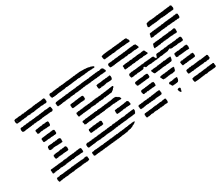
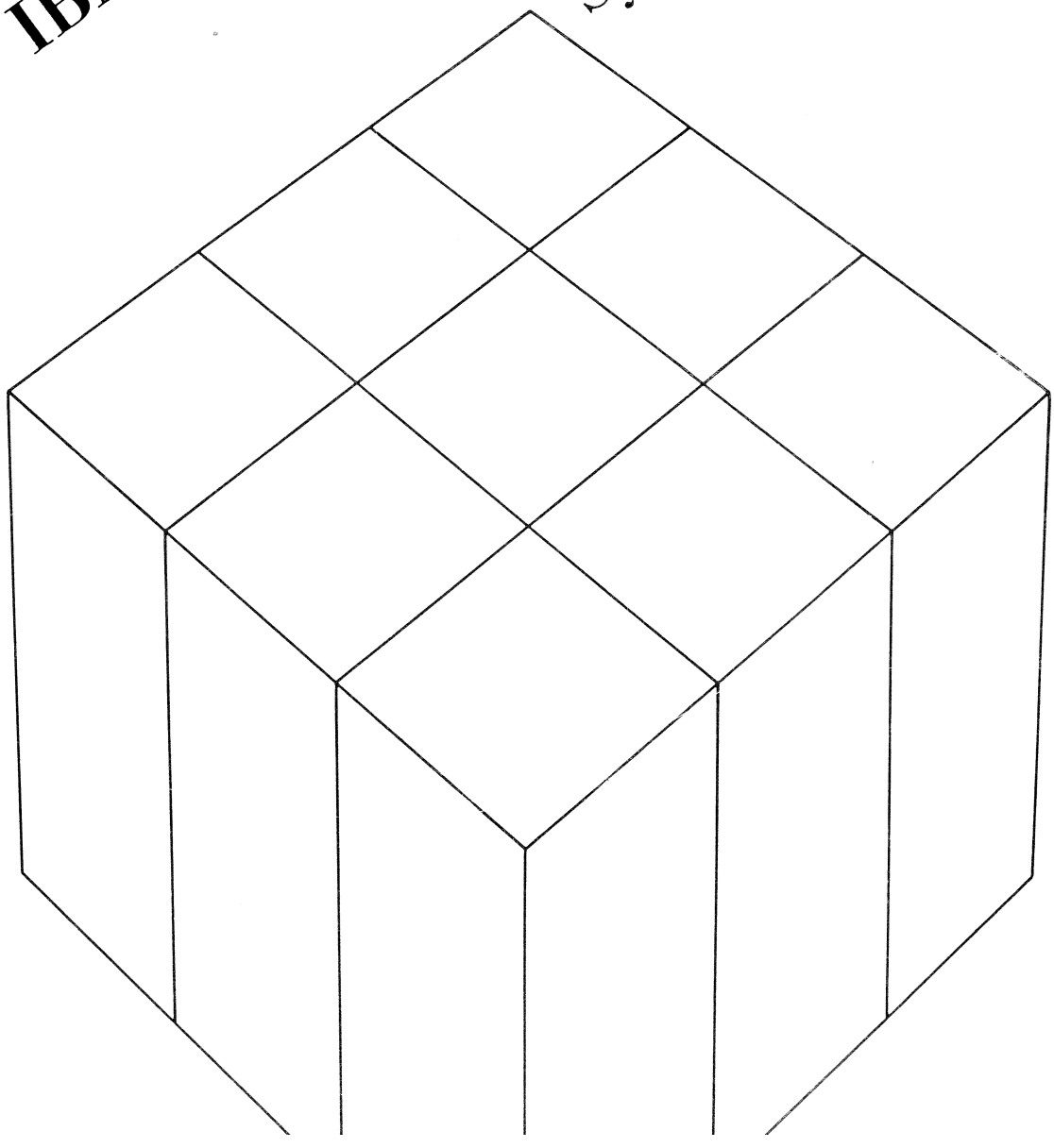


IBM Virtual Storage Extended Advanced Functions

System Control Statements



IBM Virtual Storage Extended Advanced Functions

System Control Statements

Version 2 Release 1

Program Number 5666-301

Order Number SC33-6198-02
File No. S370/4300-36

Third Edition (June 1987)

This edition is a major revision of the manual SC33-6198-1. It applies to Version 2, Release 1 of Virtual Storage Extended/Advanced Functions, Program Number 5666-301, and to all subsequent releases until otherwise indicated in new editions of Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30XX and 4300 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this document is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the addresses given below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed either to:

IBM Corporation
Dept. 6R1BP
180 Kost Road
Mechanicsburg, PA 17055, USA

or to:

IBM Deutschland GmbH
Dept. 3164
Schoenaicher Strasse 220
D-7030 Boeblingen, Federal Republic of Germany

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This manual is provided for those who need to know about the control statements that relate to VSE/Advanced Functions, the operating system support for a VSE installation. The manual consists of the following:

- Section 2, "Initial Program Load" on page 2-1 and Section 3, "Job Control and Attention Routine" on page 3-1 are of interest to anyone using the system, including system analysts, programmers, and operators. Detailed attention routine, job control statement, and job control command formats are given.

- Section 4, "Linkage Editor" on page 4-1 and Section 5, "Librarian" on page 5-1 are of interest to persons responsible for maintaining the resident system. These sections fully describe the control statements for the linkage editor and librarian programs.

- Section 6, "Edited Macro Service Program (ESERV)" on page 6-1 is of interest to programmers using the Assembler language. It describes the control statements necessary to de-edit and update edited macros.

- Section 7, "System Buffer Load (SYSBUFLD)" on page 7-1 is of interest to users who have an IBM 1403U or PRT1 printer attached to their system. The section describes the purpose of SYSBUFLD and how to use it.

- The Appendix contains a summary of the linkage editor control statements.

The control statements for the VSE/Advanced Functions utility programs are not described in this manual. You will find them, together with examples, in the prerequisite publication:

VSE/Advanced Functions System Management Guide.

Related publications are listed in the bibliography at the back of this book.

Contents

Figures

Summary of Amendments

For a complete overview of the functions which have become available since Version 2, Release 1, Modification Level 0 of VSE/Advanced Functions, refer to the publication *VSE/Advanced Functions Planning and Installation.* The amendments cover:

- Support of the IBM 3480 tape unit;
- Support of the IBM 4248 printer in native mode;
- Passing of multiple commands to the librarian program;
- An additional FORMAT specification in the librarian PUNCH command;
- An additional DISP specification in the job control DLBL statement.
- Support of the IBM 9370 processors and their related I/O devices:
  - IBM 9332 — An FBA disk device;
  - IBM 9335 — An FBA disk device;
  - IBM 9347 — A streaming-mode tape device;
  - IBM 4224 — A 3268-compatible terminal printer;
  - IBM 4234 — A 3262-3/13-compatible terminal printer;
  - IBM 4245-12D/20D
- Support of the IBM 3720 Communications Adapter.

Section 1. Introduction

This manual contains descriptions of VSE/Advanced Functions system control statements and commands. These statements and commands are grouped by function as shown below.

Initial Program Load

Before a job can be entered into the system for execution, the supervisor and the job control program must be loaded into storage. To do this, the operator starts the system by following the initial program load (IPL) procedure.

Job Control

After the system has been successfully started by means of the IPL procedure, it is ready to accept input for execution. Job control statements are entered on SYSRDR, job control commands at SYSLOG.

Job control runs in any virtual partition. It performs its functions between jobs and job steps. It is not present in the partition while an application program is being executed.

Attention Routine

When IPL is complete, and the system is running, the attention routine is available at all times. It allows the operator to alter certain system values, query the status of the system, and influence the execution of jobs in the system.

The functions of the attention routine are requested by entering attention routine commands at the console. Some attention routine commands are identical to job control commands, and both types of command are described in alphabetical order in Section 3, "Job Control and Attention Routine."

Linkage Editor

Before execution in storage, all programs must be placed in a sublibrary by the linkage editor. An exception to this rule is a single-phase program link-edited with the OPTION LINK. This is linked in VIO space and loaded from there into the partition.

The linkage editor prepares a program for execution by editing the output of a language translator into phase format. The linkage editor also combines separately assembled or compiled program sections or subprograms into phases.

Librarian

The librarian program is used to maintain data which must be readily available to the system, such as programs and cataloged procedures. These data are organized in libraries, which are subdivided into sublibraries, which in turn contain the data, organized in units called members. Each member is identified unambiguously by a library name, sublibrary name, member name and member type. Library, sublibrary and member names may be freely chosen. The member type depends on the type of data contained in the member, as follows:

A-Z, 0-9, S, #, @ for source programs, which are to serve as input for a language translator;

OBJ for modules, output from a language translator;

PHASE for executable program phases, ready for loading and execution in storage;

PROC for cataloged procedures;

DUMP for storage dumps.

A "type" other than these five may be used for members containing any user data.

Edited Macro Service Program (ESERV)

When an assembler macro has been processed (edited) by the assembler, it can no longer be updated directly. Any alterations must be made in the source, and this must be edited again before the changes can become effective. If the source of a macro is no longer available, the edited macro must be de-edited. The ESERV program de-edits macros, and gives you the opportunity to update them at the same time.

System Buffer Load (SYSBUFLD)

SYSBUFLD is a service program for users with IBM 1403U, 3203, 5203, and PRT1 printers. It can be executed as a job or job step to load the Forms Control Buffer (FCB) and/or the Universal Character Set Buffer (UCB) of these printers.

Control Statement Conventions

The conventions used in this publication to illustrate control statements are as follows:

1. Uppercase letters and punctuation marks (except as described in items 3 through 5 below) represent information that must be coded exactly as shown. The X' ' notation for hexadecimal value is no longer required but is still supported for compatibility reasons.

2. Lowercase letters and terms represent information that must be supplied by the programmer. Numeric characters are represented by n or m, alphameric characters by x or y.

3. Information contained within brackets [ ] represents an option that can be included or omitted, depending on the requirements of the program. Options separated by |, or stacked options, represent alternatives, one and only one of which **may** be chosen.
   For example:

   ```
   [A|B]
   ```

   Defaults are *underlined.*

4. Options contained within braces { } represent alternatives, one of which **must** be chosen. For example:

   ```
   {A|B}
   ```

5. An ellipsis (...) indicates that a variable number of items may be included.

6. Items *underlined* represent the option assumed (default) if a parameter is omitted (see 3).

7. Parentheses must be coded as shown.

8. Frequent abbreviations:

   a. cuu represents a hexadecimal number indicating the channel (c) and unit (uu) address of the device being specified.

b. volser represents the six-character identifier (the volume serial number) of a tape or disk volume. If you specify less than six characters, the value passed to the system is padded to the left with zeros, unless you enclose the specification in quotes. In this case, the value is padded to the right with blanks. For example,

the specification	is passed to the system as
VOL1	00VOL1
'VOL1'	VOL1__

Bear in mind that these two specifications will not match when compared by label checking routines. The IPL program always pads to the right with blanks.

For this manual, alphameric characters are defined to include the following: A - Z, 0 - 9, @, $, and #.

In case of any difference between the conventions given in this manual for control program functions and those appearing in IBM-supplied VSE component publications, observe the deviations given in the component publication.

Section 2. Initial Program Load

Operation of the system is initiated through an initial program load (IPL) procedure from the resident disk pack.

When the system enters the wait state, the operator must specify the device to be used for SYSLOG (by pressing END/ENTER), and type in the name of the supervisor to be loaded, together with other information. The format of this information is described in "The Supervisor Parameters Command" on page 2-3.

The IPL program reads the supervisor into low storage. If a read error occurs while the supervisor is being read, the system goes into a wait state and an error code is set in the first word of processor storage. The IPL procedure must then be restarted.

After successfully reading in the supervisor, the system enters the wait state a second time. The operator then causes an interrupt which, in turn, causes IPL to read its commands from the IPL communication device.

The IPL procedure can be automated almost completely by making use of the Automated System Initialization (ASI) facility. This facility allows all control statements and commands needed for the complete operating system start-up to be read from a sublibrary. For guidance on how to code and catalog an ASI procedure, refer to *VSE/Advanced Functions System Management Guide*. For information on how to execute an ASI procedure, refer to *VSE/Advanced Functions Operation*.

The following section describes the supervisor parameter command, followed by the IPL commands in alphabetical order:

ADD - to define I/O devices

DEF - to assign system logical units

DEL - to delete I/O devices

DLA - to define a label information area

DLF - to define the lock file

DPD - to define the page data set

SET - to set the system date and time

SVA - to increase the SVA size

SYS - to set options relating to disk file protection, channel queue entries, supervisor buffers, job accounting, access control, printer buffer-image loading, and number of sublibraries, and to set up areas for SDAID and other system monitoring functions.

ADD and DEL must precede any other IPL command, except the SET and SYS commands. DLF (if specified) must be the first command after ADD and DEL. SVA must be the last IPL command.

The Supervisor Parameters Command

This is the first command to be entered at the console during an interactive IPL, or the first command in the ASI IPL procedure. At the console, it must be entered in response to the message:

```
0I03D ENTER SUPERVISOR PARAMETERS [OR ASI PARAMETERS]
```

Because the system cannot accept any other command at this time, or as the first command in the procedure, the supervisor parameter command has no operation field, just operands. These are as follows:

```
┌─────┬──────────────────────────────────────────────────────────────────────────────┐
│ cuu,│[sup-name][,P|N][,LOG|NOLOG][,VPOOL={nK|mM}][,VIO={nK|mM}]                    │
└─────┴──────────────────────────────────────────────────────────────────────────────┘
  └──── Valid only in an ASI procedure.
```

Additional operand for supervisors generated with Mode = 370:

```
┌──────────────────┐
│[,VSIZE={nK|mM}]  │
└──────────────────┘
```

cuu
: This operand is valid **only** in an ASI **IPL procedure**. It specifies the physical address of the console device to be assigned to SYSLOG. (During interactive IPL, SYSLOG is defined by pressing END or ENTER on the appropriate device.)

sup-name
: Specifies the name of the supervisor to be loaded. The default supervisor name is $$A$SUP1.

P|N
: Specifies whether certain supervisor routines are to be made pageable or not. If you specify P, or omit the operand, the routines are made pageable. If you specify N, the routines are fixed in processor storage.

LOG|NOLOG
: Controls the logging of IPL commands at the console. If you specify LOG, or omit the operand, the system writes all IPL commands to SYSLOG. If you specify NOLOG, only invalid commands are listed.

VPOOL= nK|mM
: Specifies the size of the VPOOL in kilobytes (K) or megabytes (M). **n** and **m** must be decimal integers. If nK is specified, the system rounds **n** to the next higher multiple of 32 (for ECPS:VSE Mode and VM Mode) or 64 (for 370 Mode). The system default value is 64K.

VIO= nK|mM

This operand is **not** valid when the specified supervisor is generated with **Mode = VM**. It specifies the size of the VIO work area in kilobytes (K) or megabytes (M). **n** and **m** must be decimal integers, and they must be greater than or equal to the corresponding value in the VPOOL operand. If nK is specified, the system rounds **n** to the next higher multiple of 32 (for ECPS:VSE Mode) or 64 (for 370 Mode). The default VIO size is the current VPOOL size. The maximum size is 40M.

VSIZE= nK|mM

This operand is valid **only** when the specified supervisor is generated with **Mode = 370**. It specifies the maximum total size of all virtual areas which can be allocated in the system. These areas are:

- The supervisor;
- The SVA (Shared Virtual Area);
- The virtual partitions (ALLOC n values);
- The shared partitions (ALLOC S values).

If nK is specified, the system rounds the value to the next higher multiple of 64K. The default value is a minimum value which depends on certain supervisor generation options. The maximum value is 40M.

ADD

The ADD command is used to define the physical I/O devices attached to the system. The device addresses are entered into the PUB table. Either a single device or a series of devices of the same type can be added with one command.

Operation	Operands
ADD	cuu[:cuu|..cuu] [(S)] ,device-type [,mode|,SHR] [,EML]

cuu

Indicates the channel and unit number of the device(s) to be added. Leading zeros can be omitted (for example, ADD 00C,2501 may be coded as ADD C,2501).

The format cuu:cuu or cuu..cuu indicates that a series of devices of the same type, starting with the first cuu and ending with the second cuu is to be added. For example

```
ADD 130:137,3330
```

defines eight 3330 devices with addresses 130 through 137. If this type of specification is used for 2703 devices, all addresses must designate either Start/Stop or BSC lines.

(S)

Indicates that the device can be switched (that is, physically attached to two adjacent channels). The designated channel is the lower of the two channels. For the device on which the lock file resides, S must not be specified together with SHR.

device-type

Specifies the device type code of the device to be defined; see device type codes in Figure 2-2 on page 2-22.

Note: If you specify device type CTCA for a channel-to-channel adapter:

- *For* **cuu**, *specify the address of the line attached to the adapter;*
- *Do not specify any of the optional operands (S),* **mode** *or SHR.*

mode

This specification has different meanings for different device types, as follows:

Tape devices

mode specifies the mode setting (see ASSGN Statement).

If it is omitted, the following values are assigned:

C0 for 9-track tapes (2400, 3410 series);
D0 for 9-track tapes (3420, 3430 series);
60 for 8809 Magnetic Tape Unit;
60 for 9347 Magnetic Tape Unit;
90 for 7-track tapes;
00 for the 3480 Tape Subsystem.

3284, 3286, 3287, 3288, 3289
mode must be entered as 01 (see also Figure 2-2 on page 2-22).

3480 Tape Subsystem
mode can be:

00 for buffered write mode (default);
20 for unbuffered write mode.

3284,3286,3287
mode is required for these printers when they are used as console printer for a 3277 operator console. The required specification is 02 or 04.

2702
mode specifies SADxx (Set Address) requirements:

00 for SAD0 (default)
01 for SAD1
02 for SAD2
03 for SAD3

1270, 1275, 1419, 1419P, and 1419S
mode specifies the external interrupt bit associated with magnetic ink or optical character readers. The settings 01 through 20 correspond to the external interrupt code in low storage byte 87, bits 7 through 2 respectively:

01 byte 87 bit 7
02 byte 87 bit 6
04 byte 87 bit 5
08 byte 87 bit 4
10 byte 87 bit 3
20 byte 87 bit 2

3704/3705/3725
mode is required and specifies the type of channel adapter:

01 Type 1/4 channel adapter
02 Type 2/3 channel adapter

3705
For the 3705 SDLC Integrated Communications Adapter for the 4300 processors, **mode** must be 10.

2703

For the 2703 BSC Integrated Communications Adapter for the 4331 and 4361 Processors, **mode** can be:

40 EIB mode (Error Index Byte is to be set.)
00 non-EIB mode

Model 158, 3031, 3033 Processors

If you want to use, for example, a 3277 as operator communication device on a Model 158 or 3031, you can define the PA1 (request) key to be the PF1 key by setting bit zero of the device specification byte to 1 (for example X'80') using the **mode** specification:

```
ADD 140,3277,80
```

SHR

Indicates that the device to be added may be shared by two or more CPUs. SHR is valid only for the 33xx CKD device types and the 3370 FBA device. For the device on which the lock file resides, SHR may not be specified together with S. For ready disk units, the SHR option is reset during IPL if the disk unit is physically not shareable.

EML

Indicates that an emulated device is being added. That is, the specified device-type code does not correspond to the actual device type at the specified address. The EML operand causes IPL to ignore device type sensing, and add the device as the type specified in the ADD command.

DEF

The DEF command, which is mandatory, is used to assign a physical device to

- SYSREC, the logical device for the system recorder file, the hardcopy file, and the system history file
- SYSCAT, the logical device for the VSE/VSAM master catalog

Operation	Operands
DEF	SYSREC={cuu|volser}[,SYSCAT={cuu|UA|volser}]

cuu
: Indicates the channel and unit number of the physical device to be assigned.

SYSCAT=cuu
: May be specified only if VSE/VSAM is installed in the system.

SYSCAT=UA
: Is the default value.

volser
: Indicates the unique volume serial number of the disk to be assigned.

The assignments cannot be changed until the next IPL.

DEL

The DEL command is used to delete one or more of the I/O devices previously defined with the ADD command.

Operation	Operands
DEL	{cuu[:cuu|..cuu]}

cuu

Indicates the channel and unit number of the device(s) to be deleted.

The format cuu:cuu or cuu..cuu indicates that a series of devices of the same type, starting with the first cuu and ending with the second cuu is to be deleted. For example,

```
DEL 130:133
```

causes devices 130, 131, 132, and 133 to be deleted.

DLA

The DLA command, which is mandatory, defines or references a label information area. This area may be located on any disk device. Format and layout of the label information area are determined by the system.

The DLA command must be entered after the ADD (and DEL) commands and before the SVA command. Only one valid DLA command may be entered.

If you want to use previously created standard labels, enter either:

- A short form of the DLA command with only the NAME= and UNIT= or VOLID= operands, or
- A long form of the DLA command with exactly the same operands as the command used to create the standard label area.

To clear the standard label area, enter the long form of the DLA command with the same CYL and NCYL (or BLK and NBLK) operands, but a different NAME operand. The system issues the message "OVERLAP ON filename". Reply "DELETE" to format the new label area.

If you specify CYL and NCYL (or BLK and NBLK) different from those of an existing label area, the system issues the message "DUPLICATE NAME ON VOLUME". If you want to use the new extent, reply "DELETE" to this message.

Operation	Operands
DLA	NAME=areaname{,UNIT=cuu|,VOLID=volser}[,DSF={Y|N}] [,CYL=n[,NCYL=m]|,BLK=n[,NBLK=m]]

NAME=areaname
: Specifies the name of the label area, which can be one to eight alphameric characters. When the label area is first created, this name is entered in the VTOC of the device indicated in the UNIT or VOLID operand (in the form: DOS.LABEL.FILE.cpu-id.areaname). When referring to the label area during subsequent IPLs, use only the NAME and UNIT or VOLID operands.

UNIT=cuu
: Specifies the channel and unit number of the device containing the label area. The device type may be different from that of the SYSRES device. This operand may be specified together with VOLID.

VOLID=volser
: Identifies the unique volume serial number of the device containing the label area. This operand may be specified together with UNIT.

DSF=Y|N
: Specifies whether the label area is to be data-secured. If the operand is omitted, DSF = Y (Yes) is assumed.

CYL=n

Indicates, for CKD devices, the sequential number of the cylinder, relative to zero, where the label area is to begin. **n** must be a decimal number with one to three digits, with a minimum value of 1.

BLK=n

Indicates, for FBA devices, the sequential number of the block, relative to zero, where the label area is to begin. **n** must be a decimal number with one to ten digits, with a minimum of 2, but see the note on the NBLK operand below.

NCYL=m

Defines, for CKD devices, the size of the label area in number of cylinders. **m** must be a decimal number with one to three digits. If this operand is omitted, the device dependent default size is used. The maximum value for **m** is also device dependent. The table below gives the appropriate default and maximum values for the supported disk devices.

Device Type	NCYL Specification: Default	Maximum
2314	2	12
3330	2	13
3340	3	20
3350	1	8
3375	3	20
3380	2	16

NBLK=m

Defines, for FBA devices, the size of the label area in number of blocks. **m** must be a decimal number with a minimum of 12 and a maximum of 992.

Note: The sum of the NBLK specification and the BLK specification must be less than 558 000.

If this operand is omitted, the default, which is 200 blocks, is used.

DLF

The DLF command is used to define or reference the cross-system communication file (lock file). This file must be present when two or more VSE systems share disk storage devices. The DLF command is required if the supervisor was generated with disk sharing support and if devices are present which are defined with the SHR option in the ADD command.

The lock file has to be on a disk drive which is physically shared with all systems linked in the disk sharing environment. The device on which the lock file resides must not be defined as switchable. If used, the DLF command must be the first command after the ADD (and DEL) commands.

Operation	Operands
DLF	{UNIT=cuu|VOLID=volser}[,CYL=n[,NCYL=n]|,BLK=n[,NBLK=n]] [,DSF={Y|N}][,TYPE={N|F}][,NCPU=n]

UNIT=cuu
: Specifies the channel and unit number of the device containing the lock file. This operand can be used together with VOLID.

VOLID=volser
: Identifies the unique volume serial number of the disk containing the lock file. This operand can be used together with UNIT.

  No operands other than UNIT or VOLID are needed if an existing lock file is to be used. If, however, a new lock file is to be created or if a real-location is required, the following operands are also needed:

CYL=n
: Specifies, for CKD devices, the sequential number of the cylinder, relative to zero, where the lock file is to begin. **n** must be a decimal number with one to three digits.

NCYL=n
: Specifies how many cylinders of a CKD device are to be allocated to the lock file. The default is 1. See Figure 2-1 on page 2-13.

BLK=n
: Specifies, for FBA devices, the sequential number of the block, relative to zero, where the lock file is to begin. **n** must be a decimal number with a minimum of 2.

NBLK=n
: Specifies how many blocks of an FBA device are to be allocated to the lock file. The default is 80. See Figure 2-1 on page 2-13.

DSF=Y|N

Specifies whether the lock file is to be data-secured. If the operand is omitted, DSF = Y (Yes) is assumed.

TYPE=N|F

N, which is default, indicates that the lock file is not to be formatted. If you specify TYPE = N, but the lock file does not exist on the specified device or volume, the system ignores the operand and formats a new lock file.

F indicates that the system should format the lock file during IPL. Use this option only when a new lock file must be formatted, for example, because of an error in the existing one. Be sure to enter a DLF command with the TYPE = F operand at only one of the CPUs sharing the lock file.

NCPU=n

Specifies the number of machines, real or virtual, which share disk storage. Valid specifications for **n** are 2..31. The default is 4.

Note: If you want to use a previously created lock file, enter either:

A DLF command with only the UNIT= and/or VOLID= operands, or
A DLF command with exactly the same operands as the command used to create the existing lock file.

To reformat the existing lock file, enter the long form of the DLF command with the same CYL and NCYL (or BLK and NBLK) operands, but with the operand TYPE= F.

If you specify CYL and NCYL (or BLK and NBLK) different from those of an existing lock file, the system issues the message "DUPLICATE NAME ON VOLUME". If you want to use the new lock file, reply "DELETE" to this message.

The maximum number of resources that can be locked by a lock file of a given size can be calculated by the following formulae.

- For FBA devices:
  Number of resources = NBLK * 508 ÷ (12 + NCPU)

- For CKD devices:
  Number of resources = NCYL * 508 ÷ (12 + NCPU) * D,

  where D is a device-type-dependent factor of:
  308 for IBM 3330
  144 for IBM 3340
  810 for IBM 3350
  480 for IBM 3375
  690 for IBM 3380

Figure 2-1. Capacity of Lock File

DPD

The DPD command, which is mandatory, is used to define the page data set. The command is invalid if the VM/370 linkage facility is included in your system.

Operation	Operands
DPD	{UNIT=cuu|VOLID=volser},{CYL=n|BLK=n}[,NCYL=m|,NBLK=m] [,TYPE={N|F}][,DSF={Y|N}]

The operands of the DPD command may be given in any order.

UNIT=cuu
: Specifies the channel and unit number of the device that is to contain the page data set. You may specify this operand together with VOLID.

VOLID=volser
: Identifies the volume serial number (one to six alphabetic or numeric characters) of the disk pack that contains the page data set. If you do not specify VOLID, the volume serial number is not checked. You may specify this operand together with UNIT.

CYL=n
: Specifies, for CKD devices, the sequential number of the cylinder, relative to zero, where the page data set is to begin (in decimal). A specification of CYL = 0 indicates that the page data set extent is to begin on cylinder 0, track 1.

BLK=n
: Specifies, for FBA devices, the sequential number of the block, relative to zero, where the page data set is to begin. **n** must be a decimal number with a minimum of 2.

NCYL=m
: Specifies, for a multi-extent CKD page data set, the size of one page data set extent (in number of cylinders). **m** must be a decimal number with up to three digits.

NBLK=m
: Specifies, for a multi-extent FBA page data set, the size of one page data set extent (in number of blocks). **m** must be a decimal number with a minimum of 4.

TYPE=N
: TYPE = N is the default and indicates that the page data set need not be formatted. The TYPE operand is ignored for FBA devices.

  If TYPE = N is specified, but the page data set does not exist, or the extent limits have been changed, TYPE = N is ignored and the page data set is formatted during IPL.

TYPE=F

Indicates that the page data set is to be formatted during IPL. Formatting during IPL is required if the page data set has been damaged. A page data set on a shared CKD device is always formatted. The TYPE operand is ignored for FBA devices.

DSF=Y|N

Indicates whether the page data set is to be data-secured. Yes is the default. For multi-extent page data sets, the DSF specification is valid for the **first** extent definition only; it is ignored for any further extent definitions.

For each extent of a multi-extent page data set, a separate DPD command has to be entered. After each command, the operator will be prompted to enter the next extent definition until

- the entire virtual storage is mapped on the specified extents, or
- the maximum number of extents allowed (which is 15) is exceeded, or
- the operator enters a DPD command without the NCYL/NBLK operand, in which case the complete remaining storage will be mapped on this extent.

Up to 15 extents can be specified, which may reside on different volumes; up to three extents may be allocated on one volume. The various extents can be placed on different CKD device types, or can be mixed with FBA device extents.

If the size specified in the NCYL/NBLK operand is larger than the size actually needed for the page data set, the free cylinders/blocks are available to the user.

SET

The SET command, which is optional, is used to set the system date, the time-of-day (TOD) clock, and the system time zone. It is required only if the TOD clock has not been set since the last POWER ON; IPL will then prompt the operator to enter the SET command. The command may be entered at any time before the SVA command.

Operation	Operands
SET	[DATE=mm/dd/yy,CLOCK=hh/mm/ss][,ZONE={EAST|WEST}/hh/mm]

DATE=mm/dd/yy
: Specifies the date in months (1-12), day of the month (1-31), and year (last two digits of the year).

  After IPL this format can be changed to dd/mm/yy with the STDOPT command.

CLOCK=hh/mm/ss
: Specifies the local time-of-day in hours, minutes and seconds.

ZONE=EAST/hh/mm
: Specifies that the installation is located at a geographical position east of Greenwich.

ZONE=WEST/hh/mm
: Specifies that the installation is located at a geographical position west of Greenwich.

hh/mm
: Indicates the difference in hours and minutes between local time and Greenwich Mean Time. hh may be in the range 0-23, mm in the range 0-59.

The operands that have to be specified with the SET command depend upon the state of the TOD clock. The following groups can be distinguished:

1. If the TOD clock is in the set state, the command may be given in one of the three forms:

   ```
   SET ZONE=
   SET DATE= ,CLOCK=
   SET DATE= ,CLOCK= ,ZONE=
   ```

2. If the TOD clock is in the not-set state, the command **must** be given in one of the two forms:

   ```
   SET DATE= ,CLOCK=
   SET DATE= ,CLOCK= ,ZONE=
   ```

3. If the TOD clock is inoperative, the command must be given in the form:

```
SET DATE= ,CLOCK=
```

Note: If the TOD clock is in the set state, message 0I30I is printed. If the TOD clock is in the not-set state, message 0I31I is printed. If the TOD clock is inoperative, messages 0I32I and 0I31I are printed.

The time-of-day clock should always hold the exact time, that is, the time that has elapsed since January 1, 1900, 00.00 hrs.

SVA

This command is mandatory and must be the last command entered during the IPL procedure. It is used to allocate space within the SVA into which the user can later load his phases. The values specified in the SVA command are added to the system SVA space requirements, which depend on the supervisor being used.

All operands are optional. If the operands are not entered during IPL, there will be no space reserved in the SDL and SVA for user phases. However, an SVA of sufficient size to contain the required set of system phases and the default system GETVIS area will be created.

Operation	Operands
SVA	[SDL=n][,PSIZE=nK][,GETVIS=nK]

SDL=n
: Specifies the decimal number of entries in the system directory list to be reserved for user phases and IBM-supplied phases, in addition to the phases loaded automatically during IPL. For a list of those phases that are automatically loaded into the SVA during IPL, refer to *VSE/Advanced Functions Planning and Installation.* Do not specify entries for these phases, as this is done by IPL. The maximum number that can be specified is 963, minus the number of SDL entries for the automatically loaded phases. Note that only phases from IJSYSRS.SYSLIB and those generated with the SVA operand in the linkage editor PHASE statement can be loaded into the SVA at IPL.

PSIZE=nK
: Specifies the size of the area within the SVA which is to be reserved for user phases. **n** must be a decimal number and a multiple of 2. The specified size should be large enough for the user phases and for a maintenance area which is required when a phase with a copy in the SVA is replaced.

  Do not specify space for the phases loaded automatically into the SVA during IPL, as IPL will reserve the necessary space.

GETVIS=nK
: Indicates the size of the **additional** system GETVIS area which you can specify beyond the size allocated by the system. **n** must be a decimal number and a multiple of 2.

SYS

The SYS command, which is optional, specifies:

- System action when printers not READY for automatic buffer load;
- The number of supervisor buffers used for I/O processing;
- The number of channel queue entries to be allocated;
- Whether disk file protection should be active;
- Whether job accounting should be active;
- The size of the shared area for system monitoring functions;
- Whether security checking should be active;
- The number of sublibraries which can be assigned.

Operation	Operands
SYS	[BUFLD={YES|IGNORE]}[,BUFSIZE=n][,CHANQ=n][,DASDFP={YES NO}] [,JA={YES|NO}][,SDSIZE=nK][,SEC={YES|NO}][,SUBLIB=m]

BUFLD=YES | IGNORE
: Specifies what action the system is to take if a printer which supports automatic print-control-buffer loading is not READY during IPL.

  If you specify BUFLD = YES, or omit the operand, IPL stops, issues a console message, and waits for the requested operator action before continuing.

  If you specify BUFLD = IGNORE, IPL continues without waiting for operator action.

BUFSIZE=n
: Specifies the **number** of supervisor buffers to be used for I/O processing. For details, refer to *VSE/Advanced Functions System Management Guide.* The operand is invalid if the VM/370 linkage facility is included in the system (that is, the system is running in VM mode).

  n must be a decimal number with a maximum of seven digits. The following table shows the minimum and default values for BUFSIZE. These are dependent on whether the supervisor is generated with the option FASTTR or not.

FASTTR=	Default	Minimum
NO	n = 60	n = 10
YES	370 mode: n = 60 + (p - 2) * 20 ECPS:VSE mode: n = 120 + (p - 2) * 40 (where p = number of partitions specified in NPARTS)	n = 30 n = 60
p = number of partitions specified in NPARTS		

FASTTR and NPARTS are parameters of the FOPT and SUPVR generation macros, respectively.

CHANQ=n

Specifies the number of channel queue entries to be allocated. If you omit the operand, the system allocates the appropriate number of channel queue entries for the number of partitions active and the type and number of devices added.

MINIMUM	n = p + a
DEFAULT	n = (2 + d) * p + a + 15 (If this expression yields a value greater than 255, the maximum value of 255 is used)
MAXIMUM	n = 255
p= number of partitions specified in NPARTS operand d= number of disk devices added by ADD commands a= total number of ADDed devices	

DASDFP=YES|NO

Specifies whether disk file protection should be active. If you omit the operand, the system does not activate file protection.

JA=YES|NO

Specifies whether job accounting should be activated. If YES is specified, CPU-times and SIOs for all devices are accounted.

SDSIZE=nK

Specifies the size of a shared V=R area for system monitor functions, for example SDAID. This operand is valid only when the system is running in 370-mode. Valid specifications for **n** are:

Minimum: 0
Maximum: 256
Default: 48.

SEC=YES|NO
: Specifies whether access control should be active. If YES is specified, the system carries out access authorization checking, and, if VSE/Access Control Logging and Reporting (ACLR) is available, access logging is activated.

SUBLIB=m
: Specifies the number of sublibraries which may be attached to the whole VSE/Advanced Functions system at any one time. m must be a decimal integer from 10 to 2000. If the operand is omitted, the system uses the default value of 100.

Device Class	Actual IBM Device	Device Type Code
Automatic Data Unit	3644 Automatic Data Unit (attached via the Loop Adapter feature)	3791L
Audio Response Units	7770 Audio Response Unit	7770
Card Punches	1442N2 Card Punch 2520B2 Card Punch 2520B3 Card Punch 2540 Card Punch 3525 Card Punch	1442N2 2520B2 2520B3 2540P 3525P
Card Readers	2501 Card Reader 2540 Card Reader 3505 Card Reader	2501 2540R 3505
Card Read Punches	1442N1 Card Read Punch 2520B1 Card Read Punch 2560 Multifunction Card Machine 3525 Card Punch (with optional read feature) 5424/5425 Multifunction Card Unit	1442N1 2520B1 2560 3525RP 5425
Disk	2311 Disk Storage Device (see Note) 2314 Direct Access Storage Facility (see Note) 2319 Disk Storage Facility (see Note) 3310 Direct Access Storage Device (see note) 3330 Disk Storage Model 1 and 2, or 3333-1 3330 Disk Storage Model 11 3340 Disk Storage (General) 3340 Disk Storage with RPS feature (with or without 3340 Data Module, Model 35) 3340 Disk Storage with RPS feature (with or without 3340 Data Module, Model 70) 3344 Direct Access Storage 3344 Direct Access Storage with RPS 3340 Disk Storage without RPS feature (with or without 3340 Data Module, Model 35) 3340 Disk Storage without RPS feature (with or without 3340 Data Module, Model 70) 3350 Direct Access Storage 3370 Direct Access Storage Device (see note) 3375 Direct Access Storage 3380 Direct Access Storage 9332 Disk Storage 9335 Disk Storage	2311 2314 2314 FBA 3330 3330B 3340 3340R 3340R 3340 3340R 3340 3340 3350 FBA 3375 3380 FBA FBA
Note: When emulating a 2311 or 2314/2319 disk on an FBA device, submit the appropriate ADD commands for both device types. (This does not apply if the processor is a 4361.)		

Figure 2-2 (Part 1 of 4). Device Type Codes

Device Class	Actual IBM Device	Device Type Code
Diskette Storage Devices	3540 Diskette Input/Output Unit 4331 Diskette Drive, Feature No. 3401 7443 Service Record File (on 3031 service support console)	3540 3540 7443
Display Operator Consoles and Console Printers	3277 Model 2 Display Console 3278 Model 2A Display Console 3279 Model 2C Color Display Console 3284,3286 or 3287 Console Printer for Models 138, 148 and 158 in 3277 Display Mode (mode must be entered as 04) 3284, 3286, or 3287 Console Printer (With 327x Control Unit (local non-SNA), (mode must be entered as 02)	3277 3277 3277 3277 3277
Displays Stations	3277 or 3278 Model 2A Display Station (mode need not be entered) 3277 or 3278 Model 2A Display Station (attached in burst mode to a multiplexer channel; mode need not be entered) 3279 Model 2A Color Display Station 8775 Display Terminal (attached via the Loop Adapter feature)	3277 (local 3270) 3277B (local 3270) 3277B 3791L
Magnetic Ink Character Recognition (MICR) Devices	1255 Magnetic Character Reader 1259 Magnetic Character Reader 1419 Magnetic Character Reader 1419 Dual Address Adapter Primary Control Unit 1419 Dual Address Adapter Secondary Control Unit	1419 1419 1419 1419P 1419S
Magnetic Tape Devices	9-track Magnetic Tape Units, 2400 series 7-track Magnetic Tape Units, 2400 series 9-track 3410 Magnetic Tape Units 7-track 3410 Magnetic Tape Units 9-track 3420 Magnetic Tape Units 7-track 3420 Magnetic Tape Units 9-track 8809 Magnetic Tape Unit 9-track 3422 Magnetic Tape unit 9-track 3430 Magnetic Tape unit 3480 Tape Subsystem 9347 Magnetic Tape Unit	2400T9 2400T7 3410T9 3410T7 3420T9 3420T7 8809 3430 3430 3480 8809

Figure 2-2 (Part 2 of 4). Device Type Codes

Device Class	Actual IBM Device	Device Type Code
Optical Readers	1270 Optical Reader Sorter	1419
	1275 Optical Reader Sorter Primary Control Unit	1919P
	1275 Optical Reader Sorter Secondary Contr.Unit	1419S
	1287 Optical Reader	1287
	1288 Optical Page Reader	1288
	3881 Optical Mark Reader	3881
	3886 Optical Character Reader	3886
	3890 Optical Character Reader/Sorter	3890
Printers	1403 Printer	1403
	1403 Printer with UCS feature	1403U
	1443 Printer	1443
	3200 Laser Beam Printer (Supports 8000 Kanji Character Set)	3800
	3211, 3203-4, 3203-5, 3289 Model 4, and 3262 Models 1, 5, 11 printers	PRT1
	3800 Printing Subsystem	3800
	3800 Printing Subsystem with Burster-Trimmer--Stacker (BTS)	3800B
	3800 Printing Subsystem with BTS and additional CGS	3800BC
	3800 Printing Subsystem with additional Character Generation Storage (CGS)	3800C
	3800 Model 3 Channel Attached Page Printer	AFP
	4245 Line Printer	PRT1
	4248 Line Printer in 3211 compatibility mode	PRT1
	4248 Line Printer in native mode	4248
Terminal Printers	3482, 3486, 3287 or 3288 Printer with 3272 Control Unit or 3274-x1B Control Unit (Mode must be entered as 01) -see Note.	3277
	When attached in burst mode to a multiplexer Channel: (Mode must be entered as 01) -see Note.	3277B
	3262 Model 3 Printer with 3272 Control Unit attached in burst mode to a Multiplexer Channel: (Mode must be entered as 01) -see Note.	3277B
	3284, 3286, 3287 or 3288 Printer with 3274-x1D Control Unit (Mode must be entered as 01) - see Note	3277
	3289 Printer (except Model 4) with 3274-x1B Control Unit (Mode must be entered as 01) -see Note.	3277
	When attached in burst mode to a multiplexer Channel: (Mode must be entered as 01) -see Note.	3277B
	4245-12D/20D:	
	As CRT console printer (mode=02)	3277
	Attached via 3274-1D control unit (mode=04)	3277
	Attached via WSA as system printer	PRT1
	4224 As CRT console printer (mode=02)	3277
	Attached via 3274-1D control unit (mode=04)	3277
	4234 As CRT console printer (mode=02)	3277
	Attached via 3274-1D control unit (mode=04)	3277
Note: These terminal printers cannot be assigned to SYSLST.		

Figure 2-2 (Part 3 of 4). Device Type Codes

Device Class	Actual IBM Device	Device Type Code
Printer Keyboard	3210, 3215 Console Printer Keyboards	1050A
	Model 138/148/158 Console in printer-keyboard mode with 3284, 3286, or 3287 Console Printer attached	1050A
Reader/ Inscriber	3890 Document Processor (Only PIOCS support)	3890
Tele-communication Lines	2701 Data Adapter Unit	2701
	Model 135 Integrated Communications Adapter (ICA)	2701
	Model 138 Integrated Communications Adapter (ICA)	2701
	2702 Transmission Control Unit	2702
	2703 Transmission Control Unit	2703
	Model 138 Integrated Communications Adapter	3701
	3704 Communications Controller	3704
	3705 Communications Controller	3705
	3720 Communications Controller	3720
	3725 Communications Controller	3725
	3791 Local Communications Controller	3791L
	3274-x1A Control Unit	3791L
	3274-x1B Control Unit	3277
	3274-x1D Control Unit	3277
	SDLC ICA on 4300 processors (mode must be entered as 10)	3705
	BSC ICA on 4300 processors	2703
	4331 Communications Adapter for BSC or Start/Stop lines	2703
	3704/3705 Communications Controller in Emulation Mode	2703
	3138 ICA for BSC or Start/Stop lines	2701
	Channel-to-Channel Adapter	CTCA
Unsupported Devices	Unsupported, no burst mode on multiplexer channel	UNSP
	Unsupported, with burst mode on multiplexer channel	UNSPB

Figure 2-2 (Part 4 of 4). Device Type Codes

Section 3. Job Control and Attention Routine

This section contains descriptions, formats, and usages of the job control commands and statements, and attention routine commands, which are identified as follows:

Job control statement - JCS

Job control command - JCC

Attention routine command - AR

Figure 3-1 on page 3-2 contains the commands and statements grouped by function, and also indicates the programs or routines for which they are valid.

Type of Command or Statement	Operation	Valid for JCS	AR	JCC
Job Identification	JOB /&	X X		
User Identification	ID	X		X
File Definition	DLBL EXTENT TLBL /* /+	X X X X X		
Library Definition	LIBDEF LIBDROP LIBLIST	X X X		X X X
Pass Information to Operator	*	X		
Job Stream Control	BATCH CANCEL PAUSE PRTY START STOP TPBAL UNBATCH	 X 	X X X X X X 	 X X X X X see Note
Setting Symbolic Parameters	SETPARM PROC	X X		X X
Conditional Job Control	/. GOTO IF ON	X X X X		 X X X
Setting System Parameters	ALLOC ALLOCR MSECS NPGR SET SIZE STDOPT	 X	X X X X 	X X X X X X
Note: Valid only in a foreground partition.				

Figure 3-1 (Part 1 of 2). JCS, JCC, and AR by Function

Type of Command or Statement	Operation	Valid for JCS	AR	JCC
Pass Information to Program	DATE	X		
	OPTION	X		
	OVEND	X		X
	UPSI	X		
Execution of Program	EXEC	X		X
	RSTRT	X		
Operator Communications	ALTER		X	
	DSPLY		X	
	DUMP		X	
	END or ENTER key		X	X
	IGNORE		X	X
	LOG		X	X
	MAP		X	X
	MODE		X	
	MSG		X	
	NEWVOL		X	
	NOLOG		X	
	ONLINE		X	
	RC		X	
	REPLID		X	
	SETMOD		X	
	UNLOCK		X	
	ZONE	X		
Control of I/O System	ASSGN	X		X
	BANDID		X	
	CLOSE	X		X
	DVCDN			X
	DVCUP			X
	FREE		X	
	HOLD			X
	LFCB		X	
	LISTIO	X		X
	LUCB		X	
	MPXGTN		X	
	MTC	X		X
	PWR	X		X
	RESERV		X	
	RESET	X		X
	ROD			X
	SETDF		X	
	SETPRT	X		X
	UCS			X
	VOLUME		X	

Figure 3-1 (Part 2 of 2). JCS, JCC, and AR by Function

Job Control Statements

Job Control statements must conform to the following formatting rules.

- **Identifier**. Two slashes (//) identify the statement as a control statement. They **must** be in columns 1 and 2. At least one blank must immediately follow the second slash.

  Exceptions:

  /. (label statement)
  /& (end-of-job statement)
  /* (end-of-data file statement)
  /+ (end-of-procedure statement)
  * (comment statement)

- **Operation**. Describes the operation to be performed. At least one blank follows its last character.

- **Operands**. May be blank or may contain one or more entries. If a statement includes two or more operands, they must be separated by commas. The last term **must** be followed by a blank, unless its last character is in column 71. Any blank within the operand is considered an end-of-operand indication, and no further processing of that statement occurs, unless the operand is within quotes. Exceptions are the IF and ON statements, which have blanks between the condition expression and the action specification.

Job Control and Attention Routine Commands

Job control commands and attention routine commands contain the operation code, at least one blank, and then the specified operands. The operands are separated by commas. The operation code usually begins in column 1 of the command, but this is not required.

- Job control commands (JCC) are issued between jobs or job steps and are entered through SYSRDR or SYSLOG. (Job control statements, on the other hand, are usually coded as part of the input stream and are entered through SYSRDR.)

- Attention routine commands (AR) can be issued from the console at any time.

Continuation of Commands and Statements

When job control statements are entered through SYSRDR, job control will accept continuation cards or lines only for the DLBL, EXEC, IF, LIBDEF, LIBDROP, LIBLIST, PROC, SETPARM, SETPRT and TLBL statements. In these statements, up to six continuation lines are accepted. The operands of the line to be continued may be:

- Entered up to and including column 71, or;
- Interrupted after the comma or equals sign separating two operands. Any columns between the interruption and column 72 must contain blanks.

A character string enclosed in single quotes (') is regarded as a single operand. Do not interrupt it before column 71, even if it contains commas or equal signs.

Position 72 of the line to be continued must always contain a non-blank continuation character (usually a C). The continuation line must start in column 16. For example:

```
1               16                                          72
// LIBDEF PHASE,SEARCH=MYLIB.MYSUBA,                        C
                CATALOG=YOURLIB.YSUBA,                      C
                TEMP
```

When entered through SYSLOG, all job control statements and commands (except those which have no separating commas, and the overriding statements which follow an EXEC statement with the OV operand), and all attention routine commands can be continued on subsequent lines. The existence of a continuation line is indicated by a minus sign immediately following the last delimiting comma on the current line. The command or statement is then continued at the start of the next line. Example:

```
ALLOC R,F1=128K,-
F2=228K,F3=128K,-
F4=128K
```

Continuation lines may also be entered on SYSLOG in the same way as on SYSRDR.

Job Control Statements Summary

A brief description of the job control statements follows.

ASSGN	Used at execution time to assign a specific device address to the symbolic unit name used.
CLOSE	Closes either a system or a programmer logical unit assigned to tape, disk, or diskette.
DATE	Contains a date that is put in the communications region.
DLBL	Contains file label information for disk or diskette label checking and creation.
EXEC	Indicates the end of job control statements for a job step and that the job step is to be executed.
EXEC PROC	Calls a cataloged procedure and defines values for symbolic parameters.
EXTENT	Defines each area, or extent, of a disk file or diskette volume.
GOTO	Causes JC to skip all following statements (except JOB, /&, / +) up to the specified label statement.
ID	Used to specify user identification and password.
IF	Causes skipping or execution of the following statement dependent on the specified condition.
JOB	Indicates the beginning of control information for a job.
LIBDEF	Defines library chains.
LIBDROP	Drops library chain definitions.
LIBLIST	Lists library chain definitions.
LISTIO	Used to get a listing of I/O assignments on SYSLOG or SYSLST.
MSECS	Changes or displays the time-slice for partition balancing.
MTC	Controls operations on magnetic tapes.
NPGR	Defines the number of programmer logical units which may be assigned to a partition.
ON	Causes specified action to be done if the specified condition is true after any step in the following job stream.
OPTION	Sets one or more of the job control options.

Figure 3-2 (Part 1 of 2). Job Control Statements Summary

OVEND	Indicates that no more overwrite statements will follow for the respective procedure.
PAUSE	Causes a pause immediately after processing this statement, or at the end of the current job step.
PROC	Defines and initializes symbolic parameters in a procedure.
PWR	Passes a PRELEASE or PHOLD command to POWER.
RESET	Resets I/O assignments to the standard assignments.
RSTRT	Restarts a checkpointed program.
SETPARM	Assigns a character string or return code to the specified parameter.
SETPRT	Loads the IBM 3800 buffers.
STDOPT	Resets system defaults.
TLBL	Contains file label information for tape label checking and writing.
UPSI	(User Program Switch Indicators) Allows the user to set program switches that can be tested.
ZONE	Initializes the zone field in the communications region.
	DELIMITER STATEMENTS:
/.	Label statement.
/*	Indicates the end of a data file.
/&	Indicates the end of a job.
*	Job control comments.
/+	Indicates the end of a procedure or librarian End-of-Data.
	THE FOLLOWING DOS OR DOS/VS JOB CONTROL STATEMENTS ARE NO LONGER SUPPORTED UNDER VSE/ADVANCED FUNCTIONS:
	DLAB
	LBLTYP
	TPLAB
	VOL
	XTENT

Figure 3-2 (Part 2 of 2). Job Control Statements Summary

If an invalid job control statement is entered, a message is issued so that the programmer or operator can correct the statement in error.

Whenever an invalid statement is indicated, the entire statement must be reissued to be effective. This rule applies even if only one operand was invalid. It also applies if the statement itself was correct, but could nót be executed because the appropriate system environment was not available. For example, if an OPTION LINK is entered without a SYSLNK assignment, the OPTION statement is invalid. You must re-enter the OPTION statement after assigning SYSLNK.

Sequence of JCS and JCC

The job control statements for a specific **job** always begin with a JOB statement and end with a /& (end-of-job) statement. A specific job consists of one or more **job steps**. Each job step is initiated by an EXEC statement. Preceding the EXEC statement are any job control statements necessary to prepare for the execution of the specific job step. One limitation on the sequence of statements preceding the EXEC statement is that DLBL statements must immediately precede the corresponding EXTENT statements. If the DLBL and EXTENT statements for a temporary SYSLNK area are in the job stream, they should precede the OPTION LINK or OPTION CATAL statement.

Conditional Job Control

Normally, the statements or commands in a job stream are read by job control in the sequence in which they are entered on SYSRDR or SYSLOG, and the job steps are executed in this order.

However, you can cause the system to execute or bypass parts of the job stream conditionally, dependent on the result of previously executed steps within the same job.

The result of a job step can be reflected in a return code between 0 and 4095, which must be set by the executed program

- by placing the desired return code in register 15 and branching at the end of the program to the return address which was supplied in register 14 when the program was invoked. The programmer must take care that register 15 is set correctly, otherwise the job flow will be unpredictable. (Note that the first two bytes of Register 15 are not part of the return code. Bit 0 of the register indicates whether a dump is required, the rest of the two bytes is reserved.)
- by the EOJ macro
- by the DUMP macro
- by an equivalent method in high-level programming languages.

If a return code greater than 4095 is issued, job control assumes a return code of 4095. If no return code is issued, a return code of zero is assumed. The return code can be tested (see statements IF, ON), and the sequence of execution altered if appropriate (see statements GOTO, /. label), or the step parameters for a fol-

lowing step can be set accordingly (see statement SETPARM). If the job control program receives a return code greater than or equal to 16, it terminates the job, unless an ON statement specifying a different action for this return code has been given. Such an action may be the execution of a certain part of the job stream after abnormal termination or cancellation of the job. The actions to be taken in the case of abnormal termination must be specified in the statements:

ON &RC&Ar.16... if a high return code is encountered;
ON &ABEND... if the job terminates abnormally; or
ON &CANCEL... if the job is canceled.

Parameterized Procedures

In order to make the handling of job streams easier and more flexible, VSE/Advanced Functions Version 2 provides facilities for altering, not only the sequence of execution of job control statements, but also the values in their operands.

The value, or part of the value, in the operand field of a job control statement can be modified at execution time if it is coded as a symbolic parameter (see below), and the appropriate new value is assigned to it

- In a PROC statement (if the parameter is in a cataloged procedure)
- In a // EXEC PROC statement (if the parameter is in a cataloged procedure)
- By a SETPARM command or statement which precedes it in the job stream
- By a SETPARM command or statement entered by the operator.

The value you assign to a symbolic parameter must be a character string. It can be 0 to 50 characters long. If the string contains national or special characters, it must be enclosed in quotes, which the system does not take as part of the value of the string. A string consisting of only alphabetic and/or numeric characters does not need enclosing quotes. Quotes may not be used within the string itself. An ampersand (&) within the string must be coded as a double ampersand (&&) to avoid confusion with the delimiters of symbolic parameters.

The current value of a symbolic parameter can also be tested in an IF statement, giving you the possibility of influencing the sequence of execution of a job.

Symbolic Parameters

A symbolic parameter is a name consisting of one to seven alphanumeric characters, of which the first must be alphabetic. This name may be freely chosen, and the same symbolic parameter may occur any number of times in a job stream.

Avoid using "OV" as a symbolic parameter name. If it is coded as the first parameter passed in an EXEC statement, job control will interpret it as the Override operand.

When a symbolic parameter is used in the operand of a job control command or statement you must indicate to job control that it is a symbolic parameter by preceding it with an ampersand (&). The end of the parameter must be indicated by a period (.), unless it is followed by a delimiter, that is, a non-alphameric character.

The operation and comments fields of job control commands or statements may not contain symbolic parameters, only the operand field. However, symbolic parameters may occur anywhere in the operand field. This includes operands in quotes, for example the file-ID in DLBL statements or the PARM operand in EXEC PROC statements.

Symbolic parameters can be used only between a // JOB statement and the end of the job.

Here is an example of an ASSGN statement with symbolic parameters:

```
// ASSGN SYS001,&UNIT.&VOLUME.
```

Here, you could omit the periods, like this:

```
// ASSGN SYS001,&UNIT&VOLUME
```

because the symbolic parameters are ended by the non-alphameric characters '&' and blank, respectively. Let us assume that this statement is contained in a cataloged procedure named PROC1, and that SYS001 should normally be assigned to the physical unit address 380, and that you do not require any special tape volume. You should code the procedure PROC1 as follows:

```
// PROC UNIT=380,VOLUME=
...
...
// ASSGN SYS001,&UNIT&VOLUME
```

When the procedure is called, job control will substitute 380 for &UNIT, and ignore the VOLume parameter. The job will be executed as if the statement read:

```
// ASSGN SYS001,380
```

If, for a particular run of the application, you want to use the device address 381, and use a particular tape volume, for example 888888, you must assign the appropriate values in the // EXEC PROC statement, as follows:

```
// EXEC PROC=PROC1,UNIT=381,VOLUME=',VOL=888888'
```

The job will then be executed as if the statement read:

```
// ASSGN SYS001,381,VOL=888888
```

That is, the values given in the EXEC PROC statement override those in the PROC statement.

You may want to assign SYS001 to unit 382 and use volume 777777 if, for example, the preceding job step ended with a return code higher than 4. In this case, you would code your procedure PROC1 as follows:

```
// PROC UNIT=380,VOLUME=
...
...
// IF $RC>4 THEN
// SETPARM UNIT=382,VOLUME=',VOL=777777'
// ASSGN SYS001,&UNIT&VOLUME
...
```

If the preceding step does end with a return code greater than 4, the job will be executed as if the statement read:

```
// ASSGN SYS001,382,VOL=777777
```

That is, the values assigned in the SETPARM statement override those given in the PROC and EXEC PROC statements.

A job control operand may consist of several symbolic parameters, or it may consist partly of a symbolic parameter, partly of a literal specification. The value assigned to the symbolic part is concatenated with the literal part, as follows:

PARAMETER	ASSIGNMENT	EXECUTED AS
&SIZE.(80)	SIZE=BLOCK	BLOCK(80)
&SIZE(80)	SIZE=BLOCK	BLOCK(80)
&LIBNAME.LIB	LIBNAME=PRIV1	PRIV1LIB
SYS&NUM	NUM=003	SYS003
&A..B	A=X	X.B
&C&UU	C=2,UU=81	281

You may also assign to a symbolic parameter a value consisting of several job control statement operands, as follows:

PARAMETER	ASSIGNMENT	EXECUTED AS
&OPERAND	OPERAND='380,VOL=666666'	380,VOL=666666

Nested Procedures

A cataloged procedure can call another cataloged procedure. That is, an EXEC PROC statement may occur within a procedure. Procedure A "contains" procedure B if the statement EXEC PROC = B occurs in procedure A. If the statement EXEC PROC = C occurs in procedure B, then procedure B contains procedure C, and procedure A also contains procedure C. Or conversely:

Procedure B is contained in procedure A;

Procedure C is contained in procedure A and in procedure B.

In general, any cataloged procedure may call any other cataloged procedure, with the following exceptions:

- A procedure must not call itself. That is, procedure A must not issue the statement EXEC PROC=A.

- A procedure must not call a procedure in which it is contained. That means, in the example above, that procedure B must not issue the statement EXEC PROC=A, and procedure C must not issue the statements EXEC PROC=A or EXEC PROC=B.

- All procedures in one nesting must have been cataloged with the same DATA= operand on the CATALOG statement (either all with DATA=YES or all with DATA=NO).

Procedures can be nested at up to 16 levels. Nesting Level 0 denotes the job control statements read from SYSRDR or SYSLOG. Level 1 denotes procedures called by an EXEC PROC statement on SYSRDR or SYSLOG. Level 2 denotes procedures called from Level 1 procedures, and so on up to Level 15. In the example above, assuming that EXEC PROC=A was issued from SYSRDR, procedure A is Level 1, procedure B is Level 2 and procedure C is Level 3.

Scope of Symbolic Parameters

A symbolic parameter is normally valid only at the nesting level on which it is defined. If defined by a SETPARM statement issued from SYSRDR, the parameter is valid until End-of-Job. If the SETPARM statement is in a procedure, the parameter it defines is valid until End-of-Procedure.

If a parameter is defined in an EXEC PROC or PROC statement, it is valid until End-of-Procedure.

Note that a parameter can be passed to a lower-level procedure, for example from a Level 1 procedure to a Level 2 procedure. The parameter will then remain valid after the lower-level procedure ends, but will cease to be valid at the end of the procedure in which it was defined.

For detailed information on nested procedures and the passing and substitution of parameters, see *VSE/Advanced Functions System Management Guide.*

Printing of Job Control Statements and Commands

Job control statements and commands are printed on SYSLOG and/or SYSLST after they have been processed, as follows:

- symbolic parameters are substituted,

- the active data in columns 16 - 71 of continuation cards are chained together,

- unnecessary blanks are removed, where this will not affect performance,

- double ampersands and quotes are reduced to single ones,

- continuation cards which were not processed by JC because they were in error are printed in their original format to facilitate debugging,

- statements that have become longer than 120 characters (because of chaining of continuation cards) are printed line by line (120 characters per line).

If you wish to have your JC statements or commands printed out in the format in which you coded them, you can specify the operand LOGSRC in the OPTION statement of the job. In this case, the system will write each statement which contains symbolic parameters or a continuation twice, once in the source form, as coded, and once in the form described above.

Command and Statement Formats (JCS, JCC, AR)

Detailed descriptions of the formats and functions of individual JCS, JCC, and AR statements and commands follow in alphabetic order. If the JCS and JCC or JCC and AR formats coincide, this is indicated under " Type."

ALLOC (Allocate Storage to Partitions)

The ALLOC command allocates virtual and processor storage to the partitions of a VSE system.

ALLOC is available as a job control and as an attention routine command. The initial allocation of storage to foreground (Fn) partitions **must** be made using the **job control** command. Later alterations of partition size may be made using the attention routine command.

The layout of virtual storage differs between 370 mode (multiple virtual address spaces) and ECPS:VSE and VM mode (single virtual address space). This means that the range of valid specifications in the space-id operand differs, too.

For details of the virtual storage layout in the various modes, see *VSE/Advanced Functions System Management Guide.*

Operation	Operands	Type
ALLOC	[space-id,]part={nK|mM}[,part={nK|mM}]...	JCC, AR

`space-id`
Indicates in which address space the specified amount of storage for the named partition(s) is to be allocated. Valid specifications are:

370 Mode:

1, 2 or 3
In the specified virtual address space

S
In the shared virtual address space

R
In the real address space (processor storage).

ECPS:VSE Mode or VM Mode:

1
In virtual address space

R
In real address space.

The default value of space-id for all modes is 1.

Note: An ALLOC R command for a given partition can be given only after an ALLOC n or ALLOC S for the same partition.

part

Indicates the partition to which storage is to be allocated. Valid specifications are:

BG, F1..F9, FA, FB.

Restriction: If the system is generated with less than 12 partitions, the number of foreground (Fn) partitions is reduced accordingly. For a 5-partition system, for example, valid specifications are BG and F1..F4.

nK

Specifies, in kilobytes, the amount of storage to be allocated. Valid specifications are 0 or an integer. Specifying 0 means that the entire storage allocated to the named partition is freed. The partition can no longer be used.

The system rounds up the specified integer to a multiple of:

64 in 370 mode for virtual address spaces;

4 in 370 mode for real address space;

2 in ECPS:VSE mode for real and virtual address space;

4 in VM mode.

The resulting rounded-up value must be at least

4K in real address space allocations, and
128K in virtual address space allocations.

mM

Specifies, in megabytes, the amount of storage to be allocated. Valid specifications are 0 or an integer. Specifying 0 means that the entire storage allocated to the named partition is freed. The partition can no longer be used.

The upper limit for nK or mM is the amount of space available for new allocation in the specified address space. At least one partition must be specified.

After IPL, the BG partition has a default size of 1 megabyte. Partitions must always be allocated or reallocated explicitly; the size of an unspecified partition is **not** changed.

To delete a foreground partition from the system, you must issue an UNBATCH command in the partition, and then an ALLOC command specifying a size of 0K or 0M for the respective partition. No allocation takes place when the ALLOC command would move the start address of a partition upwards and/or the end address downwards while that partition is active. (Exception: The end address of the partition in which job control is processing an ALLOC command may be moved downwards as long as the virtual storage allocated to the partition does not drop below 128K.)

ALTER (Alter Contents of Virtual Storage)

The ALTER command allows the operator to alter 1 to 16 bytes of virtual storage, starting at the specified hexadecimal address. After the command has been entered and the END/ENTER key pressed, the hexadecimal representation of the information to be placed in storage should be entered on the device assigned to SYSLOG. Two hexadecimal characters (0 through F) must be entered for each byte to be changed. If an odd number of characters is entered, the last character is ignored and its associated byte is unaltered.

Operation	Operands	Type
ALTER	[space-id,]address	AR

`space-id`
Specifies in which address space the alteration at the given address is to be made. Valid specifications are:

R, 1..3 in 370 mode;

1 in ECPS:VSE mode and VM mode.

The default value for all modes is 1.

`address`
Indicates the six-digit hexadecimal address, with leading zeros if necessary, at which storage alteration is to start. The highest address that can be specified is 16 MB minus 15 (FFFFF0).

If the specified address is within the supervisor area or the shared virtual area (SVA), a message is issued and the operator has the option to cancel the command or to change the address.

If the specified address is within an address range which has not been allocated to a partition, the command is ignored and a corresponding information message issued.

If the bytes to be altered cross the boundary from a valid to an invalid address space, the command is ignored and a corresponding information message is issued.

ASSGN (Assign Logical Unit)

The ASSGN command or statement assigns a logical I/O unit to a physical device. Multiple logical units are allowed to be assigned to one physical unit within the same partition. Assignments are effective only for the partition in which they are issued. No physical devices except disks can be assigned to (shared by) several active partitions at the same time.

The job control **statement** (type JCS) is temporary. It remains in effect only until the next change in assignment or until the end of job, whichever occurs first. At the completion of a job, a temporary assignment is automatically restored to the permanent assignment for the logical unit.

The job control **command** (type JCC) is permanent. It remains in effect until the next permanent assignment, a DVCDN command, or re-IPL of the system, whichever occurs first. A CLOSE command for a system logical unit on disk or diskette also removes a permanent assignment. See also the description of the TEMP/PERM operands.

Operation	Required Operands:	Type
[//] ASSGN	SYSxxx, {cuu|(address-list)|UA|IGN|SYSyyy| device-class|device-type}	JCS, JCC
	Optional Operands for Disk Devices Only: [,TEMP|,PERM][,VOL=volser][,SHR]	
	Optional Operands for Diskette Devices Only: [,TEMP|,PERM][,VOL=volser]	
	Optional Operands for Tape Devices Only: [,mode|,ALT][,TEMP|,PERM][,VOL=volser]	
	Optional Operands for Device Types 2560 and 5424/5: [,H1|,H2][,TEMP|,PERM]	
	Optional Operands for Any Other Devices: [,TEMP|,PERM]	

Note: If you use two or more optional operands in an ASSGN statement, they must be entered in the sequence shown here.

SYSxxx

Represents the logical unit name. It can be one of the following:

SYSRDR
SYSIPT
SYSIN
SYSPCH

SYSLST
SYSOUT
SYSLNK
SYSLOG
SYSnnn

SYSCAT and SYSREC can only be assigned with the DEF command during IPL.

For compatibility reasons, an assignment for SYSREC entered from SYSRDR is ignored and processing continues; if entered from SYSLOG, the assignment is rejected.

SYSnnn represents all the other logical units in the system. For nnn, specify a decimal number from 000 to 254.

Restrictions: The type of device assignment is restricted under certain conditions:

1. If one of the system logical units SYSRDR, SYSIPT, SYSLST or SYSPCH is assigned to a disk device or diskette, the assignment must be permanent and follow the DLBL and EXTENT statements.

2. If SYSRDR and SYSIPT are to be assigned to the same disk device or diskette, SYSIN must instead be assigned and this assignment must be permanent.

3. SYSOUT is only valid for a tape unit and must be assigned permanently.

4. SYSLOG can only be assigned permanently.

5. If SYSIPT is assigned to a tape unit, it should be a single file and a single volume.

6. You may not assign SYSLOG to a 3278 Model 2A or 3279 Model 2C with a message area of 16 lines if IPL was done from a 3277, 3278 or 3279 with a message area of 20 lines.

7. SYSLOG cannot be assigned to a console printer (3284, 3286, 3287, 3288).

8. ASSGN SYSLOG,UA and ASSGN SYSLOG,IGN are not accepted.

9. If a system logical unit is assigned to a tape, disk, or diskette, the unit must be closed (using the CLOSE command) before it can be reassigned.

10. When SYSOUT is assigned to a magnetic tape device it must not be the permanent assignment of either SYSLST or SYSPCH. Before assigning a tape drive to a system output unit (SYSOUT, SYSLST, SYSPCH), all previous assignments of this tape drive to any system input units and to any programmer units (input or output) must be

permanently unassigned. The assignment of SYSOUT must always be permanent.

Also, before assigning a tape to a system input unit or any programmer unit, all previous assignments of this tape to any system output unit must be permanently unassigned.

11. A programmer logical unit cannot be assigned to SYSLST if SYSLST has been assigned to tape or disk before.

12. ASSGN SYSRDR and ASSGN SYSIPT are allowed within a cataloged procedure. SYSRDR assignments, and SYSIPT assignments in a procedure with a DATA=YES specification, become effective on returning to JC level 0. SYSIPT assignments in a procedure with DATA=NO specification become effective immediately.

13. In a system with 16 channels, ASSGN SYSxxx,FBA cannot be used to address unit BA on channel F. Use ASSGN SYSxxx,X'FBA' to distinguish the cuu specification from the device class specification FBA (for Fixed Block Architecture).

`cuu|X'cuu'`

Indicates the physical unit address to which the specified logical unit is to be assigned.

c = channel number
uu = unit number

The form X'cuu' must be used when the physical address of the specified device is FBA (that is, channel F, unit BA), to distinguish it from the device class FBA (for Fixed Block Architecture disk unit). Otherwise, the `X' '` can be omitted.

When you assign a 4248 printer using the physical unit address, you must check whether it is in the mode you require. Programs can use this printer in native mode only if it was added with the device type '4248' at IPL. 4248 printers in 3211 mode and added as 'PRT1' cannot be used in native mode.

`(address-list)`

You can specify a list of up to six device addresses in the form cuu, separated by commas and enclosed in parentheses. In this case the system searches only the addresses specified in the address list for a free unit, starting with the first specified device address. Once a free unit is found, it is assigned to SYSxxx for the job in which the assignment is made.

Note: If the address list contains the addresses of a 3480 Tape Subsystem and another tape unit, do **not** *specify a mode setting (***mode** *operand) in the ASSGN statement.*

For disks, if SHR is specified, the first unit in the list is assigned, even if previously assigned. (See Figure 3-3 on page 3-23.)

UA

Indicates that the logical unit is to be unassigned. Any operation attempted on an unassigned device cancels the job.

IGN

For certain American National Standard and DOS/VS COBOL application programs (for sequential input files), and for FORTRAN, the IGN option unassigns the specified logical unit, and ignores any subsequent logical IOCS command (OPEN, GET, etc.), issued for that unit. This allows you to disable a logical unit that is used in a program without removing the code for that unit. You can then execute the program as if the unit did not exist. This may be especially helpful when debugging a program.

For assembler language application programs, IGN indicates that the logical unit is to be ignored. With files processed by logical IOCS, the OPEN to the file is ignored, the DTF table is not initialized (for example, IOREG, extent limits), and the IGNORE indicator is set on in the DTF table. It is your responsibility to check this indicator and bypass any I/O commands (GET, PUT, etc.) for this file.

The IGN option is not valid for SYSRDR, SYSIPT, or SYSIN, nor for PL/I programs. The IGN option can be made temporary by specifying the TEMP option.

When using ASSGN IGN for associated files, all logical units of the associated files must be assigned IGN.

Additional information about IGN is in the " OPEN(R)" section of *VSE/Advanced Functions Application Programming: Macro Reference.* IGN restrictions for users of American National Standard and DOS/VS COBOL and of RPG II are given in the associated Program Product publications for the compiler being used.

SYSyyy

This may be any system or programmer logical unit, except SYSCAT (see SYSxxx, above). If this operand is specified, SYSxxx is assigned to the same device to which SYSyyy is currently assigned. This type of specification is particularly helpful because the specification of SYSxxx,SYSyyy is considerably shorter than the full specification.

Examples:

```
// ASSGN SYS001,3350,PERM,VOL=RAFT01,SHR
// ASSGN SYS003,SYS001
// ASSGN SYSLNK,SYS001
```

device-class

In this case the specification of READER, PRINTER, PUNCH, TAPE, DISK, CKD, FBA, or DISKETTE is allowed for the devices listed further below. Do not, however, use a generic assignment for a dummy device to be used as input or output device in a VSE/POWER-supported partition.

The system searches for the first unassigned unit within the specified device-class and assigns it to SYSxxx (see Figure 3-3 on page 3-23).

This type of specification might be used if the exact configuration of the installation is not known or not important. However, if a configuration consists of mixed device types of the same device-class, such as 3330's and 3340's, then either **device-type** or **address-list** should be used. If your installation includes disk drives with and without the fixed head feature, such as the 3348 Model 70F Data Module or the 3344 Direct Access Storage, do not use **device-class** or **device-type**. Instead, use **cuu** (or **address-list)** to specify the drives with the feature, so as to avoid job cancellation. For more information, see *IBM 3340 Fixed Head Feature User's Guide.*

If a configuration includes FBA and CKD disk devices, specification of DISK will assign any disk device (FBA or CKD) to the logical unit SYSxxx. The parameters CKD and FBA permit more detailed specification of the disk device to be selected.

The specific device types to which each device class applies are listed below.

READER
1442N1, 2501, 2520B1, 2540R, 2560, 2596, 3505, 3525RP, 5425

PRINTER
PRT1, 1403, 1403U, 1443, 3800, 3800B, 3800C, 3800BC, 3200, 4248

PUNCH
1442N1, 1442N2, 2520B1, 2520B2, 2520B3, 2540P, 2560, 2596, 3525P, 3525RP, 5425

TAPE
2400T9, 3410T9, 3420T9, 3430

DISK
FBA, 2311, 2314, 3330, 3330B, 3340, 3340R, 3350, 3375, 3380

Note: If you have both models of device type 3380 installed, and want a logical unit assigned to a 3380 Model E, specify DISK (not 3380), and specify the serial number of the volume you want to use in the VOL operand.

CKD
2311, 2314, 3330, 3330B, 3340, 3340R, 3350, 3375, 3380

FBA
(Fixed-block architecture disk devices cannot be assigned by device-type.)

DISKETTE
3540

`device-type`

This can be any device type code shown in Figure 2-2 on page 2-22. Do not, however, use this specification for a dummy device to be used as input or output device in a VSE/POWER supported partition. The system searches for the first free unit of the specified device type. When a free unit is found, it is assigned to SYSxxx (See Figure 3-3 on page 3-23).

Use this specification if you are interested only in the specific type of device, and not in the physical unit. For disks, if SHR is specified, the first unit of the specified device-type is assigned, even if previously assigned. If your installation includes disk drives with and without the fixed head feature, such as the 3348 Model 70F Data Module, the 3344 Direct Access Storage or the 3375 Direct Access Storage, do not use **device-class** or **device-type**. Instead, use **cuu** (or **address-list**) to specify the drives with the feature, so as to avoid job cancellation. For more information, see *IBM 3340 Fixed Head Feature User's Guide.*

If you have both models of the 3380 on your system, you cannot use the type '3380' in the ASSGN statement. Use the class 'DISK', and specify the required volume in the VOL operand.

When you assign a 4248 printer using the device type 4248, note that only those units added with the device code '4248' at IPL are available. A 4248 added as 'PRT1' will not be selected.

For a 3800 printing subsystem, you can use assignment by device codes as follows:

Specified code	is valid for			
	3800	3800B	3800C	3800BC
3800	X	X	X	X*,**
3800B		X		X*
3800C			X	X**
3800BC				X

* The job cannot use the additional character generation storage feature.

** The job cannot use the Burster-Trimmer-Stacker feature.

Specification of the device class PRINTER may select a 3800 from a list of printers; however, the existence of the two optional hardware features (the Burst-Trimmer-Stacker and additional character generation storage) cannot be assumed.

The three types of multiple device specification, device-list, device-class and device-type, assign the first available unit which matches the specification. However, the search order is different in each case:

device-list searches the units in the order specified in the list;

device-class searches the units of the specified class in ascending order of device-type code;

device-type searches the units of the specified type in ascending order of physical unit address.

Figure 3-3 shows an example of how the system scans the attached physical units with three different types of tape specification in the ASSGN statement/command.

Device		Search Order		
Physical Unit	Device Type	TAPE	2400T9	(280,281, 183,382)
181	2400T9	1	1	
182	2400T9	2	2	
183	2400T9	3	3	3
280	3410T9	8		1
281	3410T9	9		2
282	3420T9	10		
283	3420T9	11		
381	2400T9	4	4	
382	2400T9	5	5	4

Figure 3-3. Device Search Order

mode

Specifies mode settings for magnetic tapes (see Figure 3-4 on page 3-24). If the tape on the specified unit is at load point, the new mode setting becomes effective at once. Otherwise, it becomes effective the next time load point is reached. If **mode** is not specified at IPL time, the system assumes:

90 for 7-track tapes
C0 for 9-track tapes (2400, 3410 series)
D0 for 9-track tapes (3420 series)
D0 for 9-track tapes (3430 series)
00 for the IBM 3480 Tape Subsystem
60 for the 9-track 8809 and 9347 Magnetic Tape Units

For 800 bpi single-density 9-track tapes, a specification of C8 reduces the time required to OPEN an output file.

With dual-density tape units, the mode setting of D0 (6250 bpi) does not take effect for input tapes that were written at 1600 bpi (mode setting C0).

The standard mode is set during IPL. If the mode setting (different from, or the same as the standard mode) is specified in a temporary ASSGN statement, it becomes the current mode setting. This mode stays in effect until a subsequent assignment with a new mode or until EOJ. When the current job ends, the standard mode is restored, provided the unit was not

unassigned during the job. The mode specification in a permanent ASSGN becomes the standard mode. If **mode** is not specified for a new job, the mode is the same as the standard mode or the mode specified in the last permanent assignment.

Density (bpi)	Parity	Convert Feature	Translate	Mode
200	odd	on	off	10
200	odd	off	off	30
200	odd	off	on	38
200	even	off	off	20
200	even	off	on	28
556	odd	on	off	50
556	odd	off	off	70
556	odd	off	on	78
556	even	off	off	60
556	even	off	on	68
800	odd	on	off	90
800	odd	off	off	B0
800	odd	off	on	B8
800	even	off	off	A0
800	even	off	on	A8
800	Single-density 9-track tapes			C8
800	Dual-density 9-track tapes			C8
1600	Single or dual-density 9-track tapes			C0
6250	Single/dual density, 9-track			D0
1600	Streaming: high speed and long gap			90
(for 8809 and 9347)	Streaming: high speed and short gap (Not applicable for 9347)			30
	Start-Stop: low speed and long gap			50
	Start-Stop: low speed and short gap			60
IBM 3480 Tape Subsystem: - Buffered write mode - Unbuffered write mode				 00 20

Figure 3-4. Mode Settings for Tapes

ALT

Indicates an alternate magnetic tape unit that is used when the capacity of the original assignment is reached. This operand can only be specified for programs using logical IOCS. The specifications for the alternate unit are the same as those of the original unit. The characteristics of the alternate unit must be the same as those of the original unit. The original assignment and an alternate assignment must both be permanent or both be temporary assignments. Multiple alternates can be assigned to one symbolic unit.

The system does not adjust the tape mode of the alternate unit to that of the original unit. Therefore, if tape modes are different, it is advisable to first assign the units with equal tape modes and then to reassign with the ALT operand.

Using multivolume tape files without specifying ALT mode can cause performance degradation, because the first tape has to be rewound and unloaded before the next tape can be mounted.

If the original unit is reassigned, the alternate unit must also be reassigned. The ALT operand is invalid for SYSRDR, SYSIPT, SYSIN, SYSLNK, and SYSLOG.

H1

Indicates that input hopper 1 will be used for input on the 2560, 5424, or 5425. If neither H1 nor H2 is specified, H1 is assumed.

H2

Indicates that input hopper 2 will be used for input on the 2560, 5424, or 5425. Note that hopper specifications are significant only for device independent files associated with the logical units SYSIPT, SYSRDR, SYSIN, and SYSPCH. In all other cases they are invalid.

If both hoppers are used, they must be assigned to the same partition.

PERM|TEMP

Indicates whether the assignment should be permanent (PERM) or temporary (TEMP). It is thus possible to override the // specification or omission.

A permanent assignment overrides the current assignment and deletes the stored permanent and all alternate assignments.

This operand must be entered at the position shown in the syntax description.

VOL=no.

Specifies the volume serial number of the device required. This option may be specified only for tapes, disks, and diskettes.

If VOL is specified, the system searches for the first unit in the requested sequence and, if the unit is ready (for a tape, if it is at load point and not already assigned) checks the volume label to see if the required volume is mounted. If not, the next unit is checked, and so on until the proper volume serial number is found or until the end of the specified sequence is reached. The requested volume must be mounted on the unit specified in the message 1T50A MOUNT volser ON X'cuu'.

Note: If, while reading the volume label, the job is canceled (for example, because of I/O errors), the device has to be reset to its initial status (using DVCDN and DVCUP commands) before the unit can be reassigned with a generic assignment.

If a volume serial number specified for a disk device does not match the actual volume serial number, the system notifies the operator and allows him to correct the assignment statement.

Note: In a mixed device configuration, specification of TAPE,VOL or DISK,VOL may cause the system to issue a request for a volume to be mounted on the first device that becomes available. Thus, the system may request a 9-track tape to be mounted on a device that can only accommodate 7-track tapes. Likewise, a request may be issued for a 2316 disk pack to be mounted on a 3330 or 3340. Therefore the parameters **device-type** *or* **address-list** *should be used in a mixed device environment.*

SHR

This option can be specified only for disk devices and is meaningful only in combination with **address-list, device-class**, and **device-type** (see the descriptions of these operands.) It means that the unit can be assigned to a disk device which is already assigned. If the option is not specified, the system assigns the unit to a disk device not yet assigned. Therefore, unless a private device is required, it is recommended to use the SHR operand in combination with generic assignments.

BANDID (Mount or Query 4248 Print Band)

This command applies only to IBM 4248 printers. It allows you to find out which print band is mounted, or to specify which band is to be mounted.

Operation	Operands	Type
BANDID	cuu[,band-id][,FOLD][,NOCHK]	AR

cuu
: Specifies the channel and unit number of the printer for which you issue the command.

band-id
: Specifies the identifier of the print band that is to be mounted.

  If you omit the operand, then you get a message indicating the identifier of the currently mounted band.

FOLD
: Causes lowercase characters to be printed as uppercase characters.

NOCHK
: Causes a data check to be suppressed if it resulted from a mismatch between a print character and the band-image buffer.

BATCH (Start or Continue Processing)

The BATCH command activates or continues processing in one of the foreground partitions or continues processing in the background partition. The function of the BATCH command is exactly the same as that of the START command. If the specified partition is available, job control reads the operator's next command from SYSLOG. When the operator desires to give control to another command input device, he makes an assignment to SYSRDR or SYSIN, and presses the END or ENTER key.

If the specified partition has been made inactive by an UNBATCH command, it is made active. If the partition was temporarily halted by a STOP command, it is restarted. If the partition is in operation, it continues, and message

```
1P1nD AREA NOT AVAILABLE
```

is issued to the operator. In either instance, attention routine communication with the operator terminates following the BATCH command.

Operation	Operands	Type
BATCH	[BG|Fn]	AR

BG
: Indicates that the background partition is to be reactivated.

Fn
: Indicates that the specified foreground partition is to be activated or restarted after having been stopped by a STOP command.

If the operand is omitted, BG is assumed.

CANCEL (Cancel Job or I/O Request)

The CANCEL command, when used as a job control command, cancels the execution of the current job in the partition in which the command is given.

When issued as an attention routine command, it may be used for two purposes:

- To cancel an I/O request on a device for which operator intervention was requested.
- To cancel the execution of the current job in the specified partition and, optionally, to override the dump options existing for that partition.

The AR CANCEL command is accepted only when the attention routine is available. If the attention routine is not available when you want to enter the AR command, enter the RC command (Request Communication), and enter the CANCEL command in response to the message 1I40I READY.

Operation	Operands	Type
CANCEL	none	JCC
CANCEL	cuu	AR
CANCEL	{BG|Fn}[,DUMP|,PARTDUMP|,NODUMP] [,SYSDUMP|,NOSYSDUMP][,FORCE]	AR

cuu
: Indicates that the I/O request for the specified device is to be canceled.

BG
: Indicates that the background job is to be canceled.

Fn
: Indicates that the specified foreground job is to be canceled.

DUMP
: Causes a dump of the registers, of the supervisor, of the partition, the used part of the system GETVIS area, and of the SVA phase in error (if the error occurred in the SVA).

PARTDUMP
: Causes a dump of the registers, of supervisor control blocks, of the partition, of areas acquired through GETVIS in the partition, and of the SVA phase in error (if the error occurred in the SVA).

NODUMP
: Suppresses the DUMP option.

NOSYSDUMP
: Indicates that dumps are to be written on SYSLST. The form "NOSYSDMP" is accepted for compatibility reasons.

SYSDUMP
: Indicates that dumps are to be written to the dump sublibrary which is defined for the appropriate partition. If no LIBDEF DUMP statement is in effect for the partition in question, or if the defined sublibrary is full, the system assumes the option NOSYSDUMP. The form "SYSDMP" is accepted for compatibility reasons.

FORCE
: Causes the Cancel command to be carried out immediately, even if a critical system function has requested a delay. Any action specified in an ON $CANCEL statement is **not** carried out.

  Note: Use the FORCE operand **with caution**, *and only when a Cancel command without FORCE has failed to terminate the job.*

  The use of this operand can cause critical system functions to be interrupted. This, in turn, can lead to inconsistencies in the system (for example, library directories not updated).

If the CANCEL command is given for a partition in which the subsystem VSE/POWER or VSE/OCCF is active, a message is issued to the operator asking him to verify whether he really wants to cancel that partition.

If a JOB statement was specified for the job to be canceled, the remaining statements and data will be skipped up to /& or to the label specified in an ON $CANCEL GOTO statement for the job. (If FORCE is specified, the ON $CANCEL statement is not carried out.) If the JOB statement was omitted, the records will not be skipped, except during the execution of a procedure. In this case, all records up to /+ or the next JOB statement are skipped.

CLOSE (Close Output Logical Unit)

The CLOSE **command** is used to close either a system or programmer logical unit assigned to a tape, or a system logical unit assigned to a disk or diskette.

The CLOSE **statement** is used to close either a system or programmer logical unit assigned to tape. It applies only to temporarily assigned logical units.

The logical unit can optionally be reassigned to another device, unassigned, or, in the case of a magnetic tape file, switched to an alternate unit. When SYSxxx is a system logical unit (SYSLST, SYSPCH, etc.), one of the optional parameters **must** be specified. When closing a programmer logical unit (SYS000-SYS254), no optional parameter need be specified. When none is specified, the programmer logical unit is closed and the assignment remains unchanged.

Closing a magnetic tape unit causes the system to write a tapemark, an EOV trailer record, and two tapemarks, and to rewind and unload the tape. The trailer record contains no block count, and later access by logical IOCS may result in a 4131D message, which can be ignored.

Operation	Operands	Type
[//] CLOSE	SYSxxx [,cuu[,mode]|,UA|,IGN|,ALT|,SYSyyy |,device-class|,device-type]	JCC,JCS

SYSxxx

For the CLOSE command only: For disk or diskette: SYSIN, SYSRDR, SYSIPT, SYSPCH, or SYSLST.

For both the statement and the command: For magnetic tape: SYSPCH, SYSLST, SYSOUT, or SYS000-SYS254.

cuu

Specifies that, after the logical unit is closed, it will be assigned to the specified channel and unit. **c** is the channel number, **uu** is the unit number, in hexadecimal. In the case of a system logical unit, the new unit will be opened if it is either a disk, diskette, or a magnetic tape at load point.

mode

Device specification for mode settings on 7-track and 9-track tape. The specifications are shown under " ASSGN." If **mode** is not specified, the mode settings remain unchanged. The LISTIO command may be used to determine the current mode settings for all magnetic tape units.

UA

Specifies that the logical unit is to be (permanently) unassigned after the file has been closed.

IGN

Specifies that the logical unit is to be (permanently) unassigned after the associated file has been closed. Any subsequent references to the unit will be ignored until a new ASSGN is given for the unit, or IPL is performed. This operand is invalid for SYSRDR, SYSIPT, or SYSIN.

ALT

Specifies that the logical unit is to be closed and an alternate unit is to be opened and used. This operand is valid only for system output logical units (SYSPCH, SYSLST, or SYSOUT) currently assigned to a magnetic tape unit.

SYSyyy

Specifies that, after SYSxxx is closed, it will be assigned to the physical device to which SYSyyy is currently assigned (and to which it remains assigned). If SYSxxx is a system logical unit, it will be opened if the target device is a disk, diskette, or magnetic tape at load point, and if SYSxxx is not already assigned.

device-class

Indicates that after the logical unit is closed, it will be assigned to the first available unit within the specified device class. The device classes and the device types to which they apply are listed below:

READER
1442N1, 2501, 2520B1, 2540R, 2560, 2596, 3505, 3525RP, 5425

PRINTER
PRT1, 1403, 1403U, 1443, 3800, 3800B, 3800C, 3800BC, 3200, 4248

PUNCH
1442N1, 1442N2, 2520B1, 2520B2, 2520B3, 2540P, 2560, 2596, 3525P, 3525RP, 5425

TAPE
2400T7, 2400T9, 3410T7, 3410T9, 3420T7, 3420T9, 3430

DISK
FBA, 2311, 2314, 3330, 3330B, 3340, 3340R, 3350, 3375, 3380

CKD
2311, 2314, 3330, 3330B, 3340, 3340R, 3350, 3375, 3380

FBA
(Fixed-block architecture disk devices cannot be closed by device-type.)

DISKETTE
3540

`device-type`

Indicates that after the logical unit is closed, it will be assigned to the first free unit of the specified device type. The device type codes which you can specify are shown in Figure 2-2 on page 2-22.

DATE (Override System Date)

The DATE statement places the specified date temporarily in the communication region. This date overrides the date given in the SET command, either during or after IPL. The date can be used by the program for identifying printed output. The date specified in the DATE statement is reset at the end of the job to the date corresponding to the last SET command.

Operation	Operands	Type
// DATE	{mm/dd/yy|dd/mm/yy}	JCS

```
mm = month (01-12)
dd = day   (01-31)
yy = year  (00-99)
```

When the DATE statement is used, it applies only to the current job being executed, except for disk and tape output file labels for which the date from the SET command is used. Job control does not check the operand except for a length of eight characters. If no DATE statement is used, job control supplies the date given in the last SET command. If a job or job step executes past midnight, the date given in the DATE statement is not incremented.

DLBL (Disk Label Information)

The DLBL statement (disk label information) contains file label information for disk or diskette label checking and creation.

Operation	Operands	Type
// DLBL	filename,['file-ID'],[date],[codes][,DSF] [,BLKSIZE=n][,BUFSP=n][,CAT=filename] [,CISIZE=n][,DISP=disposition][,RECORDS=n] [,RECSIZE=n]	JCS

Continuation lines are accepted for the DLBL statement.

`filename`

This can be from one to seven alphameric characters, the first of which must be alphabetic, @, # or $. This unique filename is identical to the symbolic name of the program DTF that identifies the file.

For VSE/VSAM, filename is identical to (1) the dname of the FILE (dname) parameter and (2) the DDNAME = filename parameter of the access method control block (ACB) in the processing program that identifies the file. If the DDNAME parameter is omitted, the filename must be contained in the symbolic name (label) field of the ACB.

`'file-ID'`

This is the unique name associated with the file on the volume. This can be from one to 44 characters, contained within quotes, including file-ID and, if used, generation number and version number of generation. If fewer than 44 characters are used, the field is left-justified and padded with blanks. If this operand is omitted, **filename** is used. The diskette uses a maximum of eight characters in file-ID.

For VSE/VSAM, **file-ID** must be specified when an existing (input) file is being processed. The file-ID is identical to the name of the file, specified in the DEFINE command and listed in the VSE/VSAM catalog. For VSE/VSAM, the file-ID must be coded according to the following rules:

- One to 44 characters long, enclosed in quotes (');
- Characters must be alphameric (A-Z, 0-9, @, $, or #) or hyphen (-) or plus zero (+0);
- After each group of eight or fewer characters, a period (.) must be inserted;
- No embedded blanks are allowed;
- The first character of the file-ID and the first character following a period must be alphabetic (A-Z) or @, $ or #.

`date`

This can be from one to eight characters indicating either the retention period of the file in days in the format dddd (0-9999), or the absolute expiration date of the file as a Julian date in the format 19yy/ddd or 20yy/ddd. (yy = last two digits of year, ddd = day of the year.) The format yy/ddd is also accepted. For example, 85/032 is interpreted as 1985/032. If 00/ddd is specified, ddd is treated as a retention period in days.

The format 20yy/ddd is not accepted for diskette files.

If this operand is omitted, a 7-day retention period (based on the date entered in the SET command) is assumed. If this operand is present for an input file, it is ignored.

For VSE/VSAM, this parameter overrides the expiration date specified in the DEFINE command. However, VSE/VSAM files or data spaces can only be deleted through the DELETE command even though the expiration date has been reached.

`codes`

This is a two to four character field indicating the type of file label, as follows:

`SD`
for sequential disk or for DTFPH with MOUNTED=SINGLE

`DA`
for direct access or for DTFPH with MOUNTED=ALL

`DU`
for diskette

`ISC`
for indexed sequential using load create

`ISE`
for indexed sequential using load extension, add, or retrieve

`VSAM`
for all Virtual Storage Access Method files

If this operand is omitted, SD is assumed.

`DSF`

This operand indicates that a data secured file is to be created or processed. At OPEN time, if a data-secured file is accessed, a warning message is issued to the operator, who then decides whether the file may be accessed.

For the diskette and for VSE/VSAM this operand is ignored by OPEN. All VSE/VSAM files are data secured. The DSF operand is not required

for an **input** file, and it does not invoke the support if the file was not originally created as a data secured file.

BLKSIZE=n

This operand permits specification of a block size different from that given in the DTFSD macro for sequential disk files. This allows the user to utilize a more effective blocking factor. The parameter is ignored for all DTF types except DTFSD. It is not valid for VSE/VSAM files, or files on FBA devices. The value specified for n must not exceed 65,536. If the file contains blocked fixed-length records, **n** must be:

- For input files: a multiple of the RECSIZE value;
- For output Files: 8 + a multiple of the RECSIZE value. The additional 8 bytes allow for a count field for system use.

Note that this parameter will be accepted by Job Control, but the job will later be canceled if the value specified is not a multiple of the RECSIZE value. The job will also be canceled if the BLKSIZE operand in the DTFSD macro is specified for TYPEFLE = WORK. For further details on the DTFSD macro, see *VSE/Advanced Functions Application Programming: Macro User's Guide.*

BUFSP=n

If a VSE/VSAM file is to be processed, this operand specifies the number of bytes of virtual storage (0-999999) to be allocated as buffer space for this file. It overrides the values specified for BUFSP in the ACB macro and for BUFFERSPACE in the DEFINE command (only if the value of the DLBL BUFSP operand is greater than the value of the BUFFERSPACE parameter).

CAT=filename

This operand is valid in a DLBL statement for a VSE/VSAM file only. It specifies the filename (1 to 7 alphameric characters) of the DLBL statement for the catalog owning this VSE/VSAM file. The system searches only this catalog for the file-ID when the VSE/VSAM file is to be opened. Specify this operand only if you want to override the system's assumption that the job catalog or, if there is no job catalog, that the master catalog owns the file.

In a system **with** a job catalog specify nothing for the job catalog, a private name for a private user catalog, or IJSYSCT for the master catalog.

In a system **without** a job catalog specify nothing for the master catalog, or a private name for a private user catalog.

The only Access Method Services commands that use the CAT operand to specify a private user catalog are the PRINT, REPRO, VERIFY, and DELETE ERASE commands.

CISIZE=n

This operand permits specification of a control interval size for SAM files on FBA devices or in VSAM space to improve storage utilization. The

size overrides that specified (or defaulted) in the respective DTF macro. The specified size must be a number from 512 to 32,768 and a multiple of the FBA block size; if it is greater than 8K, it must be a multiple of 2K.

This operand is valid only for DLBL statements with the code SD.

DISP=disposition

This operand is valid only in a DLBL statement for a VSE/VSAM file. It permits specification of the data set disposition. The three positional keywords tell VSAM how the file is to be:

- Opened (keyword1);
- Closed after normal termination of the job step (keyword2);
- Closed after abnormal termination or cancellation of the job step (keyword3).

disposition can be specified in one of the following formats:

```
keyword1
(keyword1,keyword2)
(keyword1,keyword2,keyword3)
(keyword1,,keyword3)
(,keyword2)
(,keyword2,keyword3)
(,,keyword3)
```

where:

keyword1 may be NEW or OLD;
keyword2 may be DELETE, KEEP or DATE;
keyword3 may be DELETE or KEEP.

For the meaning of these keywords, and the default values, see *VSE/VSAM Programmer's Reference.*

RECORDS=n

This operand is only valid for VSE/VSAM space management for SAM feature files. It permits specification of the number of records for the primary and secondary data set allocation. The operand can be specified in one of two formats:

```
RECORDS=n
RECORDS=(n,n1)
```

where **n** indicates the number of records for the primary data set allocation, and **n1** the number of records for the secondary data set allocation. **n** must not be zero.

The RECORDS and RECSIZE operands must either both be specified or both be omitted. For further details, see *VSE/VSAM Programmer's Reference.*

RECSIZE=n

This operand is valid only for VSE/VSAM space management for SAM feature files. It permits specification of the average record length of the file. The value specified for **n** must not be zero. The RECSIZE and RECORDS operands must either both be specified or both be omitted. For further details see *VSE/VSAM Space Management.*

DSPLY (Display Virtual Storage)

The DSPLY command allows the operator to display 16 bytes of virtual storage, starting at the specified hexadecimal address, on the device assigned to SYSLOG. Two characters (0-9, A-F) appear on SYSLOG for each byte of information; these characters represent the hexadecimal equivalent of the current information in virtual storage.

Operation	Operands	Type
DSPLY	[space-id,]address	AR

`space-id`
Indicates in which address space the specified address is to be displayed. Valid specifications are:

R, 1..3 in 370 mode;
1 in ECPS:VSE and VM mode.

To display virtual storage in a **shared** area (370 mode only) specify the space-id of any **existing** virtual address space.

`address`
Specifies the six-digit hexadecimal address, with leading zeros if necessary, at which the storage display is to start. The highest address that can be specified is 16MB minus 15 (FFFFF0).

If the specified address is within an invalid address area, the command is ignored and a corresponding information message is issued.

If the 16 bytes to be displayed cross the boundary from a valid to an invalid address area, only the bytes in the valid address area are displayed, and a corresponding information message is issued.

DUMP (Dump Storage Areas)

The DUMP command allows the operator to dump specified areas of virtual storage on a printer or a tape device.

Operation	Operands	Type
DUMP	{SUP|BG|Fn|SVA|[space-id,]addr-addr|BUFFER}, cuu	AR

The first operand specifies which areas of storage are to be dumped, as follows:

SUP
: The control registers and the supervisor areas.

BG, Fn
: The PSW, general and floating-point registers from the partition save area, and the active real or virtual partition specified.

SVA
: The SVA, including the system GETVIS area and the VIO V-pool area.

space-id
: Part of the address space specified by this operand. Valid specifications are:

  R, 1..3 for 370 mode;
  1 for ECPS:VSE and VM mode.

  The default value for all modes is 1.

addr-addr
: The virtual storage between the specified addresses in the address space indicated by the space-id operand. If any active real or virtual partition, or a part of such a partition, lies between the specified addresses, its PSW and associated registers are dumped.

BUFFER
: The contents of the SDAID buffer. This operand is accepted only if the dump is directed to a tape device.

cuu
: Specifies the device on which the output is to be written. It can be a printer or tape device, unless BUFFER was specified in the first operand. In this case, only a tape unit address is accepted. Tape output is written without repositioning the tape to allow for several dumps per tape. The output is written after the preceding DUMP command output to allow for several dumps per file. For information on dump handling, refer to *VSE/Advanced Functions Service Aids.*

If cuu designates a printer, the printer should not, at the time of the dump, be used by the partition to which it is assigned, since this could result in interspersed partition and dump output.

Note: *When cuu is assigned to a 3800 Printing Subsystem, you must ensure that the 3800 settings are appropriate for the expected output.*

DVCDN (Device Down)

The DVCDN command informs the system that a device is no longer available for system operation. It is used when a device is to be serviced or becomes inoperative. This command may be given in any partition, and the specified device is made unavailable for all partitions.

Operation	Operands	Type
DVCDN	cuu	JCC

cuu

c is the the channel number and **uu** the unit number, in hexadecimal, of the device to be made unavailable.

Note: The system does not accept the cuu of a device on which SYSRES, SYSREC, SYSCAT, the internally assigned system logical unit SYSDMP or the page data set resides.

If a permanent or temporary assignment exists for the device specified in the command, any logical units assigned to it are unassigned.

If a sublibrary on the specified device is part of a sublibrary chain (specified in a LIBDEF statement), the DVCDN command is not accepted.

The DVCDN command does not close files associated with logical units, and after the DVCDN command has been issued, files on a disk or diskette unit cannot be closed or reassigned to another disk or diskette unit. Therefore, if the unit is a disk or diskette unit, first attempt to close any files associated with logical units currently assigned to the device, using the CLOSE command.

If an alternate assignment exists for the device, it is removed when the DVCDN command is issued. A DVCUP command must be issued before the device can be used again.

DVCUP (Device Up)

The DVCUP command informs the system that a device which was inoperative is now available again for system operation. As all assignments for this device were removed by the preceding DVCDN command, the device must be reassigned by an ASSGN statement or command.

Operation	Operands	Type
DVCUP	cuu	JCC

cuu

c is the channel number and **uu** the unit number, in hexadecimal, of the device to be made available.

END or ENTER (End of Input)

The END or ENTER command must be issued whenever the operator has finished typing an input line. It causes the communication routine to return control to the mainline job. END applies to CPU models with a printer keyboard console. ENTER applies to CPU models with a display console.

JCC and AR Format

Press the END or ENTER key.

EXEC (Execute Program or Procedure)

The EXEC command or statement indicates either:

- The end of control information for a job step and the beginning of execution of a program.
- That a cataloged procedure is to be retrieved from a sublibrary by job control.

Note: If the access control function is active, and a program is to be executed while a protected sublibrary is part of the LIBDEF PHASE,SEARCH chain in the partition, this sublibrary must be opened so that authorization checking can be done. This means that the labels for this library must be available in the label information area when the EXEC statement is issued.

Operation	Operands	Type
[//] EXEC	[[PGM=]progname][,REAL][,SIZE=size][,GO] [,PARM='value']	JCS,JCC
[//] EXEC	PROC=procname [,parname[=[value]]][...]	JCC,JCS
[//] EXEC	PROC=procname[,OV]	JCS,JCC

The operands must be entered in the specified order.

The **statement** can be issued from SYSLOG or from SYSRDR. Control returns to the unit from which the statement was issued. The **command** with a program name can be issued from SYSLOG or SYSRDR. With a procedure name, the command can be issued only from SYSLOG. Control returns to the unit from which the command was issued. For the rules governing the return of control after the execution of a procedure, see Figure 3-5 on page 3-50.

Continuation lines are accepted for the EXEC statement.

`PGM=progname|progname`
: Represents the name of the program to be executed. The program name corresponds to the first or only phase of the program in the library. If the program to be executed has just been processed by the linkage editor, the program name is omitted and the PGM keyword cannot be used.

`REAL`
: Indicates that the program will be executed in real mode. If REAL is not specified, the program is always executed in virtual mode.

  This operand is ignored if the system is running in VM Mode.

`SIZE=size`
: The SIZE operand can be specified in combination with REAL or without REAL.

1. If specified **with** REAL, it gives the size of that part of the partition's processor storage that will be needed by the program. The remaining part of the allocated processor storage can be used as additional storage (GETVIS area) for other modules or data required by the program in that partition. The program obtains this additional storage by issuing GETVIS macros with the required amount of storage as an operand; it releases the storage by issuing FREEVIS macros.

   If the SIZE operand is omitted and REAL is specified, the entire processor storage of the partition is reserved for the program.

2. If used **without** REAL, it specifies the size of that part of the virtual partition that will be directly available to the program. The remainder of the partition may be used as additional storage (GETVIS area) for other modules or data required by the program in that partition. The system always allocates a minimum partition GETVIS area of 48K bytes.

   Certain programs have partition GETVIS requirements beyond 48K bytes, such as VSE/VSAM programs, ISAM programs using the ISAM Interface Program (IIP), programs using RPS support, or programs using sequentially organized disk files. Through the SIZE operand, you can temporarily change the size of the partition GETVIS area. The new GETVIS area size is the total partition size minus the value specified in the SIZE operand. The SIZE specification is accepted only if it yields a GETVIS area larger than 48K.

   If the SIZE (and the REAL) operand is omitted, either the whole virtual partition minus the minimum GETVIS area, or the default GETVIS area as specified by a preceding SIZE command, is reserved for the job initiated with EXEC.

The SIZE operand can be specified in the following formats:

SIZE={nK|mM}
SIZE=AUTO
SIZE=(AUTO,{nK|mM})
SIZE=phasename
SIZE=(phasename,{nK|mM})

where **n** or **m** must be greater than zero and **n** must be a multiple of 4 (if not, the system rounds the value up to the nearest 4K boundary). **nK** or **mM** must not exceed the size of the partition (as defined by ALLOC) minus the minimum partition GETVIS area of 48K bytes.

Note: If you specify a SIZE which is less than the storage which the program in fact requires, the GETVIS area may be overlaid, with unpredictable results.

AUTO indicates that the program size, as calculated by the system from information in the sublibrary directory, is to be taken as the value for SIZE. Use caution in specifying SIZE=AUTO in the following case: When phases belonging to the same program (multi-phase) or same application (for example, payroll) use generic phase names (identical first four

characters), the size of the phase with the highest ending address found with that generic name will be used.

Note: Do not specify SIZE=AUTO for programs that dynamically allocate storage during execution (such as linkage editor, librarian program, and compilers).

AUTO,{nK|mM} indicates that job control must take the program size plus nK or mM bytes as the value for SIZE.

phasename indicates that the length of the specified phase, increased by its relative load address in the partition, is to be taken as the value for SIZE, regardless of other phases with the same first four characters in their names.

phasename,{nK|mM} indicates that the length of the specified phase, increased by its relative load address in the partition, **plus nK or mM bytes**, is to be taken as the value for SIZE. If this value is not a multiple of two, it is rounded up.

Note: If this operand is not AUTO, nK or mM, the system assumes that it is a phase name.

GO

Specifies, for a language translator step, that the program is to be link-edited and executed automatically after it has been compiled. Only the source program data and any additional input data for the execution step are required after the language translator step. If a serious error is encountered in either the language translator step or the link-edit step, the input stream is flushed to the end-of-job (/&) statement.

Note: This type of execution can be used only for single-phase programs.

PARM='value'

Specifies information which is to be passed to the program at execution. **value** can be up to 100 characters in length, enclosed in quotes. (The enclosing quotes are not passed to the program.) An quote within **value** must be coded as two single quotes.

The information given by value is stored into the partition GETVIS area. Therefore, if EXEC REAL is in effect, the SIZE operand must be specified to set up the partition GETVIS area.

For information on how to access the PARM value from an assembler program, see *VSE/Advanced Functions System Management Guide.*

PROC=procname

Represents the name of the procedure to be retrieved from a sublibrary.

If the procedure name begins with $$, the system substitutes a partition-related character for the second $. The character that is substituted is related to the partition in which the procedure is invoked, that is,

```
0 for the BG partition
B for the FB partition
A for the FA partition
9 for the F9 partition
. . .
1 for the F1 partition.
```

The procedure corresponding to this name is then retrieved for execution. There are three methods of addressing symbolic parameters in the EXEC PROC statement or command:

parname1 = value1

parname2

parname3 = &parname3

`parname1`

Specifies the name of a symbolic parameter which is to be substituted in the specified procedure. It must consist of 1 to 7 alphanumeric (including national) characters, and the first character must be alphabetic. The & at the beginning of the symbolic parameter as coded in the called procedure must not be coded in the EXEC statement. For example, if you want to substitute the symbolic parameter &PARM1 with the value PAYROLL, you must code PARM1 = PAYROLL.

`value1`

Specifies the actual value which is to be inserted in the specified procedure in place of the specified symbolic parameter. It must be a string of up to 50 characters. If the string is alphanumeric, no enclosing quotes are necessary. If it contains national or special characters, it must be enclosed in quotes, which will not be passed to the procedure. No quotes are allowed in the string itself.

If value1 is a null string (PARM1 = '' or PARM1 = ,) the specified parameter is ignored in the called procedure.

`parname2`

Is the name of a symbolic parameter which is to be passed to a lower-level procedure and back. The value assigned to the parameter on the higher JC level at the time of the call will be valid for the lower-level (called) procedure, and if it is altered in the lower-level procedure by SETPARM, the new value will also be valid for the higher JC level on return.

`parname3`

Is the name of a symbolic parameter which is to be passed to a lower-level procedure. The value assigned to the parameter on the higher (calling) JC level at the time of the call is valid for the lower level (called) procedure, but can be altered there with no effect on the corresponding parameter on the higher level. The symbolic parameter name after the equals sign must be coded *with* the ampersand (&).

OV

Indicates that overriding statements follow EXEC. This operand must not be used when the called procedure contains symbolic parameters or calls a nested procedure, or if it contains IF, ON, GOTO or SETPARM statements.

When used in this manner, **OV** is a positional operand, and must be placed immediately after the procedure name in the EXEC statement. Figure 3-5 illustrates the rules for entering overriding statements.

For more details on the procedure overriding feature, the nested procedures and symbolic parameters, refer to *VSE/Advanced Functions System Management Guide*.

If EXEC is a	entered from	then	control returns at End-of-Procedure to	Overriding stmts. must be entered from
statement	SYSRDR		SYSRDR	SYSRDR
statement	SYSLOG		SYSLOG	SYSRDR
command	SYSLOG		SYSLOG	SYSLOG

Figure 3-5. Procedures: Return of Control and Input of Overrides

EXTENT (Disk or Diskette Extent Information)

The EXTENT statement defines each area, or extent, of a disk or diskette file. One or more EXTENT statements must directly follow each DLBL statement, except for VSAM files and for single-volume input files for sequential disk on a disk or diskette, provided the DEVADDR parameter has been specified in the DTF table.

Note: The EXTENT statements should be checked carefully because an invalid field causes the default options or the values entered by the previous EXTENT statement to be overwritten by the valid entries of the flagged statement.

System files on disk (SYSIPT, SYSRDR, SYSLST, SYSPCH) and SYSLNK (always on disk) must have only **one** extent.

Multiple EXTENT statements are valid for system files on diskette. Valid parameters are symbolic unit, serial number, and type. The other parameters will be ignored.

Operation	Operands	Type
// EXTENT	[logical-unit], [serial-number],[type], [sequence-number], [relative-track|block], [number-of-tracks|blocks], [split-cylinder-track]	JCS

No comma need be coded for an EXTENT statement without any operands.

`logical unit`

A six-character field indicating the logical unit (SYSxxx) of the volume for which this extent is effective. If this operand is omitted, the logical unit of the preceding EXTENT, if any, is used. If this operand is omitted on the first or only EXTENT statement, the symbolic unit specified in the DTF is assumed. A symbolic unit included in the extent information for SAM, DAM, ISAM, or diskette files, however, overrides the DTF DEVADDR = SYSnnn specification.

This operand is not required if a system file with IJSYSxx as filename is specified. The following IJSYSxx filenames in a DLBL statement cause their corresponding default symbolic units to be specified in the EXTENT statement:

File Name	Default Logical Unit
IJSYSIN	SYSIN (SYSRDR/SYSIPT)
IJSYSPH	SYSPCH
IJSYSLS	SYSLST

IJSYSLN	SYSLNK
IJSYSRS	SYSRES
IJSYSxx	SYS0xx

The operand is also optional for a user file defined with a DTF DEVADDR = SYSnnn. If SYSRDR or SYSIPT is assigned, this operand must be included.

In multivolume SAM, DAM, ISAM, and diskette files, each different logical unit must be assigned to a separate physical device. In multi-extent SAM, DAM, ISAM, and diskette files, all extents on one physical unit must have the same logical unit number. For SAM and DAM files, the EXTENT statements must be in consecutive ascending order.

User programs may use, in addition to programmer logical units, the following system logical units:

SYSIPT and SYSRDR for input

SYSLST and SYSPCH for output

`serial number`

From one to six characters indicating the volume serial number of the volume for which this extent is effective. If fewer than six characters are used, the field is padded on the left with zeros, unless you enclose it in quotes, in which case it is padded on the right with blank characters.

If this operand is omitted, the volume serial number of the preceding EXTENT is used. Therefore, when a multivolume file is being processed, the volume serial number of the first volume is assumed for the entire file, unless you specify this field for the first extent of each following volume. If no serial number was provided in the EXTENT statement, the serial number is not checked and it is your responsibility if files are destroyed because the wrong volume was mounted. The serial number must be specified if the Access Control Facility is active in the system. This operand is also required for VSE/VSAM file extents.

You must also specify the serial number if the system has been IPLed with SEC = YES in the SYS command. Failure to do so will cause the job to be canceled with message 0S21I.

For the diskette, this operand specifies that the associated file will be found on this volume. If the parameter is omitted, the OPEN routines assume that the volume that was mounted is the correct one. Label checking will be done for input files and space will be allocated for an output file.

One EXTENT statement must be submitted for each volume of an input file, and sufficient EXTENT statements must be submitted for output files to ensure that enough volumes are present to contain the file.

`type`

One character indicating the type of the extent, as follows:

1 - data area (no split cylinder)

2 - independent overflow area (for indexed sequential files)

4 - index area (for indexed sequential files)

8 - data area (split cylinder, for SAM files only, but not on FBA devices)

If this operand is omitted, type 1 is assumed. 1 is the only valid type specification for diskette files.

For indexed sequential files, enter the extent information in the following order:

1. Master index (type 4) and sequence number 0.
2. Cylinder index (type 4) and sequence number 1.
3. Prime data area (type 1) and sequence number 2, 3, ..., n.
4. Independent overflow area (type 2) and sequence number (n + 1).

where n is the sequence number of the last prime data area extent.

Note also that the master and the cylinder index must be in adjacent areas on the same logical unit.

`sequence number`

One to three characters containing a decimal number 0 to 255 indicating the sequence number of this extent within a multi-extent file. Extent sequence number 0 is used for the master index of an indexed sequential file. If the master index is not used, the first extent of an indexed sequential file has the sequence number 1. The extent sequence number for all other types of files begins with 0. If this operand is omitted for the first extent of ISAM files, the extent is not accepted. For SAM, VSE/VSAM, or DAM files, this operand is not required. This parameter is ignored for diskette files.

`relative track|block`

For **CKD** devices, this operand is one to five characters indicating the sequential number of the track, relative to zero, where the data extent is to begin. If this field is omitted on an ISAM file, the extent is not accepted. This field is not required for SAM input files (the extents from the file labels are used). This field must be specified for DAM input files.

When using split cylinder files, this parameter designates the beginning of the split as well as the first track of the file.

To convert an actual address (in cylinders and tracks) to a relative track address, and vice versa, use the following formulae:

```
Actual to Relative

TC x cylinder number + track number = RT

Relative to Actual

RT / TC = cylinder number,
          remainder is track number
```

where RT is the relative track, and TC is the number of tracks per cylinder for the device type in question, as shown in Figure 3-6.

Device Type	Tracks per Cylinder
2311	10
2314, 2319	20
3330, 3333	19
3340, 3344	12
3350	30
3375	12
3380	15

Figure 3-6. Number of Tracks per Cylinder for Disk Devices

Example: Track 5, cylinder 150 on a 3380 = relative track 2255.

For **FBA** devices, this operand is a number from 2 to 2,147,483,645 which specifies the physical block at which the extent is to start.

For **VSE/VSAM**, this operand must be specified when a data space or a file with the UNIQUE option is being created. This operand is not required, and it is ignored if it is specified, when a VSE/VSAM file is created within an existing data space. In this case, the space for the file is sub-allocated by VSE/VSAM from direct-access extents it already owns. This operand is also not required for VSE/VSAM input files because the extents are obtained from the VSE/VSAM catalog.

`number of tracks|blocks`

For **CKD** devices, this operand is one to five characters indicating the number of tracks to be allocated to the file. For SD input, this field may be omitted, provided the 'relative track' field is also omitted. For an indexed sequential file, the number of tracks for prime data must be a multiple of the number of tracks per cylinder of the disk device used. For details, see Figure 3-6.

The number of tracks for a split cylinder file must be the product of the number of cylinders for the file and the specified number of tracks per cylinder for that file.

For **FBA** devices, this operand is a number from 1 to 2,147,483,645 which specifies the number of physical blocks in the extent.

This operand and **relative track|block** must either both be present or both be omitted. If the operands are present in an initial EXTENT statement, they must also be specified in all succeeding EXTENT statements. If they are omitted, they are ignored in all succeeding EXTENT statements.

`split cylinder track`

A one or two-digit decimal number, indicating the upper track number for the split cylinder in SAM files (for CKD devices only). The minimum specification is 0, the maximum is device-dependent, and is 1 less than the number of tracks per cylinder (see Figure 3-6 on page 3-54) for the device in question.

FREE (Reset RESERV Command)

The FREE command is used to reset the RESERVED status (as caused by the RESERV command) of the specified device. The command may be issued for all disk devices on the system.

Operation	Operands	Type
FREE	cuu	AR

cuu
Indicates the channel and unit number of the device to be freed.

GOTO (Skip to Label)

The GOTO statement causes all statements in the following job stream to be skipped, up to the specified label statement. It is accepted only within a job.

Operation	Operands	Type
[//] GOTO	label	JCC,JCS

label
: Specifies the operand of the /. statement at which execution of the current job is to continue. Code $EOJ to skip all statements up to end-of-job.

The job stream cannot be searched backwards, and the target label statement must be on the same level as the GOTO statement, that is, both outside a procedure or both in the same procedure.

JC does not check for duplicate labels. If two or more label statements are coded with the same operand, execution will continue after the first one to be found.

All JCL statements between the GOTO and the target label statement are ignored, except for the following:

- statements entered from SYSLOG.
- /+ (End-of-Procedure) -
  if this is encountered before the specified label statement is found, the rest of the job is skipped.
- // JOB and /& statements -
  if these are encountered, the job and its called procedure(s) are terminated.

Note: If you enter the GOTO statement from SYSLOG, the system gives you the opportunity to enter further statements (except GOTO and EXEC PROC) from SYSLOG. When you enter a blank line, JC switches back to SYSRDR and searches for the specified label.

HOLD (Hold Assignments and LIBDEFs)

The HOLD command is used to hold assignments or sublibrary definitions (LIBDEF) before you issue a command to unbatch a foreground partition. The partitions may be specified in any sequence; at least one partition must be given.

Operation	Operands	Type
HOLD	Fn[,Fn]...	JCC

n indicates the desired partition.

ID (User-ID and Password)

The ID statement or command is used to specify the user identification and the user's password. This information is checked against the contents of the user profile and, depending on the result of this check, the job is allowed to run or it is canceled.

An ID statement or command is required if a job uses resources protected by the access authorization facility of VSE; an ID statement should precede any ASSGN statements.

Operation	Operands	Type
[//] ID	USER=user-id, PWD=password	JCS,JCC

`user-id`
Specifies the user identifier, which must be four alphameric characters.

`password`
Specifies the password of the user, which can be three to six alphameric characters.

Neither the user-id nor the password will be displayed on SYSLOG or SYSLST. If the ID statement/command causes an error message, it is logged in the following format to avoid disclosure of the password:

```
ID (PARAMETERS SUPPRESSED)
```

IF (Check Local Condition)

The IF statement is a local conditional function. When it occurs in the job stream, the specified condition is checked. If it is true, the following statement is executed; if not, the following statement is skipped.

The IF statement is accepted only within a job.

Continuation lines are accepted for the IF statement.

Operation	Operands	Type
[//] IF	condition [operator condition] THEN	JCS,JCC

condition
Specifies a condition to be checked. It may be expressed in one of the following forms:

```
$RC   comparator   n
$MRC  comparator   n
pname comparator value
```

where:

$RC
Specifies the return code of the preceding job step.

$MRC
Specifies the maximum return code of all preceding steps within the current job.

pname
Specifies the name of a parameter to be compared.

comparator
Specifies the comparison to be done. This can be one of the following six possibilities:

Comparison:	Specified as:
Equal to	= or EQ
Not equal to	¬= or NE
Greater than	> or GT
Less than	< or LT
Greater or equal	>= or GE
Less or equal	<= or LE

If both comparands are numbers, the system does an arithmetic comparison. If one or both of the comparands contains any non-numeric character, the system does a logical comparison. This is carried out in the length of the longer comparand, and the shorter comparand is padded on the right with blank characters. If one of

the comparands is a null string, only the comparators =, ¬ =, EQ and NE are accepted.

n

Specifies a decimal integer from 0 to 4095.

value

Specifies a character string of 0 to 50 characters. If the string contains special characters, it must be enclosed in quotes. No quotes are allowed within the string. You can, of course, specify a symbolic parameter for this operand, for example,

```
IF PARM1>&PARM2 THEN
```

Or you can use a null string, For example:

```
IF PARM1='' THEN
```

operator

Specifies a logical operator which connects two conditions. The valid specifications are: OR, |, AND, &. The statement following the IF statement is executed when:

- The conditions are connected by OR or |, and one or both of them is true,
- The conditions are connected by AND or &, and both of them are true.

The logical operators OR, |, AND, & must be preceded and followed by a blank character.

You may enter an IF command from the console. In this case, the "following statement" is the next command you enter from the console, or the next statement from SYSRDR, if you enter a null line (just press END or ENTER) at the console.

Note: If the statement following the IF is a JOB, /& or /+ statement, it is not skipped, even when the condition in the IF statement is false.

For an example of the use of the IF statement, see Figure 3-12 on page 3-154.

IGNORE (Ignore Abnormal Condition)

Whenever an abnormal condition arises, the operator will be notified by an appropriate message on SYSLOG. Depending on the situation, he may have to ignore the condition by entering an IGNORE command. This is indicated under "Operator Action" in *VSE/System Package, Messages and Codes* for each applicable message.

Operation	Operands	Type
IGNORE	none	JCC,AR

The IGNORE command has no operand.

JOB (Identify Job)

The JOB statement indicates the beginning of control information for a job.

Operation	Operands	Type
// JOB	jobname [accounting information]	JCS

`jobname`
The name of the job. Must be one to eight alphameric characters (0-9, A-Z, #, $, @) or slash (/), hyphen (-), or period (.). When a job is restarted, the jobname must be identical to that used when the checkpoint was taken. Any user comments can appear on the JOB statement following the jobname (through column 71). The time of day appears in columns 73-100 when the JOB statement is printed on SYSLST. The time of day is printed in columns 1-28 on the next line of SYSLOG.

In both cases the format is

DATE mm/dd/yy,CLOCK hh/mm/ss

mm/dd/yy can also appear in the format dd/mm/yy, if this was specified in the STDOPT command.

`accounting information`
If the job accounting interface has been specified during system installation, the 16 characters of user information are moved to the job accounting table. If accounting information is specified, it must be separated from the job name by a single blank. If the job accounting interface has not been specified during system generation, any information specified after the job name is ignored.

Notes:

1. *If the JOB statement is omitted from the job stream, no duration and/or date is printed at end of job (when the /& statement is read).*

2. *The start time that the job control program displays is taken from the time-of-day clock (job step start time). The stop time for any given step is the start time for the next step.*

3. *Symbolic parameters are not accepted in the JOB statement.*

The layout of the job accounting table is described in *VSE/Advanced Functions System Management Guide.*

LFCB (Load Forms Control Buffer)

The LFCB command causes the system to load a buffer image, stored as a phase in the system sublibrary IJSYSRS.SYSLIB, into the forms control buffer (FCB) of the specified printer. The command can be used for any printer on which forms skip operations are controlled by an FCB.

For an IBM 4248 printer in native mode, however, the horizontal copy function cannot be activated or deactivated by the LFCB command.

If you have VSE/POWER in your system, use the *$$LST control statement with the FCB operand for this purpose. During the time the printer in question is printing the output of a program, this command should be used with extreme caution, as there is no way of predicting when the printer will be finished printing the output under control of the buffer image currently contained in the FCB.

For a printer in operation it is recommended that the operator use this command if, for example, printing the output for a particular program started under control of the wrong FCB image and he is able to correct this by issuing the command.

Operation	Operands	Type
LFCB	cuu,phasename[,FORMS=xxxx][,NULMSG]	AR

cuu
: Specifies the channel and unit number of the printer whose FCB is to be loaded.

phasename
: Specifies the name of the phase that contains the applicable buffer load image. For detailed information on the contents and format of this phase refer to the section Section 7, "System Buffer Load (SYSBUFLD)" on page 7-1.

FORMS=xxxx
: Specifies the installation-defined forms number xxxx of the paper that is to be used with the new FCB image. For xxxx, substitute from one to four alphameric characters. If the new FCB image requires a change of forms, this operand must be specified to ensure proper system operation.

NULMSG
: Specifies that the printing of a buffer load verification message is to be suppressed. If NULMSG is specified, the system continues processing immediately after the FCB load operation has been completed, and the operator is unable to verify that the contents of the FCB match the forms to be used.

LIBDEF (Define Sublibrary Chain)

The LIBDEF statement defines which sublibraries are to be searched for members of a specified type or types, and, where appropriate, the sublibrary in which new phases or dumps are to be stored. The defined sequence is referred to as a search chain. Different chains can be defined for different member types, or a common chain can be established for all types except DUMP. The specified sublibraries are searched in the sequence as entered in the LIBDEF command or statement.

The system sublibrary IJSYSRS.SYSLIB is always added at a default position in the search chain, unless it is explicitly included at a different position in the chain. For details, see "Phase Chaining" on page 3-67.

Notes:

1. *The librarian program does not use the information given in LIBDEF statements to access sublibraries.*

2. *Continuation lines are allowed for this statement.*

Operation	Operands	Type
[//] LIBDEF	{type|*} [,SEARCH=(lib.sublib,...)] [,CATALOG=lib.sublib][{,TEMP|PERM}]	JCC,JCS

`type`

Defines the member types for which this LIBDEF statement applies. For type, specify:

PHASE

To define a sublibrary chain to be used for loading or fetching program phases for execution. Only members of the type PHASE are searched for. The CATALOG operand specifies the library and sublibrary in which phases are to be cataloged by the linkage editor.

OBJ

To define a sublibrary chain to be used by the linkage editor when searching for object modules. Only members of the type OBJ are searched for. The CATALOG operand is not applicable.

SOURCE

To define a sublibrary chain to be used, for example by language translators, when searching for one of the predefined " SOURCE" types (A-Z, 0-9, #, $, @). The CATALOG operand is not applicable.

PROC

To define a sublibrary chain to be used by Job Control when searching for a procedure to be executed. Only members of the

type PROC are searched for. The CATALOG operand is not applicable.

*Note: If a LIBDEF PROC... or LIBDEF *... statement is cataloged in a procedure, this procedure must fulfill the following criteria:*

- *It must reside in the system sublibrary IJSYSRS.SYSLIB;*
- *It must not be nested;*
- *It must not contain an EXEC PROC statement after the LIBDEF statement.*

*

To indicate that the LIBDEF statement applies to all member types except DUMP and user types. That is, a common chain for all other member types is established. If the CATALOG operand is specified, it will apply for members of the type PHASE only. See note under type PROC.

DUMP

To define a sublibrary to be used by the system when a dump is to be produced and the option SYSDUMP is in effect, or a CANCEL command with the SYSDUMP operand is issued. You must use the keyword CATALOG if you specify DUMP as the type operand.

`SEARCH=lib.sublib`

Is required if you specified OBJ, SOURCE or PROC in the type operand. With type PHASE, or * you must specify SEARCH or CATALOG or both. The specified sublibraries will be searched in the sequence in which they are specified in this operand.

For all types of member except system phases, the system sublibrary IJSYSRS.SYSLIB is added at the end of the chain by default, unless you specify it explicitly at another position in the operand list. If the system sublibrary is added by default, then access control (if active) checks only the universal access rights for the system sublibrary. Any higher individual access right to this sublibrary is ignored.

For a LIBDEF statement with the type PHASE, the system directory list (SDL) may also be specified explicitly in the operand list. The default chaining sequence used when searching for phases depends on whether they are system phases or not. For details, see "Phase Chaining" on page 3-67.

Note: You may specify a list of up to 15 sublibraries in one search chain. The sublibrary names in the list must be separated by commas. If you specify only one sublibrary, it need not be enclosed in parentheses.

CATALOG=lib.sublib
: Is applicable for LIBDEF statements with the type PHASE, DUMP or * only. It specifies the library/sublibrary into which the linkage editor or DUMP output is to be cataloged. There is no system default.

TEMP | PERM
: Specify the duration of the definition given in the statement. If you specify TEMP, the defined chain will be dropped:

- At end-of-job, or
- When overridden by a new LIBDEF...TEMP statement or command, or
- When overridden by a LIBDROP...TEMP statement or command.

If PERM is specified, the chain will remain valid until:

- The partition is deactivated by an UNBATCH command, or
- A LIBDROP...PERM statement or command is issued, or
- A new LIBDEF statement overrides the definition wholly or in part.

If you omit both of these operands, TEMP will be assumed by default.

If both a TEMP and a PERM LIBDEF statement have been issued for a given member type, the following rules apply:

- For SEARCH, the TEMP search chain is placed logically before the PERM chain.
- For CATALOG, the TEMP library definition is used. If a sublibrary protected by the access control function is specified in a **permanent** LIBDEF command or statement, this sublibrary **must** have a universal access right of connect or higher. For a **temporary** LIBDEF, the normal security checking is done. That is, the universal access right or the individual access right of the user who enters the command, whichever is the greater, is used.

Phase Chaining

The search chain for phases includes the system directory list (SDL), and is different for "system" and "non-system" phases. In this context, "system phases" are phases which:

1. Have a name starting with a dollar sign ($), or;
2. Are being loaded with the SYS = YES operand in the LOAD macro.

The search chains are:

- For "non-system" phases: SDL -- TEMP chain -- PERM chain -- IJSYSRS.SYSLIB.
- For "system" phases: SDL -- IJSYSRS.SYSLIB -- TEMP chain -- PERM chain.

If you do not wish the SDL to be searched first, you must specify it at the appropriate position in the operand list of the LIBDEF statement for phases. SDL can be specified **only in a temporary** LIBDEF statement. You must not place IJSYSRS.SYSLIB before the SDL in a search chain.

For the access control aspects of the LIBDEF statement, see the chapter " Data Protection" in the *VSE/Advanced Functions System Management Guide.*

LIBDROP (Drop Sublibrary Chain)

The LIBDROP statement resets the library search and catalog definitions set up by one or more previous LIBDEF statements. Continuation lines are allowed for this statement.

Operation	Operands	Type
[//] LIBDROP	{type|*}[,SEARCH][,CATALOG][{,TEMP|PERM}]	JCC,JCS

type
: Specifies the member type for which the search and catalog definitions are to be reset. It corresponds to the type operand in the LIBDEF statement.

*
: Specifies all types except DUMP. If operands SEARCH and CATALOG are both specified, all previous PERM or TEMP definitions for search chains and catalog libraries will be dropped.

*Note: If a LIBDROP PROC... or LIBDROP *... statement is cataloged in a procedure, this procedure must fulfill the following criteria:*

- *It must reside in the system sublibrary IJSYSRS.SYSLIB;*
- *It must not be nested;*
- *It must not contain an EXEC PROC statement after the LIBDROP statement.*

SEARCH
: Specifies that only the search chain is to be dropped.

CATALOG
: Specifies that only the sublibrary defined in the CATALOG operand of a previous LIBDEF statement is to be dropped.

Note: If neither SEARCH nor CATALOG is specified, both chains are dropped.

TEMP|PERM
: Specify whether the TEMP or PERM definition is to be dropped. The system default is TEMP.

LIBLIST (Query Sublibrary Chains)

The LIBLIST statement causes the library definitions set up with the LIBDEF statement to be displayed on SYSLOG or SYSLST. Continuation lines are allowed for this statement.

Operation	Operands	Type
[//] LIBLIST	{type|*}[,BG|Fn|*][,SYSLST|SYSLOG]	JCC,JCS

`type`
Specifies the member type for which the library definitions are to be displayed. It corresponds to the type operand of the LIBDEF statement.

`*`
Specifies all types except DUMP, and causes the library definitions of all LIBDEF statements to be displayed.

`BG|Fn|*`
Specify the partition for which the current library definitions are to be displayed. BG denotes the background partition, and Fn a foreground partition, whereby n is the number of the partition, and * means all partitions. If these operands are all omitted, the output shows the library definitions for the partition in which the LIBLIST command is entered.

`SYSLST|SYSLOG`
Specify the output device to be used for displaying the library definitions. If this operand is omitted, SYSLST will be used, unless the LIBLIST command was entered at SYSLOG, in which case the information will be displayed there.

LISTIO (Query I/O Assignments)

The LISTIO command or statement causes the system to print a listing of I/O assignments. The listing appears on SYSLOG (for the command format) or SYSLST (for the statement format). If SYSLST is not assigned, the // LISTIO statement is ignored.

Operation	Operands	Type
[//] LISTIO	listtype	JCS,JCC

listtype can be one of the following:

ALL
: Lists the physical units assigned to all logical units (of all partitions).

ASSGN
: Lists the physical units assigned to all system and programmer logical units of the partition from which the command is issued. Unassigned units are **not** listed.

BG
: Lists the physical units assigned to all logical units of the background partition. See Note 1.

cuu
: Lists the logical units assigned to the specified physical unit. See Note 2.

DOWN
: Lists all physical units specified as inoperative.

Fn
: Lists the physical units assigned to all logical units of the specified foreground partition. See Note 1.

NPGR
: Lists the number of programmer logical units allocated to each partition.

PROG
: Lists the physical units assigned to all programmer logical units of the partition from which the command is issued. See Note 1.

SYS
: Lists the physical units assigned to all system logical units of the partition from which the command is issued.
See Note 1.

SYSxxx
: Lists the physical units assigned to the specified logical unit of the partition from which the command is issued (invalid for SYSOUT and SYSIN).

UA
: Lists all physical units not currently assigned to a logical unit.

UNITS
: Lists the logical units assigned to all physical units. See Note 2.

Notes:

1. *Unassigned logical units are listed as UA.*

2. *Unassigned or inoperative physical units are listed as UA or DOWN, respectively.*

An example of a listing produced by the LISTIO command is shown in *VSE/Advanced Functions Operation.*

LOG (Log JC Statements)

The LOG command or statement causes the system to log all job control commands and statements occurring within the scope of the LOG. The scope of the LOG depends on which form is used, as follows:

- The attention routine command LOG affects all partitions. Columns 1 to 72 of all logged commands and statements are written to **SYSLOG**. LOG remains effective until an attention routine NOLOG command (for all partitions) or a job control NOLOG command (for the one partition) is given.

- The job control command LOG affects the partition in which it is issued. Columns 1 to 72 of all logged commands and statements are written to **SYSLOG**. LOG remains effective until a job control NOLOG command for the same partition, or an attention routine NOLOG command for all partitions, is given.

- The job control statement // LOG affects only the statements which follow it in the job in which it is issued. Columns 1 to 80 of all logged commands and statements, including the // LOG statement itself, are written to **SYSLST**. // LOG is effective until end-of-job, or until a // NOLOG statement occurs in the same job.

  Note: The // LOG statement has the same effect as the // OPTION LOG statement. The two statements are interchangeable, and each can be reset by either the // NOLOG or the // OPTION NOLOG statement.

To have job control statements and commands of a given job logged only on SYSLST, use the job control NOLOG command in the appropriate partition and the // LOG statement in the job.

The // LOG statement can be used to print comments on SYSLST.

The LOG command or statement suppresses OPTION ACANCEL. That is, it prevents jobs being canceled because of an unsuccessful ASSGN or LIBDEF attempt.

Operation	Operands	Type
LOG	none	JCC,AR
//LOG	none	JCS

The LOG command has no operand.

LUCB (Load Universal Character-Set Buffer)

The LUCB command causes the system to load the buffer image, contained in the named phase, into the universal character set buffer (UCB) of the specified printer. The printer must be ready or in operation. The command can be used for any printer equipped with the UCS feature, except the 1403U, and the 4248 printer in native mode (that is, added with the device type 4248 at IPL).

For the 4248, use the attention routine command BANDID.

While a printer is printing the output of a program, this command should be used with caution. There is no way of knowing when the printer has finished printing the output under control of the buffer image currently contained in the UCB.

For a printer in operation it is recommended that the operator use this command if, for example, printing the output for a particular program started under control of the wrong UCB image and the operator is able to correct this condition by issuing the command.

Operation	Operands	Type
LUCB	cuu,phasename[,FOLD][,NOCHK] [,TRAIN=xxxxxx][,NULMSG]	AR

cuu
: Specifies the channel and unit number of the printer whose UCB is to be loaded.

phasename
: Specifies the name of the phase which contains the applicable buffer load image. For detailed information on the contents and format of this phase refer to the section Section 7, "System Buffer Load (SYSBUFLD)" on page 7-1.

FOLD
: Causes lowercase characters to be printed as uppercase characters.

NOCHK
: Causes data checks resulting from mismatches between print-line characters and the UCB to be suppressed.

TRAIN=xxxxxx
: Indicates that the print train identified by xxxxxx is to be mounted on the printer. The system inserts this operand in an action message. The train identification xxxxxx may be from one to six characters in length. If a new train (or chain) must be installed, this operand is required to ensure proper system operation.

NULMSG

Specifies that the printing of a buffer load verification message is to be suppressed. If NULMSG is specified, the system continues normal processing immediately after the UCB load operation has been completed and the operator is unable to verify that the contents of the UCB match the print train (or chain) mounted on the printer.

MAP (Map Storage Areas)

The MAP command produces, on SYSLOG, a map of all storage areas in the system, with their sizes and starting addresses. The execution mode (virtual or real) and the priority of the partitions are shown, and the name of the job currently running in each partition is given.

The MAP command has no operands. The output always contains the same information elements.

Operation	Operands	Type
MAP	none	JCC,AR

The map of the allocated storage areas is basically the same for 370-mode, ECPS:VSE-mode and VM-mode. The 370-mode version has two additional columns, SPACE and R-ADDR, and an additional line, UNUSED, for each address space. These additions deal with the multiple address space layout. The form of the output is shown in the following examples.

370-mode MAP output example:

```
SPACE AREA PRTY V-SIZE GETVIS V-ADDR R-SIZE R-ADDR NAME

  S   SUP          448K              0   300K      0 $$A$SUP3
  1   BG V    4   1920K   128K  70000   128K  70000 NO NAME
  1   F9 V    4    464K    48K 270000    64K  90000 NO NAME
  1   F8 V    4   4608K  1536K 2F0000   256K  A0000 ICCFSTRT
  1   FA I    4    128K   384K 8F0000    64K  E0000
  1   FB I    4    128K   384K 970000    64K  F0000
  1   UNUSED       576K        9F0000
  2   F7 V    4    432K   592K  70000   128K 100000 NO NAME
  2   F6 V    4   4096K   960K 170000   128K 120000 CICSF6
  2   UNUSED      4224K        660000
  3   F5 V    4   3264K   832K  70000   128K 140000 NO NAME
  3   F4 V    4   3584K   512K 470000   256K 160000 CICSF4
  3   UNUSED      2112K        870000
  S   F1 V    1    512K   256K A80000    64K 1E2000 IPWPOWER
  S   F2 V    2   1280K   768K B40000   200K 1B0000 VAEVTAM
  S   F3 V    3    128K   128K D40000    64K 1A0000 NO NAME
  S   SVA         1560K  1000K D80000   176K
      AVAIL      10880K                1920K
      TOTAL      40960K                3940K
```

This is how the available virtual storage is calculated in the example above:

```
Total of V-SIZE column:                29464K
Minus UNUSED Space:
(576K + 4224K + 2112K)               - 6912K

                                      22552K
Plus total of GETVIS column:         + 7528K

Equals total used virtual storage:    30080K

Virtual storage in system (TOTAL line):        40960K
Minus total used virtual storage:             -30080K

Equals available virtual storage (AVAIL line): 10880K
```

ECPS:VSE-mode and VM-mode MAP output example:

```
AREA PRTY V-SIZE GETVIS V-ADDR R-SIZE NAME
SUP         224K              0   180K $$A$SUPD
BG V   12   208K    48K  38000     0K BATCH
F3 V    6  2048K  1024K  78000     0K CICS1
F5 V    7  3584K  1536K 378000     0K CICS2
F7 R    3   176K    48K 878000   224K MICR
F4 V    1   512K   256K 8B0000    52K POWER
F2 V    4  1024K   512K 970000   100K VTAM
F1 V    2   384K   128K AF0000     0K OCCF
AVAIL      2624K        B70000  3470K
SVA        1280K   768K E00000    70K
TOTAL     16384K                4096K
```

Explanation of Keywords:

SPACE

370-mode only: The space-ID of the address space in which the respective area is located. The space-ID R does not occur; real allocations appear in the R-SIZE and R-ADDR columns.

AREA

This column identifies to which storage area the data in the respective line refer, as follows:

```
SUP     = supervisor area
BG, Fn = partition
R       = partition with real storage allocation
V       = partition without real storage allocation
I       = partition inactive
AVAIL   = address space still available for allocation
SVA     = shared virtual area
TOTAL   = maximum possible allocation in the system
UNUSED = 370-mode only: Non-allocated storage in the
           indicated address space
```

PRTY

The partition priority.

V-SIZE
: The amount of virtual storage in the area. For the partitions, this is the value specified in the SIZE command for the partition. If no SIZE command has been used, V-SIZE is the partition size, as specified in the ALLOC command, minus 48K.

  In the SUP line, the size of the pageable part of the supervisor.

GETVIS
: Size of permanent GETVIS space in the area. This is partition size minus V-SIZE (ALLOC specification minus SIZE command specification). If no SIZE command has been given for the partition, GETVIS is 48K.

V-ADDR
: The starting address of the area in virtual address space. When referring to part of an area in 370-mode, for example in a DUMP, DSPLY or ALTER command, use the space-ID in the SPACE column.

R-SIZE
: The amount of real storage in the area. In the SUP line, the size of the non-pageable supervisor.

R-ADDR
: **370-mode only**: the starting address (in address space R) of the real storage in the area.

NAME
: The name of the supervisor (SUP line), or the job currently running in the respective partition. If the job accounting file has become full, this column shows the job name 'JOB ACCT' instead of the name of the current job. When no JOB statement has been entered in the partition, this field contains NO NAME.

MODE (Alter Recording Mode)

The MODE command allows you to alter the recording mode. It provides the following options for controlling recoverable machine check interrupts (MCI):

- The mode of recording for unlabeled and nonstandard labeled tape can be reset.
- The recording mode for a particular device other than a teleprocessing device can be set to intensive or diagnostic, or no recording mode can be specified.
- The mode that the system is operating in (the status of the system) can be requested.
- An EFL threshold value can be specified to override the IBM-supplied value.
- The MODE command can also be used to place the control storage ECC of a System/370 Model 145 or 148 in threshold mode.

The MODE command is a notational command. Only one blank is allowed between the MODE keyword and the first operand. Operands of the MODE command must be continuous, that is, no blanks are allowed between or within operands. The STATUS operand cannot have any other operand, before or after it. The total length of the MODE command must not exceed 30 characters.

Operation	Operands	Type
MODE	{IR| CR| CE,cuu[,I[,xx,y]|,D[,xx,y]|,N]| R| STATUS| HIR{[,R|,Q][,E=eeee][,T=tttt]}| ECC{,M|,C}[,R|,Q|,TH][,E=eeee][,T=tttt]}	AR

In a VM/370 environment the R, HIR, and ECC operands of the MODE command are not accepted and will cause message 1I94I COMMAND IGNORED IN VM/370 ENVIRONMENT to be issued.

For the 4300 Processors, only the operands IR, CR, and CE may be used. All other operands are invalid.

For the Models 135/138, the only valid MODE commands are:

```
MODE CE,...
MODE STATUS
MODE ECC,Q
MODE ECC,R
```

The meanings of the operands are:

IR

Recording mode for nonstandard labeled and

CR

unlabeled tape. Specify Individual Recording (IR) if you wish to record and then reset the tape error statistics at each tape OPEN. Specify Combined Recording (CR) to accumulate all the statistics from nonstandard labeled and unlabeled tape on a specific tape unit until a standard labeled tape is opened. Then one recording of the statistics from all the nonstandard labeled and unlabeled tapes is made on SYSREC, and the statistical counters are reset in the PUB2 table.

CE, cuu

The recording mode for a device at physical location cuu may be reset. The possible recording modes are:

Normal

Normal recording is carried out if you specify CE,cuu without the further operands I, D or N. Note that there is no operand "Normal."

I

Intensive. Normal recording continues. In addition, the next seven errors of a particular type (xx,y) or the next seven errors of any type (if xx,y is not specified) are recorded. The number of I/O retries required for success is not recorded.

D

Diagnostic. Normal recording continues. In addition, the next seven errors of a particular type (xx,y) or the next seven errors of any type (if xx,y is not specified) are recorded. The number of I/O retries required for success is also recorded.

N

No recording. Recording for the specified device is switched off until the next IPL. (The recording mode cannot be reinstalled by a new MODE command specifying I or D.

When the recording mode parameter is the last parameter of the MODE command, a check is made to see if all errors are recorded. When in intensive or diagnostic mode, it is possible to check for only one type of error. Indicate the bit to be examined with:

```
(xx,y)  where y is the bit (0-7) and xx the byte
        (0-31) of sense data to be checked.
```

STATUS

A report is printed on SYSLOG which indicates:

- The type of facility used (HIR,ECC)
- System mode of operation
- Current error count
- Error count threshold

- Current elapsed time
- Number of buffer pages deleted.

A buffer page is a 32-byte work area in control storage that is used by the Model 155-II, 158, and 3031 hardware program.

The status report formats are:

For Models 135/138

ECC,{R|Q}

For Models 145/148

HIR,{R|Q},aaaa/eeee,bbbb/tttt
ECC,{R|Q},{M|C},aaaa/eeee,bbbb,tttt

where:
aaaa = current error count
eeee = error count threshold
bbbb = current elapsed time
ttttt = total threshold

For Models 155-II/158/3031

HIR,{R|Q},aaaa/eeee,bbbb/tttt
ECC,{R|Q,aaaa/eeee,bbbb/tttt
BUF DLT = xxx

where:
aaaa = current error count
eeee = error count threshold
bbbb = current elapsed time
tttt = time threshold

```
xxx  = total number of inoperable buffer pages
       deleted.
```

HIR

Hardware Instruction Retry. This operand changes the mode of the HIR facility to R or Q and/or modifies the error count threshold and/or time threshold.

Note: When HIR is placed in quiet mode, ECC also goes into quiet mode.

ECC

Error Correction Code. This operand changes the mode of the ECC facility to R or Q, and/or modifies the error count threshold and/or time threshold. ECC,R and ECC,Q are the only valid modes of diagnosis for the Models 135/138. If ECC is specified for a Model 145 or 148, M or C must also be specified. ECC can also place the Model 145/148 control storage in threshold mode.

Note: Use of the Error Correction Code (ECC) in full recording mode may cause severe system degradation. Thus, the ECC,M/C,RR operand combination of the MODE command should only be used by the customer engineer or at his request.

R

Recording Mode

MODE R - places both HIR and ECC in recording mode.

MODE HIR,R - places HIR in recording mode.

MODE ECC,R (Models 155-II/158, and 3031) - if HIR is already in recording mode, it places ECC in recording mode.

MODE ECC,M,R (Models 145/148) - if HIR is already in recording mode, available processor storage is placed in recording mode.

MODE ECC,C,R (Models 145/148) - if HIR is already in recording mode, control storage is placed in recording mode.

Q

Quiet Mode

MODE HIR,Q - places both HIR and ECC in quiet mode.

MODE ECC,Q (Models 135/138, 155-II/158, and 3031) - places ECC in quiet mode.

MODE ECC,M,Q (Models 145/148) - places real storage in quiet mode.

MODE ECC,C,Q (Models 145/148) - places control storage in quiet mode.

M or C

Real or control storage: M or C is only valid for Models 145/148. They must be specified when ECC is specified. M indicates real storage and C control storage.

TH

Threshold Mode: On the next occurrence of an ECC control storage error, control storage is placed in quiet mode. TH is only valid for the Models 145/148 if ECC,C is specified. TH places the Models 145/148 control storage ECC in threshold mode. If TH is specified, T = tttt must also be specified.

E=eeee

E: 8 (IBM-supplied value) through 9999

T=tttt

T: 8 (IBM-supplied value) through 9999

Note: Whenever HIR is in quiet mode, ECC mode must not be changed.

MPXGTN (Set Byte-Multiplex Gating)

The MPXGTN command sets gating for devices running on byte-multiplex (MPX) channels. Use the command to set gating ON if you experience overrun problems on a byte-MPX channel.

Operation	Operands	Type
MPXGTN	{ON|OFF}	AR

ON
Turns gating for MPX channels on.

OFF
Turns gating for MPX channels off.

Non-buffered or overrunable devices can overrun if more than one of them are attached to the same byte-MPX channel. More than one device operating in burst mode on one byte-MPX channel can cause the same problem. When gating is turned on, the system allows only one burst-mode or overrunable device to use a byte MPX-channel at any time.

MSECS (Change or Query Time Slice)

The MSECS command displays or changes the time slice for partition balancing.

Operation	Operands	Type
MSECS	n	JCC
MSECS	[n]	AR

n

Specifies the new time slice in milliseconds, and must be an integer from 100 to 10000.
If you use the attention routine command without the operand, the system displays the current time slice in milliseconds.

The Attention Routine Command MSECS can be entered at any time, but the Job Control Command can be used only within an ASI procedure. In the Job Control Command MSECS you must code a valid value for n.

MSG (Communicate With Program)

The MSG command transfers control to an operator communications routine for which linkage has been established with a STXIT macro.

Operation	Operands	Type
MSG	[BG|Fn]	AR

BG
: Indicates that communication with the background partition is desired. This is assumed by default if the operand is omitted.

Fn
: Indicates the desired foreground partition.

If the program in the specified partition has not established operator communication linkage, a message is printed on SYSLOG informing the operator of this condition.

MTC (Magnetic Tape Control)

The MTC command or statement controls magnetic tape operations.

Operation	Operands	Type
[//] MTC	opcode,{cuu|SYSxxx}[,nn]	JCC,JCS

opcode
Specifies the operation to be performed as explained in Figure 3-7.

SYSxxx
Specifies the logical unit to which the tape is assigned.

cuu
Specifies the channel and unit number.

nn
Is a decimal number from 1 through 99 that indicates the number of times the specified operation is to be performed. The default is 1.

Opcode	Meaning	Possible Use
BSF	Backspace File	Backspace one file so tape is positioned for reading the tapemark preceding the file backspaced.
BSR	Backspace Record	Backspace record.
DSE	Data Security Erase	(See Note)
ERG	Erase Gap	Erase gap.
FSF	Forward Space File	Used when restarting a program. The tape is positioned beyond tapemark following the file spaced over.
FSR	Forward Space Record	Locate a specific record within a file.
RUN	Rewind and Unload	Rewind and unload a tape on a specific unit.
REW	Rewind	Rewind a tape on a specific unit.
WTM	Write Tape Mark	Write a tapemark on an output file.

Note: Data security erase (3400-series only). This command erases a tape from the point at which the tape is positioned when the operation is initiated up to the end-of-tape reflective marker. If data is written after the end-of-tape reflective marker, the data must be erased with [//] MTC ERG SYSxxx.

Figure 3-7. Operation Codes for MTC Statement

If the MTC DSE command is issued when the tape is at load point, the contents of the tape, including the volume label, are erased completely. In such a case the tape must be reinitialized or a tapemark must be written on it before it can be used again.

The partition that issued the [//] MTC DSE command is placed in the wait state until the end-of-tape reflective marker is reached.

NEWVOL (Alter Volume Assignment)

If an assignment specifying VOL= was given for a disk or tape unit and the system cannot find the requested volume on that unit, then the system prints message 1T50A on SYSLOG, requesting the operator to mount the desired volume. The partition enters the wait state. The operator can now either mount the proper volume, make the device ready, and issue the NEWVOL attention command to indicate that processing may continue with the new volume, or - if the volume cannot be mounted - he can cancel the mount request by specifying the IGNORE operand.

Operation	Operands	Type
NEWVOL	[BG|Fn][,IGNORE]	AR

BG|Fn
: Indicates the partition for which the new volume was mounted. If no operand is specified, BG is assumed. If the specified partition is not waiting for a volume to be mounted, an error message is printed on SYSLOG.

IGNORE
: Specifies that the mount request is to be ignored. This causes message 1T40D to be displayed, after which you can either give a new assignment, or cancel the job, or enter any other command.

NOLOG (Suppress JC Logging)

The NOLOG command or statement stops the listing of job control commands and statements (except ALLOC, DVCUP, HOLD, IGNORE, JOB, MAP, PAUSE, PRTY, SIZE, STOP, UNBATCH /*, /& and /+) that occur within the scope of the NOLOG. The scope depends on the form of the NOLOG, as follows:

- The attention routine command NOLOG affects all partitions. It is effective until an attention routine LOG command (for all partitions) or a job control LOG command (for the issuing partition only) is given.
- The job control command NOLOG is affects only the partition in which it is issued.
- The job control statement // NOLOG affects only the commands and statements following it in the job in which it is issued. It overrides any previous // LOG statement for the remainder of the job. The // NOLOG statement itself is written to SYSLST.

  Note: The // NOLOG statement has the same effect as the // OPTION NOLOG statement. These statements are interchangeable. Each can be used to reset either a // LOG or a // OPTION LOG statement.

Operation	Operands	Type
NOLOG	none	JCC,AR
// NOLOG	none	JCS

The NOLOG command has no operand.

If a generic ASSGN statement or command is suppressed by NOLOG, the message indicating which physical unit the system has selected is still displayed on SYSLOG.

NPGR (Number of Programmer Logical Units)

The NPGR command defines the number of programmer logical units which may be allocated in a given partition.

Operation	Operands	Type
NPGR	[BG=m][,Fn=m]	JCC

BG,Fn

Specify the partition for which you wish the system to allocate the given number (m) of programmer logical units.

The operand BG can only be specified in the BG partition itself, and only when no other partition has been started since IPL.
The operand Fn can be specified in any partition, but only before partition Fn itself is started for the first time after IPL.

m

A decimal integer from 10 through 255. The total number of logical units for all partitions must not, however, exceed the number specified in the NPGR parameter at supervisor generation.

30 programmer logical units (SYS000 to SYS029) are available for each partition by default. The programmer logical unit names used **must** be in the range SYS000 to SYSnnn, where nnn = m - 1.

ON (Set Global Condition)

The ON statement is a global conditional function. During execution of a job in which an ON statement occurs, the specified condition is tested at the end of each job step following the ON statement. If the condition is true, the specified action is taken, otherwise processing continues with the next statement.

The ON statement is accepted only within a job.

Operation	Operands	Type
[//] ON	condition [operator condition] action	JCS, JCC

Condition
: Specifies the condition under which the action specified in the ON statement is to be taken. It may be expressed in one of the following forms:

```
$RC comparator n
$CANCEL
$ABEND
```

where:

$RC
: Specifies the return code of the preceding step.

comparator
: Specifies the comparison to be done. This can be one of the following six possibilities:

Comparison:	Specified as:
Equal to	= or EQ
Not equal to	¬= or NE
Greater than	> or GT
Less than	< or LT
Greater or equal	>= or GE
Less or equal	<= or LE

n
: Specifies a decimal integer from 0 to 4095, which is to be used for comparison with the return code.

$CANCEL
: Specifies that the action is to be taken if the CANCEL command is given for the job.

Note: If the job is canceled with the operand FORCE in the CANCEL command, the action specified in the ON statement is **not** *carried out.*

$ABEND
Specifies that the action is to be taken if the step terminates abnormally.

`operator`
Specifies a logical operator which connects two conditions. The valid specifications are: OR, |, AND, &. Each condition has the format described above. The specified action is taken at End-of-Job-Step when:

- The conditions are connected by OR or |, and one or both of them is true;
- The conditions are connected by AND or &, and both of them are true.

The logical operators (OR, |, AND, &) must be preceded and followed by a blank character.

`action`
Specifies the action to be taken if the specified condition is true at End-of-Job-Step. It may be expressed in one of the following forms:

```
GOTO label
CONT°INUE
```

where:

GOTO label
Specifies that processing is to continue at the specified label statement. For rules on the use of this operand, see the explanation of the GOTO statement on page 3-57.

CONTINUE
Specifies that processing should continue if the specified condition is true. CONTINUE is not valid with the conditions $CANCEL or $ABEND.

Whenever a job starts, the following default ON-conditions are in effect:

```
ON $RC<16 CONTINUE
ON $RC>=16 GOTO $EOJ
ON $ABEND GOTO $EOJ
ON $CANCEL GOTO $EOJ
```

ON-conditions remain in effect up to End-of-Job if they are specified outside of a procedure; if they are specified in a procedure, they are in effect up to the end of that procedure. If two or more ON statements at the same level specify the same condition, the action specified in the last one to be issued is carried out. If you are using nested procedures, JC will check the condition of an ON statement on the level at which it is specified, and on all lower levels.

If you specify GOTO label as the action to be taken, the target label statement specified must be on the same level as the ON condition GOTO label statement,

that is, either both in the same procedure, or both outside a procedure (on JC level 0).

If an ON condition occurs in a called (lower) procedure, the target of an associated GOTO will be searched for only in the procedure (or JC level 0) where the ON statement was specified, starting after the // EXEC PROC statement which called the procedure in which the condition was raised.

If a CANCEL command with the FORCE operand is given, or a job terminates abnormally while job control is running, no ON-conditions are checked. The rest of the job is skipped.

For an example of the use of the ON statement, see Figure 3-13 on page 3-155.

ONLINE (Simulate DEVICE READY)

The ONLINE command is used to simulate a 'device ready' status for a device.

Operation	Operands	Type
ONLINE	cuu	AR

cuu

Indicates the channel and unit number, in hexadecimal, of the particular device.

OPTION (Set Temporary JC Options)

The OPTION statement specifies one or more job control options which temporarily override the system defaults. These may have been changed by the STDOPT command. Consult your system programmer to find out which defaults apply on your system.

Operation	Operands	Type
// OPTION	option [,option]...	JCS

The options specified in the OPTION statement remain in effect until a contrary option is encountered or until a JOB or a /& control statement is read. In the latter case, the options are reset to the system default values or those established by the STDOPT command.

Each operand of the OPTION statement is processed separately. If processing of one operand fails, the **following** operands are **not** processed. The options specified **before** the failing operand, however, do **take effect**.

If you enter conflicting operands in one OPTION statement, the last one overrides the first.

The options, which can appear in any order, are as follows:

ACANCEL
: Indicates that the job must be canceled (instead of awaiting operator intervention) if an ASSGN or LIBDEF command fails. This may happen because of an undefined device, invalid device status, unassignable unit, or conflicting I/O assignments.

  The ACANCEL option is suppressed when either the LOG command has been issued by the operator or a LOG statement for the partition is effective.

NOACANCEL
: Suppresses the ACANCEL option. The system awaits operator intervention in the case of an unsuccessful assignment.

ALIGN
: The assembler aligns constants and data areas on proper boundaries and checks the alignment of addresses used in machine instructions.

NOALIGN
: Suppresses the ALIGN option.

CATAL
: A phase or program is permanently cataloged in a library at the completion of a link-edit run. CATAL also sets the LINK option. (See also Note 1.)

DECK
Language translators produce object modules on SYSPCH. If LINK is specified, the DECK option is accepted by the PL/I, VS/FORTRAN, and DOS/VS COBOL compilers, and the assembler.

NODECK
Suppresses the DECK option.

Specify only one of the following operands DUMP, PARTDUMP and NODUMP in one OPTION statement.

DUMP
Dumps the registers, supervisor area, partition, the used part of the system GETVIS area, the SVA phase in error (if the error occurred in the SVA), and the phase load list. The dump is taken in the case of an abnormal program end (such as program check). (See also Notes 3 and 4).

PARTDUMP
Dumps selected areas of storage. For full information, refer to *VSE/Advanced Functions Service Aids.*

The dump is taken in the case of an abnormal program end.

NODUMP
Suppresses the DUMP or PARTDUMP option.

EDECK
The assembler punches all valid source macro definitions in edited format on SYSPCH. These macro definitions are framed with the proper CATALog and BKEND statements which are needed by the librarian to catalog the macro definitions as source members of type E library.

NOEDECK
Suppresses the EDECK option.

ERRS
The FORTRAN, DOS/VS COBOL, and PL/I compilers summarize all errors in the source program on SYSLST.

NOERRS
Suppresses the ERRS option.

JCANCEL
Indicates that the system should skip to End-of-Job (instead of waiting for operator intervention) if a job control error occurs.

NOJCANCEL
Suppresses the JCANCEL option, and is the system default. The system waits for operator intervention if a job control error occurs.

LINK
Indicates that the object module is to be link-edited. When the LINK option is used it must always precede an EXEC LNKEDT statement in

the input stream. Only a single phase can be link-edited. (See also Note 1.)

NOLINK
Suppresses the LINK option.

LIST
Language translators write the source module listing. The assembler also writes the hexadecimal object module listing, and the assembler and FORTRAN write a summary of all errors in the source program. All are written on SYSLST.

NOLIST
Suppresses the LIST option. This option overrides the printing of the external symbol dictionary, relocation list dictionary (RLD), and cross-reference (XREF/SXREF) lists.

LISTX
The COBOL compiler produces a PROCEDURE DIVISION map on SYSLST. The PL/I and FORTRAN compilers produce the object modules on SYSLST.

NOLISTX
Suppresses the LISTX option.

LOG
Lists columns 1-80 of all control statements and commands on SYSLST. Control statements and commands are not listed until a LOG option is encountered.

If LOG is the only option you want to specify, consider using the // LOG statement, which has the same effect and is shorter.

NOLOG
Suppresses the listing of all valid control statements and commands on SYSLST until a LOG option is encountered. If SYSLST is assigned, invalid statements and commands are listed.

If NOLOG is the only option you want to specify, consider using the // NOLOG statement, which has the same effect and is shorter.

LOGSRC
This operand takes effect only when the option LOG is already in effect. It causes job control statements which contain symbolic parameters to be printed twice: once in source form (as coded), once with substituted symbolic parameters (as processed by job control.)

When the options LOGSRC and LOG are in effect, all statements skipped during execution are written to SYSLST. The reason for skipping them is indicated by the following character strings in columns 82 to 98:

/. labelnam Skipped because of a GOTO statement.

********** Skipped because of an IF statement.

***/. $EOJ *** Skipped because the job was canceled.

NOLOGSRC
Suppresses the LOGSRC option. If the option LOG is in effect, job control statements are printed once, showing the substitution of any symbolic parameters.

There is no corresponding operand for the STDOPT statement. The system default is NOLOGSRC.

PARSTD
All disk, diskette, and tape label statements submitted after this point are written into the partition standard subarea of the system's label information area, in order to be available to all subsequent jobs in the current partition until another PARSTD option without operand or with = DELETE is submitted.

OPTION PARSTD remains in effect until one of the following occurs:

1. End of job or job step.
2. OPTION USRLABEL is specified.
3. OPTION STDLABEL is specified.

OPTION PARSTD followed by a /& clears the partition standard subarea (see Note 2 below).

PARSTD = ADD
All label information submitted after this option will be stored permanently into the partition standard subarea of the label information area without overwriting the information that already exists in that subarea.

PARSTD = DELETE
This option must be followed by the **filename(s)** of the DLBL statement(s) to be deleted from the partition standard subarea of the label information area. The last (or only) filename must be followed by /*. After the /* the option PARSTD = ADD becomes effective. PARSTD (or STDLABEL) = DELETE must be specified as the **last** option of the OPTION statement.

PARSTD = Fn
All label information submitted after this option will be stored permanently into the specified partitions standard subarea of the label information area. The option can be submitted only in the background, and the partition specified by Fn must be inactive.

RLD
The assembler writes the relocation list dictionary on SYSLST. This option is suppressed if NOLIST is specified.

NORLD
Suppresses the RLD option.

STDLABEL

All disk, diskette, and tape labels submitted after this point are written into the system standard subarea of the label information area, to be available to all subsequent jobs in all partitions until another STDLABEL option without operand or with =DELETE is submitted. STDLABEL is not accepted by job control operating in a foreground partition. OPTION STDLABEL remains in effect until one of the following occurs:

1. End of job or job step.
2. OPTION USRLABEL is specified.
3. OPTION PARSTD is specified.

OPTION STDLABEL followed by a /& clears the system standard subarea (see Note 2 below).

Note: If OPTION STDLABEL is submitted while other partitions are executing, an attempt to open a file using a standard label will result in an open failure.

STDLABEL=ADD

All label information submitted after this option will be stored permanently into the system standard subarea of the label information area without overwriting the information that already exists in that subarea. The label information is accessible by all partitions, but can only be submitted in the background partition.

STDLABEL=DELETE

This option must be followed by the filename(s) of the DLBL statement(s) to be deleted from the system standard subarea of the label information area. The last (or only) filename must be followed by /*. After the /* the option STDLABEL=ADD becomes effective.

STDLABEL=DELETE can be submitted only from the background. STDLABEL (or PARSTD) with =DELETE must be specified as the **last** option of the OPTION statement.

SUBLIB=DF

Directs the assembler and ESERV program to retrieve non-edited macros and copy-books from sublibrary members of type D instead of from sublibrary members of type A, and to retrieve edited macros from sublibrary members of type F instead of from sublibrary members of type E. IBM uses the sublibrary members of types D and F to distribute macros and copy source code for programs that are to be executed in a teleprocessing network control unit. The option remains in force until end-of-job or a // OPTION SUBLIB=AE statement.

SUBLIB=AE

Redirects the assembler and the ESERV program to retrieve non-edited macros and copy books from sublibrary members of type A and to retrieve edited macros from sublibrary members of type E.

SYM
: The COBOL compiler produces a DATA DIVISION map on SYSLST; the PL/I compiler produces the symbol table on SYSLST.

NOSYM
: Suppresses the SYM option.

SYSDUMP
: Indicates that dumps are to be written to the dump sublibrary which is active for the partition. If no dump sublibrary has been defined for the partition (by a LIBDEF DUMP command), the option SYSDUMP is ignored, and the NOSYSDUMP option comes into effect. The old form of this operand (SYSDMP) is accepted for compatibility reasons.

NOSYSDUMP
: Indicates that dumps are to be written on SYSLST. The old form of this operand (NOSYSDMP) is accepted for compatibility reasons.

SYSPARM='string'
: Specifies a value for the assembler system variable symbol &SYSPARM. &SYSPARM gets the value of the string, which is enclosed by quotes. The string can contain 0-8 EBCDIC characters. One internal quote must be represented by two quotes. (Job control removes one of them when setting the value.) The surrounding quotes are not included and the length of &SYSPARM is determined by the resulting string.

TERM
: Error messages are written on SYSLOG (applies only to compilers that support this function).

NOTERM
: Suppresses the TERM option.

USRLABEL
: All disk, diskette, and tape labels submitted after this point are written temporarily (for one job or job step) into the partition temporary sub-area of the label information area. Label information submitted with subsequent jobs or job steps overrides this specification.

Specify only one of the following operands XREF, SXREF and NOXREF in one OPTION statement.

XREF
: The assembler writes the symbol cross-reference list on SYSLST.

SXREF
: The assembler writes the symbol cross-reference list on SYSLST; printing of all unreferenced labels is suppressed.

NOXREF
: Suppresses the XREF or SXREF option.

NOFASTTR
Suppresses fast CCW translation for the current job. (Note that FASTTR is a system generation option only.)

48C
Specifies the 48-character set on SYSIPT (for PL/I).

60C
Specifies the 60-character set on SYSIPT (for PL/I).

Note 1: Any assignment for SYSLNK after the occurrence of the OPTION statement cancels the LINK and CATAL options. These two options are also canceled after each occurrence of an EXEC statement with a blank operand.

Note 2: By storing the label information for a disk file in the label information area, the system relates that file to the type of the device which is assigned to the pertinent logical unit when this file is processed for the first time. A later attempt to use this label information for the same file (and extent) on a different device type causes the system to cancel the job.

Note 3: If SYSLST is assigned to a 3211 printer, the indexing feature of the device must be used with care. Shifting the print line to the left or too far to the right causes characters to be left out from every printed line of the dump.

Note 4: If SYSLST is assigned to a 3800 Printing Subsystem, DUMP sets the 3800 to its system default for the character arrangement table, and restores the original status at the end of the dump.

OVEND (Override End)

The OVEND statement or command applies to cataloged procedures only. It is used to indicate that no more overriding statements will follow for the procedure specified in the preceding EXEC PROC=procname,OV statement.

Operation	Operands	Type
[//] OVEND	[comment]	JCS,JCC

The OVEND statement has no operand.

For the use of overriding statements and the rules that apply to temporary procedure modification, refer to the *VSE/Advanced Functions System Management Guide*.

PAUSE (Suspend Processing)

The PAUSE **statement** causes a pause immediately after processing this statement.

The PAUSE **command** causes a pause at the end of the current job step.

The PAUSE statement or command always appears on SYSLOG. If SYSLOG is assigned to a line printer, the PAUSE statement or command is ignored. At the time SYSLOG is unlocked for input, the operator can cause processing to be continued by pressing END/ENTER.

Operation	Operands	Type
[//] PAUSE	[any user comment]	JCS,JCC
PAUSE	[BG|Fn][,EOJ]	AR

BG or Fn
: Indicates the partition in which processing is to be interrupted.

EOJ
: Indicates that the interruption will occur at the end of the current **job**. In this case EOJ must be preceded by a partition identifier. If the EOJ operand is omitted, the interruption will occur at the end of the current **job step**.

PROC (Procedure)

The PROC command or statement, when used, is the first line of a cataloged procedure. It is required only when the procedure contains symbolic parameters to which you want to assign initial values. If you omit the PROC command or statement from a procedure, the system assumes a PROC statement without operands, that is, any symbolic parameters in the procedure are without a default value, and must be defined either in an EXEC PROC statement or in a SETPARM statement.

The operands of a PROC command or statement must not contain symbolic parameters. Continuation lines are accepted for the PROC statement.

Operation	Operands	Type
[//] PROC	[parname=[value]][,...]	JCS,JCC

`parname`
The name of the symbolic parameter (without a leading &) to which you want to assign the specified value.

`value`
The value you want to assign to the specified symbolic parameter.

For rules governing the format of symbolic parameters and their values, see "Symbolic Parameters" on page 3-9.

Note: The PROC statement is accepted only within a job.

PRTY (Query and Change Partition Priorities)

The AR PRTY command allows the operator to:

Display the priority sequence of the partitions in the system;

Change that sequence.

In both cases also the current status (if active) of the TP Balancing (TPBAL) function is displayed.

The JCC PRTY command can be used only in the BG during ASI (Automated System Initialization) to modify the priority sequence of the partitions in the system. It must be issued before the first // JOB statement.

Operation	Operands	Type
PRTY	none	AR
PRTY	partition,partition[,partition]...	AR,JCC
PRTY	partition=partition[=partition]...	AR,JCC

The AR PRTY command **without** an operand displays, on SYSLOG, the current priorities of all partitions. The first partition in the list has the lowest priority, the last one the highest.

The operands can be specified in two forms, as shown above, or in a mixed form, to provide for priority setting and partition balancing together, as indicated below.

`partition,partition,partition ...`
Specifies the desired sequence of processing priority. The first partition you specify receives the lowest priority, the last partition the highest priority. For example,

```
PRTY BG,F3,F1,F2
```

causes the system to give the lowest priority to the background partition and the highest priority to the F2 partition.

The number of partition identifiers in the command must coincide with the number of partitions in the system.

partition = partition = partition ...
Specifies that dynamic partition balancing is to be used for the partitions which you list with a separating equals sign(=). Partitions so specified are treated as an entity within which the supervisor checks processor usage at regular intervals and reassigns priorities such that the partition with the highest processor usage is given lowest priority.

Mixed format:

The command

```
PRTY BG,F1=F2=F3=F4,F5,F6
```

specifies highest priority for partitions 5 and 6, lowest priority for partition BG, and partition balancing for partitions F1 to F4.

Notes:

1. *If ACF/VTAM is used, the partition in which it is running should not be specified for partition balancing.*

2. *If teleprocessing balancing is active, the partition(s) that can be subject to performance degradation will be displayed on SYSLOG, regardless of whether operands are specified with the PRTY command or not. See also the TPBAL command.*

3. *If VSE/POWER is used, the POWER partition must have a higher priority than the POWER-controlled partitions.*

4. *You can specify only one group of partitions for dynamic partition balancing.*

PWR (Pass POWER Command)

The PWR job control statement makes it possible to pass the commands PRELEASE and PHOLD to POWER at any point in the job stream. The operand of the PWR statement is taken as a POWER command, and its syntax is checked by the POWER routine.

If the POWER processing routine is active, a message is sent to the operator.

Operation	Operands	Type
[//] PWR	{PRELEASE...|PHOLD...}	JCS,JCC

PWR
: Specifies that the rest of the statement is a POWER command.

PRELEASE | PHOLD
: Are the POWER commands which will be accepted. For their syntax requirements, see the applicable VSE/POWER publication.

Notes:

1. *The second operand of the POWER command must not be a single character (CLASS) or ALL.*

2. *Symbolic parameters cannot be used in this statement.*

RC (Request Communication)

The operator can use the RC command if he wants to enter an AR command when the attention routine is not available. In this case, entering the RC command forces the system to:

Terminate processing of any previous attention routine command and

Accept any new attention routine command from the console.

Operation	Operands	Type
RC	none	AR

The RC command has no operand.

REPLID (Query Reply-IDs)

The REPLID command displays the reply-IDs (and partition indicators) of all messages for which replies are still pending.

For information on how to use this command, refer to the section on operator to system communication in the manual *VSE/Advanced Functions Operation.*

Operation	Operands	Type
REPLID	none	AR

The REPLID command has no operand.

RESERV (Reserve Device for VSAM)

The RESERV command reserves a device for VSAM space management usage. This means that the device cannot be assigned any more in the system. Also, a DVCDN command for the device will be rejected. The reserved status can be reset only by a FREE command.

The command may be issued for all disk devices on the system.

Operation	Operands	Type
RESERV	cuu	AR

cuu
: Indicates the channel and unit number of the device to be reserved.

RESET (Reset ASSGNs and LIBDEFs to Permanent Values)

The RESET command or statement resets temporary sublibrary definitions (LIBDEFs) and I/O assignments to their permanent values in the partition in which RESET was submitted. For information on temporary and permanent assignments and sublibrary definitions, refer to the ASSGN and LIBDEF statements.

When the physical device affected by RESET is a magnetic tape drive, the current mode set in the PUB table is set to the standard mode set for the device. The standard mode set is established during IPL and may be modified by a permanent ASSGN with a mode operand.

Operation	Operands	Type
[//] RESET	{SYS|PROG|ALL|SYSxxx}	JCC,JCS

SYS
: Resets all system logical unit assignments and library search chain definitions to their permanent values.

PROG
: Resets all programmer logical units to their permanent assignments.

ALL
: Resets all logical unit assignments and library chain definitions to their permanent values.

SYSxxx
: Resets the specified logical unit to its permanent assignment. SYSIN or SYSOUT cannot be specified.

ROD (Record on Demand)

The ROD command records all statistical data record counters for all non-telecommunication devices on the recorder file on SYSREC. The buffer containing the last console messages is written to the hard copy file. The command must not be issued until all jobs in the partitions have finished executing.

Operation	Operands	Type
ROD	none	JCC

The ROD command has no operand.

RSTRT (Restart Checkpointed Program)

The RSTRT statement is available for checkpointed programs. A programmer can use the CHKPT macro instruction in his program to write checkpoint records. The maximum number of checkpoints that can be taken is decimal 9999. The checkpointed information includes the registers, tape-positioning information, a dump of the program, and a restart address.

The restart facility allows the operator to continue execution of an interrupted job at a point other than the beginning. To do so, submit a RSTRT command followed by the job control statements originally used for the job.

Operation	Operands	Type
// RSTRT	SYSxxx,nnnn[,filename]	JCS

SYSxxx
: Logical unit name of the device on which the checkpoint file is stored. This unit must have been previously assigned.

nnnn
: Identification of the checkpoint record to be used for restarting. It corresponds to the checkpoint identification used when the checkpoint was taken. The serial number is supplied by the checkpoint routine and printed on SYSLOG when the checkpoint was taken.

filename
: The name of the disk checkpoint file to be used for restarting. It must be identical to the filename of the DTFPH to describe the disk checkpoint file and the fifth parameter of the CHKPT macro instruction. This operand applies only when specifying a disk as the checkpoint file.

See *VSE/Advanced Functions Application Programming: Macro Reference* for further details on the CHKPT macro instruction.

When a checkpoint is taken, the completed checkpoint is noted on SYSLOG. Restarting can be done from any checkpoint record, not just the last. The jobname specified in the JOB statement must be identical to the jobname used when the checkpoint was taken. The proper I/O device assignments must precede the RSTRT control statement.

Assignment of input/output devices to logical unit names may vary from the initial assignment. Assignments are made for restarting jobs in the same manner as assignments are made for normal jobs. System mode (370 or ECPS:VSE), run mode (virtual or real), and storage allocations and boundaries for the partition must be the same for the restart run as for the original, checkpointed run.

SET (Set Program Control Values)

The SET command sets controls for the execution of programs. Except for SET UPSI, the SET command should precede the JOB statement of the first job for which the specified control value is to be effective.

Operation	Operands	Type
SET	[DATE=date][,HC={YES|NO|CREATE}][,LINECT=count] [,RCLST=number][,RCPCH=number][RF={YES|CREATE}] [,SDL][,UPSI=config]	JCC

DATE=date

Sets the system date permanently to the specified value. This subsequently resets the JOB date when a new job is run.

date can have the formats:

mm/dd/yy or dd/mm/yy

mm specifies the month; **dd** specifies the day; **yy** specifies the year. The first format is the system default; this default can be changed to the second format using in the STDOPT command.

The DATE parameter may be specified only if the clock is not operational.

HC=YES|NO|CREATE

Defines the status of the hard-copy file (IJSYSCN) on SYSREC. It may be specified only after IPL and before the first JOB statement.

YES

Indicates that a hard-copy file exists in the system, and that it is to be opened. YES is the default.

NO

Indicates that no recording is to be performed on the hard-copy file. Can be specified only if a console printer is attached.

CREATE

Instructs the system to create a hard-copy file; the file is created and opened as soon as the first JOB statement is read.

LINECT=count

Sets the standard number of lines to be printed on each page of SYSLST. Specify an integer between 30 and 99.

RCLST=number

Specifies the number of records remaining to be written on a SYSLST extent on disk before a warning is issued to the operator that the extent is nearly full. Specify any decimal number from 100 through 65535.

If no value is given, the system sets RCLST equal to 1000. RCLST is ignored if SYSLST is assigned to a diskette.

Note: This warning is issued only between job steps. If the extent limits are exceeded before the job step ends, this job is terminated.

RCPCH=number

Specifies the minimum number of records remaining to be written on a SYSPCH extent on disk before a warning is issued to the operator that the extent is nearly full. It may be any decimal number from 100 through 65535.

If no value is given, the system sets RCPCH equal to 1000. RCPCH is ignored if SYSPCH is assigned to a diskette.

Note: This warning is issued only between jobs. If the extent limits are exceeded before the job ends, this job is terminated.

RF=YES | CREATE

Defines the status of the recorder file (IJSYSRC) on SYSREC. It may be specified only after IPL and before the first JOB statement.

YES

Indicates that an active recorder file exists. The system opens this file when the first JOB statement is encountered.

CREATE

Instructs the system to create a recorder file when the first JOB statement is encountered.

SDL

This operand indicates that phase names are to be added to the system directory list and, optionally, that phases are to be loaded into the SVA, including phases that are to be moved from the SVA to the logical transient area in order to be executed.

For performance reasons, you may have to create SDL entries not only for your own phases, but also for frequently used IBM-supplied phases. For printing a list of the phases loaded automatically into the SVA during IPL, refer to *VSE/Advanced Functions Planning and Installation.*

SET SDL may be issued at any time after IPL, but it must always be issued from the background partition. If several SET SDL commands are entered, the new phases specified are added to those already in the SDL. An existing entry is replaced only if a following SET SDL command specifies the same phase name as an earlier command.

To build the SDL, job control reads the names of the phases which are to go into the SDL. The system searches for the requested phase in the active LIBDEF PHASE,SEARCH chain, if any, and in the system sublibrary IJSYSRS.SYSLIB. If it does not find this phase, it creates a dummy entry. This is filled when a phase is cataloged with that name.

If you want to create an SDL entry for a non-SVA-eligible phase, this phase must be in IJSYSRS.SYSLIB.

If the SET SDL command is entered from the console, the operator is prompted for the phase names. If the command is entered from SYSRDR, the phase names must be on SYSIPT. This implies that, if the SET SDL command and the phase names are in a JCL procedure, this procedure must be cataloged with the operand DATA=YES in the librarian CATALOG command.

The phase names must be specified in one of the following formats:

1. phasename[,SVA]

   The operand SVA takes effect only if the named phase is SVA-eligible. It indicates that the phase itself is to be placed in the shared virtual area, in addition to having an entry inserted in the SDL. For phases in private sublibraries, always specify SVA.

2. phasename,MOVE

   This format is valid only for B- or C-transient phases. MOVE indicates that the specified phase is to be loaded into the SVA in order to be moved from there to the respective transient area when the phase is to be executed. A phase specified with MOVE must be self-relocatable.

After the last phasename, you must enter a /* statement to indicate the end of the input.

The maximum number of SDL entries is specified during IPL with the SVA command. If the maximum number is exceeded, a message is issued and all following statements are flushed until a /* or /& statement is encountered.

The SET SDL command, phase names, and /* can be placed in an ASI JCL procedure. For information on how to build an ASI (automatic system initialization) procedure, see *VSE/Advanced Functions System Management Guide.*

Note: As the SVA is automatically formatted during IPL (which also means that the SDL is created each time), the operands SVA(nK,mK) and SDL=CREATE, which previously existed, are no longer required. For compatibility reasons, the system will simply ignore these operands if they are specified.

Refer to *VSE/Advanced Functions Planning and Installation* for details on the IBM supplied SDL load procedure.

`UPSI=config`

Sets the bit configuration of the UPSI byte in the communications region. Specify one to eight characters, either 0, 1, or X. Bit positions containing

0 are set to 0; positions containing 1 are set to 1; positions containing X are unchanged. Unspecified rightmost positions are assumed to be X.

The UPSI byte is reset to zero by a JOB or /& statement.

SETDF (Set 3800 Printer Defaults)

The SETDF command allows the operator to set and/or reset default values for the IBM 3800 Printing Subsystem or to display the default values. The command is valid only for a 3800. The following values can be defaulted:

- Bursting or continuous forms stacking;
- One character arrangement table;
- The forms control buffer name;
- The forms overlay name;
- The paper forms identifier;
- The copy modification name;
- The setting of all hardware defaults with one command.

For further information on the 3800 and on the various ways that you can use its defaults, see the *DOS/VSE IBM 3800 Printing Subsystem Programmer's Guide.*

The length of the SETDF operator command is limited to one line of 71 characters. However, defaults are retained from one command to another (that is, if CHARS is set by one command and the next command sets FCB, then both are now set). Coding 'keyword=,' for an individual parameter makes the hardware default effective only for the specified keyword.

Issuing the SETDF command does not change job or program originated settings of the 3800. Instead, the parameters are saved such that when a user specifies DFLT=Y or keyword=* in a SETPRT job control statement or SETPRT macro, the SETPRT routine sets the predefined defaults.

Operation	Operands	Type
SETDF	{3800|cuu}[,BURST=[Y|N]][,CHARS=[table-name]] [,FCB=[fcb-name]] [,FLASH=[overlay-name]] [,FORMS=[forms-name]] [,LIST] [,MODIFY=[copymod-name]] [,RESET]	AR

3800
: Specifies that all 3800 printers will be set with the specified default values of SETDF, or (if LIST is specified) the defaults for all of these printers will be displayed.

cuu
: Specifies the channel and unit number of the 3800 whose default values are to be set or displayed by SETDF.

BURST=,
: No change in the threading of the forms is requested.

Y. Specifies that the printed output is to be burst into separate sheets with the edges trimmed.

N. Specifies that the printed output is to be in continuous fanfold mode. If BURST has not been specified since the system was initialized, BURST = N is assumed.

CHARS=,
The default for the character arrangement table is reset to the hardware default Gothic-10 folded table.

table-name specifies the 1- to 4-character suffix of the name of the default character arrangement table. Only the first character arrangement table can be defaulted; multiple table names are not allowed.

FCB=,
The default for the forms control buffer is reset to the hardware default FCB.

fcb-name specifies the 1- to 4-character suffix of the name of the default FCB.

FLASH=,
No flashing is done.

overlay-name specifies the 1- to 4-character name of the forms overlay frame to be used as the default.

FORMS=,
The operator is requested to load the forms named STANDARD when the default is needed.

forms name specifies the 1- to 4-character name of the forms to be used.

LIST
Specifies that the established default settings are to be displayed at the operator console. If blanks are shown for the value of a displayed keyword, this indicates the hardware default (with the exception of the BURST keyword. The default for BURST is indicated by an N.)

MODIFY=,
No copy modification is done.

copymod-name specifies the 1- to 4-character suffix of the name of the modification phase to be loaded from the library into the 3800.

RESET

Sets all keywords to the hardware defaults, which are:

- BURST = N
- A Gothic 10-pitch folded character arrangement table
- A 6-lines-per-inch FCB with channel-1 code on the first printable line, no other channel codes, and the forms length determined by the paper loaded;
- No forms overlay flashing
- No specific forms requested
- No copy modification done.

SETMOD (Set 8809 Tape Mode)

The SETMOD command, which is valid for the 8809 Magnetic Tape Unit only, can be used to adjust the tape speed to the actual I/O traffic at any time during processing by switching from 'Streaming' mode to 'Start-Stop' mode and vice versa.

Operation	Operands	Type
SETMOD	cuu[,mode]	AR

cuu
: Specifies the channel and unit number of the 8809.

mode
: Can be one of the following:

  90 (or HL) = High Speed, Long Gap (Streaming)
  30 (or HS) = High Speed, Short Gap (Streaming)
  50 (or LL) = Low Speed, Long Gap (Start-Stop)
  60 (or LS) = Low Speed, Short Gap (Start-Stop)

  If the mode operand is omitted, the default mode setting of 60 (or LS) is assumed.

SETPARM (Set Symbolic Parameter)

The SETPARM statement enables you to define a symbolic parameter and/or assign a value to it. This value can then be tested in an IF statement, or used by job control in subsequent statements.

If, for example, you code PARM1 = SYS001 for pname = value, the symbolic parameter &PARM1 in a subsequent statement will be substituted with the value 'SYS001' by job control.

Continuation lines are allowed for this statement.

Operation	Operands	Type
[//] SETPARM	pname=[{value|$RC\|$MRC}][,...]	JCC,JCS

pname
: Is the name of the symbolic parameter which you want to define, or to which a value is to be assigned. It may consist of 1 to 7 alphameric characters, of which the first must be alphabetic.

value
: Specifies a character string of up to 50 characters. If the string contains national or special characters, it must be enclosed in quotes. For detailed information on the format of strings assigned to parameters, see "Symbolic Parameters" on page 3-9. You can specify a null string as the value of a symbolic parameter, either by omitting value, $RC and $MRC, or by coding two quotes ('') in place of the character string.

$RC
: Specifies the return code of the last job step which was executed. It is assigned to the parameter as a string of four characters.

$MRC
: Specifies the maximum return code of all preceding job steps. It is assigned to the parameter as a string of four characters.

Note: The SETPARM statement is accepted only within a job.

SETPRT (Set 3800 Printer Values for Job)

The SETPRT job control statement or command sets user-specified control values for the IBM 3800 Printing Subsystem. These values are reset at the end of the current job to the installation's default values as specified in the SETDF attention routine command, or to the hardware defaults if SETDF has not been issued. For more information on the 3800 and its use, see the *DOS/VSE IBM 3800 Printing Subsystem Programmer's Guide.*

At least one of the optional operands must be specified. Continuation lines are accepted for the SETPRT statement.

Operation	Operands	Type
[//] SETPRT	SYSxxx[,BURST={N|Y|*}] [,CHARS={table-name|*|(table name,...)}] [,COPIES=number][,DCHK={B|U}] [,DEBUG={NORM|TERM|DUMP|TRAC}] [,DFLT={N|Y}] [,FCB={fcb-name|*(fcb-name,V)|(*,V)}] [,FLASH={overlay-name[,count]|(,count)| (*[,count]} [,FORMS={forms-name|*}][INIT={N|Y}] [,MODIFY={copymod-name|*| (copymod-name, table name)}] [,SEP=O][,TRC={N|Y}]	JCC,JCS

SYSxxx
: Logical unit identifier for the 3800 printer to be set up. This operand is always required. SYSxxx can be SYSLST, SYSLOG, or SYSnnn. The logical unit must have been previously assigned to a 3800.

BURST=
: If the operand is omitted, no change to the threading is requested.

  Y specifies that the operator should thread the forms through the Burst-Trimmer-Stacker.

  N specifies that the operator should thread the forms to the continuous forms stacker.

  * specifies that the system default BURST setting is requested.

CHARS=

If the operand is omitted, the character arrangement table is not changed unless INIT = Y is coded.

table-name specifies the 1- to 4-character suffix of the name of the character arrangement table.

(table-name,...) specifies up to four names, separated by commas and enclosed in parentheses. However, see Note under the MODIFY operand. Embedded null values, such as CHARS = (AA,,BB) or CHARS = (,AA), are not allowed. For the names of the IBM-supplied character arrangement tables, see the *DOS/VSE IBM 3800 Printing Subsystem Programmer's Guide*.

* specifies that the system default character arrangement table is requested. If the operator has not specified a default for CHARS, the hardware default Gothic-10 folded table is used.

COPIES=

If the operand is omitted, the number of copies is not changed unless INIT = Y is coded.

n specifies the number of copies of each page to be reproduced before printing the next page. It can be a value from 1 to 255.

DCHK=

If the operand is omitted, data checks are blocked.

B specifies that data checks are to be blocked. This means that unprintable characters in the data transmitted to the 3800 are printed as blanks.

U specifies that data checks are allowed.

DEBUG=

NORM sets a return code in register 15 and returns to the caller on any exit from the SETPRT routines. This is in effect if DEBUG is omitted from all preceding SETPRTs in the job.

TERM sets a return code in register 15 and cancels the activity for return codes higher than 4. For return codes 0 and 4, TERM has the same effect as NORM.

DUMP sets a return code in register 15 and cancels the job with a dump, for a return code higher than 4.

TRAC dynamically traces, on SYSLST, the activity of the SETPRT routines and then cancels the job with a dump if the SETPRT return code is greater than 4. Tracing requires 12K of GETVIS space.

DFLT=

N is the default specification for this keyword and does not establish 3800 default setup. **Y** specifies that the printer is to be set with the defaults that were specified by the operator in the SETDF command. It is equivalent

to coding * for each of the operands BURST, CHARS, FCB, FLASH, FORMS, and MODIFY that are not specified.

FCB=

If the FCB operand is omitted, the FCB is not changed unless INIT=Y is coded. **fcb-name** specifies the 1- to 4-character suffix of the name of the FCB. The length of the form defined by FCB must match the length of the form loaded in the 3800, as specified with the FORMS operand.

V requests FCB verification. The FCB contents are formatted and printed on the 3800. Data checks are blocked, and translate table zero is used for printing the FCB verification page.

* specifies that the system default FCB is requested. If the operator has not specified a default FCB, the hardware default FCB is 6 lines per inch with a channel-1 code defined on the first printable line, and the length set equal to that of the form currently loaded.

FLASH=

overlay-name is the 1- to 4-character name of the forms overlay frame that the operator will be requested to insert in the 3800.

count is the number (from 0 to 255) of copies to be flashed with the overlay, beginning with the first copy of the first transmission. If 0 is specified, the specified forms overlay frame is mounted or remains mounted but is not flashed. A specification of FLASH=(,100), for example means to flash the current forms overlay frame for 100 copies.

If no count is specified, all copies are flashed.

* requests the system default forms overlay. If the operator has not specified a default, no flashing occurs.

FORMS=

forms-name is the 1- to 4-character forms identifier. If the specified forms are not already loaded, a message to the operator requests the specified forms. If the new form has a length different from the previous form and a new FCB is not specified, the 3800 loads the hardware default FCB. This can cause erroneous results later. To avoid this problem, specify a new FCB when loading forms of a new length.

* requests the system default forms. If the operator has not specified a FORMS default, form STANDARD is requested.

INIT=

Y specifies that the printer be reset to hardware defaults of a 6-lines-per-inch FCB with channel-1 code in the first printable line, a Gothic-10 folded character arrangement table, one copy, and no flashing of forms overlays. Copy modification is cleared, and burster threading, forms, and blocking or unblocking of data checks are not changed. If TRC=Y is not also coded, then lines written to printer files opened after this SETPRT should not contain table reference characters unless such an

inclusion is specified in the DTFxx macro. The TRC indicators for an open printer file are not changed. (Any of these actions can be overridden with other keywords.)

N is the default and does not reset the 3800 to hardware defaults.

MODIFY=

If the operand is omitted, then the currently-loaded copy modification phase is used, unless INIT = Y is also coded. **copymod-name** specifies the 1- to 4-character suffix of the name of the copy modification phase that was assigned when the phase was built.

table-name specifies the 1- to 4-character name of the character arrangement table to be used when the 3800 prints the copy modification text. This character arrangement table need not be one of those specified with the CHARS operand. However, see Note below. If table-name is omitted, the first character arrangement table specified with the CHARS operand (or the default, if none is specified) is used.

* requests the system default copy modification. If the operator has not specified a MODIFY default, any existing copy modification is eliminated.

SEP=

Omission of the SEP operand indicates that no data set separation is required. **O** indicates that, if the burster-trimmer-stacker is being used, the 3800 should offset-stack the pages that follow from the pages that were previously transmitted. If the continuous forms stacker is being used, the 3800 changes the marking on the perforation edge from one line to two lines or vice versa.

TRC=

N indicates that, for any DTFPR or DTFDI operand after this SETPRT, data lines do not contain table reference characters unless specified in the DTF macro. The table reference character will not be prefixed to each data line when presented to the access method.

Y indicates that the first character of each output data line (after the optional print control character) given to the access method is a table reference character. This applies only to PUT macros with DTFPR or DTFDI.

Note: The total number of character sets referenced by character arrangement tables in both the CHARS and MODIFY operands cannot exceed the number of writable character generation modules available on the 3800 (either two or four). If a character set is referenced by multiple character arrangement tables and graphic character modification is not used, then only one copy of that character set is loaded into the 3800. If a character set is referenced by two character arrangement tables and one is modified by graphic character modification and the other is not, then two character sets are loaded.

Example

The following example shows the use of the SETPRT job control statement to set up the 3800 Printing Subsystem with the physical unit address 118:

```
// JOB D63SETP              SET UP 3800 Printer
// ASSGN SYS010,118         ASSIGN SYS010 TO 3800 PRINTER
// SETPRT SYS010,BURST=Y,DCHK=B,SEP=0,TRC=Y,                C
   FORMS=X,FLASH=(TEST,2),FCB=(STD6,V),                     C
   CHAR=(X,XX,XXX,GF12),MODIFY=(CMO1,FM12),COPIES=4
/&
```

The operands of the SETPRT statement specify:

- BURST = Y specifies that the operator will be asked to thread the forms through the Burster-Trimmer-Stacker.
- DCHK = B specifies that data checks are to be blocked.
- SEP = 0 specifies that the burst pages from this job are to be offset in the stacker from those of the previous job.
- TRC = Y specifies that the first character of each output data line (following the optional print control character) is a table reference character.
- FORMS = X specifies that the forms named X are to be used for printing this job.
- FLASH = (TEST,2) specifies that the first 2 copies of each page printed are to be flashed with the forms overlay named TEST.
- FCB = (STD6,V) specifies that the forms control buffer phase named STD6 is to be used, and that the FCB contents are to be formatted and printed for verification by the operator.
- CHARS = (X,XX,XXX,GF12) specifies the names of the four character arrangement tables that are to be loaded into the 3800.
- MODIFY = (CMO1,FM12) specifies that the FM12 character arrangement table, which uses Format 12-pitch characters, is to be used to print the copy modification named CMO1.
- COPIES = 4 specifies that 4 copies of each page of the file are to be printed in a group before printing 4 copies of the next page.

SIZE (Program Size)

The SIZE command is used to specify the amount of contiguous virtual storage in a partition which is reserved for program execution. The rest of the partition is available as partition GETVIS area.

The SIZE command has a function similar to the SIZE operand of the EXEC statement. The difference is that:

- The SIZE **command** makes a permanent change which lasts until another SIZE command is issued, or until the next IPL.
- The SIZE **operand** of the EXEC statement is effective only for the current job step.

The SIZE operand of the EXEC statement is still effective for its own job step after a SIZE command has been issued.

If a program running in real mode needs GETVIS space, the SIZE operand of the EXEC statement has to be specified. The SIZE command does not provide GETVIS space for a program running in real mode.

The SIZE command is not accepted for an active partition which is using its GETVIS space.

Operation	Operands	Type
SIZE	partition={nK|mM}[,partition={nK|mM}]...	JCC,AR

`partition`
Indicates the partition (BG, F1, F2, ...) for which storage is to be reserved.

`nK|mM`
Specifies the amount of storage to be reserved for program execution in kilobytes (nK) or megabytes (mM). The remainder of the partition is available as partition GETVIS area. **n** should be a multiple of 4. If not, the system rounds it up to the next higher multiple of 4. The minimum value is 80K; the maximum permissible value is the partition size (as specified by ALLOC) minus the minimum partition GETVIS area, which is 48K.

START (Start or Continue Processing)

The AR START command activates or continues processing in the specified partition. The function of the START command is exactly the same as that of the BATCH command.

The JCC START command can only be used to start a partition which has not yet completed its ASI (Automated System Initialization) job control procedure.

Operation	Operands	Type
START	[BG|Fn]	AR
START	Fn	JCC

BG
: Indicates that the background partition is to be reactivated.

Fn
: Indicates that the specified foreground partition is to be activated, or restarted after having been stopped by a STOP command.

If the operand is omitted in the AR command, BG is assumed.

STDOPT (Standard JC Options)

The STDOPT command or statement sets or resets the permanent job control options which were established at system initialization (system defaults). The permanent options established are identical with the default values of the STDOPT command. If no STDOPT command is given, all the default values are valid. The STDOPT command can be given only in the background partition, but the values specified apply to **all** partitions.

If an option is reset, its new value becomes effective in a partition after the next /& or JOB statement is issued in that partition. (Exceptions: LINES becomes effective immediately, DATE can become effective earlier in a partition if a GETIME macro is issued in that partition.)

An option specified with STDOPT can be **temporarily** overridden in one partition by the OPTION statement. (Exception: LINES can only be overridden by the SET LINECT command; DATE cannot be overridden.)

Operation	Operands	Type
[//] STDOPT	option[,option]...	JCC,JCS

The options, which can appear in any order, are as follows (the first specification, which is printed bold, is always the system default value):

ACANCEL = NO|YES
: Specifies whether job control is to cancel jobs automatically (ACANCEL = YES) or to wait for operator intervention (ACANCEL = NO) after an unsuccessful attempt to assign a device. (Note that the LOG command overrides an ACANCEL = YES specification.)

ALIGN = YES|NO
: Specifies whether the assembler is to align data on halfword or fullword boundaries, according to the type of instruction used.

CHARSET = 48C|60C
: Specifies either the 48- or 60-character set for PL/I translator input on SYSIPT.

DATE = MDY|DMY
: Specifies the format of the date:

```
MDY=month/day/year
DMY=day/month/year.
```

DECK = YES|NO
: Specifies whether or not language translators are to produce object modules on SYSPCH.

DUMP = YES|NO|PART
Specifies whether or not a dump of the registers and virtual storage is to be taken in the case of an abnormal program end. PART specifies that a dump of the major supervisor control blocks and the virtual storage of the partition is to be taken.

EDECK = NO|YES
Specifies if the assembler is to create and punch edited macros on SYSPCH.

ERRS = YES|NO
Specifies whether or not language translators are to summarize all errors in source programs on SYSLST. Assembler and PL/I always assume ERRS = YES.

FASTTR = NO
Specifies that fast CCW translation is to be switched off for all partitions. This operand overrides the FASTTR = YES supervisor option, which may have been specified at system generation.

There is no default value for this operand, and you **cannot** specify YES.

Note that, when you have submitted the STDOPT command or statement with the FASTTR = NO operand, only a new IPL can re-activate fast CCW translation. If you want to suppress fast translation temporarily, use the OPTION statement with the operand NOFASTTR (see the description of the OPTION statement).

HCTRAN = YES|NO
Specifies whether the output from PRINTLOG and LISTLOG is to be translated to all upper case (YES) or in mixed upper and lower case. Default is YES.

JCANCEL = NO|YES
Specifies whether or not the system should terminate the job abnormally when a job control error occurs (JCANCEL = YES), or wait for operator intervention (JCANCEL = NO).

LINES = 56|nn
Specifies the number of lines per page on SYSLST. The minimum is 30, the maximum is 99. (If job control is running in another partition at the same time, the new value becomes effective in that partition when the next page is started.)

LIST = YES|NO
Specifies whether or not language translators are to write source module listings and diagnostics to SYSLST.

LISTX = NO|YES
Specifies whether or not language translators are to write hexadecimal object module listings on SYSLST.

LOG = YES|NO
Specifies whether or not all job control statements are to be listed on SYSLST. Invalid statements and commands will be listed on SYSLST in any case if it is assigned.

RLD = NO|YES
Specifies whether or not the relocation dictionary information is to be printed.

SXREF = NO|YES
Specifies whether the assembler is to print short cross-reference lists on SYSLST. The printing of unreferenced labels is suppressed. Do not specify SXREF together with XREF.

SYM = NO|YES
SYM = YES specifies that the PL/I compiler is to produce a symbol and offset table listing on SYSLST, or that the COBOL compiler is to produce a data division glossary.

SYSDUMP = NO|YES
YES indicates that dumps are to be written to the dump sublibrary which has been defined for the partition by a LIBDEF DUMP command. If you specify SYSDUMP = YES when no dump sublibrary has been defined for the partition, the system ignores the specification and uses the SYSDUMP = NO option. SYSDUMP = NO specifies that dumps are to be written to SYSLST. For compatibility reasons, the keyword may be entered as SYSDMP.

TERM = NO|YES
Specifies whether messages from a compiler are to be displayed on SYSLOG.

XREF = YES|NO
XREF = YES specifies that the assembler is to write symbolic cross-reference lists on SYSLST, or that American National Standard COBOL is to produce a cross-reference listing.

Do not specify SXREF together with XREF. If you specify XREF = YES|NO the SXREF operand defaults to NO.

STOP (Stop Processing)

The STOP command indicates that there are no more jobs to be executed in the partition in which the command is given.

Operation	Operands	Type
STOP	none	JCC

The STOP command has no operand.

This command removes the partition from the system's task selection mechanism, but the partition remains active. Job control remains in the partition and can be restarted by the START or BATCH attention routine command.

TLBL (Tape Label Information)

The TLBL statement contains file label information for the checking and writing of tape labels. The TLBL statement may be used with both EBCDIC and ASCII files. For more information about tape labels, refer to *VSE/Advanced Functions Data Management Concepts*.

Operation	Operands	Type
// TLBL	filename,['file-id'],[date], [file-serial-number], [volume-sequence-number], [file-sequence-number], [generation-number], [version-number] [,DISP=NEW|OLD|MOD]	JCS (for EBCDIC files)
// TLBL	filename,['file-id'],[date], [set-identifier], [file-section-number], [file-sequence-number], [generation-number], [version-number] [,DISP=NEW|OLD|MOD]	JCS (for ASCII files)

`filename`
This can be from one to seven alphameric characters, the first of which must be alphabetic. This unique filename is identical to the name of the program DTF that identifies the file.

`'file-id'`
One to seventeen alphameric characters, contained within quotes, indicating the unique name associated with the file on the volume. On output files, if this operand is omitted, the name specified for *filename* is used. On input files, if the operand is omitted, no checking of file identifiers will be done.

`date`
Output files:

- A retention period in days written as a decimal number (0-9999), or;
- The absolute expiration date of the file as a Julian date in the format:

  `19yy/ddd or 20yy/ddd`

  (yy = last two digits of year, ddd = day of the year.)

  The format yy/ddd is also accepted. For example, 87/032 is interpreted as 1987/032.

If the operand is omitted, a 0-day retention period is assumed. The current system date is always used as the creation date for output files.

Input files: The creation date of the file can be specified as a Julian date in the format:

```
19yy/ddd or
20yy/ddd or
yy/ddd
```

(yy = 00-99, ddd = 1-365). If this date does not match the actual creation date in the label of the tape which is being accessed, processing is interrupted and the system issues a message. If the creation date is omitted, no checking is done for input files.

`file-serial-number (EBCDIC)`
`set identifier (ASCII)`

One to six alphameric characters indicating the volume serial number of the first (or only) reel of the file. For input and output tapes, specify the six-digit volume serial number given to the tape reel when it was initialized.

The operand may be omitted. In this case, no checking is done.

`volume-sequence-number (EBCDIC)`
`file-section-number (ASCII)`

A one to four-digit decimal number specifying the volume of a multi-volume file at which you wish to start processing. If this operand is omitted, 0001 is used.

`file-sequence-number`
A one to four-digit decimal number specifying the file of a multi-file volume at which you wish to start processing. If the operand is omitted, 0001 is used.

`generation-number`
A one to four-digit number specifying the generation number of the file to be processed. If the operand is omitted on output, the system inserts blanks in the appropriate label field. If it is omitted on input, the generation number on the file is not checked by the system.

`version-number`
A one or two-digit decimal number specifying the version number of the file to be processed. The version number is an extension of the generation number, and the same rules govern its use.

`DISP=NEW|OLD|MOD`
This operand specifies whether a new output file is to be created or an existing file extended. It is meaningful only if:

The file is to be written by an assembler program assembled under VSE/Advanced Functions Release 1, or a later release, and

The file is defined in the program using the DTFMT macro with the parameters TYPEFLE=OUTPUT and FILABL=STD.

The specifications have the following meanings:

NEW -
Specifies that the file is to be created. This is assumed if the DISP operand is omitted.

OLD -
Specifies that the file already exists and is to be extended. The tape is positioned behind the last record of the existing file. If the file does not already exist, an error situation occurs.

MOD -
Specifies conditional extension or creation of the file. If the file-id in the TLBL statement matches the file-id in the HDR1 label on the tape, DISP=OLD is assumed, and the file is **extended**. Otherwise DISP=NEW is assumed, and the file is **created**.

For information on the use of the TLBL statement, see *VSE/Advanced Functions Data Management Concepts*.

TPBAL (Telecommunication Balancing)

The TPBAL command allows the operator to change the status of the TP balancing function. Processing is delayed in the specified number of partitions of the lowest priority.

Without an operand, the TPBAL command displays, on the device assigned to SYSLOG, what partitions are currently being affected by the function.

This command has no effect if the supervisor has been loaded with MODE = VM.

Operation	Operands	Type
TPBAL	{n}	AR

n

specifies the number of partitions in which processing can be delayed. The system responds by displaying the partitions that are affected under the new status.

A specification of zero puts the function out of effect.

The maximum number that can be specified is the number of partitions in the system minus one.

UCS (Load Universal Character-Set Buffer)

The UCS command causes the 240-character universal character set contained in the phase specified by phasename to be loaded for the 1403U printer. The 240 EBCDIC characters correspond to the 240 print positions on 1403U trains.

It is the user's responsibility to assemble, link-edit, and catalog his UCS buffer phases, and to mount the new chain or train before the UCS command is executed. The UCS command is not logged on SYSLST.

For further details of phase names, UCB load formats, etc., see the description of the SYSBUFLD program later in this manual.

Operation	Operands	Type
UCS	SYSxxx,phasename[,FOLD][,BLOCK][,NULMSG]	JCC

SYSxxx
: The logical unit assigned to the printer, which must be an IBM 1403 Printer with the UCS feature.

phasename
: The name of the phase containing the 240 EBCDIC characters to be loaded, followed by an 80-character verification message.

FOLD
: Signifies that the buffer is to be loaded in such a way as to print lower case bit configurations as upper case characters.

BLOCK
: permits any code not represented in the UCS buffer to print as a blank without causing a data check stop.

NULMSG
: Signifies that the 80-character verification message is not to be printed after the buffer is loaded. If NULMSG is not specified after the UCS buffer has been loaded, the program skips to channel 1, prints 80 characters in the phase specified by the operand phasename, and again skips to channel 1. This is to identify the phase, if the phasename is incorporated in the verification message. If a chain/train can be identified by a unique character, this character may be included in the verification message to verify that the mounted chain or train is compatible with the UCS buffer contents.

UNBATCH (Deactivate Foreground Partition)

The UNBATCH command terminates foreground processing and releases the partition (making it inactive). It resets all assignments of the partition to UA, except those for SYSRES, SYSREC and SYSCAT. All temporary and permanent library definitions (LIBDEFs) are dropped. Use the HOLD command if assignments are not to be reset.

Operation	Operands	Type
UNBATCH	none	JCC

The UNBATCH command has no operand.

Following the UNBATCH command, the attention routine accepts BATCH or START commands for the affected partition.

Restrictions:

UNBATCH is accepted only:

- From foreground partitions not controlled by VSE/POWER.
- From SYSLOG - you can gain control of SYSLOG following a PAUSE or STOP command or
  a // PAUSE statement.
- When no job is active in the partition, that is, after a /& statement has been processed.
- When all tape or disk files assigned to system logical units have been closed.

UNLOCK (Release Locked Resources)

The UNLOCK command is used to release all resources locked by the specified system. This command should only be used when that system has become inoperative with locks still contained in the lock file. The UNLOCK command is not valid for the system on which it is entered.

Operation	Operands	Type
UNLOCK	SYSTEM=sys-id	AR

`sys-id`

Specifies the CPU-ID of the CPU which has become inoperative. The command will release all locks belonging to the named system. The operator can obtain the CPU-ID from the console printout of the failing system. During system start, IPL message 0I04I identifies the CPU-ID.

In order to reduce the risk of entering a wrong system-id which would destroy all locks set by the named system, the UNLOCK command causes a verifying message to be displayed on the system console to which the operator has to respond with either YES or NO.

UPSI (User Program Switch Indicators)

The UPSI statement allows you to set program switches that can be tested by applications during execution.

Operation	Operands	Type
// UPSI	string	JCS

`string`
Is a string of one to eight characters, which correspond to the bit positions of the UPSI byte in the communication region. The specified character string must consist of the characters 0, 1 and X. If you code a 0 in the operand, the corresponding bit in the UPSI byte is set to 0. If you code a 1 in the operand, the corresponding bit in the UPSI byte is set to 1. If you code an X in the operand, the corresponding bit in the UPSI byte remains unchanged. Unspecified rightmost positions in the operand are assumed to be X.

Job control clears the UPSI byte at end of job.

VOLUME (Query Mounted Volumes)

The VOLUME command provides the operator with a short summary of the volumes mounted on disk devices, together with an indication of whether or not a volume is in use, whether a device is shared by another system, and whether it is reserved for VSAM space management usage.

Operation	Operands	Type
VOLUME	[c|cu|cuu]	AR

c
: Channel address. Information is to be supplied for all disk units on the specified channel.

cu
: Channel and control unit address. Information is to be supplied for all devices on the specified channel and control unit.

cuu
: Device address. Information is to be supplied for the specified device only.

If the operand is omitted, the information is given for all disk volumes at present mounted on the system.

ZONE (Set Time Zone)

The ZONE statement defines the time difference between local time and Greenwich Mean Time. If no ZONE statement is supplied, job control supplies the zone defined in the IPL SET command.

Locations that are on Greenwich Mean Time need not specify the ZONE statement.

To obtain correct job accounting information, the // ZONE statement should be entered between the /& and the // JOB statement.

Operation	Operands	Type
// ZONE	{EAST|WEST}/hh/mm	JCS

EAST
: A geographical position east of Greenwich.

WEST
: A geographical position west of Greenwich.

hh/mm
: A decimal value that indicates the difference in hours (00 to 23) and minutes (00 to 59) between local time and Greenwich Mean Time.

/. (Label Statement)

The label statement defines a point in the job stream up to which you may want to skip JC statements using a GOTO statement or the GOTO action of an ON statement. When a GOTO is raised, processing continues at the JC statement following the **/. label** statement specified.

Operation	Operands	Type
/.	label	JCC, JCS

Column 1 contains a slash (/) and column 2 a period (.). Column 3 must be blank.

`label`
: is a name consisting of one to eight alphanumeric characters. The first character must be alphabetic. Symbolic parameters are not allowed in this statement.

The name you specify for label is used as the operand in the corresponding GOTO. The /. label statement must be coded on the same JC level as the GOTO, that is, both must be within the same procedure, or both outside a procedure (on JC level 0).

/+ (End-of-Procedure)

The /+ statement marks the end of a job control procedure. It must be included as the last statement when a procedure is cataloged.

The /+ statement can also be entered on SYSLOG to end the procedure currently running in the appropriate partition.

JCS Format
/+ [comments]

Column 1 contains a slash (/) and column 2 a plus sign (+). Column 3 must be blank.

When used as delimiter on a cataloged procedure, the /+ statement is neither logged nor listed when the procedure is retrieved and included in the job stream. Instead, the following message is written:

EOP procedurename

where procedurename is the name of the called procedure.

/* (End-of-Data File)

The end-of-data file statement must be the last statement of each input data file on SYSRDR and SYSIPT. /* is also recognized for files that Logical IOCS reads from a card reader that is not assigned to SYSIN or SYSIPT.

For an input file on an IBM 5424/5425 MFCU, the /* card must be followed by a blank card.

JCS Format
/* [comments]

Column 1 contains a slash (/) and column 2 an asterisk (*). Column 3 must be blank.

/& (End-of-Job)

The end-of-job statement must be the last statement of each job.

JCS Format
/& [comments]

Column 1 contains a slash (/) and column 2 an ampersand (&). Column 3 must be blank. If a program attempts to read past the /& on SYSRDR or SYSIPT, an error message is issued. Any comments, beginning in column 35, are printed at end of job. If a job updates a system directory, comments included on the /& statement are not printed.

The end-of-job statement is printed on SYSLST in the following format:

print positions 1-3 contain EOJ;
print positions 5-12 the job name;
print positions 14-33 the maximum return code, if conditional JCL was used, otherwise user comments, if any;
print positions 35-72 blanks or any user comments; and
print positions 73-118 the date, time-of-day, and job duration in the following format:

DATE mm/dd/yy,CLOCK hh/mm/ss,
DURATION hh/mm/ss

mm/dd/yy can also appear in the order: dd/mm/yy, if this was specified in the STDOPT command.

On SYSLOG, the date, time of day, and job duration (the amount of time elapsed between the start and the end of a job) appear in the same format, occupying 46 positions, on the line following the end-of-job statement.

Any temporary control values, such as values of symbolic parameters, ON-conditions, options, assignments and LIBDEFs are reset every time this statement is encountered.

The stop time that the job accounting routines store in the job accounting table is the same as that given for CLOCK at end of job.

End-of-job information is not printed on SYSLST if // OPTION NOLOG has been specified. The NOLOG statement itself is logged on SYSLST.

* (COMMENTS)

The content of the comment statement is printed on SYSLOG. If followed by a PAUSE statement, the statement can be used to request operator action.

JCS Format
* [comments]

Column 1 contains an asterisk. Column 2 is blank. The remainder of the statement (through column 72) contains any user comments.

Job Control Statement Examples

The figures in this section contain examples of job control statement input. In the explanation that follows each example, the numbering of the items corresponds to the numbering at the left of the statements in the example.

Figure 3-8 to Figure 3-11 on page 3-153 show four simple jobs;

Figure 3-12 on page 3-154 is an example of conditional job control using the IF statement;

Figure 3-13 on page 3-155 shows more complex conditional job control using the IF, ON, and GOTO statements;

Figure 3-14 on page 3-156 shows the use of symbolic parameters in procedures;

Figure 3-15 on page 3-158 shows the nesting of procedures and the use of parameters in nested procedures.

General Job Control Examples

The examples on pages 3-150 to 3-153 contain four sample jobs. The statements of each job are executed in the sequence as entered. The PHASE, INCLUDE, and ENTRY statements are linkage editor control statements. These statements are described in detail in Section 4, "Linkage Editor." They are included in this discussion to present a more meaningful example.

```
1 // JOB U81SDC                          UNLOAD SEQUENTIAL DISK TO TAPE
2 // ASSGN SYS004,111                            INPUT MASTER FILE
  // ASSGN SYS005,112                            (2 EXTENTS)
  // ASSGN SYS006,380,C8              BACKUP TAPE DUAL DENSITY 9-TRK
3 // DLBL SDUNLD,'SEQUENTIAL FILE',73/206,SD
  // EXTENT SYS004,333002,1,0,1900,380
  // EXTENT SYS005,333003,1,1,76,570
4 // EXEC SD008,REAL,SIZE=60K                  RUN IN REAL USING 60K
5 // MTC RUN,SYS006
6 * OPERATOR - TAPE ON 380 - LABEL, REMOVE RING AND ARCHIVE
7 /&
```

Figure 3-8. General Job Control Examples Part 1

1. JOB statement.
2. ASSGN statements for 3330 disks and for tape.
3. DLBL and EXTENT statements to define a sequential disk file with two extents on separate 3330 volumes.
4. EXEC statement for a program in a sublibrary that is to be executed in real mode, using 60K of processor storage allocated to BG.

5. MTC command to rewind and unload the tape just created.

6. Message to notify operator that tape handling is required.

7. End-of-job indicator.

```
 1 // JOB R61ASSM              OBJECT DECK TO TAPE - CATALOG
   *                                    IN SUBLIBRARY
   * CREATE A MAP OF STORAGE ON SYSLOG
 2 MAP
 3 ASSGN SYS012,UA                 CLEAR PREVIOUS TAPE ASSIGN
   // ASSGN SYSPCH,381                ASSIGN SYSPCH TO TAPE
 4 // OPTION DECK,LIST,XREF,NOEDECK
 5  CATALR MOD207
 6 // EXEC ASSEMBLY
 7 ...                                ASSEMBLER SOURCE HERE
   /*
 8 // MTC WTM,SYSPCH,02               WRITE TAPEMARK AND
   // MTC REW,SYSPCH                  REWIND SYSPCH TAPE
 9 // ASSGN SYSIPT,381          ASSEMBLER OUTPUT TO LIBR INPUT
10 // EXEC LIBR,PARM='ACCESS S=LIB1.S2'  SUBLIB FOR OBJ
   /*                                    CATALOG FROM SYSIPT
   // MTC RUN,381                     REWIND/UNLOAD SYSIPT
11 /&                                 EOJ R61ASSM
```

Figure 3-9. General Job Control Examples Part 2

1. JOB statement.

2. MAP command to print a map of storage allocations on SYSLOG.

3. ASSGN statements to release previous tape assignment, and temporarily assign SYSPCH to that tape.

4. OPTION statement to specify options that are different from the permanent options.

5. Statement that will be transferred by job control to the SYSPCH file (tape on 381). This tape can then, after creation of the object deck, be used as input to the program to catalog it as a library member of the type OBJ.

6. EXEC statement for the system assembler.

7. Source deck as input to the system assembler and /* (end-of-data).

8. MTC statements to write a tapemark and rewind the tape on 381. This tape is now positioned for later use as SYSIPT.

9. The tape on 381 is temporarily re-assigned as SYSIPT for the librarian catalog run.

10. The librarian program is called. The sublibrary in which the object module is to be cataloged is specified in the ACCESS command passed in the PARM operand. The CATALOG statement is read from SYSIPT with the assembler output.

11. End-of-job indicator with a comment. SYSIPT returns to its permanent assignment.

```
1 // JOB K13CATL                          LINK MODULES INTO A
  *                                       SUBLIBRARY
2 LIBDEF PHASE,SEARCH=(LIB1.S2,LIB1.S3),CATALOG=LIB1.S3,PERM
3 // OPTION CATAL
   PHASE PROGX03,*
4 INCLUDE MOD207
  INCLUDE
  ...                                     OBJECT DECK INCLUDED HERE
5 /*
6 ENTRY MD207B
7 // EXEC LNKEDT
8 /&
```

Figure 3-10. General Job Control Examples Part 3

1. JOB statement.
2. Permanent definition of sublibrary chains from which programs are to be loaded, and into which phases are to be cataloged.
3. OPTION statement to specify that the phase produced by the linkage editor is to be cataloged.
4. PHASE and INCLUDE statements that are input to the linkage editor. The first INCLUDE statement calls the module previously cataloged in the sublibrary and the second (with a blank operand) is followed by an object deck to be included.
5. /* indicates the end of the object deck, not the end of input to the linkage editor.
6. ENTRY statement input to the linkage editor specifying an entry point for the PHASE PROGX03.
7. EXEC statement for the linkage editor.
8. End-of-job indicator.

```
1 // JOB E40
2 // ASSGN SYSLST,PRINTER            ASSIGN SYSLST TO ANY PRINTER
3 // ASSGN SYS004,(380,381,382)   ASSIGN TO TAPE WITHIN THIS RANGE
  // ASSGN SYS006,(280,281,282)
4 // ASSGN SYS005,SYS004             ASSIGN SYS005 as SYS004
  // EXEC MYPROG
5 /&
```

Figure 3-11. General Job Control Examples Part 4

1. JOB statement.

2. ASSGN statement for SYSLST, which may be any printer.

3. SYS004 should be assigned to a tape on 380, or 381 (if 380 is not available), or 382 (if both 380 and 381 are not available). Similarly, for SYS006 and 280, 281, 282.

4. Assign SYS005 to the same unit as SYS004 (described in 32).

5. End-of-job indicator.

Conditional Job Control: Example of IF Statement

```
1 // JOB A3243              EXAMPLE OF IF STATEMENT
2 // EXEC PGM1              FIRST PROGRAM
3 IF $RC=0 THEN             TEST RETURN CODE OF PGM1
4 // EXEC PGM2              IF RC OF PGM1 WAS 0, RUN PGM2
5 /&
```

Figure 3-12. The Use of the IF Statement

Explanation of the sequence numbers in Figure 3-12:

1. JOB statement for Job A3243.

2. If program PGM1 is not canceled and does not terminate abnormally, it will set a return code from 0 to 4095. This return code is used to control the processing of the following JCL statements.

3. The return code of the preceding step is tested, and the next statement is executed only if the condition is true. Otherwise, the statement is skipped.

4. Program PGM2 is executed only if the condition of the preceding IF statement was true.

5. End-of-Job indicator. All conditional JCL information is reset to default values.

Conditional Job Control: Example of ON, IF and GOTO Statements for Abnormal Termination

```
1 // JOB A3244          EXAMPLE OF ON, IF AND GOTO STATEMENTS
2 ON $ABEND GOTO AB                  FOR ABNORMAL TERMINATION
  // EXEC PGM1
3 IF $RC > 4 THEN      TEST RETURN CODE OF PGM1
  // EXEC PGM2          IF RC OF PGM1 WAS GREATER THAN 4
4 GOTO $EOJ            SKIP ABTERM STEP - END JOB
5 /. AB                SKIP TO HERE IN CASE OF ABNORMAL TERM.
6 // EXEC ABTERM       ONLY IN CASE OF ABNORMAL TERMINATION
  /&
```

Figure 3-13. IF, ON and GOTO Statements for Abnormal Termination

Explanation of the sequence numbers in Figure 3-13:

1. JOB statement for Job A3244.

2. If any of the following steps terminate abnormally, job control skips all statements up to the label AB. This statement overrides the default condition ON $ABEND GOTO $EOJ.

3. If the return code of PGM1 is greater than 4, the next statement is executed.

4. If this statement is processed, the rest of the statements in the job are skipped. This would be the case if neither PGM1 or PGM2 terminated abnormally.

5. If the condition of the ON statement occurred (one of the steps terminated abnormally), processing continues at this point.

6. This program will be executed only if an abnormal termination occurs.

Parameterized Procedure Example

```
Job stream as submitted to job control:

 1 // JOB A3245                        EXAMPLE OF PARAMETERIZED PROCEDURE
 2 // SETPARM RC1=                          DEFINE AND NULLIFY PARAMETER
 3 // EXEC PROC=UPDAT,RC1,SER=333006         CALL PROC, PASS PARAMETERS
 9 // EXEC PGM=EVALUATE,PARM='&RC1'        CALL PROGRAM, PASS PARAMETER
10 /&

Procedure UPDAT as cataloged:

 4 // PROC DEV=3330,SER=333005          DEFINE AND INITIALIZE PARAMETERS
 5 // ASSGN SYS008,&DEV,TEMP,VOL=&SER,SHR
 6 // EXEC PGM=UPDATE                                      CALL PROGRAM
 7 SETPARM RC1=$RC                                  SAVE RC OF PGM UPDATE
 8 /+

Note: The index numbering on the left is in processing sequence.
```

```
Resulting flow of execution:

 1 // JOB A3245                        EXAMPLE OF PARAMETERIZED PROCEDURE
 2 // SETPARM RC1=                          DEFINE AND NULLIFY PARAMETER
 3 // EXEC PROC=UPDAT,RC1,SER=333006         CALL PROC, PASS PARAMETERS
 4 // PROC DEV=3330,SER=333005          DEFINE AND INITIALIZE PARAMETERS
 5 // ASSGN SYS008,&DEV,TEMP,VOL=&SER,SHR
 6 // EXEC PGM=UPDATE                                      CALL PROGRAM
 7 SETPARM RC1=$RC                                  SAVE RC OF PGM UPDATE
 8 /+
 9 // EXEC PGM=EVALUATE,PARM='&RC1'        CALL PROGRAM, PASS PARAMETER
10 /&
```

Figure 3-14. The Use of a Parameterized Procedure

Explanation of the sequence numbers in Figure 3-14:

1. JOB statement for Job A3245.

2. Definition of parameter RC1, and assignment of null string.

3. Procedure UPDAT is called. The parameter SER is defined for this call of the procedure, and passed to the procedure. The parameter SER is defined for this procedure, and the value '333006' is assigned to it. The existing parameter RC1 is passed to the the procedure.

4. Default values for the procedure UPDAT are defined. The parameter DEV is not specified in the procedure call (3), so the default value '3330' is used. The default value of SER is not used, because it has been specified in the procedure call with the value '333006'.

5. The device type and volume serial number in the ASSGN statement are specified as symbolic parameters. Job control substitutes the values '3330' and '333006' for DEF and SER, respectively.

6. Execution of the program UPDATE. This step terminates with a return code.

7. The return code of the program UPDATE is assigned to the parameter RC1. This parameter will still be available after the procedure UPDAT has been terminated, as it was passed to UPDAT in the calling EXEC statement.

8. End of the procedure UPDAT.

9. Execution of the program EVALUATE. The symbolic parameter RC1 is used as a program parameter, thus making the return code of the program UPDATE in the procedure UPDAT available to the program EVALUATE.

10. End-of-Job indicator.

Parameterized Procedures and Procedure Nesting Example

```
Job stream as submitted to job control:

 1 // JOB A3246                 EXAMPLE OF NESTED PROCEDURES
 2 // EXEC PROC=PROC1           CALL PROCEDURE PROC1
12 // EXEC PGM=LISTPGM          CALL PROGRAM LISTPGM
13 /&                           END OF JOB A3246

Procedure PROC1 as cataloged:

 3 // EXEC X240                 CALL PROGRAM X240
 4 // EXEC PROC=X24,NO=2        CALL PROC X24, PASS PARAMETER
10 // EXEC X243                 CALL PROGRAM X243
11 /+                           END OF PROCEDURE PROC1

Procedure X24 as cataloged:
 5 // PROC NO=1                 INITIALIZE PARAMETER NO TO 1
 6 // EXEC X241                 CALL PROGRAM X241
 7 IF NO=2 THEN                 TEST PARAMETER NO
 8 // EXEC X242                 CALL PROGRAM X242 ONLY IF NO=2
 9 /+                           END OF PROCEDURE X24

Note: The index numbering on the left is in processing
sequence.
```

```
flow of execution:

 1 // JOB A3246                 EXAMPLE OF NESTED PROCEDURES
 2 // EXEC PROC=PROC1           CALL PROCEDURE PROC1

 3 // EXEC X240                 CALL PROGRAM X240
 4 // EXEC PROC=X24,NO=2        CALL PROC X24, PASS PARAMETER

 5 // PROC NO=1                 INITIALIZE PARAMETER NO TO 1
 6 // EXEC X241                 CALL PROGRAM X241
 7 IF NO=2 THEN                 TEST PARAMETER NO
 8 // EXEC X242                 CALL PROGRAM X242 ONLY IF NO=2
 9 /+                           END OF PROCEDURE X24

10 // EXEC X243                 CALL PROGRAM X243
11 /+                           END OF PROCEDURE PROC1

12 // EXEC PGM=LISTPGM          CALL PROGRAM LISTPGM
13 /&                           END OF JOB A3246
```

Figure 3-15. Use of Nested Procedures

Explanation of the sequence numbers in Figure 3-15:

1. JOB statement for Job A3246.
2. Procedure PROC1 is called.
3. Execution of program X240.

4. Procedure X24 is called. The parameter NO is defined, and the value '2' is assigned to it. (Calling a procedure from another procedure is called "procedure nesting".)

5. The default value of NO = 1 is not used because this parameter has been defined in the procedure call.

6. Execution of program X241.

7. The IF statement examines the present value of the parameter NO. If the condition were not true, the next statement would be skipped. In this case, the condition is true, and the next statement is executed.

8. End of procedure X24. Control returns to procedure PROC1, at a point immediately after the EXEC PROC statement (4) which caused control to be passed to procedure X24.

9. Execution of program X243.

10. End of procedure PROC1. Control returns to SYSRDR at a point immediately after the EXEC PROC statement (2) which called PROC1.

11. Execution of program LISTPGM.

12. End-of-Job indicator.

Nesting Example

Section 4. Linkage Editor

The linkage editor prepares programs for execution and accepts as input the relocatable object modules produced by the language translators and object modules produced by the librarian PUNCH command. It processes these modules into program phases, which may be executed immediately or cataloged into a library.

If object modules from a sublibrary are to be link-edited, the sublibrary must be defined with the statement:

```
// LIBDEF OBJ,SEARCH=lib.sublib
```

The sublibrary into which phases are to be cataloged must be defined with the statement:

```
// LIBDEF PHASE,CATALOG=lib.sublib
```

If the correct search chain and catalog sublibrary definitions have been made permanently for the partition, you need not include them in the linkage editor job.

Now you can call the linkage editor program with the job control statement:

```
// EXEC LNKEDT
```

followed by the linkage editor control statements you need.

If the present run of the linkage editor will replace an existing phase that is under control of the Maintain System History Program (MSHP), the EXEC statement must be in the form:

```
// EXEC LNKEDT,PARM='MSHP'
```

Language Translator Modules

The input to the linkage editor consists of linkage editor control statements and object modules. Each module is the output of a complete language translator run. It consists of dictionaries and text for one or more control sections.

The dictionaries contain the information necessary for the linkage editor to resolve references between different modules. The text consists of the actual instructions and data fields of the module.

Six statement types can be produced, by the language translators or by the programmer, to form a module. They appear in the following order:

Stmt Type	Contents/Purpose
ESD	External symbol dictionary
SYM	Ignored by linkage editor
TXT	Text
RLD	Relocation list dictionary
REP	Replacement text supplied by the programmer if necessary.
END	End of module.

For the format of each of these statements (except SYM), see the Appendix.

The **external symbol dictionary** contains control section definitions and inter-module references. When the linkage editor has the ESDs from all modules, it can relocate the sections and resolve the references.

The **relocation list dictionary** identifies portions of text that must be modified on relocation (address constants). Unresolved address constants are set to zero in relocatable phases.

Linkage Editor Control Statements

In addition to the language translator output previously listed, input for the linkage editor includes linkage editor control statements. These statements are briefly discussed below and described in detail further on in this section.

ACTION
: Specifies options to be taken.

ENTRY
: Provides an optional transfer address for the first phase and ends the linkage editor input.

INCLUDE
: Signals that an object module or a number of CSECTs contained in an object module are to be included in the phase currently being processed.

PHASE
: Indicates the beginning of a phase. It gives the name of the phase and the storage address where it is to be loaded.

General Control Statement Format

The linkage editor control statements must be coded in the following format:

- The operation field must be preceded by one or more blanks.
- The operation field must be separated from the operand field by at least one blank position.
- The operand field is ended by the first blank position. It cannot extend past position 71.

Control Statement Placement

If more than one object module is to be processed in a single linkage editor run, the single ENTRY statement should follow the last object module. The ACTION statement(s) must be the first record(s) encountered in the input stream; otherwise, they are ignored.

A PHASE statement, if present, must precede any INCLUDE statements for modules in a sublibrary or any compiler-produced object code to be included in that phase.

For modules in a sublibrary, PHASE and INCLUDE statements precede the first compiler-generated statement (ESD or TXT), if any object code is present. See the description of the CATALOG statement in Section 5, "Librarian" on page 5-1, the LIBDEF description in Section 3, "Job Control and Attention Routine" on page 3-1, and the following examples:

Example:

```
   . . .
// OPTION CATAL
   PHASE A,*
   INCLUDE XYZ
// EXEC LNKEDT
   . . .
```

Module XYZ in the sublibrary may contain the following statements:

```
INCLUDE A1
PHASE B,*
INCLUDE B1
ESD
. . .
TXT                Assembler/compiler-generated object code
. . .
RLD
END
```

A1 and B1 are the names of modules cataloged in a sublibrary. The linkage editor will produce and catalog phase A consisting of module A1 and phase B consisting of module B1 and the object code contained in module XYZ.

Example:

```
// OPTION CATAL
   PHASE A,*
   INCLUDE XYZ
// EXEC LNKEDT
```

Module XYZ may contain the following statements:

```
   INCLUDE A1
   PHASE B,*
   INCLUDE B1
   INCLUDE B2
   END
```

A1, B1, B2 are the names of modules cataloged in a sublibrary.

PHASE and INCLUDE statements can also be read from SYSRDR or SYSIPT. Object modules can only be read from SYSIPT. An INCLUDE statement without operands causes job control to read from SYSIPT until end of file is found, after which reading from SYSRDR is resumed.

Example:

```
// OPTION CATAL
   PHASE A,*          from SYSRDR
   INCLUDE
   . . .
object code module A1
   . . .
  PHASE B,*
   . . .
                                 from SYSIPT
object code module B1
object code module B2
   . . .
/*
// EXEC LNKEDT
```

ACTION

This statement is used to indicate linkage editor options. The statement must be the first linkage editor statement in the input stream. If multiple operands are required, they can be placed in separate ACTION statements or in one ACTION statement separated by commas.

If an ACTION statement is flagged as invalid (as the result of an invalid operand, etc.) all subsequent ACTION statements submitted during the job step are ignored.

Operation	Operands
ACTION	[,MAP|,NOMAP][,NOAUTO][,CANCEL][,SMAP]

At least one blank must precede ACTION.

MAP

Requests the linkage editor to write to SYSLST a map of virtual storage, which can be used for problem determination. The map contains the name of every entry within each CSECT and the name of every CSECT within each phase. For an example of a partition storage map, together with a description of how to interpret it, see *VSE/Advanced Functions Service Aids.*

If the MAP operand is specified, SYSLST must be assigned. If the ACTION statement is not used, MAP is assumed when SYSLST is assigned, and NOMAP is assumed when SYSLST is not assigned.

NOMAP

Indicates that the MAP option should not take effect. The system lists all linkage editor error diagnostics on SYSLOG.

NOAUTO

Indicates that the AUTOLINK function is to be suppressed for the present linkage editor run.

The NOAUTO operand in a PHASE statement indicates to the linkage editor that AUTOLINK is to be suppressed for that phase only. If an entire program requires NOAUTO, then specifying ACTION NOAUTO cancels AUTOLINK during link editing of the entire program, thereby eliminating the necessity of specifying NOAUTO in each PHASE statement.

Note: When a weak external reference (WX) is encountered, it is treated in the same manner as a normal external reference with NOAUTO.

ACTION

CANCEL

Cancels the job automatically if any of messages the 2100I through 2170I occur. If this option is not specified, the job continues.

SMAP

Cancels the job automatically if any of messages the 2100I
Indicates that, in addition to the standard virtual storage map in which the control sections are ordered by load address, a listing of the CSECT names ordered alphabetically is also generated. This list may be useful if you want to locate a CSECT by its name and the phase consists of many CSECTs.

ENTRY

Every program, as input for the linkage editor, is terminated by an ENTRY statement. Job control writes an ENTRY statement with a blank operand on SYSLNK when EXEC LNKEDT is read; this causes the load address of the first phase to be used as the transfer address. It is necessary to supply the ENTRY statement only if you specifically request another entry point.

Operation	Operands
ENTRY	[entrypoint]

At least one blank must precede ENTRY.

`entrypoint`
: Specifies the name (label) of an entry point. It must be the name of a CSECT or a label definition (source ENTRY) defined in the first phase. This address is used as the transfer address to the first phase in the program. If the operand field is blank, the linkage editor uses as a transfer address the first significant address provided in an END record encountered during the generation of the first phase. If no such operand is found on the END card, the transfer address is the load address of the first phase.

INCLUDE

INCLUDE indicates that an object module is to be included for editing by the linkage editor.

Operation	Operands
INCLUDE	[modulename][,(namelist)]

At least one blank must precede INCLUDE. If both operands are omitted, the object module to be included is assumed to be on SYSIPT. Job control copies it from there to SYSLNK. INCLUDE statements with no operands are recognized only on SYSRDR.

`modulename`
Specifies the name of the module, as used when cataloged in the sublibrary. It consists of one to eight alphameric characters.

`(namelist)`
The linkage editor generates a phase from only the control sections specified in this operand. The namelist is in the following format:

(csname1,csname2,...)

It is possible to include within the same phase (an) object module(s) without the namelist option and (an) object module(s) specifying the namelist option. The total number of control sections in a namelist cannot exceed five; however, any number of INCLUDE statements can be used.

If the operand modulename is used, the object module is assumed to be in a sublibrary. The linkage editor searches all sublibraries specified in the search chain for OBJ-type members. The module name must be the same as the member name used when the module was cataloged in the sublibrary.

If both operands are used, the object module is read from the sublibrary and the indicated control section(s) are extracted.

Object modules in a sublibrary may contain INCLUDE statements at up to five nesting levels.

If only the operand namelist is used, it must be preceded by a comma. The object module(s) to be included can be in the input stream (SYSIPT) or in a sublibrary. If the operand **modulename** is not specified, each series of object modules on SYSIPT must end with a /* control statement. The linkage editor reads the object module(s) and extracts the control section(s) specified.

Note: If you use the INCLUDE statement with only the namelist to include modules from SYSIPT, each module on SYSIPT must be preceded by a further INCLUDE statement without operands.

PHASE

This statement provides the linkage editor with a phase name and an origin point for the phase.

The phase name is used to catalog the phase into a sublibrary. It is also used when the phase is retrieved for execution. The PHASE statement must precede the first object module of each phase processed by the linkage editor. Any object module not preceded by a PHASE statement is included, together with the preceding object modules, in the current phase. If no PHASE statement is used, or if the PHASE statement is in error, the linkage editor generates a dummy statement. This allows testing of the program when the LINK option is used. However, the program with the dummy PHASE statement cannot be cataloged in a sublibrary; when the CATAL option is used, the job is canceled.

Operation	Operands
PHASE	name,origin[,NOAUTO][,SVA][,PBDY]

At least one blank must precede PHASE. The operands have the following meaning:

`name`

Specifies the name of the phase. One to eight alphameric (0-9, A-Z, #, $, and @) characters are used as the phase name. For phases which are to be cataloged, use the librarian conventions for member names. The name may not be ALL, S, or ROOT.

Each single-phase program should be unique in the first four characters of its phase name, because all phases whose names start with the same four characters will be treated as a multiphase program. When a multiphase program, is being fetched, the partition must be large enough to contain the largest phase, unless you specify SIZE=name in the EXEC statement.

`origin`

Specifies the load address of the phase. The load address can be in one of the following forms:

1. symbol[(phase)][+relocation]
2. *[+relocation]
3. S[+relocation]
4. ROOT
5. displacement

Items 1 to 4 specify a relative address, item 5 an absolute address.

A phase can be made relocatable if its origin is specified as a relative address (formats 1-4 above). However, if the address is relative to another phase which is not relocatable, the new phase will not be relocatable. If the operand origin is not specified, the phase is made relocatable.

The elements that make up the various forms that specify the origin are the following:

1. **symbol:** May be a previously defined phase name, control section name, or external label (the operand of an ENTRY source statement).

   (phase): If **symbol** is a previously defined control section name or a previously defined external label that appears in more than one phase, **phase** (in parentheses) directs the linkage editor to the phase that contains the origin. The phase name must have been defined previously.

   relocation: Indicates that the origin of the phase currently being processed will be set relative to the symbol by a relocation term consisting of:
   a + or a - immediately followed by: X'hhhhhh' (one to six hexadecimal digits);
   a + or a - immediately followed by: ddddddd (one to eight decimal digits);
   a + or a - immediately followed by: nK,(where n is the number of kilobytes).

2. *: Indicates that, for the first PHASE statement processed, the origin is to be the first doubleword storage address after the partition save area, or the area assigned to the COMMON pool (if any).

   For successive PHASE statements, the linkage editor assigns the storage location immediately following the end of the preceding phase (with forced doubleword alignment) as an origin for the phase.

   relocation: Indicates relocation of the phase relative to the next storage location of the virtual partition. The format is as specified in item 1.

3. **S**: if **S** is specified, the origin is determined in the same manner as for the first PHASE statement in item 2.

   relocation: Indicates relocation of the phase relative to the start of the virtual partition as described in item 2.

4. **ROOT**: Tells the linkage editor that the phase that follows is a root phase. The storage address assigned to the root phase is determined in the same manner as the first PHASE statement in item 2. Only the first PHASE statement in the linkage editor input can specify ROOT. If a control section (CSECT) appears in the root phase, other occurrences of the same control section are ignored and all references are resolved to the control section in the root. Control sections are not duplicated within the same phase. If any subsequent phase overlays any part of the ROOT phase, a warning diagnostic is displayed on

SYSLST if ACTION MAP is in effect. Refer also to the description of the ACTION statement earlier in this section.

5. **displacement:** Allows the origin (loading address) to be set at a specified location. The origin point is an absolute address, relative to zero.

   displacement must be:

   X'hhhhhh' (one to six hexadecimal digits),
   dddddddd (one to eight decimal digits), or
   nK (where n is the number of kilobytes.

   A displacement of zero (+ 0) denotes a self-relocating program.

To link-edit a program for execution in real mode under a 370-mode system, you can

- Link-edit the program so that it can be relocated to a real partition when it is loaded.
- Write the program to be self-relocating.
- Link-edit the program with a PHASE statement that contains the absolute address of the location within the real partition where the program is to be loaded.

If a COMMON area (such as in FORTRAN programs) is used, the length of the largest COMMON is added to every phase origin, even if the origin is given as an absolute value.
COMMON is located at the beginning of the phase with the lowest origin address (if multiple phases).

NOAUTO

Indicates that the Automatic Library Lookup (AUTOLINK) feature is suppressed for the current phase. If you want to suppress it for the entire program, specify ACTION NOAUTO.

SVA

Indicates that the phase is SVA-eligible. This means that the phase must be re-enterable and relocatable. When this phase is cataloged into a sublibrary, the linkage editor will also have the phase loaded into the SVA if the phasename was listed in the SDL with the SVA option. If the linkage editor finds that a phase that is specified with the SVA option is not relocatable, an error message is issued and the SVA option is ignored.

PBDY

Indicates that the phase is to be link-edited on a page boundary. If the current link-edit address is not aligned on a page boundary, the linkage editor uses the next higher page boundary address.

Note: See VSE/Advanced Functions System Management Guide for information on partition boundary alignment.

Some examples of PHASE statements follow:

PHASE PHNAME,* + 504

This causes loading to start 504 bytes past the first double-word after the end of the partition save area, or past the end of the previous phase.

PHASE PHNAME3,PHNAME2

This causes loading to start at the same point where the loading of phase PHNAME2 started. When PHNAME2 is also the name of a previous entry point, then loading starts at the address of that entry point, aligned on a doubleword boundary.

PHASE PHNAME,CSECT1(PHNAME2)

This causes loading to start at the point where CSECT1 was loaded. CSECT1, the named control section, must have appeared in the phase named PHNAME2.

PHASE PHNAME1,S + 30K
PHASE PHNAME2,*
PHASE PHNAME3,PHNAME2

The first phase (PHNAME1) of the preceding series is loaded starting at 30K plus the length of the save area. The second phase (PHNAME2) of the series is loaded at the end of PHNAME1. The third phase (PHNAME3) is loaded at the same address as was PHNAME2.

PHASE PHNAME,ROOT

Loading begins at the first doubleword after the beginning of the the partition save area, and the area assigned to the COMMON pool (if any).

Note: *If the origin address supplied is not on a doubleword boundary, the linkage editor automatically increments that address to the next doubleword boundary.*

Section 5. Librarian

This section describes a program, contained in the VSE/Advanced Functions operating system, which services, copies and provides access to system and private libraries.

The present librarian is intended for use with VSE/Advanced Functions Version 2, Release 1, and replaces that used with previous releases of the system. The job control statements LIBDEF, LIBDROP and LIBLIST have also been changed to reflect the new librarian concept.

Note that the commands DELETE, MOVE and RENAME, cannot be carried out when the specified sublibrary is in use. A sublibrary is "in use" when:

A LIBDEF job control statement or command specifying that sublibrary is active in another partition, or

A librarian ACCESS or CONNECT specifying that sublibrary is active in another partition.

The Librarian program sets a return code each time a librarian function is attempted. If the function completes successfully, a return code of 0 or 2 is set. In the case of non-completion, a return code of 4 to 16 is set, depending on the severity of the error.

The set of librarian commands includes an ON, a GOTO and a /. label command, which allow conditional execution of following commands dependent on the success of the preceding ones.

The librarian program itself can test the return code after execution of each command. The highest return code set during a librarian call is passed to job control at the end of the librarian job step, and can be tested by conditional job control statements.

The librarian return codes are explained in the manual *VSE/Advanced Functions System Management Guide.*

Library Concept

The librarian program is used to maintain data which must be readily available to the system, such as programs and cataloged procedures. These data are organized in libraries, which are divided into sublibraries, which in turn contain the data, organized in units called members.

Each member is identified by library name, sublibrary name, member name and member type. Most librarian commands which act on members have an operand in the form "membername.membertype". The "library.sublibrary" component of the name is supplied in a preceding ACCESS or CONNECT command. Library, sublibrary and member names can be freely chosen, following the rules given in the descriptions of the DEFINE and CATALOG commands in "Librarian Commands" on page 5-9.

The member type used depends on the type of data contained in the member, as follows:

A-Z, 0-9, $, #, @
: for source programs, which are to serve as input to a language translator;

OBJ
: for object modules (output from a language translator, input for the linkage editor);

PHASE
: for executable program phases (output from the linkage editor, ready for loading into storage);

PROC
: for cataloged procedures;

DUMP
: for dumps.

A "type" other than these five can be used for members containing any user data.

Supported Equipment

The librarian supports libraries on all disk devices which are supported by VSE/Advanced Functions Version 2, Release 1, except the IBM 2311.

Compatibility with Previous Releases

The librarian takes over the functions of the programs MAINT, COPYSERV, CORGZ, CSERV, RSERV, SSERV, DSERV and PSERV, BACKUP and RESTORE, which were supported under earlier releases of VSE/Advanced Functions.

For compatibility reasons, however, a number of these "old" programs are emulated in the VSE librarian, and can run without change, with the exception of comments coded in the "old" control statements. Any comment in a command submitted to the librarian must preceded by /* and followed by */. For information on compatibility with and migration from earlier releases, see *VSE/Advanced Functions Planning and Installation.* Note that the system library is no longer default for the "old" functions, so MAINT, CORGZ, xSERV and LNKEDT jobs must contain a LIBDEF statement for the library to be used. The system logical units SYSxLB are no longer supported by the ASSGN statement or command.

However, LIBDEF, LIBDROP and LIBLIST job control statements in Version 1, Release 2 and Release 3 format are still accepted. They are automatically converted to Version 2, Release 1 format during execution.

Librarian Command Syntax

The facilities of the librarian are invoked by using the EXEC LIBR statement, followed by one or more librarian commands.

These are in free format, and may be coded anywhere between column 1 and column 72 of the input line. This applies to librarian commands entered from SYSRDR and from SYSLOG.

Separators

The command name is separated from its operands by one or more blanks.

A comma or blank character is allowed to separate operands or elements of a list. Any of the allowed separators, namely comma, blank, equals sign or colon, two periods, parentheses or comments, may be surrounded by one or more blanks.

Continuation

Command continuation is indicated by a minus sign (-) as the last non-blank character in a line. This sign is also recognized as a separator, and need not be preceded by a blank, although this is, of course, allowed. The same holds true for comments.

Comments

Comments are allowed at any place where a blank is allowed. They are identified by a "/*" at the beginning and a "*/" at the end. You may code a comment outside a command, in a line by itself. Do not begin the "/*" in

column 1, as this would be interpreted as the end-of-file indicator when entered on SYSIPT.

Abbreviation

Command names may be shortened by leaving off one or more letters from right to left, so long as the name remains unique.

Exceptions to this rule are the commands LISTD, PUNCH and GOTO. LISTD can be shortened to LD. The shortest allowed form of PUNCH is PU. GOTO cannot be shortened.

In the following sections, the part of the command name which must be coded is in capitals, the rest in lower case. For example, you may code the command name given as DEFine in one of the following forms:

DEF, DEFI, DEFIN, or DEFINE.

Operand keywords may be shortened in the same manner, so long as they cannot be confused with other keywords which are valid for the particular command. Here again, you can take the notation of the following sections as a guide.

Notation

Like all librarian key-words, LIBRARY and SUBLIBRARY can be entered in full or in any shortened form. In the following sections, which describe the librarian commands, this syntax notation is used:

for library specifications: Lib = l ...
for sublibrary specifications: Sublib = l.s ...
for member specifications: mn.mt ...

where
l is the library name (1..7 characters),
s is the sublibrary name (1..8 characters),
mn is the member name (1..8 characters), and
mt is the member type (1..8 characters).

Do not use the names IJSYSRS or IJSYSRn for your libraries or the name SYSLIB for a sublibrary. These names are reserved for the SYSRES files and the system sublibrary respectively.

The notation is otherwise the same as that described in "Control Statement Conventions" on page 1-3.

Operands

Operands are of two kinds, positional and keyword. Positional operands consist of a value only, for example, NAMEA.TYPEB, and must be coded in the position shown in the command description. Keyword operands consist of a keyword and a value separated by an equals sign, for example, LIST = YES. These operands may be coded in any order, but they must follow any positional operands in the command.

Multiple targets

Certain commands can perform the same function on several targets at once. This is indicated in the notation by dots after the operand, for example,
Lib=1 ...

In such a case you can code:

```
        LIB=LIB1 LIB2 LIB3
or      LIB=LIB1, LIB2, LIB3
or      LIB=(LIB1 LIB2 LIB3)
or      LIB=(LIB1,LIB2,LIB3).
```

From-to operands

Commands which require two targets, for example COMPARE and COPY, where the sequence of the operands determines how the function is to be carried out, can have the operands coded as:

```
        LIB=LIB1:LIB2
or      LIB=LIB1 LIB2
or      LIB=LIB1..LIB2.
```

The same principle holds true when the multiple operands or the two targets are sublibraries or members.

Generic References

In librarian commands you may make generic reference to member names and types within the sublibrary specified in a preceding ACCESS or CONNECT command. However, you need READ access right to the sublibrary, if it is protected by the Access Control Function.

If you want to specify members of all types with the name ABC, code ABC.* for mn.mt. Coding *.PROC will cause the command to apply to all members of the type PROC, whatever their names may be.

You can specify all members of a given type whose names start with the same combination of characters by coding, for example, AB*.OBJ. This would result in the members ABC, ABEND, ABSTAIN, and so on, being acted upon, provided they have OBJ as type.

Coding *.* makes all members of any type in the specified sublibrary available to the specified command.

Invoking the Librarian Program

Before you issue any librarian commands, you must activate the librarian program using the job control statement or command:

```
[//] EXEC LIBR[,PARM={'cmd-line[;cmd-line...]'|'MSHP[;cmd-line...]'}]
```

The librarian then accepts commands from SYSLOG, if the EXEC LIBR statement was issued from there, or from SYSIPT. Type in END on SYSLOG, or issue a /* statement on SYSIPT, after the last librarian command. The librarian then passes a return code to the job control program for use in conditional job control.

In the EXEC command or statement:

cmd-line
Can be any valid librarian command line. A librarian command of up to 72 characters can be entered in one command line. For longer commands, use two or more command lines, separated by semicolons (;), in the PARM operand. The normal continuation rules for librarian commands apply (see "Librarian Command Syntax" on page 5-3).

You can code any number of librarian command lines in the PARM operand, enclosed in single quotes and separated by semicolons. The only restriction is, that the number of characters between the quotes must not be greater than 100. This restriction also applies when you enter 'MSHP' followed by librarian commands. The command or commands specified in the PARM operand are carried out before any other commands which may follow the EXEC LIBR statement or command.

Because EXEC LIBR is a job control statement, you can use symbolic parameters in the librarian command lines which you insert in the PARM operand. Be sure, however, that an EXEC LIBR statement with symbolic parameters is called from within a job which starts with a // JOB statement.

MSHP
Specifies that members controlled by the Maintain System History Program (MSHP) can be modified by the librarian. Otherwise, these members can be modified only by MSHP.

Partition Size for the Librarian Program

The partition in which the Librarian is to run must have at least 256K of virtual storage allocated to it.

For libraries in VSAM managed space, a SIZE specification of at least 256K must be given for the partition. This can be given in a SIZE job control statement, or in the SIZE operand of the EXEC LIBR statement.

The specification SIZE=AUTO is not allowed in the EXEC LIBR statement.

Summary of Librarian Commands

The following is a complete list of the VSE Librarian commands:

Command Name	Command Object		
	Library	Sublibrary	Member
ACCESS		X	
BACKUP	X	X	
CATALOG			X
CHANGE		X	
COMPARE	X	X	X *
CONNECT		X	
COPY	X	X	X *
DEFINE	X	X	
DELETE	X	X	X *
GOTO			
INPUT			
LIST			X *
LISTDIR	X	X	X *
MOVE	X	X	X *
ON			
PUNCH			X *
RELEASE	X	X	
RENAME		X	X *
RESTORE	X	X	X *
TEST	X	X	X *
UPDATE			X
/. label			

'*' means: generic specification accepted.

Figure 5-1. List of Librarian Commands

Merging Sublibraries

There is no MERGE command as such in the Librarian command set. To merge two sublibraries, use a CONNECT command followed by a COPY or MOVE command with a generic member specification.

Using the COPY command leaves the "from" sublibrary intact; using the MOVE command causes the "from" sublibrary members to be deleted.

For examples, see the descriptions of the COPY and MOVE commands.

Librarian Commands

ACCESS (Specify Target Sublibrary)

```
Access          {Sublib=l.s | ?}
```

The ACCESS command specifies the sublibrary, qualified by the library name, to be used in any following command which has a member – name.member – type specification as its operand. When given before a RESTORE member command, it specifies the default target sublibrary.

The ACCESS command remains valid until

- Another valid ACCESS command is given, or
- The sublibrary specified in the command is deleted, or
- The sublibrary specified in the command is renamed, or
- The library specified in the command is deleted, either explicitly by a DELETE command, or implicitly by a MOVE or RESTORE command

The ACCESS command is valid only if the specified library and sublibrary already exist.

If you code a question mark as operand, the name of the sublibrary currently accessed is displayed on SYSLOG, if the command was entered from there, otherwise on SYSLST.

BACKUP (Backup Library or Sublibrary)

```
Backup          {Lib=l ...|Sublib=l.s ...}
                Tape={SYSnnn|cuu}
                [Restore={Online|Standalone}]
                [Include=Historyfile]
                [ID=name]
                [Header=l.s.mn.mt]
```

The BACKUP command causes libraries, sublibraries or SYSRES files to be copied to tape.

The backup function includes a reorganization of the copied library, which usually results in a faster read access after a later restore. The reorganization will be most effective if a complete library is backed up and restored.

`Lib=l`

Specifies the library to be backed up. The library names IJSYSR1 to IJSYSR9 specify SYSRES files, the name IJSYSRS specifies the IPLed system. If you specify IJSYSRS, and no DLBL/EXTENT information is available, the librarian creates a file-id of IJSYSRS.

For this operand you may code a list of library names.

`Sublib=l.s`

Specifies the sublibrary to be backed up. For this operand you may code a list of sublibrary names.

`Tape=SYSnnn|cuu`

Specify the programmer logical unit or the physical unit address respectively of the tape unit to be used for output. If the amount of data to be backed up requires multiple tapes, you must assign the alternate tape in an ASSGN job control statement.

If Restore = Online is specified, the tape is **not repositioned** before the backup starts.

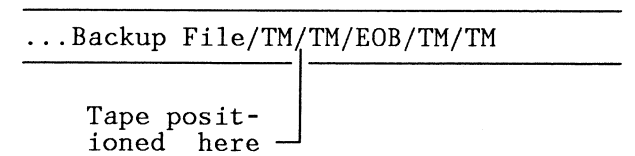
The following files are written to the backup tape:

1. Header (if specified), or empty file;
2. The Backup-file-ID record, and the system history file (if INCLUDE = HISTORYFILE was specified);
3. The backup file, containing the libraries and sublibraries specified in the command.

If Restore = Standalone is specified, the tape is first positioned at the load point, and then the following files are written to it:

1. The header (if specified) and stand-alone programs;
2. Stand-alone programs;
3. A backup-file-ID record, and the system history file (if INCLUDE=HISTORYFILE was specified);
4. The backup file, containing the libraries specified in the command.

At completion of the backup (irrespective of the RESTORE specification), two tapemarks, an End-of-Backup record, and two more tapemarks are written. The tape is left positioned after the tapemark which marks the end of the backup file, thus:



`Restore=Online|Standalone`

Specify how the backed up data are to be restored. This operand is applicable only to libraries and SYSRES files. The default value is ONLINE. A stand-alone version can be restored on-line.

If you code Restore=Standalone, the list of libraries to be backed up must contain at least one SYSRES file, because the stand-alone programs required for restoring data are available only in SYSRES files.

`Include=Historyfile`

Specifies that the system history file should also be backed up. It will be copied to tape first, before the specified libraries or sublibraries. The history file must not extend over two tapes.

`ID=name`

Defines an identification for the backup file to be created by this BACKUP command. This makes it easier to locate a particular version of backed-up data during RESTORE. If this operand is omitted, no identification is recorded.

The **name** must be 1 to 16 characters long, and enclosed in quotes. If only alphanumeric characters are used, the quotes are not needed. **name** itself must not contain quotes.

Header=l.s.mn.mt

Specifies an optional header to be written to the first file on the backup tape. The contents of the header is that of the member specified by l.s.mn.mt, which must have a record length of 80 bytes.

The header may contain, for example, information text, job control statements or copyright statements. It is skipped by the librarian during on-line restore.

If the backup is to be restored stand-alone, the header must start with the IPL bootstrap records.

CATALOG (Catalog Member)

```
CAtalog        mn.mt [Eod=xx] [Data={No|Yes}] [Replace={No|Yes}]
```

The CATALOG command causes the data following it to be cataloged under the name and type specified. The end of the data to be cataloged is usually indicated by a /+ (but see the description of the EOD operand).

An empty data set will not be cataloged.

Any member types (except PHASE and DUMP) can be cataloged into any sublibrary using this command.

If you wish to add, delete or replace lines in the member at a later date, using the librarian UPDATE command, you should provide sequence numbering in your input lines. The sequence number may be located anywhere in the line, and may be 1 to 8 characters long. The format must be:

- Numeric, with leading zeros, if you want to update the member using SEQUENCE=n or SEQUENCE=FS on the UPDATE command;
- Alphanumeric, without blank characters, if you want to update the member using SEQUENCE=NO on the UPDATE command.

The librarian provides the necessary sequence numbering if you use the following sequence of commands:

```
EXEC LIBR                          call the librarian program
ACCESS S=lib.sublib                access appropriate sublibrary
UPDATE membername.membertype       "update" the member
)END                               end update, causing renumbering
```

These commands cause the lines of the specified member to be sequence numbered in columns 77 to 80 using an increment of 10, starting at 10. These are the default values used by the librarian. Should you wish to use a different increment or a different length or position for your sequence numbering, specify the appropriate values in the SEQUENCE and COLUMNS operands of the UPDATE command.

mn.mt

Specifies the name and type under which the following data are to be cataloged. mn and mt may each be 1 to 8 characters long, and must be alphanumeric. The member type must not be PHASE or DUMP, because members of these types can be cataloged only by the Linkage Editor and Dump program respectively.

The sublibrary in which the member is to be cataloged must have been created, and must have been specified in a preceding ACCESS command, before you attempt to catalog a member into it.

If a member of the specified name and type existed in the target sublibrary before the command was carried out, its " put-in-sublibrary" time-stamp is left unchanged, and the " last-replaced" time-stamp is updated.

If such a member **did not** exist in the target sublibrary, the " put-in-sublibrary" time-stamp is set, and the "last-replaced" time-stamp is empty.

`Eod=xx`

Specifies what combination of two characters is to be used to indicate end-of-data in the following input. The default is /+. You must use an alternate end-of-data delimiter

- When the data to be cataloged contains a /+ statement, and
- When the CATALOG command and its input data are part of a job control procedure. (If a /+ were included in the SYSIPT data of a procedure, the job control program would take it as the End-of-Procedure delimiter, and not process the remainder of the procedure.)

For xx you may code any two characters except comma and blank. For further information, see the /+ command on page 3-146.

Except when cataloging members of the type OBJ or assembler macros with a MEND statement, the end-of-data record must follow the last record of the input data. The end-of-data record itself is not cataloged.

`Data=Yes|No`

Is applicable only for procedures. The member type must be PROC or a user type. You must code DATA = YES if the procedure contains SYSIPT data. The system default is DATA = NO. Procedures to be used in a set of nested procedures must be cataloged either all with DATA = YES or all with DATA = NO.

`Replace=Yes|No`

Allows conditional cataloging. If you specify REPLACE = YES, an already existing member with the same name and type as that specified will be deleted and overwritten with the new data, that is, it will be replaced by the new member.

REPLACE = NO, the default, means that the input data will not be cataloged if a member with the same name and type already exists in the sublibrary.

For compatibility with the old CATALS function, BKEND, MACRO and MEND statements are allowed instead of the EOD statement. They can start in column 1. The Librarian EOD indication may be specified in addition. It must follow the BKEND or MEND statement immediately. If a member starts with BKEND or MACRO, its end must be indicated by the corresponding BKEND or MEND statement.

A macro with input data following it must be cataloged with the following delimiting statements:

```
BKEND / MACRO / macro source statements / MEND / input data / BKEND
```

CHANGE (Change REUSE Attribute)

```
CHange  Sublib=l.s ... [REUse={Automatic|Immediate}]|
```

The CHANGE command can be used to change the REUSE attribute of a sublibrary. For details on the REUSE attribute, see the DEFINE command.

`Sublib=l.s`
: Specifies the qualified name of the sublibrary whose REUSE attribute is to be changed. You may specify several sublibraries which are to have the same attribute.

`REUse=Automatic|Immediate`
: Specifies which REUSE attribute the specified sublibrary (or sublibraries) should have from now on. If the sublibrary already has the specified attribute, the librarian program sets a return code of 4.

  When IMMEDIATE is specified, any space which is no longer in use in the sublibrary, but has not yet been freed, is freed at once.

COMPARE (Compare Libraries, Sublibraries or Members)

```
COMpare        {Lib=l [:] l ...|Sublib=l.s [:] l.s ...|mn.mt ...}
               [Data={Directory|Member}]
               [Punch={Yes|No}]
```

The COMPARE command is used to compare libraries, sublibraries or members, and provide a listing of the differences, as explained under the DATA operand below. Comparison can be made between the member names or the member contents. Only elements which are present in the first library entity are compared.

If the entities being compared are not equal, the librarian issues message L008I and sets a return code of 2.

`Lib=l`

Specifies the names of the libraries to be compared. Comparing is done for all sublibraries in each of the specified libraries. If a sublibrary corresponding to the current sublibrary in the first library is not found in the second library, a message is issued, and comparing continues with the next sublibrary.

`Sublib=l.s`

Specifies the sublibraries to be compared. An equals sign can be coded in the second operand in place of a name which is identical to one in the first operand, for example, instead of:

```
LIB1.SUBLIBA : LIB2.SUBLIBA
```

you can code:

```
LIB1.SUBLIBA : LIB2.=
```

`mn.mt`

Specifies the member(s) to be compared. The sublibraries in which the members reside must be specified in a preceding CONNECT command.

`Data=Directory|Member`

Specifies whether the directories or the member contents are to be compared. The default is DATA = DIRECTORY.

If Directory is specified or the operand is omitted, the names and types of the members which reside in the first library or sublibrary but not in the second are printed on SYSLST.

If Member is specified, the contents of the members of the specified name and type are compared record by record. Comparing stops for a member when the first mismatch is found, and the mismatching records are printed on SYSLST.

If a member is not found in the second sublibrary, a message is issued, and comparing continues with the next member, if any.

Notes:

1. *Only members with the same internal representation can be compared. Phases have a different internal representation from other member types.*

2. *Phases linked in different partitions have different starting addresses, and give a mismatch.*

Punch=Yes | No
: This operand is valid only if you specify DATA = DIRECTORY. If you specify PUNCH = YES, the system generates COPY statements on SYSPCH. The operands of these COPY statements are the names of the members found in the first sublibrary but not in the second sublibrary.

  The Librarian program writes an EOF on SYSPCH at completion of the COMPARE function.

CONNECT (Specify "From" and "To" Sublibraries)

```
CONnect          {Sublib=l.s [:] l.s | ?}
```

The CONNECT command must be used before COPY, MOVE, or COMPARE commands which have a member specification as operand. It provides the names of the sublibraries in which the members are to be found or placed.

The function is similar to that of the ACCESS command, except that CONNECT must be used before commands which require two sublibraries to be specified.

The first operand of the CONNECT command specifies the "from" or first sublibrary required for the following commands, and the second operand the "to" or second sublibrary. Both sublibraries must exist before execution of the command.

If a name in the second operand is the same as a name in the first operand, an equals sign can be coded in place of it in the second operand. For example, instead of:

```
CON S=LIB1.SUBLIBA : LIB1.SUBLIBB
```

you can code;

```
CON S=LIB1.SUBLIBA : =.SUBLIBB.
```

The CONNECT command remains valid until

- Another valid CONNECT command is given
- A sublibrary specified in the command is deleted
- A sublibrary specified in the command is renamed
- A library specified in the command is deleted, either explicitly by a DELETE command, or implicitly by a MOVE or RESTORE command.

If a question mark is specified as operand, the current CONNECT information is displayed on SYSLOG, if the command was entered from there, otherwise on SYSLST.

COPY (Copy Library, Sublibrary or Member)

```
COPy          {Lib=l [:] l ...|Sublib=l.s [:] l.s ...|mn.mt ...}
              [Replace={Yes|No}]
              [List={Yes|No}]
```

The COPY command is used to copy libraries, sublibraries or members. Copying is allowed between any supported disk devices.

This command can also be used to merge sublibraries. To do this, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
COPY    *.*
```

LIBB.SUB2 will now contain all the members previously present in either of the two specified sublibraries. Members of the same name and type which were present in both sublibraries before the merge, will remain as they were in the "to" sublibrary.

If the common members are to have the same content in the merged sublibrary as they had in the "**from**" sublibrary, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
COPY    *.*  REPLACE=YES
```

`Lib=l:l`

Specifies the libraries to be used in the copy operation. All sublibraries of the library specified first are copied to the library specified second. The to-library must exist before a copy operation can be started.

`Sublib=l.s:l.s`

Specifies the sublibraries to be used in the copy operation. The sublibrary specified first is copied into the library specified second, where it is stored under the sublibrary name specified with the second library. It is therefore possible to change the name of the sublibrary while copying it.

The REUSE attribute of the target sublibrary is retained, if this sublibrary existed. Otherwise, the attribute of the "from" sublibrary is used.

If a name in the second operand is the same as one in the first operand, it can be replaced by an equals sign. For example, instead of

LIBl.SUBLIBA : LIB2.SUBLIBA

you can code

LIBl.SUBLIBA : LIB2.=

mn.mt
Specifies the member(s) to be copied. The "from" and "to" sublibraries must be specified in a preceding CONNECT command.

Replace=Yes|No
Specifies whether copying should be conditional or unconditional.

For **unconditional** copying, specify REPLACE = YES. This means that:

- With COPY LIB = ..., all sublibraries of the from-library are copied into the to-library. In the to-library, any existing sublibraries which have the same names as copied sublibraries are overwritten. Any existing sublibraries of the to-library which are not overwritten by copied sublibraries remain in the to-library;
- With COPY SUBLIB = ..., the to-sublibrary is first emptied, if it already exists, or created, if it does not yet exist. Then all members of the from-sublibrary are copied into the to-sublibrary;
- With COPY mn.mt..., the specified member of the from-sublibrary is copied into the to-sublibrary. If a member of the specified name and type exists in the to-sublibrary, it is overwritten.

For **conditional** copying, specify REPLACE = NO or omit the REPLACE operand. This means that:

- With COPY LIB = ..., a sublibrary of the from-library is not copied if a sublibrary of the same name exists in the to-library. All other sublibraries of the from-library are copied, and all existing sublibraries of the to-library remain in that library;
- With COPY SUBLIB = ..., the from-sublibrary is not copied if the to-sublibrary already exists;
- With COPY mn.mt..., the specified member is not copied if a member if the same name and type exists in the to-sublibrary.

  When you specify members generically, each from-sublibrary member which matches the specification is compared singly with existing members in the to-sublibrary. Therefore, some of the specified members may be copied, while others are not.

LISt=Yes|No
If YES is specified, the names and types of the members copied and those of the corresponding "to" and "from" libraries and sublibraries will be printed on SYSLST.

This is especially useful if the members were specified generically or in the case of conditional copying, with REPLACE = NO.

DEFINE (Define Library or Sublibrary)

```
DEFine {Lib=l ...|Sublib=l.s ...[REUse={Automatic|Immediate}]}
       [Replace={No|Yes}]
```

The DEFINE command is used to create system libraries (SYSRES files), private libraries and sublibraries.

`Lib=l`

Specifies the name(s) of the library or libraries to be created. Library names may be 1 to 7 characters long, and must be alphanumeric. The first character must be alphabetic. Use the names IJSYSR1 to IJSYSR9 to define SYSRES files to be created. The name IJSYSRS must not be used, as it defines the IPLed system.

The device type for SYSRES files is independent of the device type of the IPLed system. The physical location of the library must be specified in DLBL and EXTENT statements. (For a library in VSAM managed space, only the DLBL statement is required.)

If a library is to be created in BAM managed space, the EXTENT statement must have a VOLID and/or logical unit specification.

If a private library is in VSAM managed space, the library space must have been defined previously with an AMS DEFINE CLUSTER command.

The library name in the DEFINE command must be the same as the filename in the DLBL statement. The type code on the DLBL statement must be SD (default) or VSAM.

A library may consist of more than one extent on different volumes, in which case the disk types must be the same. A library cannot, however, be defined on a split-cylinder extent. The minimum size for a private library extent is 1 track on CKD devices or 10 blocks on FBA devices. The maximum number of extents is 16.

For a SYSRES file, only one extent is allowed, and it may not be in VSAM managed space. On the EXTENT statement, specify track 1 as start address and 2 as the minimum number of tracks on CKD devices. On FBA devices, you must specify block 2 as start address and 28 as the minimum number of blocks in the EXTENT statement.

`Sublib=l.s`

Specifies the qualified name(s) of the sublibrary or sublibraries to be created. Sublibrary names may be 1 to 8 characters long, and must be alphanumeric.

Sublibraries are not allocated a fixed amount of space; their size at a given time is the sum of the sizes of their members.

A library can have any number of sublibraries.

REUse=Automatic|Immediate
This operand is needed in a disk sharing environment, or when several tasks share the same sublibrary. It specifies how the space occupied by deleted members is to be freed. REUSE=IMMEDIATE causes the space to be freed as soon as the members are deleted. REUSE=AUTOMATIC causes the space to be freed only when the sublibrary is in use by only one CPU in the sharing environment, and by only one task.

Replace=No|Yes
This operand controls conditional creation of libraries and sublibraries. If you specify NO, or omit the operand, no new library or sublibrary will be defined if one with the specified name already exists.

If you specify YES, any previously defined library or sublibrary is overlaid by the new one. That is, the "to" library or sublibrary is deleted implicitly. If the "to" sublibrary is on a disk volume shared by two or more CPUs, the system issues a message prompting the operator to verify this deletion.

Note that, even if REPLACE=YES is specified, the sublibrary will **not** be re-defined if one with the specified name exists and is **in use**.

DELETE (Delete Library, Sublibrary or Member)

```
DELete    {Lib=l ... | Sublib=l.s ... | mn.mt ... }
```

The DELETE command is used to delete members, sublibraries or libraries.

A sublibrary can not be deleted if it is in use in another VSE partition at the time the DELETE command is entered. If a library or sublibrary has been specified in a LIBDEF statement, the LIBDEF definition has to be dropped or changed accordingly before the DELETE command is entered. Any ACCESS or CONNECT commands which were issued for the deleted sublibraries are dropped.

If the library or sublibrary specified in a DELETE command is on a disk volume shared by two or more CPUs, the system issues a message prompting the operator to verify the deletion.

`Lib=l`
: Specifies the library or libraries to be deleted. The system library IJSYSRS may not be deleted, as it contains the IPLed system. The DELETE command causes only the corresponding VTOC entries to be removed. A library in VSAM managed space can be deleted physically only by using the VSAM Access Method Services DELETE command.

`Sublib=l.s`
: Specifies the sublibrary or sublibraries to be deleted. The system sublibrary IJSYSRS.SYSLIB cannot be deleted.

`mn.mt`
: Specifies the member(s) to be deleted. The sublibrary has to be specified in a preceding ACCESS command.

GOTO (Skip to Label)

```
GOTO    label
```

The Librarian GOTO command has the same function as the job control GOTO command. It causes the Librarian to skip all librarian commands up to the **librarian** LABEL command specified. You must not specify a **job control** label statement.

`label`

Specifies the operand of the LABEL command after which processing is to continue. The first LABEL command with the specified operand is taken, even if there are several with the same operand. The specified LABEL command must come after the GOTO command in the command stream.

Notes:

1. *The command name GOTO cannot be shortened.*
2. *The GOTO command is ignored if entered from SYSLOG.*

INPUT (Read from SYSIPT)

```
Input          SYSIPT
```

The INPUT command cause the librarian to read any following commands from SYSIPT instead of SYSLOG until the end of the current job step. This enables you, for example, to enter librarian commands at SYSLOG and data to be cataloged on SYSIPT.

The command has no effect if SYSIPT is already defined as the input device.

LIST (List Member Contents)

```
List        mn.mt ... [Unit={SYSLST|SYSLOG}] [Format=Hex]
```

The LIST command causes the contents of one or more members to be displayed on SYSLST or SYSLOG.

Phases and dumps are listed in a combined hexadecimal and character string format.

`mn.mt`
: Specifies the member(s) to be displayed. Specify the sublibrary in a preceding ACCESS command.

`Unit=SYSLST|SYSLOG`
: Specifies the output device to be used. If the LIST command is issued from SYSLOG, the default output device is also SYSLOG. If the command is issued from SYSIPT, the default output device is SYSLST.

  Note: If the output is on SYSLOG, only the first 68 positions of each line are displayed.

`Format=Hex`
: Has no effect on the output of phases and dumps. For all other members, specifying FORMAT = HEX results in the character string representation of each record of a member being followed by a two-line hexadecimal translation.

LISTDIR (List Directory Information)

```
{LISTDir|LD}    {Lib=l ...|Sublib=l.s ...|mn.mt ...|SDL}
                [Output={Full|Normal|SHort|STatus}]
                [Unit={SYSLST|SYSLOG}]
```

The LISTDIR (list directory) command is used to display the contents of a directory. The output is a list sorted in alphanumeric collating sequence.

This command can be used to check whether an entity (library, sublibrary or member) exists. If the specified entity does not exist, the librarian sets a return code of 4. This can mean nothing else when set for LISTDIR, so the non-existence of the entity can be tested by using the ON command.

Note: *If the LISTDIR command is processed, while a sublibrary is being updated from another partition, the resulting output may be incorrect. There may be discrepancies between the library directories and the sublibrary information, or between the sublibrary directories and the member information.*

`Lib=l`
Specifies that the directory information of a library or libraries is to be displayed. The information provided includes the directory contents of all sublibraries in the specified libraries.

`Sublib=l.s`
Specifies that the directory contents of a sublibrary or sublibraries is to be displayed. For sublibrary directory listings, the primary sort field is the member type, so that the members are grouped by type.

`mn.mt`
Specifies one or more members. This causes the librarian to display only those parts of the sublibrary directory which are relevant to the named member(s). The sublibrary to be used must be specified in a preceding ACCESS command.

SDL
Specifies that the system directory list is to be displayed. The OUTPUT operand is not applicable when SDL is specified.

`OUTPUT=Full|Normal|SHort|STatus`
Controls the kind and amount of information provided. The specifications FULL, NORMAL and SHORT are applicable for libraries, sublibraries and members. STATUS is applicable only for libraries and sublibraries. The operand may not be specified together with SDL. OUTPUT = NORMAL is the system default.

`Unit=SYSLST|SYSLOG`

Specifies the output device to be used. If the LIST command is issued from SYSLOG, the default output device is also SYSLOG. If the command is issued from SYSIPT, the default output device is SYSLST.

MOVE (Move Library, Sublibrary or Member)

```
Move    {Lib=l [:] l ...|Sublib=l.s [:] l.s ...|mn.mt ...}
        [Replace={No|Yes}] [LISt={No|Yes}]
```

The MOVE command works in a similar way to the COPY command, except that the data which have been moved to a target library or sublibrary are deleted from the from-location after they have been copied. When a sublibrary is moved, either explicitly or as part of a moved library, any ACCESS or CONNECT commands for the "from" sublibrary are dropped.

Libraries, sublibraries, or members can be moved.

As the moving of a sublibrary causes an implicit deletion of the "from" sublibrary, the MOVE function will not be executed if the "from" library:

- has been specified in a previous LIBDEF statement, or
- is in use by another VSE partition or task at the time the MOVE command is entered.

If the sublibrary specified in a MOVE command is on a disk volume shared by two or more CPUs, the system issues a message prompting the operator to verify the implicit deletion.

The system sublibrary IJSYSRS.SYSLIB cannot be moved.

This command can also be used to merge sublibraries. To do this, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
MOVE    *.*
```

LIBB.SUB2 will now contain all the members previously present in either of the two specified sublibraries. Members of the same name and type which were present in both sublibraries before the merge, will remain as they were in the "to" sublibrary.

If the common members are to have the same content in the merged sublibrary as they had in the "**from**" sublibrary, use a command sequence like the following example:

```
CONNECT SUBLIB = LIBA.SUB1 : LIBB.SUB2
MOVE    *.*  REPLACE=YES
```

This will leave the "from" sublibrary empty.

`Lib=l:l`

Specifies the libraries to be used in the MOVE function. All sublibraries of the library specified in the first operand will be moved to the library speci-

fied in the second operand. The to-library must exist before execution of the MOVE function.

The "from" library still exists after the MOVE, but it is empty if all its sublibraries are copied. The latter is always the case when REPLACE = YES is specified.

`Sublib=l.s:l.s`

Specifies the sublibraries to be used in the MOVE function. The sublibrary specified in the first operand will be moved to the library specified in the second operand, where it will reside under the sublibrary name specified in the second operand. The REPLACE operand specifies whether an already existing sublibrary of the same name in the to-library is to be overwritten or not. Any ACCESS or CONNECT commands for the "from" sublibrary are dropped.

The REUSE attribute of the target sublibrary is retained, if this sublibrary existed. Otherwise, the attribute of the "from" sublibrary is used.

`mn.mt`

Specifies the members to be moved. The from- and to- sublibraries must be specified in a preceding CONNECT command. Use the REPLACE operand to control the deletion of duplicate named members in the to-sublibrary.

If a member of the specified name and type existed in the target sublibrary before the command was carried out, its " put-in-sublibrary" time-stamp is left unchanged, and the " last-replaced" time-stamp is updated.

If such a member **did not** exist in the target sublibrary, the " put-in-sublibrary" time-stamp is set, and the "last-replaced" time-stamp is empty.

`Replace=Yes|No`

Specifies whether moving should be conditional or unconditional.

For **unconditional** moving, specify REPLACE = YES. This means that:

- With MOVE LIB = ..., all sublibraries of the from-library are moved into the to-library. In the to-library, any existing sublibraries which have the same names as moved sublibraries are overwritten. Any existing sublibraries of the to-library which are not overwritten by moved sublibraries remain in the to-library.

- With MOVE SUBLIB = ..., the to-sublibrary is first emptied, if it already exists, or created, if it does not yet exist. Then all members of the from-sublibrary are moved into the to-sublibrary;

- With MOVE mn.mt..., the specified member of the from-sublibrary is moved into the to-sublibrary. If a member of the specified name and type exists in the to-sublibrary, it is overwritten.

For **conditional** moving, specify REPLACE = NO or omit the REPLACE operand. This means that:

- With MOVE LIB = ..., a sublibrary of the from-library is not moved if a sublibrary of the same name exists in the to-library. All other sublibraries of the from-library are moved, and all existing sublibraries of the to-library remain in that library;
- With MOVE SUBLIB = ..., the from-sublibrary is not moved if the to-sublibrary already exists;
- With MOVE mn.mt..., the specified member is not moved if a member if the same name and type exists in the to-sublibrary.

  When you specify members generically, each from-sublibrary member which matches the specification is compared singly with existing members in the to-sublibrary. Therefore, some of the specified members may be moved, while others are not.

`LISt=Yes|No`

Specify YES to obtain a listing on SYSLST of the names and types of the members moved, together with the corresponding sublibraries.

This is especially useful if the members were specified generically or in the case of conditional moving, with REPLACE = NO.

If you specify NO, or omit the operand, no such listing is produced.

ON (Set Global Condition)

```
ON  $RC  {>|<|=}  n  {GOTO label|CONTinue}
```

The ON command allows conditional execution of librarian command streams in batch mode.

The Librarian program sets a return code each time processing of a command is completed. On successful completion, this return code is 0; in case of an error it is 2, 4, 8 or 16, depending on the severity of the error. A return code of 16 is set only when a severe error has occurred, and the Librarian program has terminated abnormally. With a return code of 0, 2, 4 or 8, processing of the Librarian commands continues.

The ON command causes the Librarian program to test the return code after each following command. If the comparison of the return code with the specified number yields the value "true", the specified action is taken.

$RC
: Indicates the return code of any following Librarian command

>|<|=
: Indicates whether the specified action is to be taken when the return code is:

  Greater than (>),
  Less than (<) or
  Equal to (=)

  the specified number.

n
: Specifies the number with which the return codes are to be compared. It must be a whole number in the range 0..9999.

GOTO label|CONTinue
: Specify the action to be taken if the specified comparison yields the value "true". The operand GOTO has the same effect as the GOTO command. Processing continues with the command following the specified label. For "label", substitute the name of a LABEL command. This label must occur **after** the ON command.

  If you specify CONTinue, processing continues with the command immediately after the command which caused the return code to be set.

Notes:

1. *The ON command is ignored if entered at SYSLOG.*

2. *There may be up to 30 active ON commands in one sequence of Librarian commands.*

3. *Each ON command remains active until it is overridden by another, or until End-of-File on SYSIPT.*

4. *A new ON command overrides all previous ON commands with the same condition, or with a condition which is included in the new one. For example:*

   ```
   ON $RC > 4 GOTO B
   ```

   overrides:

   ```
   ON $RC > 8 GOTO A
   ```

   but does not override:

   ```
   ON $RC = 4 CONT
   ```

5. *If several ON commands are active, the condition of the most recent one is tested first.*

6. *The system default ON conditions are:*

   ```
   ON $RC>0 CONTINUE
   and
   ON $RC=0 CONTINUE.
   ```

   That is, any return code will allow processing to continue.

PUNCH (Punch Member Contents)

```
PUnch          mn.mt ...    [Format={Old|Noheader}] [Eof={Yes|NO}]
```

The PUNCH command causes the contents of one or more members to be "punched" to the output device SYSPCH. Note that "punched" indicates the record format of the data produced, not to the type of storage medium used for output.

Members of type DUMP, and dumps which have been renamed to a user type, cannot be "punched".

Unless you specify FORMAT = NOHEADER, the members are prepared for re-cataloging. That is, a CATALOG command with the operand REPLACE = YES, or a PHASE statement, is placed before each member. If applicable, EOD and DATA operands are added (see CATALOG command).

`mn.mt`
Specifies the members to be punched. The sublibrary must be specified in a preceding ACCESS command.

`Format=Old`
Allows members to be transferred from a current library to a library of a VSE-Version-1 format. If FORMAT = OLD is specified, the punched members will be prepared for cataloging by the MAINT program into a VSE-Version-1 library.

Notes:

1. *If this operand is specified, the naming conventions for members must conform to the rules for the old MAINT CATALx programs.*
2. *This operand is not applicable for members which have a user type. They are not prepared for re-cataloging.*
3. *This operand is ignored for members of the type PHASE. They are punched as usual.*

`Format=Noheader`
Suppresses the punching of the CATALOG and EOD commands. Only the contents of the member are punched. This operand is ignored for members of type PHASE, and for phases which have been renamed to a user type.

`Eof=Yes|No`

If EOF = YES is specified, or the operand is omitted, an end-of-file indicator (/*) is written to SYSPCH after completion of the command.

If EOF = NO is specified, no end-of-file indicator is written. This makes it possible to collect members from different sublibraries in one punch file. Only the last member to be punched should have EOF = YES (or default).

RELEASE (Release Unused Shared Space)

```
RELease      [SPace]  {Lib=l ...|Sublib=l.s ...}
```

The RELEASE command overrides the library or sublibrary attribute AUTOMATIC. It is needed only when members have been deleted from a library or sublibrary which:

- Is shared by two or more VSE partitions; or
- Resides on a disk device shared by two or more processors.

In these cases, the space formerly occupied by deleted members is normally released only when the library or sublibrary is no longer shared. The RELEASE command causes this space to be released immediately.

Therefore, if several CPUs access the specified library or sublibrary, the RELEASE command must be given in all CPUs, to avoid inconsistency in results.

If you want to use the RELEASE command for the system sublibrary IJSYSRS.SYSLIB, first enter a SET SDL job control command at all CPUs sharing the sublibrary. The SET SDL command must be followed by the names of the phases that have been deleted (explicitly or implicitly), and that have entries in the SDL. This avoids SDL entries pointing to empty spaces.

`SPace`
: may be coded as a reminder to someone not familiar with your job stream that only free space, and not valid members, are affected by the command. Whether you omit or include the operand makes no difference to the function which is carried out.

`Lib=l`
: Specifies in which library the space formerly occupied by deleted members is to be released.

`Sublib=l.s`
: Specifies in which sublibrary the space formerly occupied by deleted members is to be released.

RENAME (Rename Sublibrary or Member)

```
REName      {Sublib=l.s [:] l.s ...| mn.mt [:] mn.mt ...}
```

The RENAME command changes the name and/or type of one or more members, or the names of one or more sublibraries. If the new name already exists, the function is not executed.

`Sublib=l.s`

Specifies the old (first operand) and new (second operand) names of the sublibrary to be renamed. The library name in the second operand must be the same as in the first, and can be specified by an equals sign. The system sublibrary IJSYSRS.SYSLIB cannot be renamed. A sublibrary can only be renamed if it is not in use in another VSE partition. If a sublibrary was specified in a LIBDEF statement, the LIBDEF statement must be dropped or changed accordingly before the RENAME command is entered. Any ACCESS or CONNECT commands for the renamed sublibraries are dropped.

The RENAME S=... command cannot be processed when the specified sublibrary is in use. If the sublibrary to be renamed is on a disk device shared by two or more CPUs, the system issues a message prompting the operator to verify that renaming is to take place.

`mn.mt`

Specifies the old (first operand) and new (second operand) names and types of the members to be renamed. The sublibrary must be specified in a preceding ACCESS command.

Remember that the member names and types must be 1..8 characters long.

If the new name or type are specified generically, the corresponding part of the "old" operand must also be specified generically. If the name or type is not to be changed, you may specify an equals sign for it in the second operand.

Note that renaming to a new type must not imply a change of internal representation. Members of the types PHASE and DUMP have a different internal representation from other types. Renaming into a user type is always possible, but you must be careful not to violate the above restriction when renaming into a predefined type.

If a member of the specified new name and type existed in the sublibrary before the command was carried out, its " put-in-sublibrary" time-stamp is left unchanged, and the " last-replaced" time-stamp is updated.

If such a member **did not** exist in the sublibrary, the " put-in-sublibrary" time-stamp is set, and the "last-replaced" time-stamp is empty.

RESTORE (Restore Backed-up Library, Sublibrary or Member)

```
REStore {Lib=l [:l] ... |
         Sublib=l.s [:l.s] ... |
         l.s.mn.mt [:l.s] ... |
         Oldlib=oldlib [:l.s] ... |
         *}
        Tape={SYSnnn|cuu}
        [LISt={Yes|No}|[Scan={No|Yes}]
        [ID={name|*}]
        [Replace={No|Yes}]
```

The RESTORE command causes the libraries, sublibraries, members or SYSRES files which were backed up using the BACKUP command to be restored to disk. The disk address and extents of the libraries are determined from DLBL, EXTENT and ASSGN information in the label area. If no label information is available for a library, it will not be restored. If several libraries, sublibraries, SYSRES files or members are to be restored, the sequence of the restoring is determined by the sequence as stored on the backup file. The RESTORE function does not reposition the backup tape after reading the specified backup files.

You may also restore members selectively out of backed-up sublibraries. Members with the same names as the restored members will be overwritten in the to-sublibrary or not, depending on the REPLACE operand of the RESTORE command.

When you restore a library or sublibrary, the creation dates in its time-stamp, and in the time-stamps of the entities it contains, are set to the time of the restore. For the members in the library or sublibrary, the "put-in-sublibrary" time-stamps are set, and the "last-replaced" time-stamps are emptied.

Libraries and sublibraries keep the same attributes (for example, REUSE, MSHP) after restore as they had before backup. There is one exception to this rule. If a sublibrary is restored into a target sublibrary which already exists, the REUSE attribute of the **target** sublibrary is kept.

When a library or sublibrary is restored, all members in it receive new date stamps. The "put-in-sublibrary" and "last update" fields both show the time and date of the restore.

A library may not be restored if it is in use.

`Lib=l`

Specifies a library to be restored. Restoring of a library or SYSRES file includes creating it. The location of the libraries to be restored must be outside the extent of the IPLed system. SYSRES files to be restored must not be on the same disk as the IPLed system. The number of extents for a library may change during BACKUP/RESTORE, the disk type may be different, and a library which resided in VSAM managed space before may be restored to non-VSAM space and vice versa. If a SYSRES file with

the name IJSYSRS is to be restored on-line, it must be given a new name, which must be of the form IJSYSR1...IJSYSR9.

`:l`

Specifies a new name for the library to be restored, and, if specified, this new name is used by the librarian to determine the label and extent information to be used in restoring the library. This label and extent information must be given before the RESTORE command, using DLBL and EXTENT job control statements. The rules for coding these statements are the same as those given in the description of the librarian DEFINE command. SYSRES files can be restored as private libraries by giving them new names other than IJSYSRn, but private libraries must remain private.

`Sublib=l.s`

Specifies a sublibrary to be restored. A sublibrary can be restored only into an existing library. Restoring a sublibrary includes creating it, if a sublibrary with the same name does not already exist in the target library. If a sublibrary with the same name does exist in the target library, this will be overwritten or not, depending on the REPLACE operand of the RESTORE command. The system sublibrary of the IPLed system (IJSYSRS.SYSLIB) cannot be deleted, so a backed up version of it must be given a new name when it is restored. The specified sublibrary may be part of a library which was backed up on tape as a whole. It is not necessary to restore the whole library for the sake of a few sublibraries.

`:l.s`

Specifies a new target sublibrary. By specifying this operand you can cause the sublibrary to be restored to a library other than that from which it was backed up, and you can give it a new name. If the library or sublibrary is the same as in the first operand, you may specify an equals sign for it. The REUSE attribute of the target sublibrary is retained, if this sublibrary existed. Otherwise, the attribute of the sublibrary from the tape is used.

`l.s.mn.mt`

Specifies the member(s) to be restored, and in which backed-up sublibrary they are to be searched for. The target sublibrary on disk must be specified in a preceding ACCESS command, or by :l.s.

This operand is positional, that is, it must be coded as the first operand after the command name.

If a member of the specified name and type existed in the target sublibrary before the command was carried out, its " put-in-sublibrary" time-stamp is left unchanged, and the " last-replaced" time-stamp is updated.

If such a member **did not** exist in the target sublibrary, the " put-in-sublibrary" time-stamp is set, and the "last-replaced" time-stamp is empty.

`:l.s`

The specified members will be copied into this sublibrary, dependent on the REPLACE operand of the RESTORE command. They may be restored to the system sublibrary (IJSYSRS.SYSLIB) of the IPLed system. The specified member(s) may be contained in libraries or sublibraries which were backed up on tape as a whole.

`*`

Specifies that all libraries and sublibraries on the backup file are to be restored.

This operand is positional, that is, it must be coded as the first operand after the command name. The "*" operand may also be specified for a backup file containing libraries of pre-Version-2 format if the operand SCAN = YES is specified with it.

`Oldlib=oldlib`

Specifies the filename of a pre-Version-2 format private library, which will be restored as a VSE librarian sublibrary under the name specified in the :l.s operand. If the specified sublibrary already exists, the members of the "oldlib" are merged into it. The REPLACE operand indicates whether members with duplicate names are to exist as they were in the sublibrary (R = NO), or as they were in the "oldlib" (R = YES). If the sublibrary specified in the :l.s operand does not exist, it is created during processing of the RESTORE command. The specification of :l.s may be omitted if the SCAN = YES operand is specified.

If there are several pre-Version 2 libraries in one backup file, use **one librarian RESTORE command for each backup file**. List the names of the "old" libraries you want restored from the file in the OLDLIB = operand of the RESTORE command. A pre-Version 2 backup file contains several libraries when it is created using an EXEC BACKUP statement followed by several BACKUP subcommands.

Note: When replacing existing members, be sure to provide enough space in the target library for both the old and new members. This is necessary because the old members are deleted only after restore of the new ones.

`Tape=SYSnnn|cuu`

Specifies the logical unit or device address of the tape containing the backup file. When using a multi-volume backup tape, specify the alternate tape(s) using the job control statement ASSGN.

If the ID = name operand is not used, the first RESTORE command specifying a newly mounted tape processes the first backup file on the tape. The tape is not repositioned. Any following RESTORE command (without an ID = name specification) begins processing at the start of the next backup file on the tape. This is true even if the first RESTORE does not process all of the first file. When the last backup file on the tape has been processed, or the name specified in the ID operand has not been found, the tape is positioned at the tapemark before the End-of-Backup

record. (For details, see the description of the librarian command BACKUP).

`LISt=Yes|No`

If you specify LIST = YES, the names of the restored libraries, sublibraries and members will be printed on SYSLST. If LIST = NO is specified no listing will be produced. If libraries or sublibraries are specified in the first operand, the default is LIST = YES; if members are specified, the default is LIST = NO.

`SCan=Yes|No`

If you specify SCAN = YES, the restore function is not performed. Instead, information about the data on the backup file is printed on SYSLST. The kind of information provided depends on what you have specified in the first operand, as follows:

If * is specified, the names of all libraries and sublibraries found on the backup file are printed. If specific library, sublibrary or member names were used in the first operand, their names are printed, and a message informs you whether or not they are present on the backup file.

The information on libraries and sublibraries also contains the space requirements for all supported disk devices. This enables you to provide appropriate DLBL and EXTENT information if required.

If you specified generic member names in the first operand, all member names which match the generic specification will be printed.

SCAN = YES can be used for private libraries in pre-Version-2 format on backup tapes. In this case, an estimate of the space required on all supported disk devices is given.

`ID=name|*`

Specifies the name of the backup file to be searched for. The name must have been specified in the BACKUP command which created the file. It may consist of 1 to 16 characters enclosed in quotes. If only alphanumeric characters are used, the quotes may be omitted.

The backup is searched from the current tape position until the specified file is found, or the last backup file has been reached, that is, until an End-of-Backup record is encountered.

If you specify ID = *, the entire backup tape is searched from the current position on.

The ID operand is not applicable if libraries from pre-Version 2 backup tapes are to be restored.

REPLACE=YES|NO

Controls the restoring of backed-up libraries and sublibraries and the merging of pre-Version 2 libraries and members into existing ones. If REPLACE = NO is specified, or the operand is omitted, the entities of the backed-up library or sublibrary are not restored if an entity with the same name already exists in the target entity. With REPLACE = YES, the entire backed-up library or sublibrary, or all specified members, are restored regardless of whether there are duplicate names.

The stand-alone version of this function restores a single SYSRES file. If the backup file contains more than one SYSRES file, one can be selected. Any private libraries on the same backup file can only be restored on-line. For information on how to run the stand-alone version of the RESTORE function, see *VSE/Advanced Functions System Management Guide.*

TEST (Test Library Integrity)

```
Test  {Lib=1 ... [Area={Space|All}][Repair={Yes|No}] |
       Sublib=1.s ... |
       mn.mt |
       Trace={Space|IO|Buffer|LEVEL1|LEVEL2|ALL|OFF]}}
      [Unit={SYSLST|SYSLOG}]
```

The TEST command should be used only on request of IBM service personnel. It checks the structure and contents of a library, sublibrary or member for consistency and correctness, and to provide a trace function for librarian services at different levels.

If TEST detects any inconsistency or incorrectness in a library, sublibrary or member, the librarian sets a return code of 2.

To use the TEST command on a resource protected by the Access Control Function, you need READ access to the affected library.

To ascertain a possible library problem, follow this procedure:

1. Run TEST LIB=1 for the library suspected of causing the problem.
2. If the TEST output does not show error lines, the problem was not caused by this library.

   If the TEST output shows errors, then:

3. Run BACKUP and RESTORE for the library.
4. Run TEST LIB=1 again for the same library.
5. If the TEST output does not show error lines, the problem is probably solved.

   If the TEST output does show errors again, then:

6. Contact your support center. There is probably a system error.

UPDATE (Alter Member Contents)

```
Update      mn.mt  [SAve=mn.mt] [SEquence={10|n|FS|NO}]
            [Column=start [:] end]
```

The UPDATE command allows you to modify the contents of a member by adding, deleting or replacing lines.

If you wish, you can save the unmodified version under a new name or type or both.

The command applies to all member types which can be created by the CATALOG command.

`mn.mt`

Specifies the member to be updated. The sublibrary must be specified in a preceding ACCESS command.

`SAve=mn.mt`

Specifies that the unmodified version of the member is to be saved under the name and type specified. If a member with the same name and type already exists in the sublibrary, the UPDATE function will not be carried out. If a new type is specified, it must not imply a change in the internal representation of the member.

`SEquence=n|FS|NO`

Controls the resequencing of the member being updated. FS may be specified only if the sequence field in the member is numeric without leading blanks.

n must be a decimal number from 1 to 999, and specifies the increment between line numbers which will be used for resequencing. The first line will be given the value n.

FS specifies fixed sequence; the current line numbers will not be changed. The updates are checked to ensure that a valid sequence is retained.

NO specifies that the order of the records in the member will not be checked. The updates must be supplied in ascending order. The sequence number may consist of any alphanumeric characters. If it is shorter than the length specified in the COLUMN operand, it must be padded on the right with blank characters. Sequencing is not checked.
If you omit the operand, SEQUENCE = 10 will be assumed by default.

`Column=start:end`

Specifies the start and end of the sequence field in the member. This may be located anywhere within the line, and can be 1 to 8 characters long. The following defaults will apply if the operand is omitted:

if SEQUENCE = n
or SEQUENCE = NO COLUMN = 77:80

if SEQUENCE = FS COLUMN = 73:78

UPDATE Subcommands

The UPDATE subcommands)ADD,)DEL and)REP have as their operand the sequence number(s) of the line(s) of the member to which the subcommand applies. If the member you wish to update has no sequence numbering, you can cause the librarian to generate it by issuing the following command sequence:

```
EXEC LIBR                          call the librarian program
ACCESS S=lib.sublib                access appropriate sublibrary
UPDATE membername.memberty         "update" the member without inpı
)END                               end update, causing renumbering
```

These commands cause the lines of the specified member to be sequence numbered in columns 77 to 80 using an increment of 10, starting at 10. These are the default values used by the librarian. Should you wish to use a different increment or a different length or position for your sequence numbering, specify the appropriate values in the SEQUENCE and COLUMNS operands of the UPDATE command.

)DEL and)REP act on the specified line(s),)ADD inserts the provided data **after** the specified line. Specification is by means of the sequence number.

The updates applied with one UPDATE command must be in ascending order. That is, the first operand of an)ADD,)DEL or)REP subcommand must be **greater than** the first operand of any preceding)ADD,)DEL or)REP subcommand.

The length and position within the line of the field containing this sequence number (the sequence field) must be specified in the COLUMN operand of the UPDATE command, and the length of the seq-no operands of the subcommands must not exceed this specification.

When using SEQUENCE = n or FS on the UPDATE command, you may omit leading zeros in the subcommand operands. With SEQUENCE = NO, the sequence number or character string specified is padded on the right with blank characters if necessary.

If you are adding or replacing lines in a member using SEQUENCE = NO, the input lines following the)ADD or)REP subcommand must contain the sequence number or character string.

When using)ADD or)REP with SEQUENCE = n, you need not provide sequence numbering in the input lines, and if you do so, you may omit leading zeros. The member in which you want add or replace lines must, of course, be sequence numbered.

Note: Update subcommands may start with a), or with a) and one blank character, for example:

```
)ADD ...
or
) ADD ...
```

)ADD (Add Line to Member)

The)ADD subcommand indicates that the lines following it are to be added to the member specified in the UPDATE command.

Operation	Operands
)ADD	seq-no

`seq-no`
Represents the sequence number of the line in the member after which the new lines are to be added. To add new lines in front of the first line of the member, code 0 for seq-no. Adding lines in front of the first line is not possible if you have specified SEQUENCE = NO in the UPDATE command.

If you specify sequence numbers in the input lines following this subcommand, be sure that their position and length correspond with the COLUMN operand in the UPDATE command.

)DEL (Delete Line form Member)

The)DEL subcommand causes the deletion of lines from the member specified in the UPDATE command.

Operation	Operands
)DEL	first-seq-no[,last-seq-no |,*]

`first-seq-no, last-seq-no`

Represent the sequence numbers of the first and last lines of a section to be deleted. If **last-seq-no** is not specified, the line represented by **first-seq-no** is the only line deleted. To delete all lines from the line specified in first-seq-no to the end of the member, specify * in place of last-seq-no.

)END (Finish Update)

The subcommand)END has no operand. Issue it to inform the system that input for the required UPDATE function is complete.

Operation	Operands
)END	none

)REP (Replace Line in Member)

The)REP subcommand indicates that lines following it are to replace existing lines in the member specified in the UPDATE command.

Operation	Operands
)REP	first-seq-no[,last-seq-no |,*]

`first-seq-no,last-seq-no`
: Represent the sequence numbers of the first and last lines of a section to be replaced. The **first-seq-no** must not be zero. Any number of new lines can be added to a member when a section is replaced. The number of lines added need not equal the number of lines being replaced. To replace all lines up to the end of the member, specify * in place of last-seq-no.

If you specify sequence numbers in the following input lines, be sure that their position and length correspond with the COLUMN operand on the UPDATE command.

/. (Label Statement)

```
/. label
```

The LABEL statement is used in conditional command streams. It marks a point in the command stream up to which commands can be skipped using a GOTO command or the GOTO action of an ON command. The Librarian LABEL statement corresponds to the job control LABEL statement.

`/.`
Indicates that this is a label. These characters must be in positions 1 and 2 of the command followed by at least one blank character.

`label`
Specifies the name of the label. This must be 1 to 8 alphanumeric characters.

You must use this name in the label operand of the GOTO command which addresses the label.

Note: The /. statement is ignored if entered from SYSLOG.

/+ (End-of-DATA)

The End-of-Data statement for input to the librarian CATALOG command is /+. This is used for data of all types, whether procedures, source code or other user data.

```
/+ [comments]
```

Column 1 contains a slash (/) and column 2 a plus sign (+). Column 3 must be blank.

The /+ statement is also used by job control as an End-of-Procedure statement. If this is the last statement of a member to be cataloged, the librarian recognizes the end of the input data and includes an End-of-Procedure mark at the end of the cataloged member.

If, however, a /+ statement must be included as part of a member to be cataloged, the CATALOG step for this member must have an End-of-Data statement other than /+. You can define the alternative End-of-Data statement in the EOD operand of the librarian CATALOG command. The need for an alterna-

tive EOD statement arises, for example, when you catalog a procedure which itself contains librarian CATALOG commands.

/* or END (Librarian End-of-Session)

These statements indicate to the librarian program that no more librarian commands follow.

Format for SYSIPT:

```
/*
```

Format for SYSLOG:

```
END
```

The SYSIPT format is used when a librarian job stream for batch execution is being prepared. The SYSLOG format is used to end an interactive librarian session at the system console.

When the librarian program receives either form of the End-of-Session command, gives control of the partition to the job control program. The highest return code set during the librarian session or step is passed to job control. You can test this return code using the job control statements IF and ON.

Section 6. Edited Macro Service Program (ESERV)

The ESERV program de-edits assembler macros from sublibrary members of type E, and punches and/or displays the macros in source format. It is also possible to update the source form of the macro before output. For further information on the use of ESERV, see *VSE/Advanced Functions System Management Guide.*

To run ESERV, the following job control statements are necessary:

```
// ASSGN SYSPCH,cuu
// LIBDEF SOURCE,SEARCH=lib.sublib
// EXEC ESERV
```

(If an appropriate ASSGN and LIBDEF are already valid for the partition in which ESERV is to run, these statements need not be included.)

The EXEC statement may be followed by a GENEND or GENCATALS control statement. If neither is used, GENCATALS is assumed by default.

The next control statement is required. It may be DSPLY, PUNCH or DSPCH.

To verify the macro in question, or to update it before it is displayed or punched, one or more of the following statements may then be entered:

) ADD to add statements at a specified point in the macro;

) COL to define the location and length of the sequence number field;

) DEL to delete specific statement(s);

) REP to replace specified statement(s);

) RST to indicate that sequence numbering starts at a lower number than that of the specified preceding statement within a macro;

) VER to verify the contents of a specified statement.

) END to indicate the end of update statements. This statement **must** be used if any of the above verify or update statements are used.

Column 1 of these update statements must contain a right parenthesis, and there must be one blank before and at least one blank after the operation code.

If the update commands are entered after a DSPLY, PUNCH or DSPCH statement which specifies several members, they are applied to the last-named member.

/* must be entered after the last ESERV control statement to indicate end-of-input on SYSIPT.

The syntax of all ESERV control statements is described in the following section. For further details of the update function, see the *Guide to the DOS/VSE Assembler*.

ESERV Control Statements

GENEND, GENCATALS (Specify Macro Output Format)

If used, one of these statements must follow the EXEC ESERV statement directly. If neither is used, GENCATALS is assumed.

These statements must start in or after **column 2**.

Operation	Operands
GENEND	none

This causes ESERV to place an END and a /* statement immediately after the de-edited macro on SYSPCH, so that it can be used as SYSIPT for the assembler.

Operation	Operands
GENCATALS	none

This causes a librarian catalog statement for a member "bookname.A" (or "bookname.D," if the OPTION SUBLIB = DF is in effect on the system) to be placed before each macro, and a /* to be placed after each macro. This allows the SYSPCH output to be used as SYSIPT for the librarian program to catalog the de-edited macro with the appropriate member type.

DSPLY, PUNCH, DSPCH (Specify Output Destination)

These statements must follow the GENEND or GENCATALS statement. They can act on one or more edited macros in one ESERV run.

The statements must start in or after **column 2**.

Operation	Operands
DSPLY PUNCH DSPCH	type.bookname[,type.bookname,...]

DSPLY
: De-edits macros and displays them on SYSLST.

PUNCH
: De-edits macros and punches them on SYSPCH.

DSPCH
: De-edits macros, punches them on SYSPCH and displays them on SYSLST.

type.bookname
: Specifies the member name and member type of the macro to be de-edited. The sublibrary in which this member is to be searched for must be specified previously in a LIBDEF job control statement.

Verify and update control statements may follow these statements. If several macros were specified, the verify and update functions are applied to the **last** one specified.

) ADD (Add Statement to Macro)

The) ADD statement indicates that the source statement(s) following it are to be inserted in the macro, and specifies at what position.

Operation	Operands
) ADD	seq-no[+rel]

`seq-no`

Indicates the sequence number of the macro definition statement **after** which the new source statements are to be inserted. The sequence number is 1..8 decimal digits, as specified in the) COL statement.

`+rel`

Indicates the position of the macro statement after which the new statements are to be added, **relative** to the statement specified in "seq-no."

seq-no[+rel] must be greater than the **(last-)seq-no[+rel]** of any preceding update control statement. An) ADD statement may, however, reference the same statement as an immediately preceding) VER statement.

) COL (Control Macro Statement Numbering)

The) COL statement specifies the position of the sequence number within the source statements of the de-edited macro. If it is used, this statement must immediately follow the DSPLY, PUNCH or DSPCH statement to which it applies.

Operation	Operands
) COL	startcol,n

`startcol`
Specifies the column in which the sequence number is to start. It must be a decimal integer; the valid range is 73..80. The default value is 73.

`n`
Specifies the length of the sequence number. It must be a decimal integer; the valid range is 1..8. The default value is 6.

) DEL (Delete Statement from Macro)

The) DEL statement causes deletion of one or more source statements from the de-edited macro.

Operation	Operands
) DEL	first-seq-no[+rel][,last-seq-no[+rel]]

`first-seq-no`
: Specifies the sequence number of the **first or only** source statement to be deleted from the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the) COL statement.

`last-seq-no`
: Specifies the sequence number of the **last** of a series of source statements to be deleted from the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the) COL statement.

+rel
: Indicates the position of the first or last statements to be deleted, **relative** to the statement with the specified sequence number. **rel** must be a decimal integer, and may be 1..4 digits long.

first-seq-no[+rel] must be greater than the **(last-)seq-no[+rel]** of any preceding update control statement. A) DEL statement may, however, reference the same statement as an immediately preceding) VER statement.

) END (Finish Macro Update)

The) END statement indicates the end of ESERV update or verify statements on SYSIPT. It is required in every update run.

Operation	Operands
) END	none

) REP (Replace Statement in Macro)

The) REP statement indicates that the following source statements on SYSIPT are to replace one or more existing statements in the de-edited macro.

Operation	Operands
) REP	first-seq-no[+rel][,last-seq-no[+rel]]

`first-seq-no`
: Specifies the sequence number of the **first or only** source statement to be replaced in the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the) COL statement.

`last-seq-no`
: Specifies the sequence number of the **last** of a series of source statements to be replaced in the de-edited macro. The sequence number is 1..8 decimal digits, as specified in the) COL statement.

`+rel`
: Indicates the position of the first or last statements to be replaced, **relative** to the statement with the specified sequence number. **rel** must be a decimal integer, and may be 1..4 digits long.

first-seq-no[+rel] must be greater than the **(last-)seq-no[+rel]** of any preceding update control statement. A) REP statement may, however, reference the same statement as an immediately preceding) VER statement.

) RST (Change Macro Statement Numbering)

The) RST statement causes the sequence numbers of the statements in a macro definition to restart at a lower number after the statement specified in the) RST operand.

Operation	Operands
) RST	seq-no[+rel]

`seq-no`
Specifies the sequence number of the source statement after which the new series of sequence numbers starts.

`+rel`
Indicates the position of the statement after which the new series of numbers is to start, **relative** to the statement with the specified sequence number. **rel** must be a decimal integer, and may be 1..4 digits long.

If an) ADD,) DEL or) REP operation is performed on the **last** statement in a sequence number series, the) RST statement must reference the first statement **after** the statement specified in the ADD, DEL or REP.

) VER (Verify Contents of Macro Statement)

The) VER statement causes all or part of the specified source statement in the de-edited macro to be verified against the contents of the statement following) VER statement on SYSIPT.

The first string of characters, of the length specified in the **len** operand, are compared. If the strings do not match, an error message is issued.

Operation	Operands
) VER	seq-no[+rel],len

`seq-no`

Specifies the sequence number of the source statement to be verified in the de-edited macro. It must be a decimal integer 1..8 digits long, as specified in the) COL statement.

`+rel`

Indicates the position of the source statement to verified, relative to the statement specified in "seq-no." It must be a decimal integer, 1..4 digits long. **seq-no[+rel]** must be greater than the **(last-)seq-no[+rel]** of any preceding update statement.

`len`

Specifies the length of the field to be verified. It must be a decimal integer; the valid range is 1..80.

Section 7. System Buffer Load (SYSBUFLD)

SYSBUFLD is a service program which loads UCBs (universal character set buffers) and FCBs (forms control buffers) of VSE supported line printers, except 3800. The buffer image phases for the UCB load operation must reside in a sublibrary; for the FCB load operation, the phases may reside in a sublibrary or the information may be read from SYSIPT (following the FCB control statement). For information on how to use SYSBUFLD under VSE/POWER, refer to *VSE/POWER Installation and Operations Guide.*

SYSBUFLD is executed in your job stream whenever it is necessary to change the contents of the UCB and/or FCB of a specific printer. Execution is initiated with the statement:

```
// EXEC SYSBUFLD
```

When the access control function is active (SEC = YES was specified in the IPL command SYS), note that:

- The UCB and FCB phase names you use must start with '$$B';
- The phases must be cataloged in a protected library;
- You must establish access to this.

Control Statements

Once started, SYSBUFLD reads, from SYSIPT, control statements which identify the printer and specify the buffer image to be loaded. The last statement is followed by a /* statement. The control statements are BANDID, FCB and UCB.

BANDID

The BANDID control statement can be used only for 4248 printers. Use it to ensure that the correct band for the output of the following job is mounted.

Operation	Operands
BANDID	SYSxxx,[band-id][,FOLD[,NOCHK]

SYSxxx
: Is the logical unit assigned to the 4248 printer for which the SYSBUFLD run is being performed. For SYSxxx, specify SYSLST or the programmer logical unit assigned to the printer.

band-id
: Specifies the identifier of the required print band.

  If you omit the operand, no print-band verification takes place. This is meaningful only if you want to change the output characteristics to FOLD or NOCHK.

  If you specify a band identifier, a console message tells the operator which band is needed, if this band is not already mounted. The identifier of the band needed is repeated on the printer panel.

FOLD
: Causes lowercase characters to be printed as uppercase characters.

NOCHK
: Causes a data check to be suppressed if it results from a mismatch between a print character and the band-image buffer.

FCB

The FCB control statement is used to load the FCB of a printer.

Operation	Operands
FCB	SYSxxx[,phasename[,NULMSG]]

SYSxxx
: Identifies the printer whose FCB is to be loaded. The printer must be a 3203, 4248, 5203, or PRT1 device; SYSxxx must be SYSLST or a programmer logical unit; SYSxxx may be SYSLOG if, for any reason, SYSLOG has to be assigned to a line printer.

phasename
: Specifies the name of the phase which contains the required buffer image. If the phase name is omitted, an FCB image from SYSIPT is assumed.

NULMSG

Indicates that the 80-character verification message, which follows the buffer image in the specified phase, is not to be printed. If this operand is omitted, the program loads the FCB, skips to channel 1, prints the last 80 characters of the phase, and again skips to channel 1.

If the FCB is loaded from SYSIPT, a verification message cannot be defined in the buffer image phase.

Note: Loading an FCB image phase for horizontal copy control does not turn on the horizontal copy function of IBM 4248 Printers.

UCB

The UCB control statement is used to load the UCB of a printer. For 4248 printers, use the SYSBUFLD control statement BANDID.

Operation	Operands
UCB	SYSxxx,phasename[,FOLD][,NOCHK][,NULMSG]

SYSxxx

Identifies the printer whose UCB is to be loaded. The printer must be a PRT1 device; SYSxxx must be SYSLST or a programmer logical unit; SYSxxx may be SYSLOG if for any reason, SYSLOG has to be assigned to a line printer.

phasename

Specifies the name of the cataloged phase which contains the required buffer image information.

FOLD

Indicates that the UCB is to be loaded with the folding operation code to cause printing of uppercase characters for lowercase bit combinations.

NOCHK

Suppresses data checks resulting from an attempt to print unprintable characters (during subsequent use of the printer, not during SYSBUFLD).

NULMSG

Indicates that the 80-character verification message which follows the buffer image in the specified phase is not to be printed. If this operand is omitted, the program loads the UCB, skips to channel 1, prints the last 80 characters of the phases, and again skips to channel 1.

Buffer Load Phases

The following standard UCB and FCB image phases are provided in the system.

Standard Buffer Image Phases

Printer	UCB		FCB
	Phase Name	Train Type	Phase Name
1403U	$$BUCB4	AN	-
3211 (PRT1)	$$BUCB	A11	$$BFCB
3203-4/5 (PRT1)	$$BUCB00	AN	$$BFCB00
3289-4 (PRT1)	$$BUCB10	64-character belt	$$BFCB10
3262 (PRT1)	$$BUCB22	64-character belt	$$BFCB22
4245 (PRT1)	See Note	See Note	$$BFCB23
4248 (PRT1)	See Note	See Note	$$BFCB
4248 (Native Mode)	See Note	See Note	$$BFCBWM

Figure 7-1. Standard Buffer Load Phases

Note: For these printers, the correct UCB for the mounted train is loaded automatically by microcode.

The standard FBC image phases are designed for 12 inch forms and a line density of 6 lines per inch (lpi), with:

Channel 1 on line 5
Channel 9 on line 56
Channel 12 on line 66
End of page on line 72.

Additional UCB Images

The following additional UCB images (including copies of the standard images - printed bold in the table) are supplied in object format:

Printer	Module Name	Train Type
1403U	IJBTRAN	AN or HN
	IJBTRGN	GN
	IJBTRONA	ONA
	IJBTRPAN	PCS-AN
	IJBTRPHN	PCS-HN
	IJBTRPN	PN
	IJBTRQNC	QNC
	IJBTRQN	QN
	IJBTRRN	RN
	IJBTRSN	SN
	IJBTRTN	TN
	IJBTRYN	YN
	IJBTRALA	ALA
3203-4,3203-5	IJBTVAN	AN or HN
	IJBTVGN	GN
	IJBTVOAA	OAA
	IJBTVOAB	OAB
	IJBTVODA	ODA
	IJBTVONA	ONA
	IJBTVPAN	PCS-AN
	IJBTVPHN	PCS-HN
	IJBTVPN	PN
	IJBTVQNC	QNC
	IJBTVQN	QN
	IJBTVRN	RN
	IJBTVSN	SN
	IJBTVTN	TN
	IJBTVYN	YN
	IJBTVALA	ALA
3211	IJBTRA11	A11
	IJBTRG11	G11
	IJBTRH11	H11
	IJBTRP11	P11
	IJBTRT11	T11
3262	IJBNA48	48-character
	IJBNA64	64-character
	IJBNA96	96-character
	IJBNAHI	High-performance 63-character set
3289-4	IJBBE48	48-character
	IJBBE64	64-character
	IJJBBE96	96-character
	IJB116CF	116-character (Canadian French)
	IJB128KK	128-character (Katakana)

Figure 7-2. Additional UCB Images

These additional UCB images must be link-edited before you load them. Any valid phasename may be assigned to them.

Automatic Buffer Loading During IPL

The IPL routine automatically loads the UCB and/or FCB of each operational printer with the standard image phases for the device, for example with $$BUCB00 and $$BFCB00 for the 3203-4 or 3203-5 printer. If you normally have some other train or belt mounted on the printer(s), link-edit the appropriate object-type image, so that automatic UCB loading can use the correct information. For example, if you normally use a TN chain on a 1403U printer, link-edit IJBTRTN with the phase name $$BUCB4.

For support (at IPL) of the dualling feature and of trains/belts and forms not covered by the standard buffer image phases, generate your own images phases as described below and catalog them under the standard phase names.

Creating Your Own UCB/FCB Image Phases

If you use non-standard trains or belts on your printer, or if you use special forms, you must:

1. Create the necessary UCB/FCB image phases, using the information given in Figure 7-3 on page 7-7;

2. Assemble and link-edit the new phases;

3. Ensure that the phases are stored on an accessible sublibrary.

   If they are to be loaded automatically at IPL, link-edit them with the phase names of the standard images. Catalog them in the system sublibrary IJSYSRS.SYSLIB.

Printer	Buffer	Bytes	Contents
1403U	UCB	1-240 241-320	Train image Verification message
3211 (PRT1)	UCB	1-432 433-447 448-511 512 513-592	Train image Zeros Associative field Zero Verification message
	FCB (no index- ing byte)	1-255 or 1-112, or 1-180, or 1-192 256-335	FCB image Verification message
	FCB (with in- dexing byte) See Note	1-256, or 1-181 257-336	FCB image Verification message
3203-4/5 (PRT1)	UCB	1-240 241-304 305-512 513-592	Train image Associative field Zeros Verification message
	FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image Verification message
3262 (PRT1)	UCB	1-288 289-512 513-592	Belt image Zeros Verification message
	FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image Verification message
3289-4 (PRT1)	UCB	1-256 257-512 513-592	Font offset table Zeros Verification message
	FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image Verification message

Note: If the indexing control byte is specified in the FCB image phase for a PRT1, it is used only if the printer is a 3211; otherwise, it is ignored.

Figure 7-3 (Part 1 of 2). Formats of UCB/FCB Image Phases

Printer	Buffer	Bytes	Contents
4245 (PRT1)	FCB FCB FCB	1-255, or 1-112, or 1-180, or 1-192 256-335	FCB image Verification message
4248 (PRT1 Mode)	UCB	1-432 433-447 448-511 512 513-592	Train image Zeros Associative field Zero Verification message
	FCB (no indexing byte)	1-255 or 1-112, or 1-180, or 1-192 256-335	FCB image Verification message
	FCB (with indexing byte)	1-256, or 1-181 257-336	FCB image Verification message
4248 (Native Mode*)	FCB	1-260 261-340	FCB image Verification message

* For the control characters to be used in the phase, see the IBM publication 4248 Printer Model 1 Description.

Figure 7-3 (Part 2 of 2). Formats of UCB/FCB Image Phases

Legend:

Train image
: The hexadecimal equivalent of all characters on the train (chain or belt).

FCB image
: Control characters for the FCB, as defined in "FCB Characters" on page 7-9.

Font offset table
: Table containing one entry for each possible hexadecimal combination from X'00' to X'FF'. Each entry contains the hexadecimal displacement of the appropriate printed character from the home position (X'00') of the belt (or one of the home positions if the character set is repeated on the belt).

  The home position has a displacement of X'00'. Unused hexadecimal combinations have a displacement of X'80' and cause a data check on printing. Combinations X'00' (null) and X'40' (blank) have a displacement of X'7F' and suppress printing at the corresponding position on the line.

Associative field
: This is used for suppressing invalid characters. For further details, see the hardware description of the respective printers.

Verification message
: An 80-byte message which is printed after the load (unless NULMSG is specified). This message must be included in the image phase (even if all 80 bytes contain X'40').

Loading the FCB Using SYSIPT

When the FCB is loaded using SYSIPT, there is no verification message. You supply the FCB image phase in card format immediately behind the FCB SYSxxx control statement. Each card column corresponds to a line on the forms to be used, that is card 1, column 1 refers to line 1; card 2, column 1 refers to line 81, and so on. The codes to be punched (or written to a diskette) are described in "FCB Characters."

Note that the 3211 indexing control byte cannot be specified if the FCB is loaded using SYSIPT. FCBs for IBM 4248 Printers cannot be loaded using SYSIPT.

FCB Characters

The FCB characters for the phase and SYSIPT formats are shown in the table below.

Channel	Phase format (Hex)	SYSIPT punch code
None	00	blank
1	01	1
2	02	2
3	03	3
4	04	4
5	05	5
6	06	6
7	07	7
8	08	8
9	09	9
10	0A	A
11	0B	B
12	0C	C
End of FCB	10	X
8 lines per inch	10	*

End of FCB
: In the phase format for a PRT1 printer, it is possible to combine a channel character and the end-of-FCB character in the last buffer position used, if these two conditions coincide. For example, if channel 12 is on the last line of the form, the X'10' for end-of-FCB and the X'0C' for channel 12 can be combined by coding X'1C'. This is not possible for a

non-PRT1 printer, nor is it possible when the FCB is loaded using SYSIPT.

8 lines per inch
This can be specified only for a PRT1 printer; all other printers have a hardware switch for selection of line density. If used, the * (SYSIPT format) or X'1x' (phase format, where x may be a channel specification character) must be specified in the first column of the first (or only) card or in the first buffer position, respectively.

Examples of FCB Image Phases

1. Example of the source code for a PRT1 printer FCB image phase (LPI = 6, paper size = 12 inches, used FCB positions: 12x6 = 72):

```
// JOB FCB6PRT1
// OPTION CATAL
   PHASE FCB6PRT1,*
// EXEC ASSEMBLY
   START 0
   DC    XL4'00'          FCB POSITIONS 1 TO 4
   DC    X'01'            CHANNEL 1 ON LINE 5
   DC    XL24'00'         FCB POSITIONS 6 TO 29
   DC    X'05'            CHANNEL 5 ON LINE 30
   DC    XL41'00'         FCB POSITIONS 31 - 71
   DC    X'1C'            END OF FORMS AND CHANNEL 12
*                         ON LINE 72
   DC    XL183'00'        FCB POS. 73 - 255 ZEROS
   DC    CL80'PHASE FCB6PRT1 LOADED'
*            LPI=6, PAPERSIZE=12 INCHES,
*            CHANNEL 1/5/12 ON LINE 5/30/72
   END
/*
// EXEC LNKEDT
/&
```

2. Example of a PRT1 printer FCB image phase (LPI = 8, paper size = 12 inches, used FCB positions: 12x8 = 96):

```
// JOB FCB8PRT1
// OPTION CATAL
   PHASE FCB8PRT1,*
// EXEC ASSEMBLY
   START 0
   DC     X'10'              LPI=8
   DC     XL3'00'            FCB POSITIONS 2 TO 4
   DC     X'01'              CHANNEL 1 ON LINE 5
   DC     XL54'00'           FCB POS. 6 TO 59
   DC     X'09'              CHANNEL 9 ON LINE 60
   DC     XL29'00'           FCB POS. 61 TO 89
   DC     X'0C'              CHANNEL 12 ON LINE 90
   DC     XL5'00'            FCB POS. 91 - 95
   DC     X'10'              END OF FORMS ON LINE 96
   DC     CL159'00'          FCB POS. 97 - 255 ZEROS
   DC     CL80'PHASE FCB8PRT1 LOADED'
*            LPI-8, PAPER SIZE=12 INCHES,
*            CHANNEL 1/9/12 ON LINE 5/60/90
   END
/*
// EXEC LNKEDT
/*
```

Appendix A. Linkage Editor Summary

Format of the ESD Statement

Card Columns **Content**

1
Multiple punch (12-2-9). Identifies this as a loader statement.

2 - 4
ESD -- External Symbol Dictionary statement.

11 - 12
Number of bytes of information contained in this statement.

15 - 16
External symbol identification number (ESID) of the first SD, PC, CM or ER on this statement. Relates the SD, PC, CM or ER to a particular control section.

17 - 72
Variable information.

8 positions
Name

1 position
Type code hex '00', '01', '02', '04', '05', or '0A' to indicate SD, LD, ER, PC, CM, or WX, respectively.

3 positions
Assembled origin

1 position
Blank

3 positions
Length, if an SD-type, CM-type, or a PC-type. If an LD-type, this field contains the external symbol identification number (ESID) of the SD containing the label.

73 - 80
May be used by the programmer for identification.

Format of the TXT Statement

Card Columns	Content
1	Multiple punch (12-2-9). Identifies this as a loader statement.
2 - 4	TXT -- Text statement.
6 - 8	Assembled origin (address of first byte to be loaded from this statement).
11 - 12	Number of bytes of text to be loaded.
15 - 16	External symbol identification number (ESID) of the control section (SD or PC) containing the text.
17 - 72	Up to 56 bytes of text -- data or instructions to be loaded.
73 - 80	May be used for program identification.

Format of the RLD Statement

Card Columns	Content
1	Multiple punch (12-2-9). Identifies this as a loader statement.
2 - 4	RLD -- Relocation List Dictionary statement.
11 - 12	Number of bytes of information contained in this statement.
17 - 72	Variable information (multiple items). a. Two positions - (relocation identifier) pointer to the ESID number of the ESD item on which the relocation factor of the contents of the address constant is dependent.

b. Two positions - (position identifier) pointer to the ESID number of the ESD item on which the position of the address constant is dependent.

c. One position - flag byte indicating type of constant, as follows:

Bits	Setting and Meaning
0-2	(ignored)
3	0 - a nonbranch type load constant 1 - a branch type load constant
4-5	00 - load constant length = 1 byte 01 - load constant length = 2 bytes 10 - load constant length = 3 bytes 11 - load constant length = 4 bytes
6	0 - relocation factor is to be added 1 - relocation factor is to be subtracted
7	0 - Next load constant has different R and P identifiers; therefore, both R and P must be present. 1 - Next load constant has the same R and P identifiers; therefore, they are both omitted.

The five significant bits of this byte are expanded in the RSERV printout.

d. Three positions - assembled origin of load constant.

73 - 80

May be used for program identification.

Format of the END Statement

Card Columns **Content**

1

Multiple punch (12-2-9). Identifies this as a loader statement.

2 - 4

END

6 - 8

Assembled origin of the label supplied to the assembler in the END statement (optional).

15 - 16

ESID number of the control section to which this END statement refers (only if 6-8 present).

17 - 22

Symbolic label supplied to the assembler if this label was not defined within the assembly.

29 - 32
Control section length (if not specified in last SD or PC).

73 - 80
Not used.

Format of the REP (User Replace) Statement

Card Columns	Content
1	Multiple punch (12-2-9). Identifies this as a loader statement.
2 - 4	REP -- Replace text statement.
5 - 6	Blank.
7 - 12	Assembled address of the first byte to be replaced (hexadecimal). Must be right justified with leading zeros if needed to fill the field and must be equal to or greater than the starting address of the control section (columns 14-16). Note that there is no check to determine if the assembled address is actually within this control section.
13	Blank.
14 - 16	External symbol identification number (ESID) of the control section (SD) containing the text (hexadecimal). Must be right justified with leading zeros if needed to fill the field.
17 - 70	From 1 to 11 4-digit hexadecimal fields separated by commas, each replacing two bytes. A blank indicates the end of information in this statement.
71 - 72	Blank.
73 - 80	May be used for program identification.

External Symbol Dictionary

The external symbol dictionary (ESD) contains control section definitions and inter-module references. Six types of entries are defined in the control dictionary:

ESD Type	Definition
SD	Section definition: provides control section name, assembled origin and length. Generated by a named START or a named CSECT in a source module.
WX	Generated by weak external reference (WXTRN), which has a function similar to EXTRN, except that WXTRN suppresses AUTOLINK. The linkage editor treats WX as an ER, NOAUTO.
PC	Private code: provides assembled origin and length for an unnamed control section.
LD/LR	Label definition: specifies the assembled address and the associated SD of a label that may be referred to by another module. The LD entry is termed LR (Label Reference) when the entry is matched to an ER entry.
ER	External reference: specifies the location of a reference made to another module. ER is generated by EXTRN or a V-type address constant in a source module.
CM	Common: indicates the amount of storage to be reserved for common use by different phases. CM is generated by COM in a source module.

Bibliography

This publication makes reference to the following IBM publications:

VSE/Advanced Functions System Management Guide, SC33-6191

VSE/Advanced Functions Application Programming: Macro User's Guide, SC33-6196

VSE/Advanced Functions Application Programming: Macro Reference, SC33-6197

VSE/Advanced Functions Planning and Installation, SC33-6193

VSE/Advanced Functions Operation, SC33-6194

VSE/System Package, Messages and Codes,

SC33-6181 (if you are using IBM VSE/SP Version 2), or
SC33-6310 (if you are using IBM VSE/SP Version 3)

VSE/Advanced Functions Service Aids, SC33-6195

Guide to the DOS/VSE Assembler, GC33-4024

IBM 3340 Fixed Head Feature User's Guide, GA26-1632

DOS/VSE IBM 3800 Printing Subsystem Programmer's Guide, GC26-3900

VSE/POWER Installation and Operations Guide, SH12-5329

VSE/POWER Remote Job Entry User's Guide, SH12-5328

VSE/VSAM Programmer's Reference, SC24-5145

Index

Special Characters

A

E

F

G

H

I

J

M

P

R

S

T

U

V

W

X

Z

Numerics

IBM Virtual Storage Extended/
Advanced Functions
System Control Statements
Order No. SC33-6198-02

READER'S COMMENT FORM

This form may be used to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Your comments:

Note: **Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.**

If you wish a reply, give your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC33-6198-02

CUT
OR
FOLD
ALONG
LINE

Reader's Comment Form

Fold And Tape **Please Do Not Staple** Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** Fold And Tape

If you would like a reply, please print:

Your Name _____

Company Name/Department _____

Street Address _____

City _____

State/Zip Code _____

IBM Branch Office serving you _____

IBM®

IBM Virtual Storage Extended/
Advanced Functions
System Control Statements
Order No. SC33-6198-02

READER'S COMMENT FORM

This form may be used to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Your comments:

Note: **Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.**

If you wish a reply, give your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC33-6198-02

CUT
OR
FOLD
ALONG
LINE

Reader's Comment Form

Fold And Tape **Please Do Not Staple** Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape **Please Do Not Staple** Fold And Tape

If you would like a reply, please print:

Your Name _____

Company Name/Department _____

Street Address _____

City _____

State/Zip Code _____

IBM Branch Office serving you _____

IBM®

IBM Virtual Storage Extended/
Advanced Functions
System Control Statements
Order No. SC33-6198-02

READER'S COMMENT FORM

This form may be used to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Your comments:

Note: **Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.**

If you wish a reply, give your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC33-6198-02

CUT
OR
FOLD
ALONG
LINE

Reader's Comment Form

Fold And Tape | **Please Do Not Staple** | Fold And Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 6R1BP
180 Kost Road
Mechanicsburg, PA 17055

Fold And Tape | **Please Do Not Staple** | Fold And Tape

If you would like a reply, please print:

Your Name _____

Company Name/Department _____

Street Address _____

City _____

State/Zip Code _____

IBM Branch Office serving you _____

IBM®

IBM Virtual Storage Extended/
Advanced Functions
Order Number SC33-6198-02
(File No. S370/4300-36)
Printed in U.S.A.