partition. The same rules apply to reservation of core storage for labels, using the // LBLTYP statement at link-edit time, as to background programs. Self-relocating programs can operate in any foreground partition of sufficient size. No label storage reservation is required.

Symbolic Unit Assignments

Since at system generation time no assignments of symbolic units to physical devices can be made for the foreground partitions, these assignments must be made just before the execution of the foreground job. This often involves first unassigning a physical device that is assigned to the background, because no physical device except DASD can be shared between partitions. LISTIO can be used to determine existing assignments so that conflicts can be avoided.

Foreground assignments hold only for this execution unless a HOLD command is issued, in which case the assignment remains until a RELSE or UNA command is given.

Dumping Main Storage

Abnormal termination of a foreground program causes a dump on foreground unit SYS000, provided SYS000 is a printer or non-file-protected tape. Only the problem program area that caused the dump will be printed.

Labels

Label information must be available for writing and checking labels of DASD files and labeled tape files. The information can be submitted through VOL, DLAB, XTENT, TPLAB, and TLBL commands each time a foreground program is initiated, or it can be stored on the standard label track using job control with STDLABEL via a background job. Remember that only sequential DASD or tape labels can be placed on the standard label track. Also, each time labels are written, they overlay any labels previously stored on the track. Depending on the mix, 20 to 30 labels can be stored on the standard label track.

Foreground Initiation Process

The job control program, which is used to prepare the system for the execution of background programs and operates only in the background area, is not available for initiating foreground programs. Instead, the foreground initiator provides job-control-type functions by reading and interpreting foreground initiation commands. The main functions of the foreground initiator are to assign symbolic units, to write label information on the user label track, and to initiate the loading of the program for execution.

The foreground initiator is brought into the respective foreground partition as the result of issuing a START command after pressing the request key on the printer-keyboard. The initiator will accept commands from either the printer-keyboard or a card reader. Control is initially at the printer-keyboard but can be given to the card reader following a READ command.

If foreground programs are initiated from the printer-keyboard only, the operator must key in all of the commands. The VOL, TPLAB, TLBL, DLAB, and XTENT file label information should be prepared for the operator on a form layout that will tie in with the pointer on the 1052. The layout should allow for the automatic typing of F1 and F2, which occurs before the keyboard is unlocked.

If frequent foreground initiation occurs, the use of a card reader instead of the printer-keyboard is more efficient. Thus, in a system configuration with only one card reader, the card reader must be freed from background usage. This can be accomplished by:

1. Temporarily interrupting the background job stream through a PAUSE to initiate the foreground
2. Placing the background job stream on tape or disk

DESCRIPTION OF COMMANDS

Communication from the user to the DOS system can be issued at the following times:

1. Job control -- between jobs or job steps for batch processing (background)
2. Attention (ATTN) -- at any time by pressing the request key on the printer-keyboard. Some of these commands can be issued only in a multiprogramming environment.
3. Foreground program initiation -- only in a multiprogram environment following the ATTN command, START F1 or F2
4. IPL -- at initial program loading

Appendix E is an alphabetic listing of all statements or commands and their format. The time when each is valid is indicated.

So far in this manual, job control has been the primary communication medium because the examples dealt with batch processing. Commands used for foreground initiation are discussed below. They are valid only in a multiprogramming system.

STOP Command -- Job Control

STOP blank

The STOP command suspends background processing or indicates that there are no more background jobs to be executed. It can be issued only when job control is in core. The background job is removed from the system's task selection mechanism, and job control remains dormant.

The command can be used instead of a PAUSE. It is issued before the initiation of a foreground program.

START Command -- ATTN

$$\text{START}\ \left\{\begin{array}{c}\underline{\text{BG}}\\ \text{F1}\\ \text{F2}\end{array}\right\}$$

The START command can be used to initiate a foreground program or to resume batch job processing.

Use of the BG parameter, or no operand, causes job control to read the next control statement in the background program job stream. It is effective only if a STOP command was issued previously.

F1 or F2 specifies that a foreground program is to be initiated and indicates the area to be used. The foreground initiator is loaded into the specified area, and control is given to it. Only foreground initiation commands may be issued following the START command.

This command can be given only to the ATTN routine, that is, following depression of the request key on the printer-keyboard.

READ Command -- Foreground Initiation (FI)

READ X'cuu'

The foreground initiator brings control first to the printer-keyboard for entry of the foreground initiation commands. If a card reader is available, it may be specified by a READ command so as to cause all further foreground initiation commands to be read from the designated card reader. Remember that the device cannot be assigned to any other partition at this time.

ASSGN Command -- FI

$$\text{ASSGN SYSnnn,}\left\{\begin{array}{l}\text{X'cuu'}\\ \\ \text{IGN}\end{array}\right\}\left[\left\{\begin{array}{l}\text{,X'ss'}\\ \\ \text{,ALT}\end{array}\right\}\right]$$

The ASSGN command performs the same function for a foreground program as its counterpart does for a background program -- it assigns a logical

I/O unit to a physical device. The parameters are the same except for the following two cases:

1. UA is not a valid address.
2. Only programmer symbolic units (SYSnnn as opposed to SYSxxx) are valid.

Remember that, except for DASD devices, a foreground program may not be assigned a device being used by another partition. All device assignments made for a foreground program are canceled at the end of each program unless held across jobs by a HOLD command.

HOLD Command -- FI or Job Control

$$\text{HOLD}\ \left\{\begin{matrix}\text{F1}\ [,\text{F2}]\\ \text{F2}\ [,\text{F1}]\end{matrix}\right\}$$

The HOLD command causes all I/O assignments for the foreground area(s) specified to stay in effect from one job to the next.

If only one foreground area is referenced, it is assumed that the I/O assignments for the other are not to be held.

New assignments can be made in initiation of a job to run in an area where the HOLD command is in effect. Any new assignment overrides the previous assignment to the logical unit specified.

A HOLD is terminated by a RELSE or UNA command.

RELSE Command -- FI or Job Control

$$\text{RELSE}\ \left\{\begin{matrix}\text{F1}\ [,\text{F2}]\\ \text{F2}\ [,\text{F1}]\end{matrix}\right\}$$

The RELSE command causes all I/O assignments for the foreground area(s) specified to be set to "unassigned" at the end of any job that is initiated for that area.

If only one foreground area is referenced, the I/O assignments for the other are unaffected.

UNA Command -- FI or Job Control

$$\text{UNA}\ \left\{\begin{matrix}\text{F1}\ [,\text{F2}]\\ \text{F2}\ [,\text{F1}]\end{matrix}\right\}$$

Unlike the RELSE command, which is not effective until the end of a job, the UNA command immediately causes all I/O assignments for the foreground area(s) specified to be set to "unassigned".

The foreground area specified must be currently inactive. If only one foreground area is referenced, a previous HOLD for the other remains in effect. The command is useful to free physical units currently assigned to a foreground area under the HOLD command.

Label Commands -- FI

The label information for a foreground program is entered into the system with commands that are identical to the VOL, DLAB, XTENT, TPLAB, and TLBL statements used for background programs. In fact, the // in columns 1 and 2 is acceptable (this is not so on any other FI command) but is not required. Of course, only SYSnnn units are valid.

EXEC Command -- FI

EXEC progname

The EXEC foreground initiation command performs functions for the foreground program similar to those performed by the EXEC job control statement for the background program. Note that progname is not optional, since the link-and-go operation is not available. The program must have been cataloged to the core image library previously. This command terminates the foreground initiation and causes the specified program to be loaded into main storage.

(blank page)

Example 20. Foreground Initiation

```
*    FOREGROUND INITIATION

          Operator gains control of system via a PAUSE

①   ASSGN SYSRDR,UA                                        }
     ASSGN SYSIPT,UA                                        }
②   ASSGN SYSLST,UA                                        }  Job control
③   ASSGN SYS006,UA                                        }
④   STOP                                                   }

          Operator depresses the request key                }
                                                            }  ATTN
⑤   START F1                                               }
⑥   READ X'00C'                                            }
     ASSGN SYS004,X'190'                                    }
⑦   ASSGN SYS005,X'00E'                                    }
     ASSGN SYS000,X'280'                                    }
⑧   HOLD F1                                                }  Foreground
     VOL SYS004,OUTFILE                                     }  initiator
⑨   DLAB 'SAM DISK                                 1111111'                X
                   0001,67001,67002,'            '          }
     XTENT 1,0,000177000,000187002,'111111',SYS004          }
⑩   EXEC CREATER                                           }

⑪        Operator initiates another foreground job when
          previous job in this partition is completed

     START F1
     READ X'00C'
     VOL SYS004,INFILE
     DLAB 'SAM DISK                                 1111111'                X
                   0001,67001,67002,'            '
     XTENT 1,0,000177000,000187002,'111111',SYS004
⑫   RELSE F1
     EXEC PRINTERR
```

Discussion

This example is designed to illustrate the foreground initiation commands and the general considerations discussed at the beginning of this section. The first foreground job (CREATER) generates a sequential disk file. The second foreground job (PRINTERR) prints the sequential file just created. It is assumed that the operator has gained control at the end of a background job via a PAUSE statement.

For examples of foreground initiation under other operational conditions such as two card readers, planned procedure, etc., see IBM System/360 Disk Operating System, Operating Guide (C24-5022).

① Because SYSRDR and SYSIPT are both assigned to the single card reader in the assumed system configuration, these symbolic units must be unassigned to free the card reader for use by the foreground initiator. A single ASSGN command for SYSIN is correct also.

② The foreground program requires the printer; therefore, it must be unassigned from SYSLST.

③ To provide a tape unit (X'280') for recording an abnormal program end, SYS006 is unassigned. Note that all of these assignment changes must be made while job control is in main storage, since the background assignments are the ones to be altered.

④ The background program is removed from the system selection mechanism during foreground initiation.

⑤ The operator presses the request key. When the message READY FOR COMMUNICATIONS appears on SYSLOG, the START F1 command can be issued. This command calls the foreground initiator into the named partition.

(Continued on next page)

⑥ All of the commands so far have been entered on the printer-keyboard. In order to enter foreground initiation commands through the card reader, the READ command is given, using the address of the card reader. From this point on, the initiator looks to the card reader for further commands.

⑦ Assignments are made for the required units to be used by the foreground program: a DASD unit, the printer, and SYS000 to receive an abnormal end dump. Remember that only DASD units may be shared between partitions.

⑧ Assignments for the foreground program normally are terminated when execution is complete. This HOLD statement causes the assignments to be maintained until a RELSE or UNA command is given. Therefore, like assignments for a subsequent program do not have to be reissued.

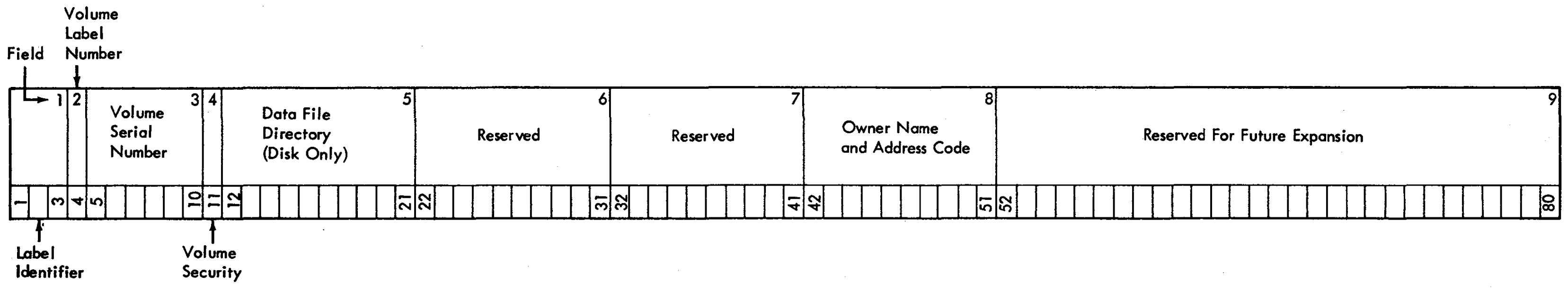
⑨ Label sets are submitted for each file in order to place label storage data into the proper foreground temporary area. Since this is a sequential file, the information could have been placed into the permanent label storage area with a prior STDLABEL background operation, thus eliminating the need to submit label sets at this time.

⑩ Foreground initiation is terminated with the EXEC statement. The named program is called into the specified partition. The program must be in the core image library. It may be self-relocating, or it may have been link edited into locations that lie within the existing foreground partition.

The card reader is released and is available for reassignment to the background. By pressing the request key, the operator calls in the ATTN routine. A START BG command will resume the background program at the point of interruption.

⑪ When the foreground program is completed, the operator may initiate another foreground program. Assuming a background job is running, the operator must consider assignments and issue STOP.

⑫ The RELSE command will unassign all foreground units upon termination of the execution of the PRINTERR program.

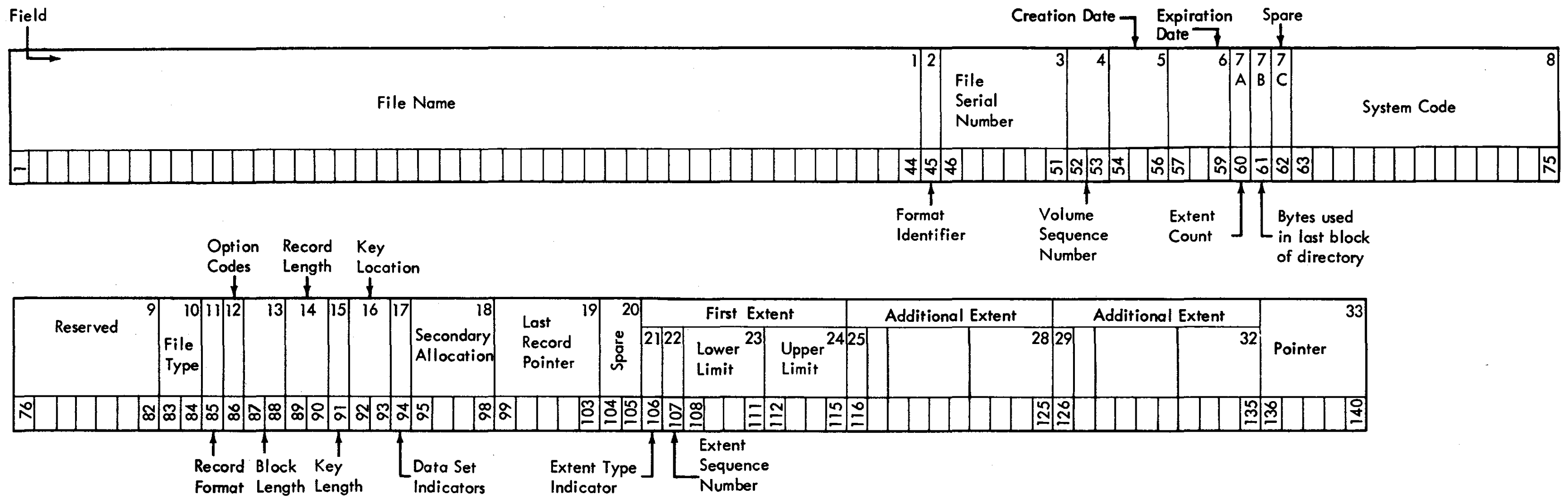


Volume Label Format (80 bytes) for Tape or DASD

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes	Must contain VOL to indicate that this is a Volume Label.
2.	VOLUME LABEL NUMBER 1 byte	Indicates the relative position (1-8) of a volume label within a group of volume labels.
3.	VOLUME SERIAL NUMBER 6 bytes	A unique identification code which is assigned to a volume when it enters an installation. This code may also appear on the external surface of the volume for visual identification. It is normally a numeric field 000001 to 999999, however any or all of the 6 bytes may be alphameric.
4.	VOLUME SECURITY 1 byte (OS/360 only)	Indicates security status of the volume: 0 = no further identification for each file of the volume is required. 1 = further identification for each file of the volume is required before processing.
5.	DATA FILE DIRECTORY 10 bytes	For DASD only. The first 5 bytes contain the starting address (CCHHR) of the VTOC. The last 5 bytes are blank. For tape files, this field is not used and should be recorded as blanks.
6.	RESERVED 10 bytes	Reserved.
7.	RESERVED 10 bytes	Reserved.
8.	OWNER NAME AND ADDRESS CODE 10 bytes	Indicates a specific customer, installation and/or system to which the volume belongs. This field may be a standardized code, name, address, etc.
9.	RESERVED 29 bytes	Reserved.

Note: All reserved fields should contain blanks to facilitate their use in the future. Any information appearing in these fields at the present time will be ignored by the Disk Operating System programs as well as the Operating System/360 programs.

APPENDIX B. STANDARD DASD FILE LABELS, FORMAT 1



Format 1: This format is common to all data files on Direct Access Storage Devices.

FIELD	NAME AND LENGTH	DESCRIPTION
1.	FILE NAME 44 bytes, alphameric EBCDIC	This field serves as the key portion of the file label. Each file must have a unique file name. Duplication of file names will cause retrieval errors. The file name can consist of three sections: 1. File ID is an alphameric name assigned by the user and identifies the file. Can be 1-35 bytes if generation and version numbers are used, or 1-44 bytes if they are not used. 2. Generation Number. If used, this field is separated from File ID by a period. It has the format Gnnnn, where G identifies the field as the generation number and nnnn (in decimal) identifies the generation of the file. 3. Version Number of Generation. If used, this section immediately follows the generation number and has the format Vnn, where V identifies the field as the version of generation number and nn (in decimal) identifies the version of generation of the file. Note: The Disk Operating System compares the entire field against the file name given in the DLAB card. The generation and version numbers are treated differently by Operating System/360.

The remaining fields comprise the DATA portion of the file label:

FIELD	NAME AND LENGTH	DESCRIPTION
2.	FORMAT IDENTIFIER 1 byte, EBCDIC numeric	1 = Format 1
3.	FILE SERIAL NUMBER 6 bytes, alphameric EBCDIC	Uniquely identifies a file/volume relationship. It is identical to the Volume Serial Number of the first or only volume of a multi-volume file.
4.	VOLUME SEQUENCE NUMBER 2 bytes, binary	Indicates the order of a volume relative to the first volume on which the data file resides.
5.	CREATION DATE 3 bytes, discontinuous binary	Indicates the year and the day of the year the file was created. It is of the form YDD, where Y signifies the year (0-99) and DD the day of the year (1-366).

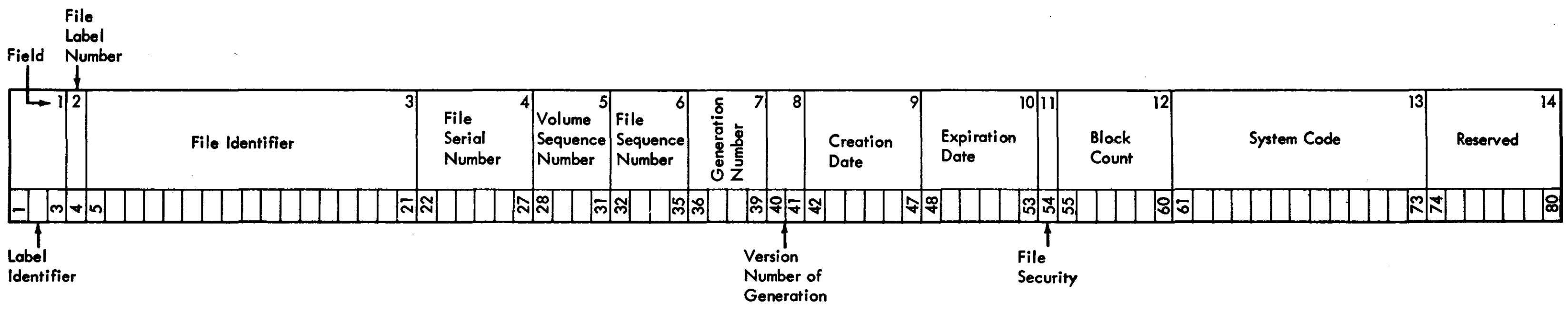
FIELD	NAME AND LENGTH	DESCRIPTION
6.	EXPIRATION DATE 3 bytes, discontinuous binary	Indicates the year and the day of the year the file may be deleted. The form of this field is identical to that of Field 5.
7A	EXTENT COUNT	Contains a count of the number of extents for this file on this volume. If user labels are used, the count does not include the user label track. This field is maintained by the Disk Operating System programs.
7B	BYTES USED IN LAST BLOCK OF DIRECTORY 1 byte, binary	Used by Operating System/360 only for partitioned (library Structure) data sets. Not used by the Disk Operating System.
7C	SPARE 1 byte	Reserved.
8	SYSTEM CODE 13 bytes	Uniquely identifies the programming system. The character codes that can be used in this field are limited to 0-9, A-Z, or blanks.
9	RESERVED 7 bytes	Reserved.
10	FILE TYPE 2 bytes	The contents of this field uniquely identify the type of data file: Hex 4000 = Consecutive organization Hex 2000 = Direct-access organization Hex 8000 = Indexed-sequential organization Hex 0200 = Library organization Hex 0000 = Organization not defined in the file label.
11.	RECORD FORMAT 1 byte	The contents of this field indicate the type of records contained in the file:

Bit Position	Content	Meaning
0 and 1	01	Variable length records
	10	Fixed length records
	11	Undefined format
2	0	No track overflow
	1	File is organized using track overflow (Operating System/360 only)
3	0	Unblocked records
	1	Blocked records
4	0	No truncated records
	1	Truncated records in file
5 and 6	01	Control character ASA code
	10	Control Character machine code
	00	Control Character not stated
7	0	Records have no keys
	1	Records are written with keys.

FIELD	NAME AND LENGTH	DESCRIPTION
12.	OPTION CODES 1 byte	Bits within this field are used to indicate various options used in building the file. Bit 0 = If on, indicates data file was created using Write Validity Check. 1-7 = unused
13.	BLOCK LENGTH 2 bytes, binary	indicates the block length for fixed length records or maximum block size for variable length blocks.
14.	RECORD LENGTH 2 bytes, binary	indicates the record length for fixed length records or the maximum record length for variable length records.
15.	KEY LENGTH 1 byte, binary	indicates the length of the key portion of the data records in the file.
16.	KEY LOCATION 2 bytes, binary	indicates the high order postion of the data record.
17.	DATA SET INDICATORS 1 byte	Bits within this field are used to indicate the following: BIT 0 If on, indicates that this is the last volume on which this file normally resides. This bit is used by the Disk Operating System.

FIELD	NAME AND LENGTH	DESCRIPTION
		BIT 1 If on, indicates that the data set described by this file must remain in the same absolute location on the direct access device. (OS/360 only) 2 If on, indicates that Block Length must always be a multiple of 7 bytes. (OS/360 only) 3 If on, indicates that this data file is security protected; a password must be provided in order to access it. (OS/360 only) 4-7 Spare. Reserved for future use.
18.	SECONDARY ALLOCATION 4 bytes, binary	indicates the amount of storage to be requested for this data file at End of Extent. This field is used by Operating System/360 only. It is not used by the Disk Operating System routines. The first byte of this field is an indication of the type of allocation request. Hex code C2 (EBCDIC B) blocks (physical records), hex code E3 (EBCDIC T) indicates tracks, and hex code C3 (EBCDIC C) indicates cylinders. The next three bytes of this field is a binary number indicating how many bytes, tracks or cylinders are requested.
19.	LAST RECORD POINTER 5 bytes discontinuous binary	points to the last record written in a sequential or partition-organization data set. The format is TTRLL, where TT is the relative address of the track containing the last record, R is the ID of the last record, and LL is the number of bytes remaining on the track following the last record. If the entire field contains binary zeros, the last record pointer does not apply.
20.	SPARE 2 bytes	Reserved.
21.	EXTENT TYPE INDICATOR 1 byte	indicates the type of extent with which the following fields are associated: HEX CODE 00 Next three fields do not indicate any extent. 01 Prime area (Indexed Sequential); or Consecutive area, etc., (i.e., the extent containing the user's data records.) 02 Overflow area of an Indexed Sequential file. 04 Cylinder Index or master Index area of an Indexed Sequential file. 40 User label track area. 8n Shared cylinder indicator, where n = l, 2, or 4.
22.	EXTENT SEQUENCE NUMBER 1 byte, binary	indicates the extent sequence in a multi-extent file.
23.	LOWER LIMIT 4 bytes, discontinuous binary	the cylinder and the track address specifying the starting point (lower limit) of this extent component. This field has the format CCHH.
24.	UPPER LIMIT 4 bytes	the cylinder and the track address specifying the ending point (upper limit) of this extent component. This field has the format CCHH.
25-28.	ADDITIONAL EXTENT 10 bytes	These fields have the same format as the fields 21-24 above.
29-32.	ADDITIONAL EXTENT 10 bytes	These fields have the same format as the fields 21-24 above.
33.	POINTER TO NEXT FILE LABEL WITHIN THIS LABEL SET 5 bytes, discontinuous binary	the address (format CCHHR) of a continuation label if needed to further describe the file. If field 10 indicates Indexed Sequential organization, this field will point to a Format 2 file label within this label set. Otherwise, it points to a Format 3 file label, and then only if the file contains more than three extent segments. This field contains all binary zeros if no additional file label is pointed to.

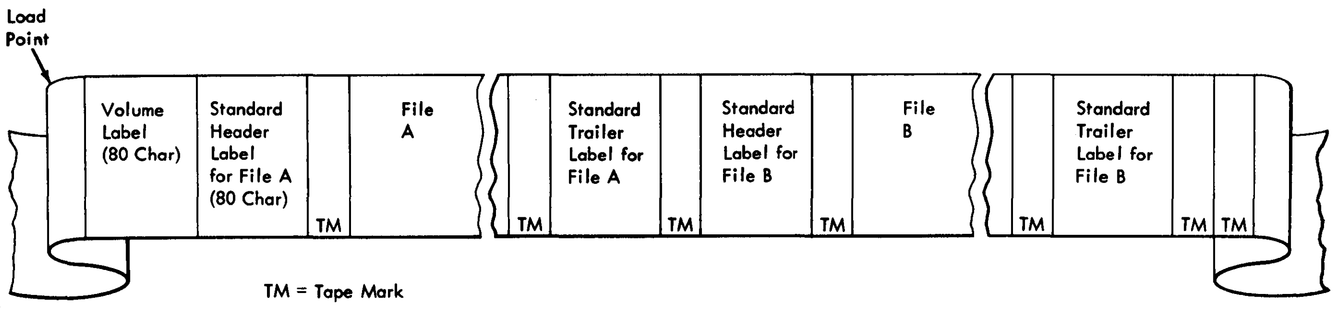
APPENDIX C. STANDARD TAPE FILE LABEL



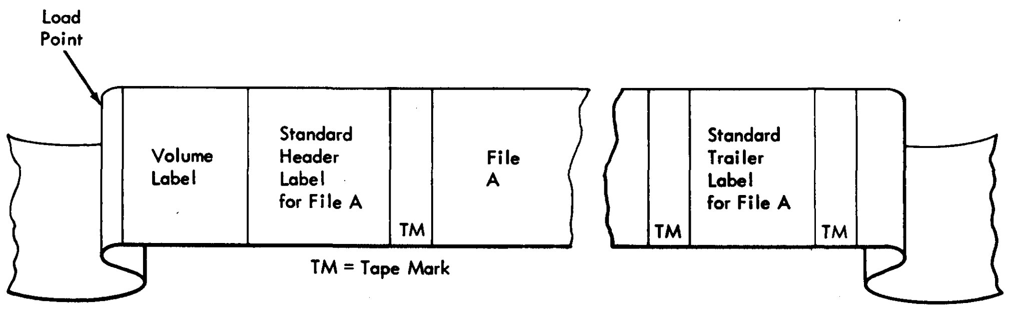
The standard tape file label format and contents are as follows:

FIELD	NAME AND LENGTH	DESCRIPTION
1.	LABEL IDENTIFIER 3 bytes, EBCDIC	identifies the type of label HDR = Header -- beginning of a data file EOF = End of File -- end of a set of data EOV = End of Volume -- end of the physical reel
2.	FILE LABEL NUMBER 1 byte, EBCDIC	always a 1
3.	FILE IDENTIFIER 17 bytes, EBCDIC	uniquely identifies the entire file, may contain only printable characters.
4.	FILE SERIAL NUMBER 6 bytes, EBCDIC	uniquely identifies a file/volume relationship. This field is identical to the Volume Serial Number in the volume label of the first or only volume of a multi-volume file or a multi-file set. This field will normally be numeric (000001 to 999999) but may contain any six aphameric characters.
5.	VOLUME SEQUENCE NUMBER 4 bytes	indicates the order of a volume in a given file or multi-file set. The first must be numbered 0001 and subsequent numbers must be in proper numeric sequence.
6.	FILE SEQUENCE NUMBER 4 bytes	assigns numeric sequence to a file within a multi-file set. The first must be numbered 0001.
7.	GENERATION NUMBER 4 bytes	uniquely identifies the various editions of the file. May be from 0001 to 9999 in proper numeric sequence.
8.	VERSION NUMBER OF GENERATION 2 bytes	indicates the version of a generation of a file.
9.	CREATION DATE 6 bytes	indicates the year and the day of the year that the file was created: Position — Code — Meaning 1 — blank — none 2-3 — 00-99 — Year 4-6 — 001-366 — Day of Year (e.g., January 31, 1965, would be entered as 65031).
10.	EXPIRATION DATE 6 bytes	indicates the year and the day of the year when the file may become a scratch tape. The format of this field is identical to Field 9. On a multi-file reel, processed sequentially, all files are considered to expire on the same day.
11.	FILE SECURITY 1 byte	indicates security status of the file. 0 = no security protection 1 = security protection. Additional identification of the file is required before it can be processed.
12.	BLOCK COUNT 6 bytes	indicates the number of data blocks written on the file from the last header label to the first trailer label, exclusive of tape marks. Count does not include checkpoint records. This field is used in trailer labels.
13.	SYSTEM CODE 13 bytes	uniquely identifies the programming system.
14.	RESERVED 7 bytes	Reserved. Should be recorded as blanks.

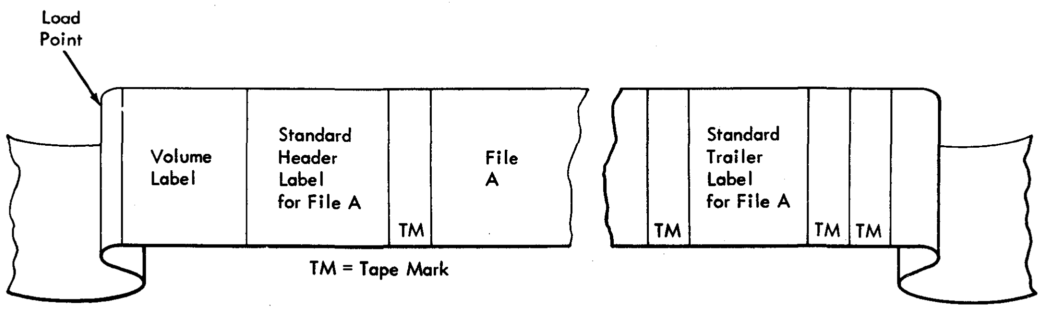
APPENDIX D. TAPE FILES WITH STANDARD LABELS


Load Point
Volume Label (80 Char)
Standard Header Label for File A (80 Char)
TM
File A
TM
Standard Trailer Label for File A
TM
Standard Header Label for File B
TM
File B
TM
Standard Trailer Label for File B
TM
TM
TM = Tape Mark

Multifile volume


Load Point
Volume Label
Standard Header Label for File A
TM
File A
TM
Standard Trailer Label for File A
TM
TM = Tape Mark

Multivolume file (first thru n-1 reels)


Load Point
Volume Label
Standard Header Label for File A
TM
File A
TM
Standard Trailer Label for File A
TM
TM
TM = Tape Mark

Single volume file or last reel of multivolume file

APPENDIX E. SUMMARY OF JOB CONTROL AND OPERATOR COMMUNICATION STATEMENTS

When accepted:
Job Ctrl. (Job Control Statements) - Between Jobs and Job Steps. Affects Background Job only. Generally effective only for given job, whereas operator JC holds until next IPL. Precede operation with //b.
JC (Job Control) - Between Jobs and Job Steps.
AR (ATTN Routine) - After pressing request key on 1052.
FI (Foreground Initiation) - After ATTN command START.
IPL - During Initial Program Loading

Label Information: The label statements for a tape file are VOL followed by TPLAB
The label statements for a DASD file are VOL followed by DLAB followed by one XTENT for each area of the file in the volume
All label statements must immediately precede the EXEC statement to which they apply

Job Ctrl.	Operator Commands						
	JC	AR	FI	IPL			
				X	Add a device to PUB table	ADD	X'cuu' [(k)], devicetype [, X'ss'] k = S (switchable) or 0-255 for priority designation (0 = highest)
	X	X			Allocate foreground program areas	ALLOC	{F1=nK [, F2=nK] | F2=nK [, F1=nK]} n=an even integer in the range 0-510
	X				Assign symbolic names to I/O devices	ASSGN	SYSxxx, {X'cuu' | UA | IGN} [{, X'ss' | , ALT}] [, TEMP]
X			X		Assign symbolic names to I/O devices	ASSGN	SYSxxx, {X'cuu' | UA | IGN} [{, X'ss' | , ALT}] UA: not valid for foreground initiation
		X			Cancel execution of current job in specified area	CANCEL	{BG | F1 | F2}
	X		X		Cancel execution of current background job or initiation of foreground job	CANCEL	blank
	X				Close a magnetic tape or DASD unit	CLOSE	SYSxxx [{, X'cuu' [, X'ss'] | , UA | , IGN | , ALT}]
X					Enter date in communication region	DATE	{mm/dd/yy | dd/mm/yy}
				X	Delete a device from PUB table	DEL	X'cuu'
X			X		Provide DASD file label information	DLAB	'label fields 1-3', c xxxx, yyddd, yyddd, 'system code' [, type] c = continuation code (any character) in col 72; enter remaining parameters (xxxx etc.) on a continuation statement beginning in column 16 xxxx = volume sequence no. yyddd, yyddd = file creation date followed by file expiration date 'system code' = 13 character string type = SD, DA, ISC, ISE
	X				Make a device unavailable for system operations	DVCDN	X'cuu'
	X				Return a device to available status	DVCUP	X'cuu'
X					Terminate job control, load program and execute	EXEC	[progname]
			X		Terminate foreground initiation, load program and execute	EXEC	progname
	X		X		Maintain I/O assignments for foreground area(s)	HOLD	{F1 [, F2] | F2 [, F1]}
X					Indicate beginning of job control data	JOB	jobname
X					Reserve storage at link edit time for tape and non-sequential disk file label processing	LBLTYP	TAPE(nn) for TOS or {TAPE [(nn)] | NSD(nn)} for DOS TOS {TAPE(nn): nn = decimal no. of pairs of VOL/TPLAB statements DOS {TAPE [(nn)]: nn ignored; use statement only if tape file label processing and no non-sequential DASD file processing NSD(nn): nn = largest number of extents per single file; use statement if any non-sequential DASD file processing}
X	X				Print list of I/O assignments	LISTIO	{SYS | PROG | F1 | F2 | ALL | SYSxxx | UNITS | DOWN | UA | X'cuu'}
			X		Print list of I/O assignments	LISTIO	{BG | F1 | F2 | UA | ALL}
	X	X	X		Log Job Control Statements and foreground initiation commands on SYSLOG	LOG	blank
	X	X	X		Print map of main storage areas	MAP	blank
		X	X		Transfer control to foreground program message routine	MSG	{F1 | F2}
X	X				Control magnetic tape operations	MTC	opcode, {X'cuu' | SYSxxx} [, nn] opcode = BSF, BSR, ERG, FSF, FSR, REW, RUN or WTM X'cuu': not valid for programmer Job Control nn = number of times operation is to be performed (in decimal form)
	X	X	X		Suppress logging of foreground initiation commands and Job Control Statements on SYSLOG	NOLOG	blank
X					Establish program options	OPTION	option 1 [, option 2, . . .] The options are listed at the end of the appendix.

When accepted:
<u>Job Ctrl.</u> (Job Control Statements) - Between Jobs and Job Steps. Affects Background Job only. Generally effective only for given job, whereas operator JC holds until next IPL. Precede operation with //b.
<u>JC</u> (Job Control) - Between Jobs and Job Steps.
<u>AR</u> (ATTN Routine) - After pressing request key on 1052.
<u>FI</u> (Foreground Initiation) - After ATTN command START.
<u>IPL</u> - During Initial Program Loading

Job Ctrl.	Operator Commands						
	JC	AR	FI	IPL			
X	X	X	X		Allow for operator intervention	PAUSE	[comments]
			X		Specify card reader for further foreground initiation commands	READ	X'cuu'
	X		X		Unassign I/O units for foregound area(s) at end of job for area	RELSE	{ F1[, F2] F2[, F1] }
X	X				Reset I/O assignment	RESET	{ SYS PROG ALL SYSxxx }
X					Provide ID & location of check-point records & initiate restart	RSTRT	SYSxxx, nnnn [, filename]
							nnnn = checkpoint record indentification filename = symbolic name if a 2311 disk is the checkpoint file
				X	Initialize date and time	SET	DATE = { mm/dd/yy dd/mm/yy } [, CLOCK=hh/mm/ss]
	X				Initialize date, clock, UPSI, etc.	SET	[DATE = { mm/dd/yy dd/mm/yy }] [, CLOCK=hh/mm/ss] [, UPSI-nnnnnnnn] [, LINECT=n1] [, RCLST=n2] [, RCPCH=n3]
							n = 0, 1 or X (unchanged) n1 = standard number of lines for output on each page of SYSLST n2 = decimal number indicating minimum number of SYSLST disk records remaining to be written before operator warning n3 = decimal number indicating minimum number of SYSPCH disk records remaining to be written before operator warning
		X			Initiate program in specified area	START	{ <u>BG</u> F1 F2 }
	X				Stop background program processing	STOP	blank
		X	X		Give interval timer support to specified area	TIMER	{ BG F1 F2 }
X			X		Provide tape file label information (in lieu of VOL, TPLAB)	TLBL	filename, ['file-ID'], [date], [file-serial-number] , [volume-sequence-number], [file-sequence-number] , [generation-number], [version-number]
X			X		Provide tape file label information	TPLAB	{ 'label fields 3-10' 'label fields 3-13' }
							fields 3-13: enter a continuous character string through column 71, continuation code (any character) in col. 72 and complete the string in a continuation statement beginning in column 16
	X				Load UCS buffer	UCS	SYSxxx, phasename [, FOLD] [, BLOCK] [, NULMSG]
	X		X		Unassign I/O units for foreground area(s) immediately	UNA	{ F1 [, F2] F2 [, F1] }
X					Set user program switches	UPSI	nnnnnnnn
							n = 0, 1 or X (unchanged)
X			X		Provide volume label information	VOL	SYSxxx, filename
X			X		Indicate limits of a file on a DASD unit	XTENT	type, sequence, lower, upper, 'serial no', SYSxxx [, B_2]
							type = 1 for data area (no split cylinders) 2 for overflow area (for indexed sequential file) 4 for index area (for indexed sequential file) 128 for data area (split cylinder) sequence = 0-255 for sequence no. of extent within multi-extent files lower/upper = lower/upper extent in the form $B_1C_1C_1C_2C_2C_2H_1H_2H_2$ B_2 = 0 for 2311, 0-9 for 2321; $B_2 = B_1$ if omitted
X					Indicate end-of-data file input for a job step	/*	ignored
X					Indicates end-of-job	/&	ignored
X					Provide programmer-to-operator comments	*	comments
	X	X	X	X	End-of-communication	Ⓑ	blank
							Ⓑ is alter code 5. If alter code 0 (Ⓒ) is issued, the command will be ignored.

<u>Meanings of Operands</u>

<u>devicetype</u> = actual device as:

2400T7	Seven track tape
2400T9	Nine track tape
1442N1	Card Reader Punch
2520B1	Card Reader Punch
2501	Card Reader
2540R	Card Reader
2540P	Card Punch
2520B2	Card Punch
2520B3	2520B1 Card Punch
1442N2	Card Punch
1403	Printer
1403U	Printer with UCS
1404	Printer
1443	Printer
1445	Printer
1050A	Printer - Keyboard
2671	Paper Tape Reader
UNSP	Unsupported device
UNSPB	Unsupported burst-mode device
2311	Disk Drive
2321	Data Cell Drive
2701	Line Adapter Unit
2702	Trans. Control Unit
2703	Trans. Control Unit
7770	Audio Response Unit
7772	Aduio Response Unit
2260	Display Unit
1285	Optical Reader

<u>X'ss'</u>

ss	Density	Parity	Translate	Convert
10	200	odd	off	on
20	200	even	off	off
28	200	even	on	off
30	200	odd	off	off
38	200	odd	on	off
50	556	odd	off	on
60	556	even	off	off
68	556	even	on	off
70	556	odd	off	off
78	556	odd	on	off
90	800	odd	off	on
A0	800	even	off	off
A8	800	even	on	off
B0	800	odd	off	off
B8	800	odd	on	off
C0	800	single density 9-track		
C0	1600	dual density 9-track		
C8	800	dual density 9-track		

The above values are for magnetic tape. If omitted X'C0' (9 track) or X'90' (7 track) is assumed. X'ss' designates the SADxxx requirement for a 2702 as follows: X'00' = SAD0, X'01 = SAD1, X'02' = SAD2, X'03' = SAD3. X'00' is the default option. This information is not accepted on the ASSGN statement.

<u>Option</u>	
LOG	Log control statements on SYSLST
DUMP	Dump registers and main storage on SYSLST if abnormal program end
LINK	Indicate object module is to be linkage edited; write output of language translator on SYSLNK
DECK	Output object module on SYSPCH
LIST	Output listing of source module on SYSLST; also object module listing and errors (Assembler)
LISTX	Output Procedure Division Map (COBOL) on SYSLST; output object module (PL/I) on SYSLST
SYM	Output symbol table (Assembler) on SYSPCH; output Data Division Map (COBOL) on SYSLST; output symbol table (PL/I) on SYSLST
XREF	Output symbolic cross reference list (Assembler and PL/I) on SYSLST
ERRS	Output summary of errors in source program (COBOL, FORTRAN and PL/I) on SYSLST
To suppress any of the above options, prefix operation with NO; e.g., LOG, NOLOG	
48C	48-character set on SYSIPT (PL/I)
60C	60-character set on SYSIPT (PL/I)
CATAL	Catalog program/phase in core image library after linkage editing
STDLABEL	Cause all <u>sequential</u> DASD or tape labels to be written on the standard label track
USRLABEL	Cause all DASD or tape labels to be written on the user label track

APPENDIX F. SYMBOLIC UNIT NAMES

Symbolic Name (Note 1)	Function	May be Assigned to	Remarks
SYSRES	System residence unit	2311 Disk Storage Drive	Assingment is established by the system during an IPL and cannot be altered until another IPL occurs.
SYSRDR	Job Control Background program input device	Card Readers: 1442,2501, 2520, or 2540 Magnetic Tape Units: 2400 Series (Note 2) Disk Storage Drive: 2311	1. Tape units may be either 7-or 9-track (dual density). If 7-track, the data conversion feature is required. 2. If the 1052 printer-keyboard is inoperable, SYSRDR must be assigned to a card reader.
SYSIPT	Processing program input device	Card Readers: 1442,2501, 2520, or 2540 Magnetic Tape Units: 2400 Series (Note 2) Disk Storage Drive: 2311	1. Tape units may be either 7-or 9-track (dual density). If 7-track, the data conversion feature is required. 2. If the 1052 printer-keyboard is inoperable, SYSIPT must be assigned to a card reader. 3. SYSIPT and SYSRDR may be assigned to the same physical device. 4. Required for system generation and maintenance, and language translators.
SYSIN	Assign SYSIPT and SYSRDR to the same physical device	Same units as SYSIPT	1. Tape units may be either 7-or 9-track (dual density). If 7-track, the data conversion feature is required. 2. If the 1052 printer-keyboard is inoperable, SYSIN must be assigned to a card reader.
SYSPCH	Punched output	Card Punches: 1442,2520, or 2540 Magnetic Tape Units: 2400 Series (Note 2) Disk Storage Drive: 2311	1. Tape units may be either 7- or 9-track (dual density). If 7-track, the data conversion feature is required. 2. If the 1052 printer-keyboard is inoperable, SYSPCH must be assigned to a card punch. 3. SYSLST and SYSPCH may be assigned to a single magnetic tape (see SYSOUT). 4. Required for system generation and maintenance, and for language translators.
SYSLST	System output unit	Printers: 1403,1404,1443, or 1445 Magnetic Tape Units: 2400 Series (Note 2)	1. Tape units may be either 7-or 9-track dual density). If 7-track, the data conversion feature is required. 2. 1404 used for continuous forms only. 3. If SYSPCH and SYSLST are assigned to a tape unit, they can be assigned to the same physical device (see SYSOUT). 4. If the 1052 printer-keyboard is inoperable, SYSLST must be assigned to a printer. 5. The 1445 printer must be used as a 1443 printer. 6. Required for system generation and maintenance, and for language functions.
SYSOUT	Assign SYSPCH and SYSLST	2400 Series Magnetic Tapes only (Note 2)	1. Tape units may be either 7- or 9-track (dual density). If 7-track, the data conversion feature is required. 2. If the 1052 printer-keyboard is inoperable, SYSOUT cannot be assigned.
SYSLNK	Compile/Link Edit and Execute system file	2311 Disk Storage Drive	1. Must be a single XTENT.
SYSLOG	Operator Messages	Printer-Keyboard: 1052 Printers: 1403,1404,1443, or 1445	1. Can be used by any program. 2. If the 1052 printer-keyboard is inoperable, SYSLOG must be assigned to a printer.
SYS000 to SYS244	I/O operations for processing programs	Card Readers: 1442,2501, 2520,. or 2540 Card Punches: 1442,2520, or 2540 Printers: 1403,1404,1443, or 1445 Magnetic Tape Units: 2400 Series Optical Reader: 1285 Disk Storage Drive: 2311 Data Cell Drive: 2321 Paper Tape Readers: 2671 Printer-Keyboard: 1052 Data Collection System: 1030 Data Communication System: 1050, or 1060 Selective Calling Stations: AT&T 83B3 Teletypewriter Terminal: AT&T Models 33 and 35 Western Union Plan: 115A Outstation Binary Synchronous Communication: System/360 (Models 30,40,50,65,75)	1. If a dump of a foreground program is desired, SYS000 must be assigned to a printer or magnetic tape unit. All storage dumps in the background area use SYSLST. 2. SYS000 through SYS009 are the minimum number of units defined in any system. 3. Tape units may be either 7- or 9-track (dual density). If 7-track, the data conversion feature is required. 4. The 1404 printer is used for continuous forms only.

Note 1. System logical units (e.g. SYSLST, SYSLNK) cannot be assigned to a foreground program.

Note 2. A tape written in 1600 bpi mode must have a tape mark written on it before this tape can be used on a 7-track or 9-track drive operating in 800 bpi mode.

APPENDIX G. SYMBOLIC UNITS REQUIRED FOR IBM-SUPPLIED PROGRAMS: Part 1 of 5

		Language Translators					Linkage Editor	Autotest
Symbolic Unit	Operand of EXEC Statement	ASSEMBLY[1]	COBOL[1]	PL/I	RPG[1]	FORTRAN[1]	LNKEDT	ATLEDT
SYSIPT	Required:	Always						
	Function:	Input for program						
	Device Type:	Card reader or tape unit, or disk						
SYSLOG	Required:	Always					Always	
	Function:	Operator Communication					Operator Messages	
	Device Type:	1052 Printer-Keyboard					1052 Printer-Keyboard	
SYSLST	Required:	Always						Yes
	Function:	Programmer messages, listing, etc.						
	Device Type:	Printer or tape unit, or disk						
SYSPCH	Required:	If DECK specified in OPTION statement[2]					No	No
	Function:	Punched output						
	Device Type:	Card punch or tape unit, or disk						
SYSRDR	Required:	Always						
	Function:	Job Control statement input						
	Device Type:	Card reader or tape unit, or disk						
SYSLNK	Required:	If LINK or CATAL is specified in the OPTION statement					Always	Always
	Function:	Receive input for Linkage Editor					Input	Output[6]
	Device Type:	Disk unit					Disk unit	Disk unit
SYS001	Required:	Always	Always		Always	Always	Always[7]	
	Function:	Mixed workfile	Workfile		Mixed workfile	Workfile	Work file	
	Device Type:	Disk or tape unit[3]	Disk or tape unit[4]		Disk or tape unit[5]		Disk or tape unit	
SYS002	Required:	Always	Always		Always	No	No	No
	Function:	Mixed workfile	Workfile		Mixed workfile			
	Device Type:	Disk or tape unit[3]	Disk or tape unit[4]		Disk or tape unit[5]			
SYS003	Required:	Always	Always		Always	No	No	No
	Function:	Mixed workfile	Workfile		Mixed workfile			
	Device Type:	Disk or tape unit[3]	Disk or tape unit[4]		Disk or tape unit[5]			
SYS004	Required:	No	Optional	No	No		No	No
	Function:		Debug pockets					
	Device Type:		Disk or tape unit[4]					
SYS005	Required:	No					No	Optional
	Function:							Output
	Device Type:							Tape unit

[1] Either DECK or LINK, but not both, may be specified in the OPTION statement for any language translator.
[2] SYSPCH is also required for the Assembler if SYM is specified in the OPTION statement.
[3] Assembler has three variants -- one using tape workfiles only, a second using disk workfiles only, and a third using mixed workfiles. The background partition must be 14K or larger for mixed workfiles.
[4] If disk is used, SYS001, SYS002, SYS003 must be disk; if tape is used, they must be tape.
[5] For mixed workfiles, the background partition must be 12K or larger.
[6] Autotest Workfile
[7] For Autotest, used only by the Autotest Linkage Editor.

APPENDIX G (continued): Part 2 of 5

		Librarian				
Symbolic Unit	Operand of EXEC Statement	MAINT	RSERV	SSERV	DSERV	CORGZ
SYSIPT	Required: Function: Device type:	When cataloging to the relocatable or source statement library Book or module input Card reader or tape unit, or disk	No	No	No	No
SYSLOG	Required: Function: Device type:	Always Operator Messages 1052 Printer- Keyboard				
SYSLST	Required: Function: Device type:	Always Programmer Messages and/or listings Printer or tape unit, or disk				
SYSPCH	Required: Function: Device type:	No	If punch function is specified Punched output. Card punch or tape unit, or disk.		No	No
SYSRDR	Required: Function: Device type:	Always Control statement input Card reader or tape unit, or disk				
SYS000	Required: Function: Device type:	No	No	No	No	No
SYS001	Required: Function: Device type:	No	No	No	No	No
SYS002	Required: Function: Device type:	No	No	No	No	Always Output Disk unit

		Disk Sort/Merge	Tape Sort/Merge	7772 Vocabulary File Utility
Symbolic Unit	Operand of EXEC Statement	DSORT	TSRT	VOC72UT
SYSIPT	Required: Function: Device type:	Always Input for program Card reader, tape unit, or disk		Always Input for program Card reader or tape unit
SYSLOG	Required: Function: Device type:	Always Operator Messages 1052 Printer- Keyboard		Always Operator messages 1052 Printer- Keyboard
SYSLST	Required: Function: Device type:	Always Programmer Messages Printer, tape unit, or disk		Always Listings Printer or tape unit
SYSPCH		Not Used		Not Used
SYSRDR	Required: Function: Device type:	Always Job Control statement input Card reader, tape unit, or disk		Always Job Control statement input Card reader or tape unit
SYSLNK		Not Used		Not Used
SYS000	Required: Function: Device type:	Optional Input, work area, or output Disk unit	No	See Note 1
SYS001	Required: Function: Device type:	Only for tape output Input, work area, or output Disk or tape unit	Always Output Tape unit	
SYS002	Required: Function: Device type:	Only for tape input Input, work area, or output Disk or tape unit (AA)	Always Input Ⓐ Tape unit Ⓗ	
SYS003	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (BB)	Always for sort, optional for merge Workfile for sort, input for merge Ⓑ Tape unit	
SYS004	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (CC)	Always for sort, optional for merge Workfile for sort, input for merge Ⓒ Tape unit	See Note 2 Input Vocabulary Files Tape unit
SYS005	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (DD)	Always for sort, optional for merge Workfile for sort, input for merge Ⓓ Tape unit	See Note 1
SYS006	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (EE)	Optional Workfile for sort, input for merge Ⓔ Tape unit	
SYS007	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (FF)	Optional Workfile for sort, input for merge Ⓕ Tape unit	
SYS008	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (GG)	Optional Workfile for sort, input for merge Ⓖ Tape unit	
SYS009	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (HH)	No	
SYS010	Required: Function: Device type:	Optional Input, work area, or output Disk or tape unit (II)	No	

Note: There are no mandatory assignments of symbolic units for DSORT with disk input/output units. Any logical unit SYSnnn may be assigned.

Ⓐ	Must be user's first input file, for merge
Ⓑ	Must be user's second input file, for merge
Ⓒ	Must be user's third input file, for merge
Ⓓ	Must be user's fourth input file, for merge
Ⓔ	Must be user's fifth input file, for merge
Ⓕ	Must be user's sixth input file, for merge
Ⓖ	Must be user's seventh input file, for merge
Ⓗ	If multi-file input with alternate drives is specified, the IOCS CLOSE routine will not switch to the alternate drive when encountering an end-of-file condition. The operator must mount the first volume of the next file on the same tape unit on which the last volume of the preceding file was mounted.

(AA)	Must be user's first tape input file (FILE A)
(BB)	Must be user's second tape input file (FILE B)
(CC)	Must be user's third tape input file (FILE C)
(DD)	Must be user's fourth tape input file (FILE D)
(EE)	Must be user's fifth tape input file (FILE E)
(FF)	Must be user's sixth tape input file (FILE F)
(GG)	Must be user's seventh tape input file (FILE G)
(HH)	Must be user's eighth tape input file (FILE H)
(II)	Must be user's ninth tape input file (FILE I)

Notes.
1. SYSnnn is used as a utility workfile. SYSppp is used to record Operative Vocabulary File. SYSnnn and SYSppp are assigned unique extents in 2311 disk storage. SYSnnn is always required for Operative Vocabulary File updating. It is required for Operative Vocabulary File building only when tables are to be created. Building an Operative Vocabulary File made up of only a residuum does not require SYSnnn.
2. If the Input Vocabulary is in the form of punched cards, it can be added to the control card deck; the resulting deck is the system input file (which may be copied on a magnetic tape) to be read from SYSIPT. If the Input Vocabulary is in the form of a magnetic tape file, it must be read from SYS004 while the control statements must be read from SYSIPT.

		Utilities								
		Card To Printer/Punch	Card To Tape	Card To Disk	Tape To Card	Tape To Tape	Tape To Printer	Tape To Disk	Tape To Data Cell	Tape Compare
Symbolic Unit	Operand of EXEC Statement	CDPP	CDTP	CDDK	TPCD	TPTP	TPPR	TPDK	TPDC	TPCP
SYSIPT	Required: Function: Device type:	Always Utility control statement input Card reader, tape unit, or disk								
SYSLOG	Required: Function: Device type:	Always Operator Messages 1052 Printer- Keyboard								
SYSLST	Required: Function: Device type:	Always Programmer Messages Printer, tape unit, or disk								
SYSPCH		Not Used								
SYSRDR	Required: Function: Device type:	Always Job Control statement input Card reader, tape unit, or disk								
SYSLNK		Not Used								
SYS000		Not Used								
SYS001		Not Used								
SYS002		Not Used								
SYS003		Not Used								
SYS004	Required: Function: Device type:	Always Input for program Card reader			Always Tape input and alternate tape input Tape unit					Always Tape to be compared
SYS005	Required: Function: Device type:	If printed output is specified Printer	Always Ⓐ Tape unit	Always Ⓑ Disk unit	No	Always Ⓐ Tape unit	Always Printer	Always Ⓑ Disk unit	Always Ⓒ Data cell	Always Tape to be compared
SYS006	Required: Function: Device type:	If punched output is specified Card punch	No	No	Always Card punch	No	No	No	No	No

Note: The DASD (direct access storage device) utility programs are not restricted to the use of SYS004, SYS005, and SYS006 for input or output. Any logical unit SYSnnn may be assigned.

Ⓐ Tape output and alternate tape output
Ⓑ Disk output and alternate disk output
Ⓒ Data cell output and alternate data cell output

APPENDIX G (continued): Part 5 of 5

Symbolic Unit	Operand of EXEC Statement	Utilities										
		Disk to Card	Disk To Disk	Disk To Printer	Disk To Tape	Disk To Data Cell	Data Cell To Data Cell	Data Cell To Printer	Data Cell To Tape	Data Cell To Disk	Clear Data Cell	Clear Disk
		DKCD	DKDK	DKPR	DKTP	DKDC	DCDC	DCPR	DCTP	DCDK	CLDC	CLRDSK
SYSIPT	Required:	Always										
	Function:	Utility control statement input										
	Device type:	Card reader, tape unit, or disk										
SYSLOG	Required:	Always										
	Function:	Operator Messages										
	Device type:	1052 Printer-Keyboard										
SYSLST	Required:	Always										
	Function:	Programmer Messages										
	Device type:	Printer, tape unit, or disk										
SYSPCH		Not used										
SYSRDR	Required:	Always										
	Function:	Job Control statement input										
	Device type:	Card reader, tape unit, or disk										
SYSLNK		Not required										
SYS000		Not required										
SYS001		Not required										
SYS002		Not required										
SYS003		Not required										
SYS004		Not required										
SYS005	Required:	No	No	Always	Always	No	No	Always	Always	No	No	No
	Function:				Ⓐ				Ⓐ			
	Device type:			Printer	Tape unit			Printer	Tape unit			
SYS006	Required:	Always	No	No	No	No	No	No	No	No	No	No
	Function:	Card output										
	Device type:	Card punch										

Note: The DASD (direct access storage device) utility programs are not restricted to the use of SYS004, SYS005, and SYS006 for input or output. Any logical unit SYSnnn may be assigned.

Ⓐ Tape output and alternate tape output
Ⓑ Disk output and alternate disk output
Ⓒ Data cell output and alternate data cell output

GC20-1685-0

Printed in U.S.A. GC20-1685-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)