File No. S360-31
Form C28-6554-1

OS

IBM Systems Reference Library

IBM System/360 Operating System

System Generation

System generation is a process that generates an IBM System/360 Operating System adapted to both the machine configuration and the data processing requirements of an installation. The system generation process is performed under the control of an existing IBM System/360 Operating System. This publication provides information on the computing system and operating system requirements for system generation, the initialization of system volumes and data sets, the macro-instructions used in specifying system generation, and the methods of including user-written programs in the operating system.

PREFACE

This publication provides system programmers with information about the system generation process provided with IBM System/360 Operating System. The reader must be familiar with the various facilities and options offered within IBM System/360 Operating System, and must have already decided upon the operating system configuration to be generated. This publication tells him how to generate the operating system configuration he has selected. The following IBM System/360 Operating System publications are required to understand and select the operating system to be generated:

IBM System/360 Operating System: Introduction, Form C28-6534

IBM System/360 Operating System: Concepts and Facilities, Form C28-6535

IBM System/360 Operating System: Storage Estimates, Form C28-6551

IBM System/360 Operating System: System Programmer's Guide, Form C28-6550

IBM System/360 Operating System: Maintenance, Form C27-6918

The following publications are required for a better understanding of the system generation process:

IBM System/360 Operating System: Job Control Language, Form C28-6539

IBM System/360 Operating System: Utilities, Form C28-6586

IBM System/360 Operating System: Assembler Language, Form C28-6514

Second Edition (October 1966)

This publication is a major revision of Form C28-6554-0 and obsoletes it. In addition to incorporating information released in Technical Newsletters N28-2136, N28-2164, N28-2152, N28-2165, N28-2186, and N28-2187, significant changes have been made.

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

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for reader's comments appears at the back of this publication. It may be mailed directly to the IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

ILLUSTRATIONS

FIGURES

TABLES

INTRODUCTION

IBM System/360 Operating System is comprised of a general library of modules that can be united in a variety of combinations to meet the particular requirements of a given installation. The operating system is tailored during the system generation process, which produces an operating system composed of the standard features incorporated in every operating system, those optional features selected from the distributed modules, and any additional features provided by the user.

The system generation process operates under an existing IBM System/360 Operating System. (IBM provides a starter operating system available for the first system generation.) Both the selection of optional features and the addition of user-written programs is specified by the user through system generation macro-instructions. Several operating system programs process the output of these macro-instructions to produce an operating system tailored to the user's specifications.

THE SYSTEM GENERATION PROCESS

IBM System/360 Operating System is composed of modules that can be united in a variety of combinations according to options specified by the user. The user selects the programming options that meet his data processing requirements and that conform to the processing, storage, and input/output facilities of his machine configuration. The selected programming options are then translated into program module requirements through the system generation process, and the modules are combined into the libraries that form the installation's new operating system.

The desired operating system is specified by the user through system generation macro-instructions. During the system generation process, several operating system programs are used to build a new operating system tailored by the specifications in the macro-instructions. These programs are executed under the control of an existing operating system.

An operating system is generated in two stages (see Figure 1). During Stage I, user-supplied macro-instructions, which describe both the installation's machine configuration and the programming options desired, are analyzed and used to generate a job stream. In Stage II, this job stream is processed to generate the libraries of modules that form the user's operating system. These libraries contain modules supplied by IBM and, optionally, modules supplied by the user.

PRODUCING THE JOB STREAM

Stage I consists of two phases. During the first phase all the macro-instructions supplied by the user are analyzed for errors. Error messages are written for each error found. If errors are not found in any of the macro-instructions, a job stream is produced during the second phase. If errors are found, the second phase is bypassed and the job stream is not produced.

PROCESSING THE JOB STREAM

During Stage II the job stream is processed by the assembler, by the linkage editor, and by utilities. The following functions are performed:

- Selected modules are assembled.
- The linkage editor combines the modules selected for inclusion in the resident portion of the control program (nucleus).
- The linkage editor processes those modules selected for combination, in order to construct members of the new libraries of the operating system.
- Utility programs complete the construction and initialization of the libraries selected for the new operating system.

The generated operating system is then ready for use.

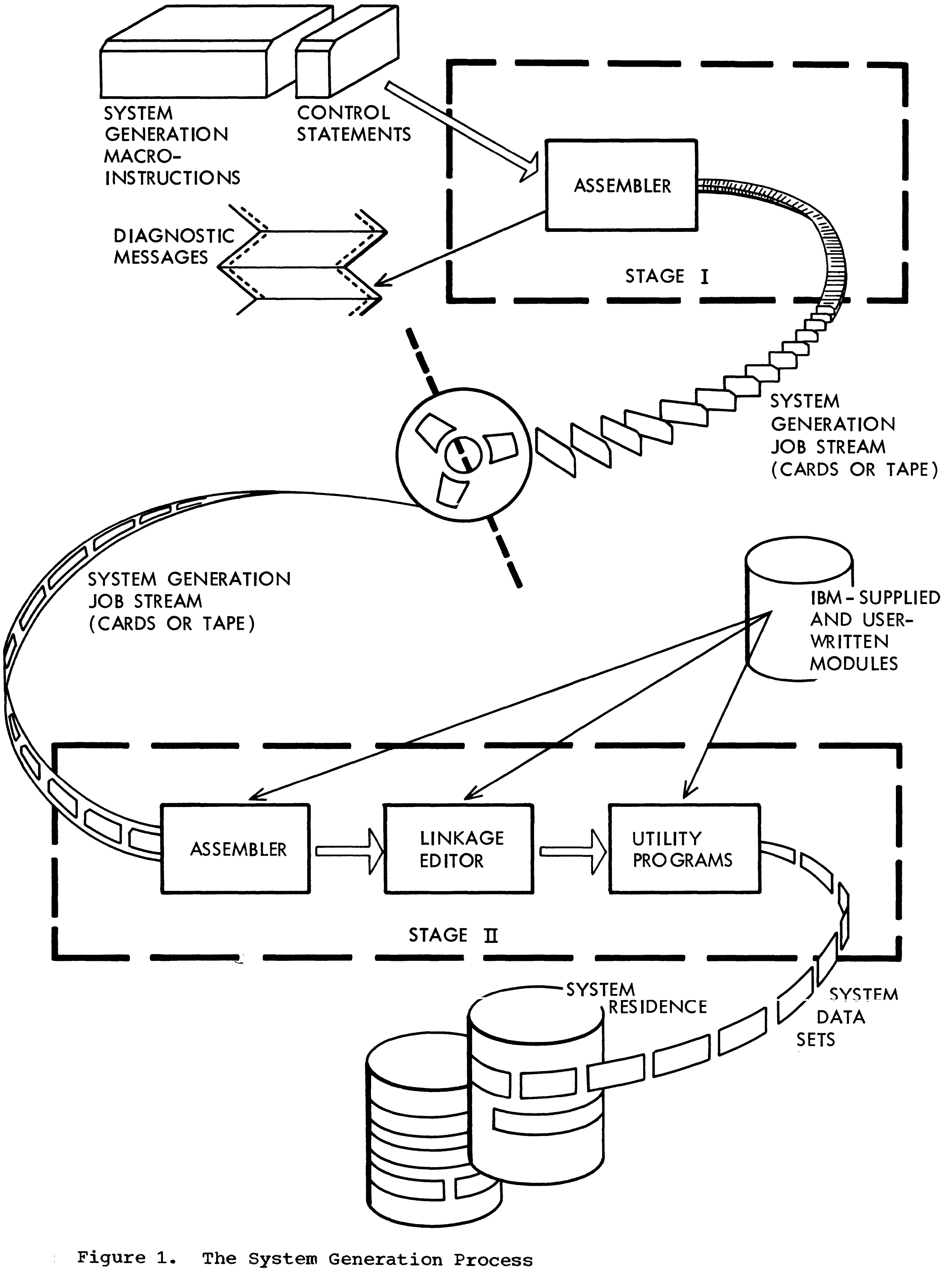

SYSTEM GENERATION MACRO-INSTRUCTIONS
CONTROL STATEMENTS
ASSEMBLER
STAGE I
DIAGNOSTIC MESSAGES
SYSTEM GENERATION JOB STREAM (CARDS OR TAPE)
SYSTEM GENERATION JOB STREAM (CARDS OR TAPE)
IBM-SUPPLIED AND USER-WRITTEN MODULES
ASSEMBLER
LINKAGE EDITOR
UTILITY PROGRAMS
STAGE II
SYSTEM RESIDENCE
SYSTEM DATA SETS

Figure 1. The System Generation Process

SYSTEM DATA SETS

The modules that comprise the operating system are contained in system data sets. The system data sets are the system catalog, the system libraries, and data sets such as SYS1.SYSJOBQE. The system libraries are partitioned data sets.

The following system data sets are required for every operating system:

- SYSCTLG (System Catalog) -- The system catalog contains pointers to all cataloged data sets.
- SYS1.NUCLEUS (Nucleus library) -- This library usually contains only one member, the resident portion (nucleus) of the control program. To include additional members, see the description of the GENERATE macro-instruction in the section entitled "Specifying the System."
- SYS1.SVCLIB (SVC library) -- The members of the SVC library are the nonresident SVC routines, the data management access methods, and the system's standard error recovery routines.
- SYS1.LOGREC -- This data set is used to record statistical data about machine errors.
- SYS1.LINKLIB (Link library) -- The members of the link library are programs and routines that can be referred to by XCTL, ATTACH, LINK, or LOAD macro-instructions, or by EXEC statements. Nonresident operating system programs, e.g., the COBOL compiler, are contained in this library.
- SYS1.PROCLIB (Procedure library) -- The members of the procedure library include those cataloged procedures used to perform certain system functions, e.g., compile-linkage edit-go.
- SYS1.SYSJOBQE -- This data set is used as a work area by the job scheduler.

The following system data sets are optional:

- SYS1.MACLIB (Macro library) -- The members of the macro library include the macro-definitions for the system macro-instructions.
- SYS1.SORTLIB (Sort library) -- The members of the sort library are the load modules from which a sort/merge program is produced at execution time.
- SYS1.COBLIB (COBOL library) -- The members of the COBOL library are load modules (COBOL subroutines).
- SYS1.FORTLIB (FORTRAN library) -- The members of the FORTRAN library are load modules (FORTRAN subprograms).
- SYS1.PL1LIB (PL/I library) -- The members of the PL/I library are load modules (PL/I subprograms).

The following system data sets are required for system generation only:

- SYS1.GENLIB (Generation library) -- This library contains the macro-definitions of the system generation macro-instructions.
- SYS1.MODLIB (Module library) -- The members of the module library are the load modules from which an operating system is generated.

SYSTEM/360 REQUIREMENTS

To generate an operating system, the installation's computing system configuration and its current operating system must fulfill certain minimum requirements. These requirements are described in the following text.

COMPUTING SYSTEM REQUIREMENTS

The following minimum System/360 configuration is required to generate an operating system:

- 64K bytes of main storage.
- One console device.
- Two IBM 2311 Disk Storage Drives.
- One card reader or one magnetic tape drive (input).
- One card punch or one magnetic tape drive (intermediate output).
- One printer or one magnetic tape drive (print output).

OPERATING SYSTEM REQUIREMENTS

System generation is performed under the control of an existing operating system and is executed as any other job. The operating system used to perform a system generation process must have system data sets, utility data sets, and, if desired, data sets that contain user-written programs to be included in the new system.

SYSTEM DATA SETS

The generating system must have the following system data sets:

- SYSCTLG (System Catalog)
- SYS1.NUCLEUS (Nucleus library)
- SYS1.SVCLIB (SVC library)
- SYS1.LOGREC
- SYS1.LINKLIB (Link library)
- SYS1.PROCLIB (Procedure library)
- SYS1.SYSJOBQE
- SYS1.MACLIB (Macro library)
- SYS1.MODLIB (Module library)
- SYS1.GENLIB (Generation library)

The link library (SYS1.LINKLIB) must include an assembler language processor, linkage editor and utilities.

If the E-design-level assembler language processor is to be used for system generation, both the macro library and the generation library must be unblocked. SYS1.MACLIB and SYS1.GENLIB can be unblocked by the IEBCOPY utility program by specifying DCB=(RECFM=FB,BLKSIZE=80,LRECL=80) in the DD statement for SYSUT2.

If the new operating system is to have COBOL, FORTRAN, PL/I, or sort/merge processors, the appropriate subroutine library or libraries (SYS1.COBLIB, SYS1.FORTLIB, SYS1.PL1LIB, SYS1.SORTLIB) must exist as cataloged partitioned data sets in the generating operating system, and the necessary modules for the processor must be included in the module library (SYS1.MODLIB). However, the generating system need not have the actual processor in order to generate it for the new system. For example, to generate a system that contains a FORTRAN compiler, the generating system need not have a FORTRAN compiler, as long as SYS1.FORTLIB exists as a cataloged partitioned data set in that system, and the modules for the FORTRAN compiler are in SYS1.MODLIB.

UTILITY DATA SETS

In addition to the system data sets, four utility data sets must be allocated space and cataloged in the generating system for use during the system generation process. Each of these data sets must be cataloged as SYS1.name, where the value of name cannot exceed eight alphameric characters, the first of which must be alphabetic. Three of these data sets must be sequential data sets. The fourth data set must be a partitioned data set. One of the sequential data sets and the partitioned data set must reside on a direct-access volume. The three sequential data sets are used during system generation by the assembler, linkage editor, and utilities. The partitioned data set is used for the storage of object modules assembled during system generation. (See "Input Deck for System Generation" in the section entitled "Specifying the System.")

USER-WRITTEN PROGRAMS

During the system generation process, the user can include his own routines in the nucleus, SVC library, and/or link library of the new operating system. The routines to be included in each of these libraries must be members of partitioned data sets cataloged in the generating system. The names under which they are cataloged are of the form SYS1.name. There must be only one partitioned data set for each of the libraries to be modified. However, if more than one library is to be modified, all routines can be included in one partitioned data set.

The user specifies the inclusion of his routines in the appropriate libraries by using the RESMODS, SVCTABLE, SVCLIB, and/or LINKLIB macro-instructions. (See the descriptions of these macro-instructions in the section entitled "Specifying the System.")

PREPARATION FOR SYSTEM GENERATION

Before an operating system can be generated, a new system residence volume, and any other direct-access volumes required, must be initialized. The volume index (SYS1) of the system catalog must be built on the new system residence volume. Space must be allocated for the appropriate system data sets in the new operating system, and the appropriate data sets must be cataloged in the new system catalog. By definition, the system residence volume is that volume on which the nucleus library, the SVC library, the IPL program, the SYS1.LOGREC data set, and the volume index of the catalog are located.

Volume initialization is performed by an independent utility program. A system utility program is used to build the volume index of the system catalog, to allocate space for system data sets, and to catalog data sets. This system utility can be executed at any time after the system residence volume is initialized.

The following paragraphs describe the initialization of the system residence volume and other required direct-access volumes, the initialization of the system data sets, and considerations on allocating space for these data sets. Detailed descriptions of the utility programs and of the control statements they require are found in the publication IBM System/360 Operating System: Utilities.

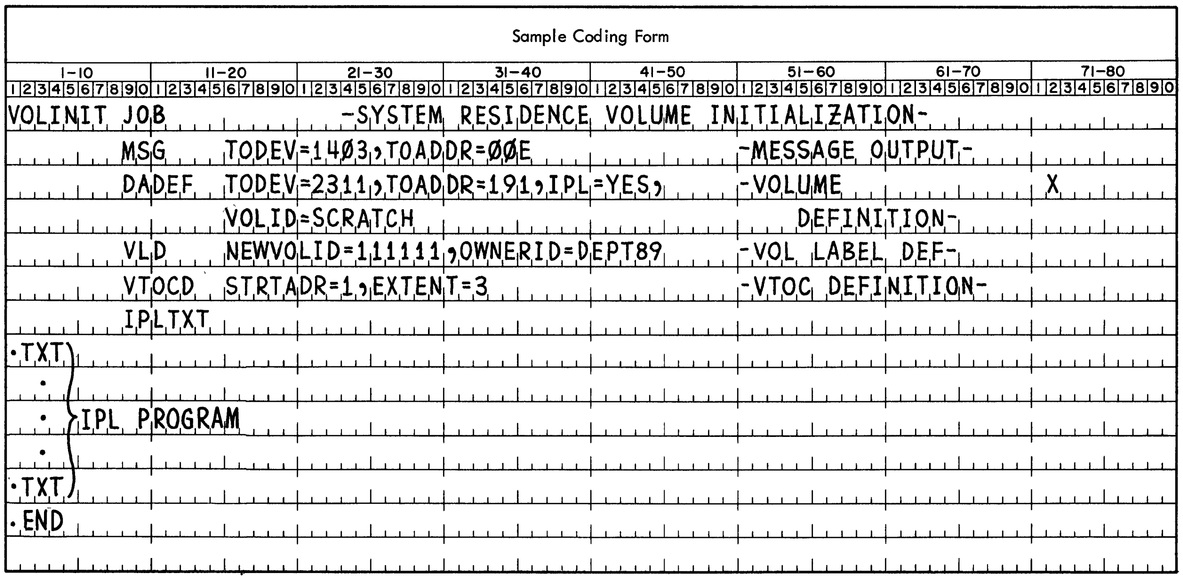
INITIALIZING DIRECT-ACCESS VOLUMES

Initialization is the process of writing home addresses, a volume label, and a volume table of contents (VTOC) on direct-access volumes. In addition, the initial program load (IPL) program must be written on the direct-access volume that is to become the system residence volume. These functions are accomplished by the direct-access device initialization (DASDI) utility program. This utility program is self-loading and operates independently of the operating system. In addition to its initialization functions, this utility program checks for defective tracks and, if any are found, assigns alternative tracks and issues appropriate messages.

Figure 2 is an example of an input deck for system residence volume initialization. In this example, the volume to be initialized resides on an IBM 2311 Disk Storage Drive. The volume serial number to be written in the label is 111111. The volume table of contents (VTOC) starts at track 1, and is three tracks long.

INITIALIZING SYSTEM DATA SETS

The initialization of system data sets is the process allocating space to the system data sets, of building the volume index of the system catalog, and cataloging system data sets in the system catalog. The contents of the system libraries are placed in the allocated space during system generation. The contents of the other data sets are not placed in the allocated space until job execution time in the generated operating system.



Sample Coding Form

```
VOLINIT JOB            -SYSTEM RESIDENCE VOLUME INITIALIZATION-
        MSG    TODEV=14Ø3,TOADDR=ØØE                -MESSAGE OUTPUT-
        DADEF  TODEV=2311,TOADDR=191,IPL=YES,       -VOLUME             X
               VOLID=SCRATCH                            DEFINITION-
        VLD    NEWVOLID=111111,OWNERID=DEPT89       -VOL LABEL DEF-
        VTOCD  STRTADR=1,EXTENT=3                   -VTOC DEFINITION-
        IPLTXT
.TXT\
  .  |
  .  >IPL PROGRAM
  .  |
.TXT/
.END
```

Figure 2. Initializing the System Residence Volume

The volume index of the system catalog is built on the new system residence volume by the IEHPROGM system utility program. This index points to the system data sets that are to form the new operating system. These data sets can be cataloged by the same utility program. Space should be allocated for these data sets (except for SYS1.LOGREC) by DD statements included in the input to this system utility program. Guidelines for the allocation of space to the system data sets are given in the section entitled "Space Allocation Planning."

The following system data sets are required, and must have space allocated on the system residence volume. These data sets need not be cataloged. However, it is recommended that at least one of these data sets be cataloged to permit references to the system residence volume without knowing its serial number. (These references can be made by specifying VOLUME=REF=dsname in a DD statement, where dsname is the name of the cataloged data set.)

- SYSCTLG (System Catalog) -- Only the volume index of the system catalog need reside on the system residence volume.

- SYS1.NUCLEUS (Nucleus library)

- SYS1.SVCLIB (SVC library)

- SYS1.LOGREC -- Space must not be allocated for this data set by the user. (Space is allocated for SYS1.LOGREC during the system generation process. To reinitialize SYS1.LOGREC, if required, after system generation, see Appendix G.)

The following system data sets are required, and must have space allocated on a direct-access volume. They need not reside on the system residence volume.

- SYS1.LINKLIB (Link library) -- This data set must be cataloged.
- SYS1.PROCLIB (Procedure library) -- It is recommended that this data set be cataloged.
- SYS1.SYSJOBQE -- This data set does not have to be cataloged.

The operating system can function without the following optional system data sets. If the user wishes to make use of the facilities they provide, they can be included in the new operating system. Space must be allocated on a direct-access volume for the optional data sets desired. They need not reside on the system residence volume. It is recommended that the data sets to be included be cataloged.

- SYS1.MACLIB (Macro library)
- SYS1.SORTLIB (Sort library)
- SYS1.COBLIB (COBOL library)
- SYS1.FORTLIB (FORTRAN library)
- SYS1.PL1LIB (PL/I library)

SPACE ALLOCATION PLANNING

This section provides guidelines for allocating space to the data sets required for the new operating system, and to the utility data sets required during system generation. Allocation of space for the system data sets is determined by:

- The selection of those system generation options that require the inclusion of data sets in the new operating system.
- The location of these data sets on the direct-access volume or volumes that will contain them.
- The estimated size of these data sets.
- Those data sets that permit secondary allocation of space.
- The space requirements of the utility data sets used during system generation.
- The machine configuration, in particular the number and type of direct-access devices available in the generating system.

Illustrations of allocation are provided with the discussion of these factors.

DATA SETS FOR THE NEW SYSTEM

Space must be allocated during the preparation for system generation for all the required system data sets, except for SYS1.LOGREC. No space should be allocated for SYS1.LOGREC because space is allocated for this data set (on the system residence volume) during the system generation process. SYS1.NUCLEUS and SYS1.SVCLIB must be allocated space entirely on the system residence volume. Space must also be allocated for any of the optional data sets (SYS1.MACLIB, SYS1.SORTLIB, SYS1.COBLIB, SYS1.FORTLIB, or SYS1.PL1LIB) that, during system generation, are specified for inclusion by the MACLIB, SORTLIB, COBLIB, FORTLIB, or PL1LIB macro-instructions.

LOCATION OF SYSTEM DATA SETS

The system data sets can be arranged in different ways on one or more direct-access volumes. For best performance, it is desirable to place the data sets on more than one volume. Whenever possible, frequently used data sets should be located on a volume other than the system residence volume. System data sets on the same volume should be arranged according to the interaction between them: the more interaction, the closer they should be placed. For example, in Figure 8, SYS1.SVCLIB and SYS1.LINKLIB are adjacent because they are frequently used data sets and are closely related. Space for SYS1.NUCLEUS is allocated last, because this data set is referenced only by the IPL program.

The placement of new system data sets is also determined by the location of the generating system data sets. The following rule may be used for allocating space to the new system data sets. During system generation all volumes necessary to contain the new system, the generating system, SYS1.MODLIB, the utility data sets that must reside on direct-access volumes; the optional libraries (e.g., SYS1.FORTLIB) for the new system, and any user-written modules (see "Inclusion of User-Written Programs") must be mounted at the same time. If there are not enough drives available to achieve the desired distribution of new system data sets, the IEHMOVE utility program can be used to distribute data sets after system generation. If SYS1.SVCLIB is changed, replaced, or moved after system generation the IEHIOSUP program must be executed. See Appendix H for information on IEHIOSUP.

Note: The location of system data sets during preparation for system generation determines their corresponding volume specifications in the system generation macro-instructions. (See the descriptions of the GENERATE, PROCLIB, MACLIB, SORTLIB, COBLIB, PL1LIB, and FORTLIB macro-instructions in "Specifying the System.")

ESTIMATED SIZES

The size of SYS1.LOGREC is determined during system generation. Information on this data set is supplied in a message during Stage II of the system generation process. The auxiliary storage requirements for the remaining system data sets can be found in the publication *IBM System/360 Operating System: Storage Estimates*. Space must be allocated for SYS1.PROCLIB in the new system. If neither IBM-supplied nor user-written cataloged procedures are to be used, a null allocation can be made for SYS1.PROCLIB; i.e., the SPACE parameter is specified as SPACE=(TRK,(0)) in the DD statement for SYS1.PROCLIB. (See "Sample Data Set Initialization.")

The maximum space that can be allocated to a system data set is one volume, excet for SYS1.SVCLIB, which may not occupy more than 1023 tracks on the system residence volume. Unless sufficient space is allocated to the system data sets, the system generation process cannot be completed successfully.

SECONDARY ALLOCATION

In addition to the space indicated for the system data sets in *IBM System/360 Operating System: Storage Estimates*, the user must allow space for the inclusion of his own programs into those data sets during (or after) system generation. Some system data sets, such as SYS1.LINKLIB, may not have multiple extents and must be contiguous, thus sufficient space must be allocated initially to them if user modules are to be included. Table 1 indicates whether multiple extents are permitted for each of the system data sets.

Table 1. System Data Sets

System Data Set	Multiple Extents Allowed
SYSCTLG	Yes
SYS1.NUCLEUS	No
SYS1.SVCLIB	No
SYS1.LOGREC	No
SYS1.LINKLIB	No
SYS1.SYSJOBQE	No
SYS1.PROCLIB	Yes
SYS1.MACLIB	Yes
SYS1.MODLIB	Yes
SYS1.GENLIB	Yes
SYS1.SORTLIB	Yes
SYS1.COBLIB	Yes
SYS1.FORTLIB	Yes
SYS1.PL1LIB	Yes

UTILITY DATA SETS

The system generation process requires four utility data sets. Two of the utility data sets must reside on direct-access volumes. The allocation of space and cataloging requirements for these data sets are described in the section "Input Deck for System Generation." The placement of these data sets and of the system data sets is described in the following section.

MACHINE CONFIGURATION

The precise distribution of data sets depends on the number and type of I/O devices on the generating system and on the desired configuration of the new system. The following examples illustrate the distribution of data sets in configurations with two, three, and four direct-access devices.

In the figures, the volumes marked #1 and #2 show the data sets in the generating system, as specified in the section entitled "Operating System Requirements." The volume marked #3 represents the system residence volume for the new system. In Figures 3, 4, and 5 all data sets for the new system have been placed on the new system residence volume (volume #3). However, because it may be desirable to have a data set on a volume other than the system residence volume, Figures 6 and 7 show space allocated for the new SYS1.LINKLIB on volumes other than the new system residence volume. In all these examples the four utility data sets required for system generation are called SYS1.UT1, SYS1.UT2, SYS1.UT3, and SYS1.OBJMOD.

Generation on Two Drives

Figure 3 shows the distribution of data sets using two 2311 Disk Storage Drives and three volumes. The scheduler, when required, automatically requests the dismounting of volume #2 and the mounting of volume #3 on drive #2.

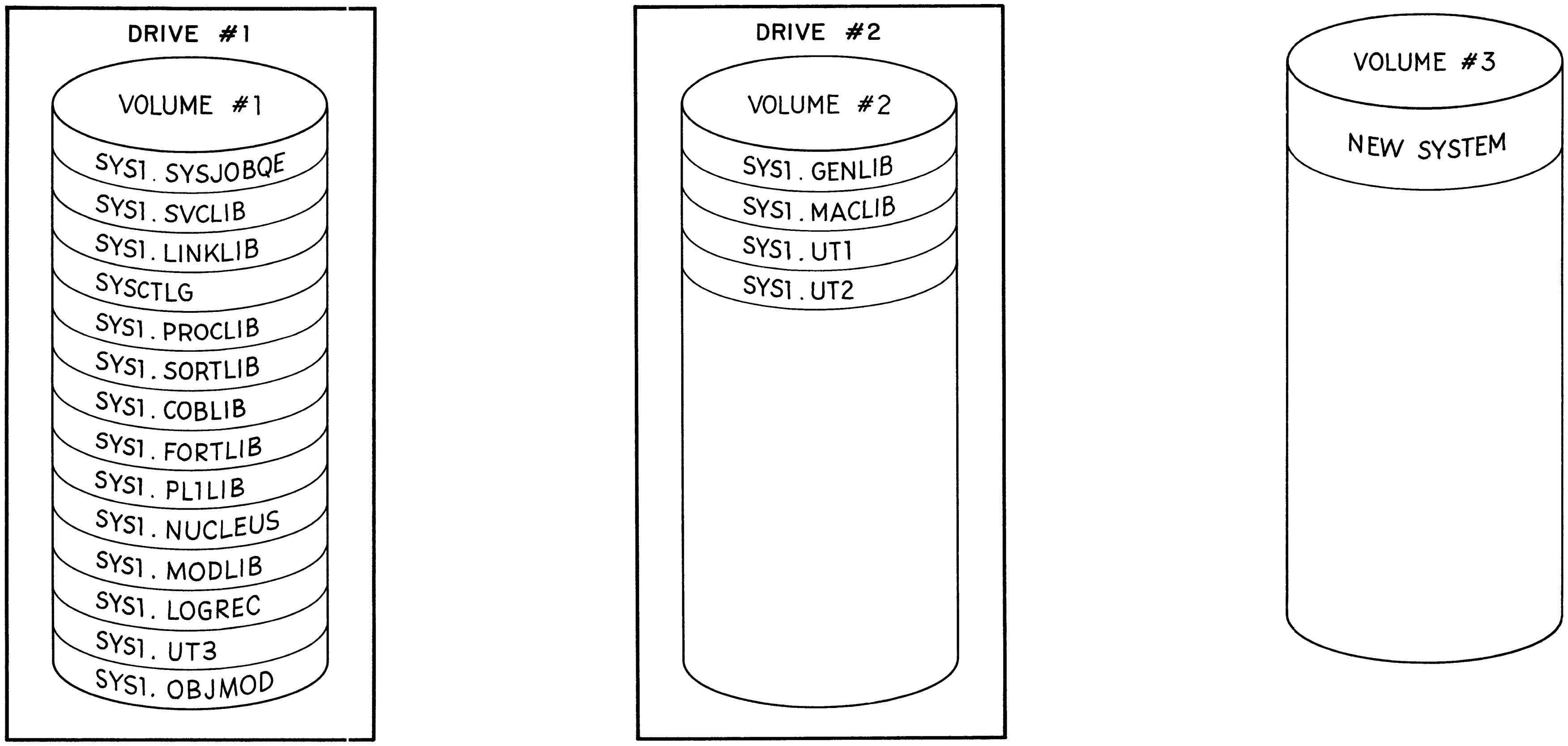


Figure 3. Generation on Two Drives

Generation on Three Drives

Figure 4 shows the distribution of data sets using three 2311 Disk Storage Drives and three volumes.

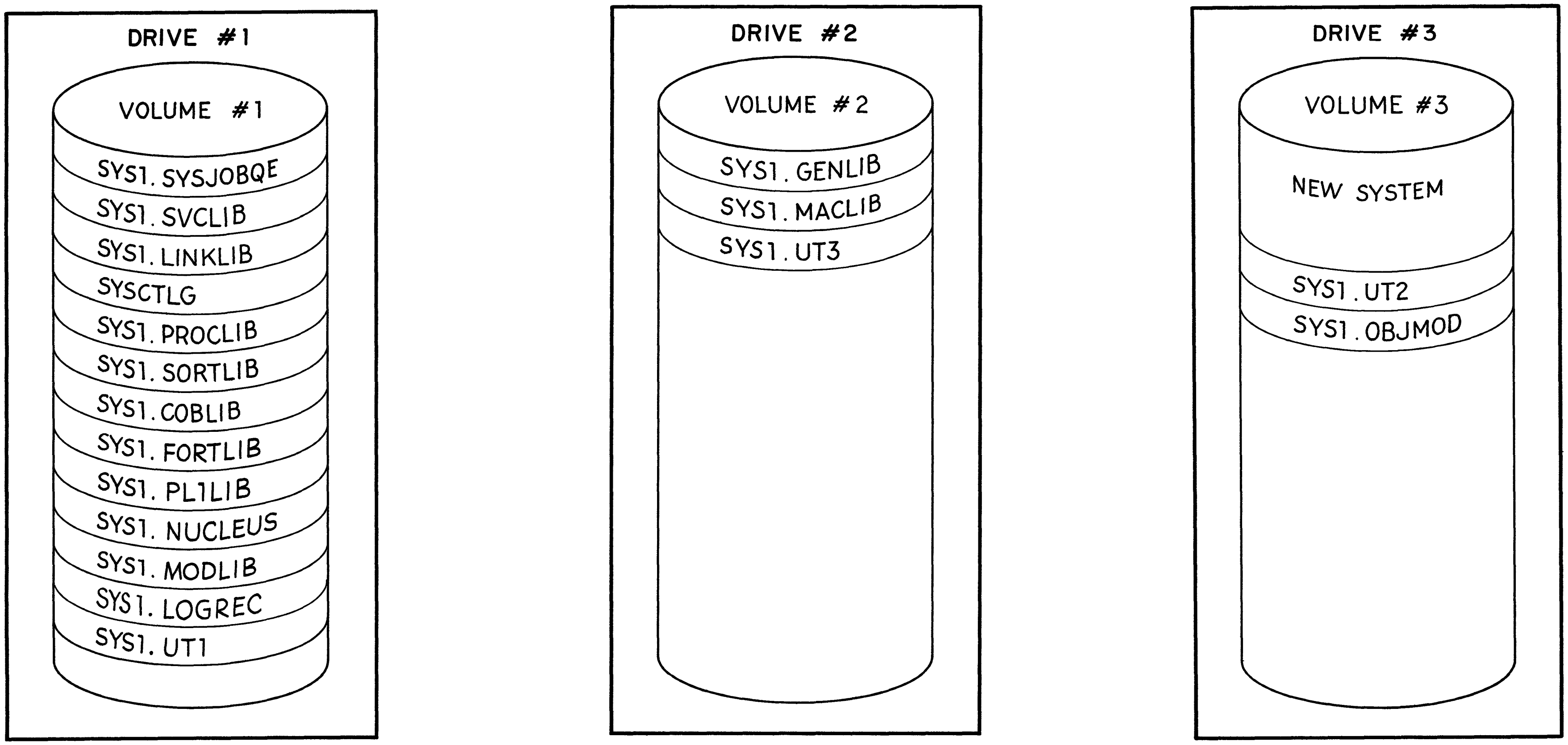


Figure 4. Generation on Three Drives

Generation on Four Drives

Figure 5 shows the distribution of data sets using four 2311 Disk Storage Drives and four volumes.

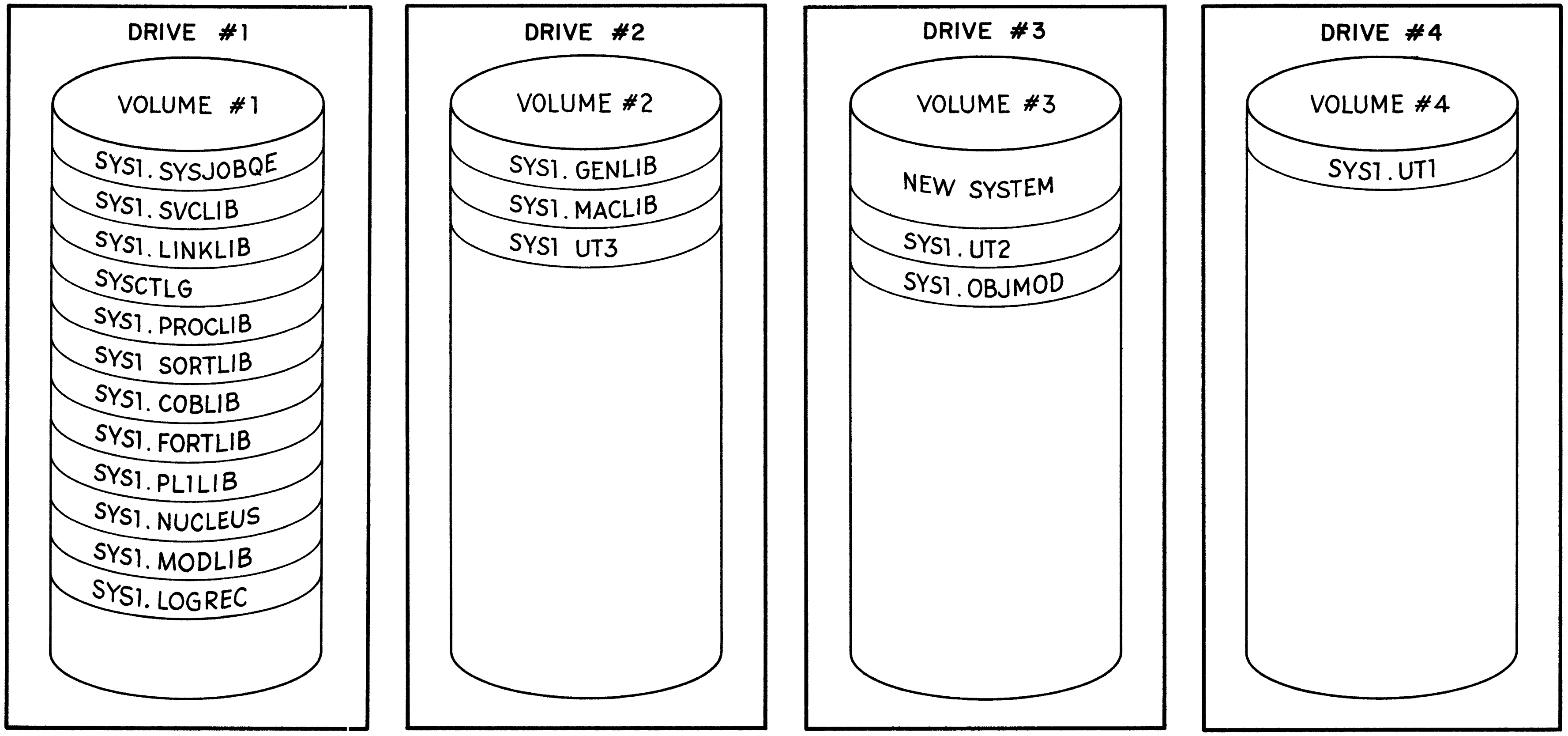


Figure 5. Generation on Four Drives

Generation on Three Drives with Multiple Volumes for New System

Figure 6 shows the distribution of data sets using three 2311 Disk Storage Drives and three volumes. The generated SYS1.LINKLIB is to be placed on the volume marked #2. (SYS1.COBLIB is not to be included in the new system.)

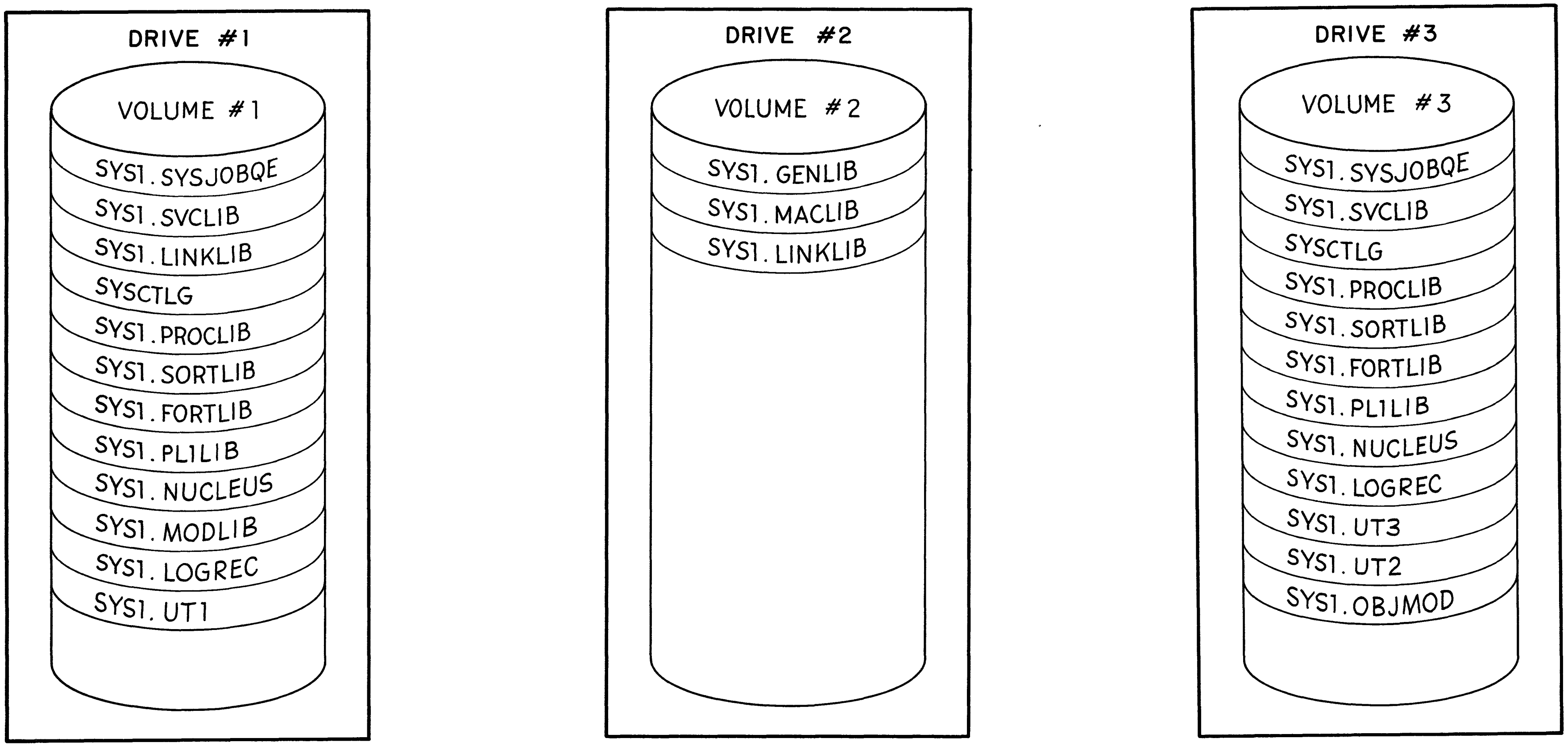


Figure 6. Generation on Three Drives with Multiple Volumes

Generation on Four Drives with Multiple Volumes for New System

Figure 7 shows the distribution of data sets using four 2311 Disk Storage Drives and four volumes. The generated SYS1.LINKLIB is to be placed on the volume marked #4. (SYS1.FORTLIB is not to be included in the new system.

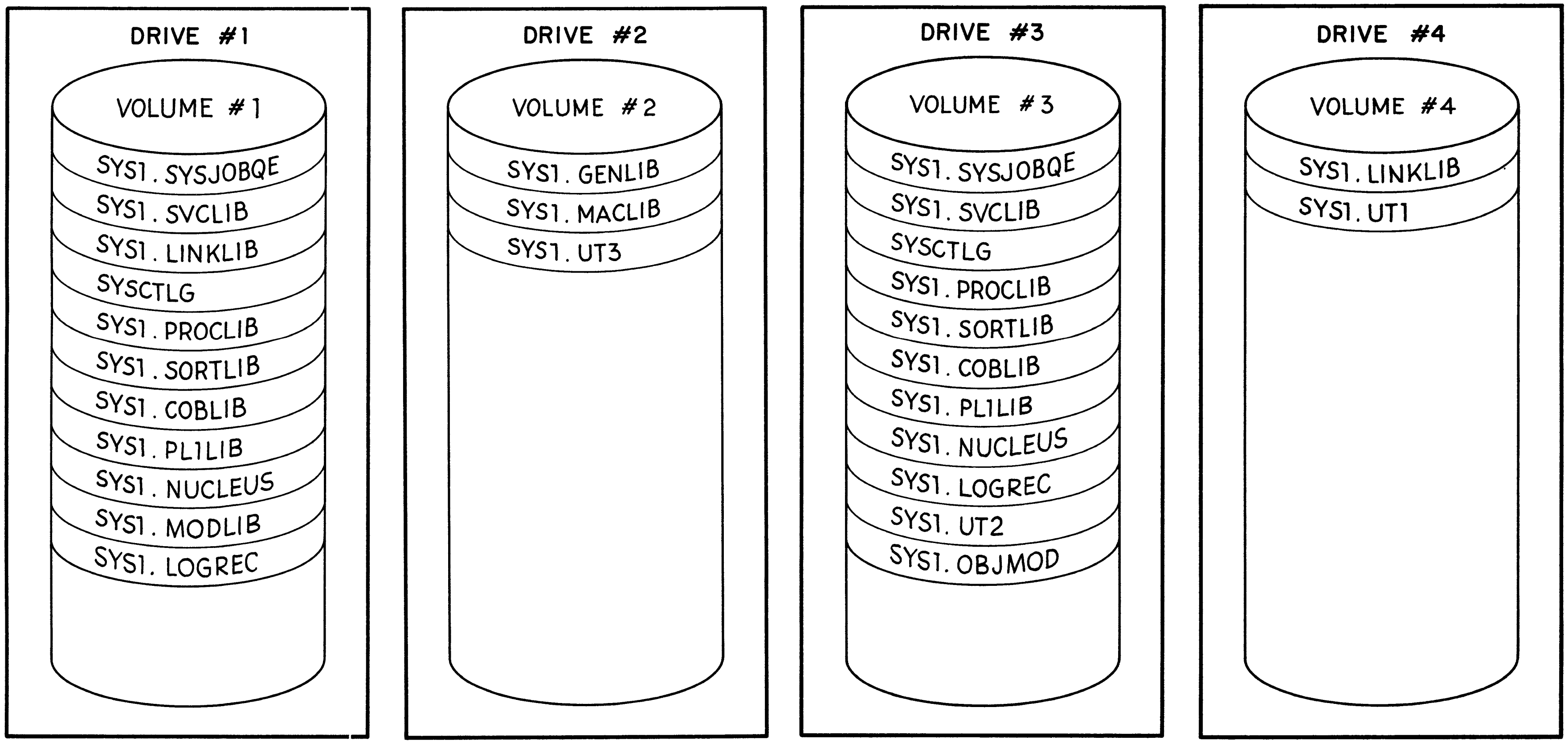


Figure 7. Generation on Four Drives with Multiple Volumes

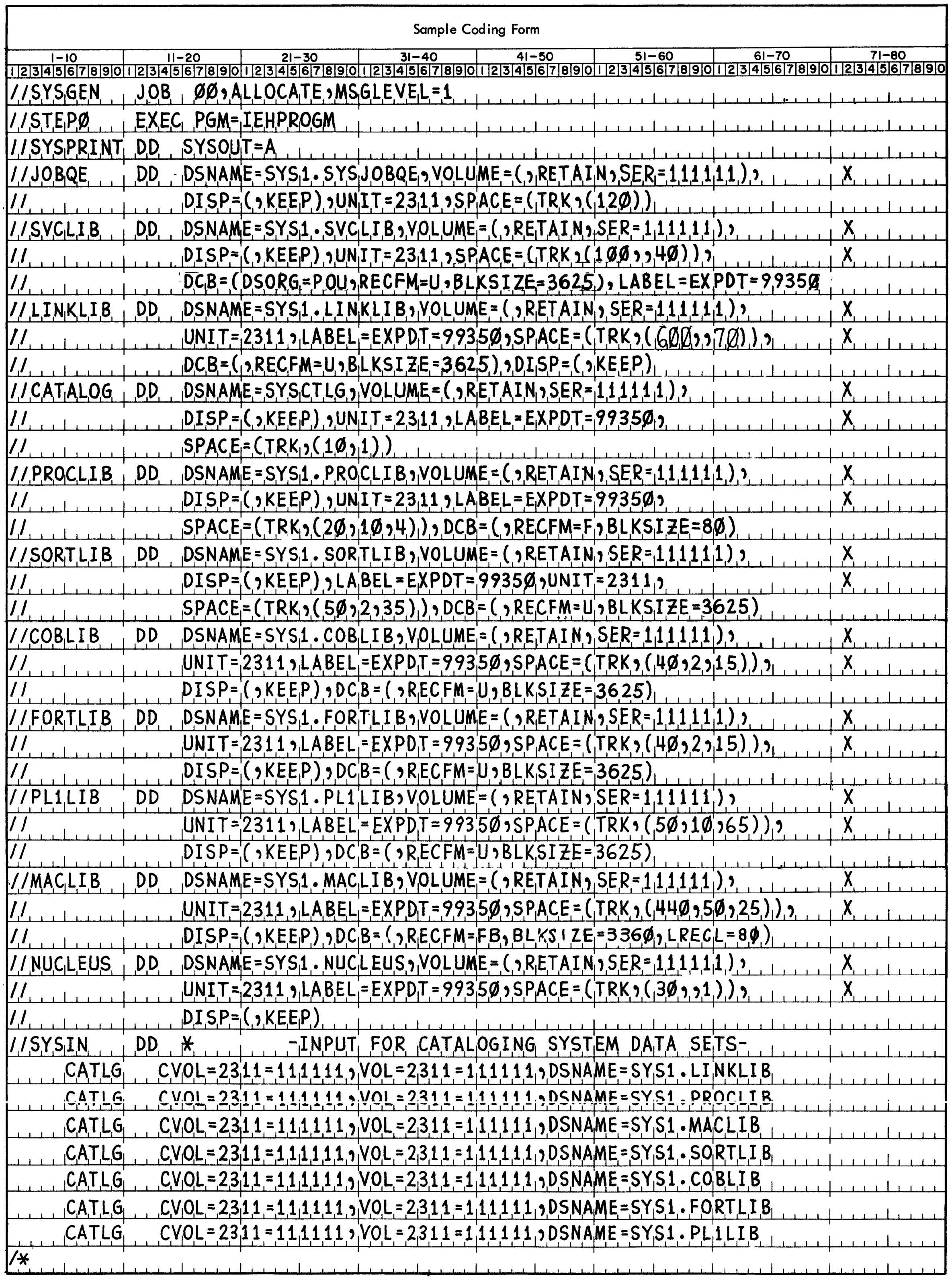
SAMPLE DATA SET INITIALIZATIONS

Figure 8 is an example of an input deck for building the system catalog and for allocating space to the system data sets. It is assumed that the system residence volume was initialized as shown in Figure 2, and that the new system requires all the optional system data sets described in "Initializing System Data Sets." These optional data sets, and SYS1.PROCLIB and SYS1.LINKLIB are to be cataloged in the new system. The new system residence volume is specified as 2311. The serial number of the system residence volume is 111111. (The numbers chosen for track allocation are for illustrative purposes only.)

Figure 9 is an example of an input deck for building the system catalog and for allocating space to the system data sets. The system data sets are to reside on two volumes. The system residence volume is specified as 2301 and its serial number is AAA111. The second volume is specified as 2311 and its serial number is AAA112. It is assumed that both volumes were previously initialized. The new system requires all optional system data sets described in "Initializing System Data Sets." These optional data sets, SYS1.PROCLIB, and SYS1.LINKLIB are to be cataloged in the new system. All system data sets, except SYS1.PL1LIB and SYS1.MACLIB, reside on the system residence volume. SYS1.PL1LIB and SYS1.MACLIB reside on the second volume (AAA112).

The expiration date for all data sets is chosen to prevent accidental deletion. This data set protection requires additional action by the user. If the current date is set in the generating system during system generation, the operator is required to override this current date each time a protected data set is opened. Another, more convenient method can be used. For the system generation job (or any time the data sets are to be modified), the current date may be set at a higher value than the expiration date specified for the protected data sets. (In this case the set date becomes the new expiration date.) This may be done by the operator from the console, or by a card in the job stream. In either case, the current date should be reset in the generating system immediately after the completion of the system generation process.

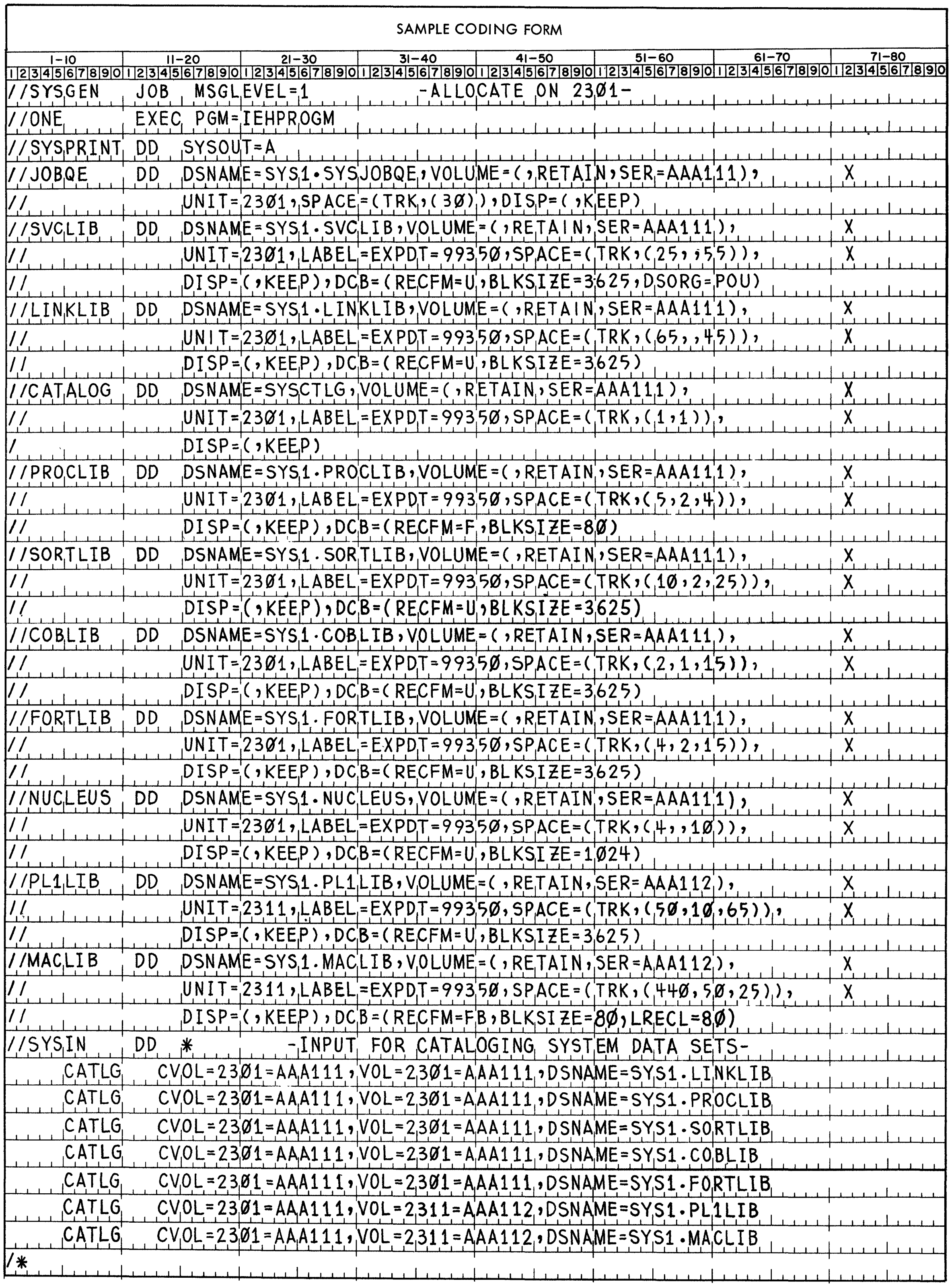
Note: If the E-design-level assembler language processor is to be included in the new system, a BLKSIZE value of 80 rather than 3360 must be specified in the DD statement that defines SYS1.MACLIB. For example, Figure 8 specifies a blocked SYS1.MACLIB; Figure 9 an unblocked one.



Sample Coding Form

```
//SYSGEN   JOB  ØØ,ALLOCATE,MSGLEVEL=1
//STEPØ    EXEC PGM=IEHPROGM
//SYSPRINT DD  SYSOUT=A
//JOBQE    DD  DSNAME=SYS1.SYSJOBQE,VOLUME=(,RETAIN,SER=111111),       X
//             DISP=(,KEEP),UNIT=2311,SPACE=(TRK,(12Ø))
//SVCLIB   DD  DSNAME=SYS1.SVCLIB,VOLUME=(,RETAIN,SER=111111),         X
//             DISP=(,KEEP),UNIT=2311,SPACE=(TRK,(1ØØ,,4Ø)),          X
//             DCB=(DSORG=POU,RECFM=U,BLKSIZE=3625),LABEL=EXPDT=9935Ø
//LINKLIB  DD  DSNAME=SYS1.LINKLIB,VOLUME=(,RETAIN,SER=111111),        X
//             UNIT=2311,LABEL=EXPDT=9935Ø,SPACE=(TRK,(6ØØ,,7Ø)),     X
//             DCB=(,RECFM=U,BLKSIZE=3625),DISP=(,KEEP)
//CATALOG  DD  DSNAME=SYSCTLG,VOLUME=(,RETAIN,SER=111111),             X
//             DISP=(,KEEP),UNIT=2311,LABEL=EXPDT=9935Ø,               X
//             SPACE=(TRK,(1Ø,,1))
//PROCLIB  DD  DSNAME=SYS1.PROCLIB,VOLUME=(,RETAIN,SER=111111),        X
//             DISP=(,KEEP),UNIT=2311,LABEL=EXPDT=9935Ø,               X
//             SPACE=(TRK,(2Ø,1Ø,4)),DCB=(,RECFM=F,BLKSIZE=8Ø)
//SORTLIB  DD  DSNAME=SYS1.SORTLIB,VOLUME=(,RETAIN,SER=111111),        X
//             DISP=(,KEEP),LABEL=EXPDT=9935Ø,UNIT=2311,               X
//             SPACE=(TRK,(5Ø,2,35)),DCB=(,RECFM=U,BLKSIZE=3625)
//COBLIB   DD  DSNAME=SYS1.COBLIB,VOLUME=(,RETAIN,SER=111111),         X
//             UNIT=2311,LABEL=EXPDT=9935Ø,SPACE=(TRK,(4Ø,2,15)),     X
//             DISP=(,KEEP),DCB=(,RECFM=U,BLKSIZE=3625)
//FORTLIB  DD  DSNAME=SYS1.FORTLIB,VOLUME=(,RETAIN,SER=111111),        X
//             UNIT=2311,LABEL=EXPDT=9935Ø,SPACE=(TRK,(4Ø,2,15)),     X
//             DISP=(,KEEP),DCB=(,RECFM=U,BLKSIZE=3625)
//PL1LIB   DD  DSNAME=SYS1.PL1LIB,VOLUME=(,RETAIN,SER=111111),         X
//             UNIT=2311,LABEL=EXPDT=9935Ø,SPACE=(TRK,(5Ø,1Ø,65)),    X
//             DISP=(,KEEP),DCB=(,RECFM=U,BLKSIZE=3625)
//MACLIB   DD  DSNAME=SYS1.MACLIB,VOLUME=(,RETAIN,SER=111111),         X
//             UNIT=2311,LABEL=EXPDT=9935Ø,SPACE=(TRK,(44Ø,5Ø,25)),   X
//             DISP=(,KEEP),DCB=(,RECFM=FB,BLKSIZE=336Ø,LRECL=8Ø)
//NUCLEUS  DD  DSNAME=SYS1.NUCLEUS,VOLUME=(,RETAIN,SER=111111),        X
//             UNIT=2311,LABEL=EXPDT=9935Ø,SPACE=(TRK,(3Ø,,1)),       X
//             DISP=(,KEEP)
//SYSIN    DD  *        -INPUT FOR CATALOGING SYSTEM DATA SETS-
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.LINKLIB
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.PROCLIB
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.MACLIB
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.SORTLIB
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.COBLIB
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.FORTLIB
     CATLG   CVOL=2311=111111,VOL=2311=111111,DSNAME=SYS1.PL1LIB
/*
```

Figure 8. Initializing the System Data Sets - 2311 System Residence



SAMPLE CODING FORM

```
//SYSGEN   JOB  MSGLEVEL=1         -ALLOCATE ON 2301-
//ONE      EXEC PGM=IEHPROGM
//SYSPRINT DD  SYSOUT=A
//JOBQE    DD  DSNAME=SYS1.SYSJOBQE,VOLUME=(,RETAIN,SER=AAA111),      X
//             UNIT=2301,SPACE=(TRK,(30)),DISP=(,KEEP)
//SVCLIB   DD  DSNAME=SYS1.SVCLIB,VOLUME=(,RETAIN,SER=AAA111),        X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(25,,5)),        X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=3625,DSORG=POU)
//LINKLIB  DD  DSNAME=SYS1.LINKLIB,VOLUME=(,RETAIN,SER=AAA111),       X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(65,,45)),       X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=3625)
//CATALOG  DD  DSNAME=SYSCTLG,VOLUME=(,RETAIN,SER=AAA111),            X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(1,1)),          X
/              DISP=(,KEEP)
//PROCLIB  DD  DSNAME=SYS1.PROCLIB,VOLUME=(,RETAIN,SER=AAA111),       X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(5,2,4)),        X
//             DISP=(,KEEP),DCB=(RECFM=F,BLKSIZE=80)
//SORTLIB  DD  DSNAME=SYS1.SORTLIB,VOLUME=(,RETAIN,SER=AAA111),       X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(10,2,25)),      X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=3625)
//COBLIB   DD  DSNAME=SYS1.COBLIB,VOLUME=(,RETAIN,SER=AAA111),        X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(2,1,15)),       X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=3625)
//FORTLIB  DD  DSNAME=SYS1.FORTLIB,VOLUME=(,RETAIN,SER=AAA111),       X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(4,2,15)),       X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=3625)
//NUCLEUS  DD  DSNAME=SYS1.NUCLEUS,VOLUME=(,RETAIN,SER=AAA111),       X
//             UNIT=2301,LABEL=EXPDT=99350,SPACE=(TRK,(4,,10)),        X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=1024)
//PL1LIB   DD  DSNAME=SYS1.PL1LIB,VOLUME=(,RETAIN,SER=AAA112),        X
//             UNIT=2311,LABEL=EXPDT=99350,SPACE=(TRK,(50,10,65)),     X
//             DISP=(,KEEP),DCB=(RECFM=U,BLKSIZE=3625)
//MACLIB   DD  DSNAME=SYS1.MACLIB,VOLUME=(,RETAIN,SER=AAA112),        X
//             UNIT=2311,LABEL=EXPDT=99350,SPACE=(TRK,(440,50,25)),    X
//             DISP=(,KEEP),DCB=(RECFM=FB,BLKSIZE=80,LRECL=80)
//SYSIN    DD  *        -INPUT FOR CATALOGING SYSTEM DATA SETS-
     CATLG   CVOL=2301=AAA111,VOL=2301=AAA111,DSNAME=SYS1.LINKLIB
     CATLG   CVOL=2301=AAA111,VOL=2301=AAA111,DSNAME=SYS1.PROCLIB
     CATLG   CVOL=2301=AAA111,VOL=2301=AAA111,DSNAME=SYS1.SORTLIB
     CATLG   CVOL=2301=AAA111,VOL=2301=AAA111,DSNAME=SYS1.COBLIB
     CATLG   CVOL=2301=AAA111,VOL=2301=AAA111,DSNAME=SYS1.FORTLIB
     CATLG   CVOL=2301=AAA111,VOL=2311=AAA112,DSNAME=SYS1.PL1LIB
     CATLG   CVOL=2301=AAA111,VOL=2311=AAA112,DSNAME=SYS1.MACLIB
/*
```

Figure 9. Initializing the System Data Sets - 2301 System Residence

During each system generation process the user can specify one of three types of generation:

- Complete Operating System generation.
- Nucleus generation.
- Processor and Library generation.

In the first type, the user specifies the generation of an operating system consisting of either a control program only, or a control program and language processors and their associated libraries. A primary control program, primary data management routines, and system utilities are always provided with this type of generation. These standard features are adapted to the installation's machine configuration during the generation process. An Operating System generation must be performed whenever the installation's machine configuration is modified, or whenever changes are to be made to the system generation options for the control program. (An Operating System generation may not be needed if only changes to the nucleus of the control program are to be made. In this case, a new nucleus can be added to the operating system through a Nucleus generation.)

In the other types of generation, the user specifies that a nucleus, or that language processors and/or their associated libraries are to be added to the operating system. During the preparation for a Processor and Library generation, the user must allocate space for, and, if desired, catalog, any new system data set to be added to the operating system. If any of the existing system data sets are to be modified, the user must have allocated sufficient space to those data sets when they were initially generated or, where permitted, provided for multiple extents in those data sets. (SYS1.NUCLEUS is the only system data set affected by a Nucleus generation.)

The system to be generated is specified by the user. Specifications are of two types: job control language statements and system generation macro-instructions. The job control language statements are those required to perform an assembly. The system generation macro-instructions describe the machine configuration of the installation, those options the user desires, and the type of generation to be performed. The programs and routines generated as a result of these specifications are placed into the appropriate operating system libraries during the system generation process.

The following sections describe the input deck required for system generation, and the conventions used to code system generation macro-instructions. A detailed description of each system generation macro-instruction and a table showing the relationship of the macro-instructions to each type of system generation are also included.

From the specifications supplied to the system generation macro-instructions, parameters are created for the job control language statements produced from Stage I and included in the job stream to Stage II. Samples of these statements are shown in Appendix F.

Note: The specifications for a system generation must be compatible with the cataloging and space allocation performed during the preparation for that generation.

INPUT DECK FOR SYSTEM GENERATION

The input deck required for system generation consists of job control language statements and system generation macro-instructions. The sequence of the deck and the job control language statements are shown in Figure 10. This figure represents a two-drive system generation. For other generations, only the values given to the VOLUME, UNIT, and SPACE keywords, and the device to be used for the job stream need vary. (In Figure 10, the generating system resides on a volume whose serial number is DLIB01; SYS1.MACLIB and SYS1.GENLIB reside on a volume whose serial number is DLIB02; and unit 182 is a 9-track magnetic tape.) It is recommended that the values given to the UNIT keywords of the DD statements be generic unit names. (See Appendix C.)

The first statement of the deck is an EXEC statement indicating that the generation process immediately follows the catalog building step described in the previous section. Alternatively, system generation can be defined as an independent job.

The four DD statements named OBJPDS, SYSUT1, SYSUT2, and SYSUT3, respectively, are used to allocate space to the four utility data sets required for the system generation process. These data sets are cataloged as SYS1.name in the generating system, where the value of name cannot exceed eight alphameric characters, the first of which must be alphabetic. These names (SYS1.name) must also be specified as the value of the corresponding keywords (OBJPDS, UT1SDS, UT2SDS, and UT3SDS) in the GENERATE macro-instruction. The data set defined by the OBJPDS DD statement must be a partitioned data set. The other three data sets must be sequential data sets, of which the one specified by the SYSUT3 DD statement must reside on a direct-access volume. Table 2 shows the values to be given to the SPACE keyword of each of the DD statements for the utility data sets according to the type of direct-access device on which they may reside.

If magnetic tape drives are available, the data sets specified by the SYSUT1 and SYSUT2 DD statements can be assigned to 9-track magnetic tape. If only one utility data set is to reside on magnetic tape, the one specified by the SYSUT1 DD statement should be chosen. If the data sets defined by the SYSUT or SYSUT2 DD statements reside on unlabeled magnetic tape, NL must be specified as the value of the LABEL keyword of the corresponding DD statement, and NL must also be specified in the UT1SDS or UT2SDS keyword of the GENERATE macro-instruction.

The DD statement named DUMMY is used to reserve space on the volume that contains the utility data set defined by the SYSUT3 DD statement for the IEHMOVE utility program. IEHMOVE does its own allocation on that volume during Stage II of system generation. The DUMMY DD statement is optional, but it should be used to ensure that the required number of tracks will be available during Stage II. (The data set created by the DUMMY DD statement is deleted at the end of Stage I.) The value to be given to the SPACE keyword of the DUMMY DD statement is determined by the type of device on which the data set specified by the SYSUT3 DD statement resides. The appropriate values are shown in Table 2.

Table 2. Space Allocation for Utility Data Sets

DD Statement	IBM 2311 Disk Storage Drive	IBM 2301 Drum Storage
OBJPDS	(TRK,(40,20,8))	(TRK,(10,2,20))
SYSUT1	(TRK,(240,20))	(TRK,(45,4))
SYSUT2	(TRK,(240,20))	(TRK,(45,4))
SYSUT3	(TRK,(250,20))	(TRK,(50,4))
DUMMY	(TRK,(160))	(TRK,(28))

The DD statement named SYSPUNCH defines the data set which is to contain the job stream produced during Stage I of system generation. If any error messages (see Appendix B) occur during system generation, the job stream is not produced. After a successful completion of Stage I, the job stream produced becomes the input to Stage II. If the device defined by the SYSPUNCH DD statement is a card punch, the operator is required to place the cards in an input device and to issue a START RDR command for that device. However, operator intervention can be eliminated by making the output (SYSPUNCH) of Stage I become the input reader to Stage II. This can be accomplished, if the value given to UNIT in the SYSPUNCH DD statement is the specific unit name of a magnetic tape drive, by inserting a // START RDR,xxx statement after the /* card of the input deck. (See Figure 9.) xxx is the specific unit name given to UNIT in the SYSPUNCH DD statement. If there were any errors during Stage I, there will be an end-of-file condition detected and the reader will be closed. Appendix D indicates some of the conditions that require operator intervention. Appendix E shows sample console listings from system generation processes.

Note: The input deck described is the input to Stage I of the system generation process. During Stage I a job stream is produced which serves as the input for Stage II. After a satisfactory completion of Stage I, the beginning of Stage II is a logical restart point. The Stage I output (SYSPUNCH) should be saved after system generation for maintenance purposes.

Sample Coding Form

```
//STEP1     EXEC PGM=ASMBLR         -SYSTEM GENERATION-
//SYSLIB    DD  DSNAME=SYS1.GENLIB,DISP=OLD
//OBJPDS    DD  DSNAME=SYS1.OBJMOD,VOLUME=(,RETAIN,SER=DLIB01),              X
//              DISP=(,CATLG),UNIT=2311,SPACE=(TRK,(40,2,8))
//SYSUT1    DD  DSNAME=SYS1.UT1,VOLUME=(,RETAIN,SER=DLIB02),                 X
//              DISP=(,CATLG),UNIT=2311,SPACE=(TRK,(240,20))
//SYSUT2    DD  DSNAME=SYS1.UT2,VOLUME=(,RETAIN,SER=DLIB02),                 X
//              DISP=(,CATLG),UNIT=2311,SPACE=(TRK,(240,20))
//SYSUT3    DD  DSNAME=SYS1.UT3,VOLUME=(,RETAIN,SER=DLIB01),                 X
//              DISP=(,CATLG),UNIT=2311,SPACE=(TRK,(250,20))
//DUMMY     DD  VOLUME=(,RETAIN,REF=*.SYSUT3),SPACE=(TRK,(160))
//SYSPUNCH  DD  UNIT=182,LABEL=(,NL)
//SYSPRINT  DD  SYSOUT=A
//SYSIN     DD  *
          .
          .
          . SYSTEM GENERATION MACRO-INSTRUCTIONS
          .
          .
          END
/*
//        START    RDR,182      (OPTIONAL STATEMENT; SEE TEXT)
```

Figure 10. Input Deck Organization for System Generation

CONVENTIONS

This section describes the conventions used to code system generation macro-instructions and the notation used in this publication to describe system generation macro-instructions.

CODING MACRO-INSTRUCTIONS

System generation macro-instructions have the following standard format:

Name	Operation	Operand
Symbolic name	Macro-instruction type	Optional and required parameters

The name symbolically identifies the macro-instruction. If included, it can contain from one through eight alphameric characters, the first of which must be alphabetic. The name must begin in the first position of the macro-instruction and must be followed by one or more blanks. Unless otherwise indicated in the description of individual macro-instructions, the name field of a system generation macro-instruction is ignored during system generation.

The operation identifies the macro-instruction. It must be preceded and followed by one or more blanks.

The operand contains parameters coded in any order and separated by commas. The operand field ends with one or more blanks placed after the last parameter. In most system generation macro-instructions, keyword parameters are used in the operand field. A keyword parameter consists of a keyword followed by an equal sign (=) and the keyword value. The keyword value must be a single value or a list of values; in the latter case, the values must be separated by commas and the list enclosed in parentheses.

Comments can be written in a system generation macro-instruction, but they must be separated from the last parameter of the operand field by one or more blanks. An entire card may be used for a comment by placing an asterisk in the first column. Extensive comments may be written by using a series of cards with an asterisk in the first column of each card. A macro-instruction that has no parameters can not have comments.

A typical system generation macro-instruction might appear as:

```
NAME OPERATION KEY1=value,KEY4=(value1,value2),KEY3=value,...
```

System generation macro-instructions are coded in columns 1 through 71 of a card. A macro-instruction that exceeds column 71 can be continued onto one or more additional cards; a nonblank character is placed in column 72 to indicate continuation. The macro-instruction can be interrupted either in column 71 or after any comma that separates parameters. The continued portion must begin in column 16 of the following card. Comments can be coded through column 71, and, if continued, must begin in column 16 of the following card. In addition, comments may appear on every card of a continued statement. Columns 73 through 80 can be used to code identification and/or statement sequence characters.

DESCRIBING MACRO-INSTRUCTIONS

The following conventions are used in this publication to illustrate the format and coding of system generation macro-instructions:

- Upper case letters, numbers, and punctuation marks must be coded by the programmer exactly as shown. Exceptions to this convention are brackets, []; braces, {}; ellipses, ...; and subscripts. These are never coded.
- Lower case letters and words represent variables for which the programmer must substitute specific information or specific values.
- Items or groups of items within brackets [] are optional. They may be omitted at the programmer's discretion. Conversely, the lack of brackets indicates that an item or group of items must be coded.
- Braces {} group related items.
- Stacked items, enclosed in either brackets or braces, represent alternative items. Only one of the stacked items should be coded.
- If an alternative item is underlined, that item is implied; that is, the operating system will automatically assume it is the programmer's choice when none of the items are coded.
- An ellipsis (...) indicates that the preceding item or group of items can be coded more than once in succession.

SYSTEM GENERATION MACRO-INSTRUCTIONS

The content of the new operating system is specified through system generation macro-instructions. Not all system generation macro-instructions are required for the system generation process. Table 3 lists the system generation macro-instructions for each type of system generation, indicating whether they are required or optional. If neither required nor optional is indicated, that macro-instruction does not apply to that type of system generation.

The type of system generation must be specified in the GENERATE macro-instruction. Table 2 also shows which macro-instructions can be issued more than once during a system generation process. All UNITNAME macro-instructions having the same NAME value must appear together in the input deck. Each IOCONTRL macro-instruction must precede in the input deck to system generation those IODEVICE macro-instructions that define devices attached to that control unit. All other system generation macro-instructions, with the exception of GENERATE, can be issued in any order. The GENERATE macro-instruction must be the last macro-instruction in the input deck for the system generation process.

Note: A primary control program, primary data management routines, and system utilities are included in every operating system generated; therefore, their inclusion is not specified in any of the system generation macro-instructions.

Table 3. System Generation Macro-Instructions

Macro-Instruction	Type of System Generation[1]		
	Operating System	Nucleus	Processor/Library
CENPROCS	Required	Required	Required
CHANNEL[2]	Required	Required	-----
IOCONTRL[2]	Required	Required	-----
IODEVICE[2]	Required	Required	-----
CTRLPROG	Required	Required	-----
SCHEDULR	Required	Required	-----
UNITNAME[2]	Optional	-----	-----
PROCLIB	Optional	-----	Optional
SUPRVSOR	Optional	Optional	Optional[3]
SVCTABLE[2]	Optional	Optional	-----
RESMODS	Optional	Optional	-----
SVCLIB	Optional	-----	Optional
LINKLIB	Optional	-----	Optional
DATAMGT	Optional	Optional	-----
SYSUTILS	Optional	-----	-----
EDITOR[2]	Optional	-----	Optional
ASSEMBLR[2]	Optional	-----	Optional
TESTRAN	Optional	Optional	-----
MACLIB	Optional	-----	Optional
SORTMERG	Optional	-----	Optional
SORTLIB	Optional	-----	Optional
COBOL[2]	Optional	-----	Optional
COBLIB[2]	Optional	-----	Optional
FORTRAN	Optional	-----	Optional
FORTLIB	Optional	-----	Optional
PL1	Optional	-----	Optional
PL1LIB	Optional	-----	Optional
RPG	Optional	-----	Optional
GENERATE	Required	Required	Required

[1]The type of system generation is specified with the GENTYPE parameter of the GENERATE macro-instruction. GENTYPE=ALL specifies the generation of an operating system. GENTYPE=NUCLEUS specifies the generation of a nucleus to be included in the operating system. (No libraries, except SYS1.NUCLEUS, are affected by this type of system generation.) GENTYPE=PROCESSOR specifies the generation of processors and/or their associated libraries to be added to the operating system.

[2]This macro-instruction can be issued more than once during a system generation process.

[3]This macro-instruction is used during a processor/library generation if, and only if, the PL1 macro-instruction is used.

CENPROCS

The CENPROCS macro-instruction is used to describe the central processing unit. This macro-instruction is required.

```
┌──────────┬─────────────┬──────────────────────────────────────┐
│ Name     │ Operation   │ Operand                              │
├──────────┼─────────────┼──────────────────────────────────────┤
│          │ CENPROCS    │ ┌       ┌30┐┐                         │
│          │             │ │       │40││                         │
│          │             │ │ MODEL=┤50├│                         │
│          │             │ │       │65││                         │
│          │             │ └       └75┘┘                         │
│          │             │ ┌         ┌E┐┐                        │
│          │             │ │         │F││                        │
│          │             │ │ STORAGE=┤G├│                        │
│          │             │ │         │H││                        │
│          │             │ │         │I││                        │
│          │             │ └         └J┘┘                        │
│          │             │ ┌         ┌STD  ┐┐                    │
│          │             │ │ INSTSET=┤COMM ├│                    │
│          │             │ │         │SCNTF││                    │
│          │             │ └         └UNIV ┘┘                    │
│          │             │ [FEATURE=(feature[,feature])]        │
└──────────┴─────────────┴──────────────────────────────────────┘
```

Operand Field:

MODEL=model
specifies the model of the central processing unit.

STORAGE
specifies the size of main storage as one of the following:

Value	Storage Size (in bytes)
E	32K
F	64K
G	128K
H	256K
I	512K
J	1024K

INSTSET
specifies the instruction set available in the central processing unit as one of the following:

Value	Instruction Set
STD	Standard
COMM	Commercial (Standard instruction set with decimal feature)
SCNTF	Scientific (Standard instruction set with floating point feature)
UNIV	Universal (Standard instruction set with decimal, floating point and storage protection features)

Note: This parameter need not be specified for model 50 or above; UNIV is always assigned in this case.

FEATURE=$feature_n$

specifies the optional features installed in the central processing unit as one or more of the following values. These values can be written in any order.

Value	Feature
TIMER	Interval timer on model 30
PROTECT	Storage protection

Example: In the following example, a CENPROCS macro-instruction is used to describe a model 40 central processing unit with a main storage size of 64K bytes. The commercial instruction set is used.

```
   CENPROCS MODEL=40,STORAGE=F,INSTSET=COMM
```

CHANNEL

The CHANNEL macro-instruction is used to describe channel characteristics. A CHANNEL macro-instruction is required for each channel of an installation's computing system.

Name	Operation	Operand

[name]	CHANNEL	ADDRESS=address
		TYPE= $\left\{\begin{array}{l}\text{SELECTOR}\\\text{MULTIPLEXOR}\\\text{HISPEEDMULTIPLEXOR}\end{array}\right\}$

Name Field:

name
    is used in system generation error messages (see Appendix B) to identify the CHANNEL macro-instruction that produced an error. If no name is entered, the macro-instruction supplies sequential identification numbers to the CHANNEL macro-instructions in the same order in which these macro-instructions are introduced in the user's input stream. These numbers are used for identification purposes instead of names. For example, if the name is omitted from the third CHANNEL macro-instruction in the input stream, the name CHAN#3 is supplied in each diagnostic message resulting from an error encountered in the macro-instruction.

Operand Field:

ADDRESS=address
    specifies the address of the channel as one of the following: 0, 1, 2, 3, 4, 5, or 6.

TYPE
    specifies the type of channel as one of the following:

Value	Channel
SELECTOR	Selector
MULTIPLEXOR	Multiplexor
HISPEEDMULTIPLEXOR	2870 Multiplexor

Note: If a 2870 multiplexor channel is specified, no burst devices should be attached to the multiplexor portion of the channel.

Example: The following example illustrates the use of the CHANNEL macro-instruction to describe a multiplexor channel whose address is 0.

```
MPX CHANNEL ADDRESS=0,TYPE=MULTIPLEXOR
```

IOCONTRL

The IOCONTRL macro-instruction is used to describe a control unit and its operating system requirements. An IOCONTRL macro-instruction is required for each control unit (listed in Table 4) in the user's computing system. A maximum of 40 IOCONTRL macro-instructions may be issued during a system generation. For assistance in choosing valid combinations of values for the UNIT, MODEL, and FEATURE keywords, refer to Table 4.

Name	Operation	Operand

[name]	IOCONTRL	UNIT=unit
		ADDRESS=address
		[MODEL=model]
		[FEATURE=(feature[,feature]...)]
		
		For UNIT=2821 Only:
		
		TRNMODE={BYTE / BURST}

Name Field:

name
is used in system generation error messages (see Appendix B) to identify the IOCONTRL macro-instruction that produced an error. If no name is entered, the macro-instruction supplies sequential identification numbers to the IOCONTRL macro-instructions in the same order in which these macro-instructions are introduced in the user's input stream. These numbers are used for identification purposes instead of names. For example, if the name is omitted from the sixth IOCONTRL macro-instruction in the input stream, the name UNIT#6 is supplied in each diagnostic message resulting from an error encountered in the macro-instruction.

Operand Field:

UNIT=unit
specifies the control unit number as one of the following: 1051, 2403, 2404, 2701, 2702, 2803, 2804, 2820, 2821, 2822, 2841, or 2848. (2816 is implied through the specification of the OPTCHAN parameter in the IODEVICE macro-instruction, and need not be specified in any macro-instruction.)

Note: The IBM 1052 printer-keyboard is attached to model 40 or above through an IBM 1052 adapter, and not through a control unit. Thus, an IOCONTRL macro-instruction is not needed in this case.

ADDRESS=address
specifies the address of the control unit. The address value consists of two hexadecimal digits whose valid range of values is 00 to 6F. This value normally corresponds to the two high order digits of the addresses of the devices attached to the control unit. If the high order digits of the device addresses differ, the lowest value must be used. For example, if the addresses of the devices attached to the control unit are 00E and 010, the value

given to the ADDRESS keyword of the IOCONTRL macro-instruction must be 00.

MODEL=model
specifies the model, if any, of the control unit as one of the following: 1, 2, 3, 4, 5, or N1.

FEATURE=feature$_n$
specifies the optional features that are present on the control unit as one of the following values. These values can be written in any order.

Value	Feature
COLBNRY	Column binary on 2821, model 1 or 4
DATACONV	Data conversion on 2403, 2404, 2803, or 2804
2-CHANSW	2-channel switch on 2820
7-TRACK	7-track compatibility on 2403, 2404, 2803, or 2804
16-DRIVE	16-drive addressing on 2403 or 2803

TRNMODE
specifies the data transmission mode for an IBM 2821 Control Unit attached to a multiplexor channel. The value selected, either BYTE or BURST, is determined by the setting of the mode switch on the 2821 CE panel. This parameter must be specified if, and only if, the value of UNIT is 2821.

Table 4. Keyword Values for IOCONTRL Macro-Instruction

UNIT	MODEL	FEATURE
1051	1 or N1	
2403	1, 2, or 3	DATACONV 7-TRACK 16-DRIVE
2404	1, 2, or 3	DATACONV 7-TRACK
2701		
2702		
2803	1	DATACONV 7-TRACK 16-DRIVE
2804	1	DATACONV 7-TRACK
2820		2-CHANSW
2821	1 or 4	COLBNRY
	2, 3, or 5	
2822		
2841		
2848	1, 2, or 3	

Example: The following example illustrates the use of the IOCONTRL macro-instruction to describe an IBM 2821 control unit, model 4, with the column binary feature. The control unit operates in the byte mode, and its address is 05.

```
IOCONTRL UNIT=2821,MODEL=4,ADDRESS=05,FEATURE=COLBNRY,TRNMODE=BYTE
```

IODEVICE

The IODEVICE macro-instruction is used to describe the characteristics of an input/output device and its operating system requirements. An IODEVICE macro-instruction is required for each uniquely addressable input/output device in the system. A maximum of 96 IODEVICE macro-instructions may be issued during a system generation. In the input deck for system generation, each IOCONTROL macro-instruction must be immediately followed by the IODEVICE macro-instructions that define devices attached to that control unit. For assistance in choosing valid combinations of values for the UNIT, MODEL and FEATURE keywords, refer to Table 5.

The value given to ADDRESS becomes the specific unit name of the device. Specific unit names are automatically assigned to the devices during system generation. Generic unit names are also provided in every operating system for each type of device specified by the UNIT parameter of an IODEVICE macro-instruction. Generic unit names are described in Appendix C.

Name	Operation	Operand
[name]	IODEVICE	UNIT=unit
		ADDRESS=address
		[MODEL=model]
		[FEATURE=(feature[,feature]...)]
		<u>For UNIT=2401,2402,2403 or 2404 Only:</u>
		[OPTCHAN=(address[,address]...)]
		<u>For UNIT=1443 Only:</u>
		$\left[\text{TRNMODE=}\begin{Bmatrix}\text{BURST}\\ \text{BYTE}\end{Bmatrix}\right]$
		<u>For UNIT=2250 Only:</u>
		$\left[\text{NUMSECT=}\begin{Bmatrix}\underline{1}\\ 0\end{Bmatrix}\right]$
		<u>For Telecommunications Devices Only:</u>
		ADAPTER=adapter
		[SETADDR=address]

<u>Name Field:</u>

name

is used in system generation error messages (see Appendix B) to identify the IODEVICE macro-instruction that produced an error. If no name is entered, the macro-instruction supplies sequential identification numbers to the IODEVICE macro-instructions in the same order in which these macro-instructions are introduced in the user's input stream. These numbers are used for identification purposes instead of names. For example, if the name is omitted from the eleventh IODEVICE macro-instruction in the input stream, the name DEV#11 is supplied in each diagnostic message resulting from an error encountered in the macro-instruction.

Operand Field:

UNIT=unit
specifies the unit number of the device as one of the following: 1030, 1050, 1052, 1053, 1060, 1403, 1442, 1443, 2250, 2260, 2301, 2311, 2401, 2402, 2403, 2404, 2501, 2520, 2540R, 2540P, 2671, 115A, 83B3, or TWX.

Note: 2540R and 2540P refer to the same IBM 2540 card read punch, but, because they are uniquely addressable, the read (2540R) and punch (2540P) functions must be specified as two different units in separate IODEVICE macro-instructions. TWX refers to either the teletype model 33 or 35.

MODEL=model
specifies the model number, if any, of the device as one of the following: 1, 2, 3, 5, 6, 7, B1, B2, B3, N1, or N2. This parameter must be specified if the unit has a model number (see Table 5).

ADDRESS=address
specifies the address of the device. The address value consists of three hexadecimal digits whose valid range of values is 000 to 6FF.

FEATURE=feature_n
specifies the optional features that are present on the device as one or more of the following values. These values can be written in any order. Features enclosed in braces { } are mutually exclusive.

Value	Feature
AUTOANSR	Automatic answering capability on 1050
AUTOCALL	Automatic calling feature on 1050
{BUFFER4K}	4096-byte buffer storage on 2250, model 1
{BUFFER8K}	8192-byte buffer storage on 2250, model 1
CARDIMAGE	Card image on 1442, 2501, or 2520
READWRITE	Simultaneous read while write on 2401, or 2402
SELCHSET	Selective character set on 1443
UNVCHSET	Universal character set on 1403
{7-TRACK}	7-track head on 2401, 2402, 2403, or 2404
{9-TRACK}	9-track head on 2401, 2402, 2403, or 2404
24ADDPOS	24 additional print positions on 1443

Note: Either 7-TRACK or 9-TRACK must be specified when the value of the UNIT keyword is 2401, 2402, 2403, or 2404

Table 5. Keyword Values for the IODEVICE Macro-Instruction

UNIT	MODEL	FEATURE
1030		
1050		AUTOANSR AUTOCALL
1052[1]	5, 6, or 7	
1053		
1060		
1403[2]	2, 3, 7, or N1	UNVCHSET
1442	N1 or N2	CARDIMAGE
1443	N1	SELCHSET 24ADDPOS
2250	1	BUFFER4K or BUFFER8K
2260	1	
2301		
2311		
2401	1, 2, or 3	READWRITE 7-TRACK or 9-TRACK
2402	1, 2, or 3	READWRITE 7-TRACK or 9-TRACK
2403	1, 2, or 3	7-TRACK or 9-TRACK
2404[3]	1, 2, or 3	7-TRACK or 9-TRACK
2501	B1 or B2	CARDIMAGE
2520	B1, B2, or B3	CARDIMAGE
2540R	1	
2540P	1	
2671	1	
115A		
83B3		
TWX		

[1]A 2150 used to connect a 1052 is addressed as the 1052, and may not be specified.

[2]A 1404 printer is supported only as a continuous form printer, and must be specified as a 1403, model 2.

[3]The READWRITE feature is implicit for the 2404 and must not be specified.

OPTCHAN=address$_n$

specifies the alternative channels through which a 2401, 2402, 2403, or 2404 may be addressed. Each value specified is the address of an alternative channel as specified in the CHANNEL macro-instruction. These values must be greater than the first digit of the value of the ADDRESS keyword. (If the device is attached to a 2403 or 2803 control unit, a 2816 is required in order to have alternative channel addressing. In this case, the presence of the 2816 is implied when specifying the OPTCHAN parameter and it must not be specified elsewhere.) A maximum of three alternative channels can be specified. If the READWRITE feature is specified for the device, one and only one alternative channel must be specified. The addresses of alternative channels can be written in any order.

Note: There must be only one IODEVICE macro-instruction for each I/O device, regardless of the number of alternative addresses given to the device. For example, if the primary address of a device is 181, and if it can also be addressed through channels 2, 3, and 4, there must not be separate IODEVICE macro-instructions defining the address of the device as either 281, 381, or 481.

TRNMODE

specifies the data transmission mode for the device. BURST specifies the burst mode of data transmission. BYTE specifies the byte mode, i.e., multiplex mode of data transmission. This parameter must be specified if, and only if, the device is on a multiplexor channel and the device specified is an IBM 1443 printer, Model N1.

NUMSECT

specifies the number of buffer sections to be allocated to a 2250 device and the inclusion of buffer management routines. 0 specifies no sections and the exclusion of buffer management routines. 1 specifies one section and the inclusion of buffer management routines.

ADAPTER=adapter

specifies the terminal control or transmission adapter used to connect a telecommunications I/O device to its control unit. This parameter must be included if, and only if, the value given to the UNIT keyword is 1030, 1050, 1060, 115A, 83B3, or TWX.

Value	Adapter or Control
IBM1	IBM Terminal Adapter Type I attaching a 1050 or 1060 to a 2701, or IBM Terminal Control Type I attaching a 1050 or 1060 to a 2702
IBM2	IBM Terminal Adapter Type II attaching a 1030 to a 2701, or IBM Terminal Control Type II attaching a 1030 to a 2702
IBMT	IBM Telegraph Adapter attaching a 1050 to a 2701, or IBM Terminal Control Type I and a Telegraph Line Adapter attaching a 1050 to a 2702
TELE1	Telegraph Adapter Type I attaching a 115A or 83B3 to a 2701, or Telegraph Terminal Control Type I attaching a 115A or 83B3 to a 2702
TELE2	Telegraph Adapter Type II attaching a TWX to a 2701, or Telegraph Terminal Control Type II attaching a TWX to a 2702

SETADDR=address
specifies the set address (SAD) command to be issued for a telecommunications device attached to an IBM 2702 Transmission Control Unit.

Value	Command
0	SADZERO
1	SADONE
2	SADTWO
3	SADTHREE

Examples: The following example illustrates the use of the IODEVICE macro-instruction to describe an IBM 1404 printer, model 2. The address of the device is 20E.

```
PRINTER2 IODEVICE UNIT=1403,MODEL=2,ADDRESS=20E
```

The following example illustrates the use of the IODEVICE macro-instruction to describe an IBM 2401 magnetic tape drive, model 3, with a 9-track head. The address of the device is 181. This device can be addressed alternatively through channel 2.

```
TAPE1 IODEVICE UNIT=2401,ADDRESS=181,OPTCHAN=2,MODEL=3,FEATURE=9-TRACK
```

CTRLPROG

The CTRLPROG macro-instruction is used to specify control program options. This macro-instruction is required.

Name	Operation	Operand
	CTRLPROG	MAXIO=number
		$\left[\text{OVERLAY=}\left\{\begin{matrix}\underline{\text{BASIC}}\\ \text{ADVANCED}\end{matrix}\right\}\right]$
		$\left[\text{FETCH=}\left\{\begin{matrix}\underline{\text{STD}}\\ \text{PCI}\end{matrix}\right\}\right]$
		[STORAGE=PARTITIONED]
		<u>For STORAGE=PARTITIONED Only:</u>
		[HITASK=size]
		[LOWTASK=(size$_1$[,size$_2$][,size$_3$])]

Operand Field:

MAXIO=number
specifies the maximum number of I/O operations that can be simultaneously processed by the new operating system. This number is the sum of those I/O operations that are being executed simultaneously and those that are concurrently queued but are not being executed. (A recommended minimum value for MAXIO is the number of uniquely addressable I/O devices in the new system.)

Note: The maximum number of channel programs that can be started when using access methods, or graphic programming services is limited by this number.

OVERLAY
specifies overlay supervisor options. BASIC specifies synchronous overlay without exclusive call checking. ADVANCED specifies synchronous overlay with error checking for invalid SEGWT instructions. If ADVANCED is specified, E must not be specified as the value of the STORAGE keyword of the CENPROCS macro-instruction.

FETCH
specifies the type of program fetch. STD specifies standard fetch. PCI specifies the use of Program Controlled Interrupt while fetching a program into storage.

STORAGE=PARTITIONED
specifies the system as partitioned. If this parameter is not specified a sequential scheduling system is assumed.

HITASK=size$_n$
specifies the size, in bytes, of the highest priority partition. The value specified must be a decimal integer, and it must also be a multiple of 8 and at least as large as the scheduler design level specified in the SCHEDULR macro-instruction.

LOWTASK

specifies the sizes, in bytes, of one, two, or three low priority partitions. The value specified must be a decimal integer, and it must also be a multiple of 8 and at least as large as the scheduler design level specified in the SCHEDULR macro-instruction.

Note: If PROTECT is specified as a value of the OPTIONS keyword of the SUPRVSOR macro-instruction, the values given to HITASK and LOWTASK must be multiples of 2048.

Example: The following example illustrates the use of the CTRLPROG macro-instruction to specify that the maximum number of I/O operations that can be processed simultaneously is 20. The basic overlay supervisor and standard fetch are included.

```
   CTRLPROG MAXIO=20
```

SCHEDULR

The SCHEDULR macro-instruction is used to specify job scheduler options. This macro-instruction is required.

Name	Operation	Operand
	SCHEDULR	$\left[\text{DESIGN=}\left\{\begin{array}{l}\underline{\text{18K}}\\ \text{44K}\\ \text{100K}\end{array}\right\}\right]$
		$\text{CONSOLE=}\left\{\begin{array}{l}\text{address}\\ \text{(I-address,O-address)}\end{array}\right\}$
		$\left[\text{ALTCONS=}\left\{\begin{array}{l}\text{address}\\ \text{(I-address,O-address)}\end{array}\right\}\right]$
		[STARTR=(A-address[,V-serial])]
		[STARTW=(A-address[,V-serial])]
		[CANCEL=(condition$_1$[,condition$_2$])]
		$\left[\text{ACCTRTN=}\left\{\begin{array}{l}\underline{\text{NOTSUPPLIED}}\\ \text{SUPPLIED}\end{array}\right\}\right]$
		$\left[\text{TSYSIN=}\left\{\begin{array}{l}\underline{\text{200}}\\ \text{556}\\ \text{800}\end{array}\right\}\right]$
		$\left[\text{TSYSOUT=}\left\{\begin{array}{l}\underline{\text{200}}\\ \text{556}\\ \text{800}\end{array}\right\}\right]$
		[WTOBFRS=value]
		[REPLY=value]

Operand Field:

DESIGN
specifies the design level (in bytes) of the job scheduler.

CONSOLE
specifies the address or addresses of the primary console device. If a composite console is used (e.g., a card reader and a printer) as the primary console, I-address specifies the address of the input device and O-address specifies the address of the output device. The address value(s) must be the same as that specified for the device(s) in the IODEVICE macro-instruction(s).

ALTCONS=address
specifies the address of an alternative console device. If a composite console is used as an alternative console, I-address specifies the address of the input device and O-address specifies the address of the output device. For sequential scheduling systems, if the primary console is a composite console, an alternative console cannot be specified. An alternative composite console can only be specified for partitioned systems. The address value must be the same as that specified for the device in the IODEVICE macro-instruction.

STARTR
specifies that a START RDR command is to be executed automatically each time the new operating system is loaded into main storage after IPL. A-address specifies the address of the I/O device to be started. The address value must be the same as that specified for the device in the IODEVICE macro-instruction. V-serial specifies the serial number of the labeled volume associated with the I/O device.

STARTW
: specifies that a START WTR command is to be executed automatically each time the new operating system is loaded into main storage after IPL. A-address specifies the address of the device to be started. The address value must be the same as that specified for the device in the IODEVICE macro-instruction. V-serial specifies the serial number of the labeled volume associated with the device.

CANCEL
: specifies conditions under which a job is to be cancelled without being executed. NONAME specifies that a job is to be cancelled if the programmer's name field is omitted from the JOB statement. NOACCNUM specifies that a job is to be cancelled if the account number field is omitted from the JOB statement.

ACCTRTN
: specifies whether or not the user supplies an accounting routine. NOTSUPPLIED specifies that the user does not intend to provide an accounting routine. SUPPLIED specifies that the user supplies (or will supply after system generation) an accounting routine.

TSYSIN
: specifies the standard magnetic tape density for the system input unit as either 200, 556, or 800 characters per inch. (This parameter applies only to 7-track magnetic tape.)

TSYSOUT
: specifies the standard magnetic tape density for the system output unit as either 200, 556, or 800 characters per inch. (This parameter applies only to 7-track magnetic tape.)

Note: If either system input or output is on 7-track magnetic tape, the drives must have the data conversion feature.

WTOBFRS=value
: specifies the number of buffers to be used by the WTO routines. Each write buffer is 144 bytes long. If this parameter is omitted, a value of 5 is assumed.

REPLY=value
: specifies the number of reply queue elements to be used by the WTOR routines. Each reply queue element is 24 bytes long. If this parameter is omitted, a value of 5 is assumed.

Note: WTOBFRS and REPLY can be specified if, and only if, STORAGE=PARTITIONED is specified in the CTRLPROG macro-instruction.

Example: The following example illustrates the use of the SCHEDULR macro-instruction to specify the 18K design level of the job scheduler. The address of the console device is 01F. The address of an alternative console is 21F. The START RDR command is to be executed after the system is loaded into main storage. The device to be started is located at 181. The standard magnetic tape density for system input and output is 200 characters per inch. The user will supply an accounting routine. In addition, the macro-instruction is used to specify the cancellation of all jobs whose account number is omitted.

```
  SCHEDULR CONSOLE=01F,ALTCONS=21F,STARTR=A-181,ACCTRTN=SUPPLIED,
                                                     CANCEL=NOACCNUM
```

UNITNAME

The UNITNAME macro-instruction is used to name a collection of I/O devices. A UNITNAME macro-instruction is required for each named collection of I/O devices in the system, except for generic unit names (see Appendix C). This macro-instruction is optional. If selected, all UNITNAME macro-instructions having the same NAME value must appear together in the input stream.

Name	Operation	Operand

	UNITNAME	UNIT=(address[,address]...)
		NAME=name

Operand Field:

UNIT=address$_n$
specifies the addresses of the I/O devices to be included in the collection. The addresses must be the same as those specified in the IODEVICE macro-instructions for these devices. The only combination of unlike device types permitted in a collection is magnetic tape and direct-access devices. The maximum number of devices that can be included in collections is determined by the following formula:

$$A=510-N$$

where:

A	is the maximum number of devices (a device may belong to more than one collection, but it must be counted as a separate device for each of the collections).
N	is the number of uniquely named collections. (The maximum value of N is 50.)

For example, if there are 40 collections, a maximum of 470 devices can be distributed among those collections.

NAME=name
specifies the name to be given to the collection of devices. The name may be from one to eight alphameric characters. A maximum of 50 uniquely named collections can be specified for the system.

Note: If the user intends to use IBM-supplied cataloged procedures, he must assign, using UNITNAME macro-instructions, certain names to collections of I/O devices. These names will be used by the IBM-supplied cataloged procedures (contained in the procedure library) to specify the I/O devices required. The names to be specified follow.

Name	Types of I/O Devices in the Collection
SYSSQ	magnetic tape, direct-access
SYSDA	direct-access
SYSCP	card punch

For further information on cataloged procedures see the publications IBM System/360 Operating System: System Programmer's Guide, and IBM System/360 Operating System: Utilities.

<u>Example:</u> The following example illustrates the use of the UNITNAME macro-instruction to assign the name TAPE to the devices located at 180, 181, 182, and 183.

```
   UNITNAME UNIT=(180,181,182,183),NAME=TAPE
```

PROCLIB

The PROCLIB macro-instruction is used to specify the inclusion of the procedure library (SYS1.PROCLIB) in the new operating system. SYS1.PROCLIB must be allocated space and, if desired, cataloged in the new system during the preparation for system generation. This library must also exist as a cataloged partitioned data set (SYS1.PROCLIB) in the operating system that is generating the new system. This macro-instruction is optional. If not chosen, a null data set must be made available to the generated system. (See "Estimated Sizes" in the section "Preparation for System Generation.") If IBM-supplied cataloged procedures are to be used, certain names must be assigned to collections of devices. Refer to the description of the UNITNAME macro-instruction for a list of these names.

Name	Operation	Operand

	PROCLIB	UNIT=name
		VOLNO=serial

Operand Field:

UNIT=name
specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the procedure library being generated.

VOLNO=serial
specifies the serial number of the volume that is to contain the procedure library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.PROCLIB during the preparation for system generation.

Note: If these parameters are omitted, the procedure library is placed on the new system residence volume. (However, if one of these parameters is specified, both must be specified.)

Example: This example illustrates the use of the PROCLIB macro-instruction to specify the inclusion of the procedure library in the operating system to be generated. The unit name is 2311. The volume serial number is 909090.

```
   PROCLIB UNIT=2311,VOLNO=909090
```

SUPRVSOR

The SUPRVSOR macro-instruction is used to select task supervisor options. This macro-instruction is optional.

Name	Operation	Operand
	SUPRVSOR	[OPTIONS=(option[,option]...)] [RESIDNT=(function[,function]...)] $\left[\text{WAIT=}\begin{Bmatrix}\underline{\text{SINGLE}}\\\text{MULTIPLE}\end{Bmatrix}\right]$ $\left[\text{TIMER=}\begin{Bmatrix}\text{TIME}\\\text{INTERVAL}\end{Bmatrix}\right]$ [TRACE=number] $\left[\text{SER=}\begin{Bmatrix}\text{SER0}\\\text{SER1}\end{Bmatrix}\right]$

Operand Field:

OPTIONS=option$_n$
specifies supervisor options as one or more of the following values. These values may be listed in any order.

Value	Option
IDENTIFY	The IDENTIFY function is to be included.
TRSVCTBL	A table containing the relative track addresses of all transient SVCs is to be stored in the resident portion of the control program.
VALIDCHK	The WAIT, POST, and GETMAIN/FREEMAIN modules are to contain extra validity checking to determine whether addresses are located within proper boundaries. The validity checking for WAIT also checks for the number of events.
COMM	Specifies communication with the operator at IPL time for the purpose of changing options specified by the RESIDNT keyword. The two options that can be changed at IPL time are BLDLTAB and ACSMETH. The communications procedure is described in the IBM System/360 Operating System: Operator's Guide publication under the discussion of NIP messages.
PROTECT	Specifies the inclusion of the protect function when the protect feature is part of the central processing unit. (PROTECT includes the VALIDCHK option.)

RESIDNT=function$_n$
specifies that one or more of the following routines, normally executed from the transient area, are to be included in the resident portion of the control program. The values are ATTACH, EXTRACT, IDENTIFY, SPIE, BLDLTAB and ACSMETH. BLDLTAB specifies that directory entries are to be made part of the nucleus at IPL time for selected linkage library modules. ACSMETH specifies that access method modules are to be loaded and made part of the nucleus at IPL time. The use of these options is discussed in the publication IBM System/360 Operating System: System Programmer's Guide.

Note: If IDENTIFY is specified as a value of the RESIDNT keyword, it need not be specified as a value of the OPTIONS keyword.

WAIT
: specifies, as either SINGLE or MULTIPLE, the number of events that can be specified in a WAIT macro-instruction.

TIMER
: specifies the inclusion of the timer function when the timer feature is part of the central processing unit. TIME provides the ability to request date plus time of day in various units of measurement. INTERVAL provides the same functions, plus the ability to request, check, and cancel intervals of time.

TRACE=number
: specifies the inclusion of an optional trace table. The value is the number of entries in the table. (See the publication IBM System/360 Operating System: System Programmer's Guide for a description of the trace table.)

SER
: specifies the system environment recording (SER) level desired. Either SER0 or SER1 may be specified. If this parameter is included, the nucleus generated is CPU dependent, that is, it can only be used in the CPU model specified in the CENPROCS macro-instruction. (This parameter is valid only on models 40, 50 and 65.) For further information on this parameter, refer to the publication IBM System/360 Operating System: Operators Guide.

Example: The following example illustrates the use of the SUPRVSOR macro-instruction to specify that a table containing the relative track addresses of transient SVCs be stored in the resident portion of the control program, that the IDENTIFY and the ATTACH functions be included in the resident portion of the control program, and that multiple events can be specified in a WAIT macro-instruction. There are 100 entries in the trace table.

```
 SUPRVSOR OPTIONS=TRSVCTBL,RESIDNT=(IDENTIFY,ATTACH),WAIT=MULTIPLE,
                                                         TRACE=100
```

SVCTABLE

The SVCTABLE macro-instruction is used to specify the number, type, and SVRB extended save area of the user-written supervisor call (SVC) routines that are to be added to the new operating system. This macro-instruction is optional.

Name	Operation	Operand
	SVCTABLE	operand[,operand]...

Operand Field: Each operand must be written in the following format:

SVC-nnn-Ta-Sb

Upper case letters and hyphens (-) must be written exactly as shown.

nnn
: specifies the SVC number. The highest SVC number that may be assigned is 255. The user must assign unique numbers to his SVC routines, and should assign them in descending order starting with 255 and ending with 200 to avoid conflict with the numbers assigned to IBM-written SVC routines.

a
: specifies the SVC type as either 1, 2, 3, or 4.

b
: specifies the size of the extended save area of the SVRB associated with the SVC routine. The value indicates the number of double words by which the SVRB is to be extended. A type 1 SVC must have a value of 0. Types 2, 3, and 4 can have a value of from 0 to 6.

Note: For each type 1 or type 2 SVC, there should be a corresponding module specified in the RESMODS macro-instruction; one module may contain more than one resident SVC routine. For each type 3 SVC, there should be a corresponding module specified in the SVCLIB macro-instruction; each module may contain only one transient SVC routine. For each type 4 SVC, there should be one corresponding module specified in the SVCLIB macro-instruction for each load module of the SVC routine. (For further information on user-written SVC routines refer to the publication IBM System/360 Operating System: System Programmer's Guide.)

Examples: The following example illustrates the use of the SVCTABLE macro-instruction to specify that four user-written SVCs are to be added to the operating system to be generated.

```
SVCTABLE SVC-255-T4-S5,SVC-254-T2-S3,SVC-253-T3-S1,SVC-252-T1-S0
```

RESMODS

The RESMODS macro-instruction is used to add user-written routines, in load module form, to the nucleus library (SYS1.NUCLEUS) to be generated. Before these load modules can be included in the nucleus library, they must be members of a partitioned data set. This data set must have been cataloged, as SYS1.name, in the operating system that is generating the new system. This macro-instruction is optional.

Name	Operation	Operand

	RESMODS	PDS=SYS1.name
		MEMBERS=(name[,name]...)

Operand Field:

PDS=SYS1.name
specifies the name of the partitioned data set that contains the load modules to be included. The value of name cannot exceed eight alphameric characters, the first of which must be alphabetic.

MEMBERS=$name_n$
specifies the simple names of the members to be included. The value of name cannot exceed eight alphameric characters, the first of which must be alphabetic. A maximum of ten load modules can be included in the nucleus library.

Note: If resident SVC routines are being included, each load module can contain more than one SVC routine. The type, number, and SVRB extended save area of each of the resident SVC routines to be included must be specified in the SVCTABLE macro-instruction. (For further information on user-written SVC routines, refer to the publication IBM System/360 Operating System: System Programmer's Guide.)

Example: This example illustrates the use of the RESMODS macro-instruction to include the load modules CONTROL and IORTN in the nucleus library. These modules are members of the SYS1.NEW partitioned data set.

```
   RESMODS PDS=SYS1.NEW,MEMBERS=(CONTROL,IORTN)
```

SVCLIB

The SVCLIB macro-instruction is used to add transient user-written routines, in load module form, to the SVC library (SYS1.SVCLIB) during system generation. Before these routines can be included in the SVC library they must be members of a partitioned data set. This data set must have been cataloged, as SYS1.name, in the operating system that is generating the new system. The number, type, and SVRB extended save area of each of the SVC routines to be added must be specified in the SVCTABLE macro-instruction. The SVCLIB macro-instruction is optional.

Name	Operation	Operand

	SVCLIB	PDS=SYS1.name
		MEMBERS=(name[,name]...)

Operand Field:

PDS=SYS1.name
: specifies the name of the partitioned data set that contains the routines to be added. The value of name cannot exceed eight alphameric characters, the first of which must be alphabetic.

MEMBERS=name$_n$
: specifies the names of the members to be added. Each name (unless referring to nonstandard label routines) must be of the form:

IGCssnnn

where:

ss
: is the number of the load module minus one, e.g., the second load module has a value of 01. The value of ss is always 00 for type 3 SVC routines.

nnn
: is an SVC number. nnn must be a signed decimal integer (e.g., 242=24B) if the routine is called directly by SVC.

The names of nonstandard label routines must begin with NS and can not exceed eight alphameric characters. With this macro-instruction, a maximum of 50 members may be added to the SVC library. (For further information on user-written SVC routines and nonstandard label routines refer to the publication IBM System/360 Operating System: System Programmer's Guide.)

Example: The following example illustrates the use of the SVCLIB macro-instruction to add the routines named IGC0025E, IGC0025D, IGC0025C, and IGC0025B to the SVC library. These routines are members of the SYS1.USERSVC partitioned data set, and are called directly by an SVC.

```
   SVCLIB PDS=SYS1.USERSVC,MEMBERS=(IGC0025E,IGC0025D,IGC0025C,
                                                      IGC0025B)
```

LINKLIB

The LINKLIB macro-instruction is used to add user-written routines, in load module form, to the link library (SYS1.LINKLIB) during system generation. Before these routines can be included in the link library, they must be members of a partitioned data set. This data set must have been cataloged, as SYS1.name, in the operating system that is generating the new system. This macro-instruction is optional.

Name	Operation	Operand

	LINKLIB	PDS=SYS1.name
		MEMBERS=(name[,name]...)

Operand Field:

PDS=SYS1.name
specifies the name of the partitioned data set that contains the routines to be added. The value of name cannot exceed eight alphameric characters, the first of which must be alphabetic.

MEMBERS=$name_n$
specifies the names of the members to be added. The value of name cannot exceed eight alphameric characters, the first of which must be alphabetic. With this macro-instruction, a maximum of 20 routines may be included in the link library during one system generation process.

Example: The following example illustrates the use of the LINKLIB macro-instruction to add the routines PAYROLL, COMPILER, and MULT to the link library. These routines are members of the SYS1.USER partitioned data set.

```
  LINKLIB PDS=SYS1.USER,MEMBERS=(PAYROLL,COMPILER,MULT)
```

DATAMGT

The DATAMGT macro-instruction allows the user to specify optional access methods. This macro-instruction is optional.

Name	Operation	Operand
	DATAMGT	ACSMETH=(method[,method]...)

Operand Field:

ACSMETH=method$_n$
specifies the optional access methods to be included as one or more of the following values. These values may be listed in any order.

Value	Access Method
BDAM	Basic direct access method (BDAM) and routines for creating a direct data set.
ISAM	Queued and basic index sequential access methods (QISAM and BISAM).
BTAM	Basic telecommunications access method (BTAM).

Example: The following example illustrates the use of the DATAMGT macro-instruction to specify that the basic direct access method is to be included in the operating system to be generated.

```
DATAMGT ACSMETH=BDAM
```

SYSUTILS

The SYSUTILS macro-instruction is used to specify the amount of main storage available to the system and data set utilities. System and data set utilities are generated with every operating system and they will operate in 15K bytes of main storage unless this macro-instruction is used to specify a larger amount. This macro-instruction is optional.

Name	Operation	Operand
	SYSUTILS	SIZE=size

Operand Field:

SIZE=size
: specifies the amount of main storage, in bytes, available to the system and data set utilities. The value specified must be an integer of from 15360 to 999999, or it may be of the form nnnnK where nnnn is an integer of from 15 to 9999 and K represents 1024 bytes. If this parameter is omitted, a value of 15360 is assumed.

Example: The following example illustrates the use of the SYSUTILS macro-instruction to specify that there are 44K bytes of main storage available to the system and data set utilities.

```
    SYSUTILS SIZE=44K
```

EDITOR

The EDITOR macro-instruction is used to specify the inclusion of the linkage editor. This macro-instruction is optional. This macro-instruction must be issued once for each design level to be generated.

IEWLE150, IEWLE180, IEWLF440, and IEWLF880 are the names of the 15K, 18K, 44K, and 88K linkage editors, respectively. The alias IEWL (used for cataloged procedures) is given to the linkage editor chosen or, if more than one is chosen, it is given to the largest.

Name	Operation	Operand
	EDITOR	DESIGN= $\begin{Bmatrix}\text{E15}\\\text{E18}\\\text{F44}\\\text{F88}\end{Bmatrix}$

Operand Field:

DESIGN
: specifies the design level of the linkage editor to be included. E15 specifies the E-design-level linkage editor that operates in 15K bytes of main storage. E18 specifies the E-design-level linkage editor that operates in 18K bytes of main storage. F44 specifies the F-design-level linkage editor that operates in 44K bytes of main storage. F88 specifies the F-design-level linkage editor that operates in 88K bytes of main storage.

Example: The following example illustrates the use of the EDITOR macro-instruction to specify the E-design-level linkage editor that operates in 18K bytes of main storage.

```
   EDITOR DESIGN=E18
```

ASSEMBLR

The ASSEMBLR macro-instruction is used to specify the inclusion of the assembler language processor. This macro-instruction is optional. However, it must be issued once for each design level to be generated.

IETASM and IEUASM are the names of the assembler E and assembler F, respectively. The alias ASMBLR is given to the assembler chosen or, if both are chosen, it is given to assembler F.

Name	Operation	Operand
	ASSEMBLR	DESIGN= $\left\{\begin{matrix} E \\ F \end{matrix}\right\}$

Operand Field:

DESIGN
specifies the design level of the assembler language processor as E or F.

Note: The E-design-level of the assembler language processor requires an unblocked macro library (SYS1.MACLIB). If the E-design-level assembler language processor is to be used for future system generations, SYS1.GENLIB must also be unblocked.

Example: The following example illustrates the use of the ASSEMBLR macro-instruction to specify the E design level of the assembler language processor.

```
  ASSEMBLR DESIGN=E
```

TESTRAN

The TESTRAN macro-instruction is used to specify the inclusion of test translator options in the new operating system. This macro-instruction is optional.

Name	Operation	Operand
	TESTRAN	PHASES=(phase[,phase]) $\left[\text{MODE=}\begin{Bmatrix}\underline{\text{NOTRACE}}\\ \text{TRACE}\end{Bmatrix}\right]$ [PAGES=numbers] [EXEC=statements]

Operand Field:

PHASES=phase$_n$
: specifies the test translator components to be included as one or both of the following values.

Value	Component
INTER	The TESTRAN interpreter facility is to be included.
EDITOR	The TESTRAN editor facility is to be included.

MODE
: specifies the inclusion of the tracing facilities of the test translator. TRACE specifies that the tracing facilities are to be included. NOTRACE specifies that the tracing facilities are not to be included. This parameter is meaningful only if INTER is specified in the PHASES parameter.

PAGES=number
: specifies the maximum number of pages to be produced during one execution of the test translator. The value specified must be a positive decimal integer smaller than 999. This parameter is required if and only if INTER is specified in the PHASES parameter.

EXEC=statements
: specifies the maximum number of test translator statements that are to be interpreted during one execution of the test translator. The value specified must be a positive decimal integer smaller than 65535. This parameter is required if and only if INTER is specified in the PHASES parameter.

Example: The following example illustrates the use of the TESTRAN macro-instruction to specify a test translator with tracing facilities. The test translator interpreter is specified. During one execution of the test translator the maximum number of pages to be written is 50, and the maximum number of test translator statements to be interpreted is 200.

```
  TESTRAN MODE=TRACE,PHASES=INTER,PAGES=50,EXEC=200
```

MACLIB

The MACLIB macro-instruction is used to specify the inclusion of the macro library (SYS1.MACLIB) in the new operating system. SYS1.MACLIB must be allocated space and, if desired, cataloged in the new system during the preparation for system generation. This library must also exist as a cataloged partitioned data set (SYS1.MACLIB) in the operating system that is generating the new system. This macro-instruction is optional.

If the new system is to contain the E-design-level assembler language processor, SYS1.MACLIB must be unblocked. Unblocking is accomplished by allocating space to SYS1.MACLIB during the preparation for system generation with a BLKSIZE value of 80 rather than 3360 (see Figures 8 and 9).

This macro-instruction may not be used if there are only two IBM 2311 Disk Storage drives in the computing system configuration used to generate the new system, unless SYS1.MACLIB is on the system residence volume of the generating system. If it is not on the system residence volume of the generating system, SYS1.MACLIB may be included after system generation in the new system with the IEBCOPY utility program.

Name	Operation	Operand

	MACLIB	UNIT=name
		VOLNO=serial

Operand Field:

UNIT=name
specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the macro library being generated.

VOLNO=serial
specifies the serial number of the volume that is to contain the macro library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.MACLIB during the preparation for system generation.

Note: If these parameters are omitted, the macro library is placed on the new system residence volume. (If one of these parameters is specified, both must be specified.)

Example: This example illustrates the use of the MACLIB macro-instruction to specify the inclusion of the macro library in the operating system to be generated. The unit name is 2311. The volume serial number is 003475.

```
  MACLIB UNIT=2311,VOLNO=003475
```

SORTMERG

The SORTMERG macro-instruction is used to specify the inclusion of sorting and merging functions in the new operating system. Either all sort/merge functions or selected sort/merge functions can be specified for inclusion. However, any function not specified through the SORTMERG macro-instruction must not be specified on sort/merge control cards at execution time. For example, if the sorting of only fixed length records is specified during system generation, the sorting of variable length records must not be requested at sort/merge execution time. If such sorting were specified, the sort/merge would be terminated because the programs for sorting variable length records would not be in the generated operating system. This macro-instruction is optional. If it is used, the EDITOR and the SORTLIB macro-instructions must also be specified.

Name	Operation	Operand
	SORTMERG	[SIZE=size] [SORTOPT=FULLIB] _If SORTOPT=FULLIB is not chosen_: RECTYPE=(rectype[,rectype]...) SORTDEV=(device[,device]) CNTLFLD=(cntlfld[,cntlfld]) [MERGE=MERGONLY] [MESSAGE=(msgopt[,msgopt])] [SORTOPT=MODPRGM]

Operand Field:

SIZE=size
: specifies, as a positive decimal integer, the maximum amount of main storage, in bytes, that can be used for sorting. This amount of main storage is used only for sorting and does not include the space required for data management functions. If this parameter is omitted, a value of 12000 is assumed.

Note: The sort/merge program operates in 17000 bytes of main storage. Of these 17000 bytes, 12000 (minimum value that can be given to SIZE) are used for sorting. The user can specify in the SIZE parameter that a larger amount of main storage be used for sorting. The sort/merge program issues a variable GETMAIN macro-instruction to request storage in which to operate. The main storage thus made available to the sort/merge program will vary from 12000 bytes to any other amount specified by the user in the SIZE parameter.

The maximum value that can be specified for SIZE is the difference between the total amount of main storage available, and the amount required for data management routines. The following formula can be used to determine the value of SIZE:

size=A-24N-Y-5000

where:

A

is the total amount of main storage available for execution. (The maximum amount is the number of bytes of main storage as specified in the CENPROCS macro-instruction, minus the bytes required for the nucleus of the operating system minus, if sort/merge is called by another program, the bytes occupied by other programs.)

N

is the maximum number of DD statements to be used in any user's sort/merge program.

Y

is a constant with a value of 1500 which must be entered into the formula if any messages are to be written on SYSOUT. Otherwise, the value of Y is 0.

SORTOPT=FULLIB

specifies that all sort/merge functions be included in the user's operating system. If this parameter is written, the RECTYPE, SORTDEV, CNTLFLD, MERGE, MESSAGE, and SORTOPT=MODPRGM parameters must be omitted. (The values of CONSOLE and ALL are assumed for the MESSAGE keyword.)

RECTYPE=rectype_n

specifies the type and length of records to be sorted or merged as one or more of the following values:

Value	Records
VAR	Variable length records.
FIXED	Fixed length records.
LONG	Records longer than 256 bytes.

Note: VAR or FIXED or both must be specified.

SORTDEV=device_n

specifies the device(s) to be used for sorting or merging as one or both of the following values: 2311, 2400.

Note: The value of 2400 stands for 2401, 2402, 2403, 2404 and 2415.

CNTLFLD=cntlfld_n

specifies control field requirement(s) for sorting and/or merging. One or both of the following values must be specified:

Value	Control Field
SINGLE	Single control fields
MULTIPLE	Multiple control fields

MERGE=MERGONLY

specifies that the merge routines of the sort/merge processor can be executed independently from the sort routines for a merging application.

MESSAGE=mesgopt$_n$

specifies the I/O device on which sort/merge messages are to be printed, as well as the type of messages to be produced. If the MESSAGE operand is omitted, no messages are printed during a sorting or merging operation. The values included in braces { } are mutually exclusive. Acceptable values for this operand are:

Value	Meaning
{PRINTER	Sort/merge messages are to be printed on a printer.
CONSOLE}	Sort/merge messages are to be printed on a console typewriter.
{ALL	All sort/merge messages are to be printed.
CRITICAL}	Only serious diagnostic sort/merge messages are to be printed.

SORTOPT=MODPRGM

specifies the inclusion, at sort/merge execution time, of user-written modification programs.

<u>Example:</u> The following example illustrates the use of the SORTMERG macro-instruction to specify the use of fixed length records, single control fields, and IBM 2311 Disk Storage Drives. The merge functions are to be included. The maximum amount of main storage to be used for sorting is 12000 bytes.

```
    SORTMERG RECTYPE=FIXED,CNTLFLD=SINGLE,SORTDEV=2311,MERGE=MERGONLY
```

SORTLIB

The SORTLIB macro-instruction is used to specify the inclusion of the sort/merge subroutine library (SYS1.SORTLIB) in the new operating system. SYS1.SORTLIB must be allocated space and, if desired, cataloged in the new system during the preparation for system generation. This library must also exist as a cataloged partitioned data set (SYS1.SORTLIB) in the operating system which is generating the new system. This macro-instruction is optional.

Name	Operation	Operand

	SORTLIB	UNIT=name
		VOLNO=serial

Operand Field:

UNIT=name
: specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the sort subroutine library being generated.

VOLNO=serial
: specifies the serial number of the volume that is to contain the sort subroutine library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.SORTLIB during the preparation for system generation.

Note: If these parameters are omitted, the sort subroutine library is placed on the new system residence volume. (If one of these parameters is specified, both must be specified.)

Example: This example illustrates the use of the SORTLIB macro-instruction to specify the inclusion of the sort subroutine library in the operating system to be generated. The unit name is 2311. The volume serial number is 654321.

```
   SORTLIB UNIT=2311,VOLNO=654321
```

COBOL

The COBOL macro-instruction is used to specify the inclusion of the COBOL compiler. This macro-instruction is optional. If it is used, either COMM or UNIV must be specified as the value of the INSTSET keyword in the CENPROCS macro-instruction. (The commercial or the universal instruction set is required for COBOL compilations and executions.) UNIV is required if either floating point literals are used at compilation time, or if exponentiation to a non-integer power or floating point numbers are used at object time. If the COBOL macro-instruction is included, the COBLIB macro-instruction must also be included. This macro-instruction must be issued once for each design level to be generated.

Name	Operation	Operand
	COBOL	$\text{DESIGN=}\left\{\begin{array}{l}\text{E}\\\text{F}\end{array}\right\}$ $\left[\text{MSGLEV=}\left\{\begin{array}{l}\underline{\text{FLAGW}}\\\text{FLAGE}\end{array}\right\}\right]$ [LINECNT=lines] For COBOL E Only: $\left[\text{DATAMAP=}\left\{\begin{array}{l}\underline{\text{DMAP}}\\\text{NODMAP}\end{array}\right\}\right]$ $\left[\text{PROCMAP=}\left\{\begin{array}{l}\underline{\text{PMAP}}\\\text{NOPMAP}\end{array}\right\}\right]$ $\left[\text{DISPCHK=}\left\{\begin{array}{l}\underline{\text{DISPCK}}\\\text{NODISPCK}\end{array}\right\}\right]$ [BUFSIZE=number] $\left[\text{EDIT=}\left\{\begin{array}{l}\underline{\text{REGED}}\\\text{INVED}\end{array}\right\}\right]$ For COBOL F Only: [SIZE=size] [BUF=number] $\left[\text{SORLIST=}\left\{\begin{array}{l}\underline{\text{SOURCE}}\\\text{NOSOURCE}\end{array}\right\}\right]$ $\left[\text{STORMAP=}\left\{\begin{array}{l}\underline{\text{NOMAP}}\\\text{MAP}\end{array}\right\}\right]$ $\left[\text{PUNCH=}\left\{\begin{array}{l}\underline{\text{NODECK}}\\\text{DECK}\end{array}\right\}\right]$ $\left[\text{TYPERUN=}\left\{\begin{array}{l}\underline{\text{LOAD}}\\\text{NOLOAD}\end{array}\right\}\right]$ $\left[\text{SEQCHK=}\left\{\begin{array}{l}\underline{\text{SEQ}}\\\text{NOSEQ}\end{array}\right\}\right]$ $\left[\text{BASIS=}\left\{\begin{array}{l}\underline{\text{BAS}}\\\text{NOBAS}\end{array}\right\}\right]$ $\left[\text{COPY=}\left\{\begin{array}{l}\underline{\text{COPY}}\\\text{NOCOPY}\end{array}\right\}\right]$ $\left[\text{SPACE=}\left\{\begin{array}{l}\underline{\text{SPACE1}}\\\text{SPACE2}\\\text{SPACE3}\end{array}\right\}\right]$

<u>Operand Field:</u> Many of the parameters of the COBOL macro-instruction specify the setting of default options at compilation time. Default options are the options that are assumed if the corresponding values of the PARM keyword are omitted from an EXEC statement in a COBOL compilation.

DESIGN
: specifies the design level of the COBOL compiler as either E or F.

MSGLEV
: specifies the default option at compilation time for the type of compilation error messages to be printed. FLAGW specifies that all warning and error messages are to be printed. FLAGE specifies that warning messages are not to be printed.

LINECNT=lines
: specifies the default option at compilation time for the number of lines to be printed on each page of the COBOL compiler output listing. The value specified must be a two-digit integer of from 10 to 99. If this value is omitted, a value of 60 is assumed.

DATAMAP
: specifies the default option at compilation time for the production of a listing of the data-names and their addresses either relative to load point for the Working Storage Section or relative to the record addresses for the File or Linkage Sections. DMAP specifies that a listing is to be produced; NODMAP specifies that the listing is not to be produced.

PROCMAP
: specifies the default option at compilation time for the production of a listing of the generated instructions for each statement in the Procedure Division. PMAP specifies that the listing is to be produced; NOPMAP specifies that the listing is not to be produced.

DISPCHK
: specifies the default option at compilation time for the generation of object code which determines if a field to be displayed exceeds the record length of the device on which it is to be written. DISPCK specifies that a check is to be made; NODISPCK specifies that no check is required

BUFSIZE=number
: specifies the default option at compilation time for the size, in bytes, of each of the six work buffers used during a COBOL compilation. The valid range of values for magnetic tape is 180 to 32000; for volumes on 2311 Disk Storage drives, 180 to 3600; for volumes on 2301 Drum Storage drives, 180 to 20000. (The maximum size is an object time option and it is not checked during system generation.) If this value is omitted, a value of 180 is assumed. The following formula can be used as a guide to determine the maximum value that can be specified to optimize the allocation of available storage for the data-name table and work buffers. (Any remainder should be ignored.)

$$\text{number}=\frac{M-30000-[(13+L)(N)]}{6}$$

where:

number
: is the size of each work buffer. If the result is less than 180, 180 must be specified.

M

is the size (in bytes) of main storage. This value should correspond to the size specified in the CENPROCS macro-instruction.

L

is the length of the average data-name.

N

is the number of data-names.

EDIT

specifies the default option at compilation time for the editing function to be used by the compiler. REGED specifies that the standard monetary editing function will be used. INVED specifies that the inverted monetary editing function will be used.

SIZE=size

specifies the default size at compilation time for the number of bytes of main storage available to the COBOL F compiler. The value specified must be an integer of from 65536 to 9999999. If this value is omitted, a value of 65536 is assumed.

BUF=number

specifies the default option at compilation time for the number of bytes of main storage to be used for buffer allocation by the COBOL F compiler. The value specified must be an integer of from 2762 to 99999. This value must be included in the value given to the SIZE parameter. If BUF is omitted and SIZE is specified, the value of BUF is calculated as:

$$\frac{SIZE-65536}{4} + 2762$$

If both BUF and SIZE are omitted, a value of 2762 is assumed for BUF.

SORLIST

specifies the default option at compilation time for the production of a listing of the COBOL source program. SOURCE specifies that the listing is to be produced; NOSOURCE specifies that the listing is not to be produced.

STORMAP

specifies the default option at compilation time for the production of a listing of the data-names and their internal attributes in the Data Division and the generated instructions for the statements in the Procedure Division. MAP specifies that a listing is to be produced; NOMAP specifies that a listing is not to be produced.

PUNCH

specifies the default option at compilation time for the production of a punched deck of the object program. DECK specifies that a punched deck is to be produced; NODECK specifies that a punched deck is not to be produced.

TYPERUN

specifies the default option at compilation time for the production of input to the linkage editor from the program being compiled. LOAD specifies that the program is to be processed by the linkage editor after compilation; NOLOAD, specifies that the program is to be compiled only.

SEQCHK
: specifies the default option at compilation time for the checking of the source program card sequence numbers. SEQ specifies that the source program card sequence numbers are to be checked; NOSEQ specifies that the source program card sequence numbers are not to be checked.

BASIS
: specifies the default option at compilation time for the possible presence of a BASIS card in the source program. BAS specifies that a BASIS card may be present in the source program; NOBAS specifies that a BASIS card is not present in the source program. (This parameter is used to optimize buffer allocation at compilation time.)

COPY
: specifies the default option at compilation time for the possible presence of a COPY and/or INCLUDE clause in the source program. COPY specifies that a COPY and/or INCLUDE clause may be present in the source program; NOCOPY specifies that neither a COPY nor an INCLUDE clause is present in the source program. (This parameter is used to optimize buffer allocation at compilation time.)

SPACE
: specifies the default option at compilation time for the line spacing on the listing obtained when the SOURCE and/or MAP options are specified. SPACE1 specifies single spacing; SPACE2 specifies double spacing, and SPACE3 specifies triple spacing.

Example: The following example illustrates the use of the COBOL macro-instruction to specify an E-design-level COBOL compiler. The number of lines to be printed in each compiler output listing is 55. Listings of data-names and their address, and listings of the generated instructions for each statement in the Procedure Division are to be produced. All warning and error messages are to be printed. The generation of object code to determine the length of fields to be displayed is not required. The size of each of the six work buffers used during a COBOL compilation is 5708. The standard monetary editing function will be used.

The formula used to compute the BUFSIZE value is as follows:

$$\frac{65536-30000-[(13+10)(56)]}{6}=5708$$

where the main storage specified is 64K, the length of the average data-name is 10, and the number of data names is 56.

```
 COBOL DESIGN=E,LINECNT=55,DISPCHK=NODISPCK,BUFSIZE=5708
```

COBLIB

The COBLIB macro-instruction is used to specify the inclusion of the COBOL subroutine library (SYS1.COBLIB) in the new operating system. SYS1.COBLIB must be allocated space and, if desired, cataloged in the new system during the preparation for system generation. This library must also exist as a cataloged partitioned data set (SYS1.COBLIB) in the operating system that is generating the new system. This macro-instruction is optional. If a combined E and F design level subroutine library is required, two COBLIB macro-instructions must be issued: one specifying DESIGN=E and one specifying DESIGN=F.

Name	Operation	Operand
	COBLIB	DESIGN=$\left\{\begin{matrix}E\\F\end{matrix}\right\}$ [UNIT=name] [VOLNO=serial]

Operand Field:

DESIGN
: specifies the design level of the subroutine library as either E or F.

UNIT=name
: specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the COBOL subroutine library being generated.

VOLNO=serial
: specifies the serial number of the volume that is to contain the COBOL subroutine library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.COBLIB during the preparation for system generation.

Note: If the UNIT and VOLNO parameters are omitted, the COBOL subroutine library is placed on the new system residence volume. (However, if one of these parameters is specified, both must be specified.) If a combined E- and F-design-level subroutine library is to be included and these parameters are specified, they should be specified identically in both COBLIB macro-instructions. If they are specified differently or they are omitted from one of the macro-instructions, the second macro-instruction processed determines where the subroutine library is to reside.

Example: This example illustrates the use of the COBLIB macro-instruction to specify the inclusion of the F-design-level COBOL subroutine library in the operating system to be generated. SYS1.COBLIB is to reside on the new system residence volume.

```
  COBLIB DESIGN=F
```

FORTRAN

The FORTRAN macro-instruction is used to specify the inclusion of the FORTRAN compiler. This macro-instruction is optional. If it is used, either SCNTF or UNIV must be specified as the value of the INSTSET keyword in the CENPROCS macro-instruction. (The scientific or the universal instruction set is required for FORTRAN compilations and executions.) This macro-instruction must be issued once for each design level to be generated.

Name	Operation	Operand
	FORTRAN	$\text{DESIGN=}\left\{\begin{matrix}\text{E}\\ \text{G}\\ \text{H}\end{matrix}\right\}$ $\left[\text{PUNCH=}\left\{\begin{matrix}\underline{\text{NODECK}}\\ \text{DECK}\end{matrix}\right\}\right]$ $\left[\text{SORLIST=}\left\{\begin{matrix}\underline{\text{SOURCE}}\\ \text{NOSOURCE}\end{matrix}\right\}\right]$ $\left[\text{STORMAP=}\left\{\begin{matrix}\underline{\text{NOMAP}}\\ \text{MAP}\end{matrix}\right\}\right]$ $\left[\text{OBJPROG=}\left\{\begin{matrix}\underline{\text{LOAD}}\\ \text{NOLOAD}\end{matrix}\right\}\right]$ $\left[\text{SORCODE=}\left\{\begin{matrix}\underline{\text{EBCDIC}}\\ \text{BCD}\end{matrix}\right\}\right]$ For FORTRAN E only: [LINELNG=length] $\left[\text{STORAGE=}\left\{\begin{matrix}\underline{\text{SPACE}}\\ \text{PRFRM}\end{matrix}\right\}\right]$ $\left[\text{SCAN=}\left\{\begin{matrix}\underline{\text{NOADJUST}}\\ \text{ADJUST}\end{matrix}\right\}\right]$ For FORTRAN E and FORTRAN H only: [SIZE=size] For FORTRAN G and FORTRAN H only: $\left[\text{OBJLIST=}\left\{\begin{matrix}\underline{\text{NOLIST}}\\ \text{LIST}\end{matrix}\right\}\right]$ [LINECNT=lines] For FORTRAN H only: $\left[\text{OPT=}\left\{\begin{matrix}\underline{0}\\ 1\\ 2\end{matrix}\right\}\right]$

Operand Field: Many of the parameters of the FORTRAN macro-instruction specify the setting of default options at compilation time. Default options are the options that are assumed if the corresponding values of the PARM keyword are omitted from an EXEC statement in a FORTRAN compilation.

DESIGN
specifies the design level of the FORTRAN compiler as E, G, or H.

PUNCH
: specifies the default option at compilation time for the production of a punched deck of the object program. DECK specifies that a punched deck is to be produced; NODECK, that a punched deck is not to be produced.

SORLIST
: specifies the default option at compilation time for the production of a listing of the FORTRAN source program. SOURCE specifies that the listing is to be produced; NOSOURCE, that the listing is not to be produced.

STORMAP
: specifies the default option at compilation time for the production of a map showing the relative location of variables, constants, etc., in the source program. MAP specifies that the map is to be produced; NOMAP, that the map is not to be produced.

OBJPROG
: specifies the default option at compilation time for the production of input to the linkage editor from the program being compiled. LOAD specifies that the source program is to be processed by the linkage editor after compilation. NOLOAD specifies that the source program is only to be compiled.

SORCODE
: specifies the default option at compilation time that indicates the character set used to keypunch the source programs to be compiled. BCD specifies the BCD character set. EBCDIC specifies the EBCDIC character set.

LINELNG=length
: specifies the default option at compilation time for the maximum print line length at object time. At compilation time, if a FORMAT statement indicates a record larger than the length specified, a warning message is issued. The value specified must be a three-digit integer of from 001 to 255. If this parameter is omitted, a value of 132 is assumed.

STORAGE
: specifies the default option at compilation time for the use of main storage during a FORTRAN compilation. SPACE specifies that 15360 bytes of main storage are used for compilations and any amount specified in the SIZE keyword in excess of 15360 bytes is used for the dictionary, overflow table, and I/O buffers. PRFRM specifies that 19456 bytes of main storage are used for compilations and any amount specified in the SIZE keyword in excess of 19456 bytes is used for the dictionary, overflow table, and I/O buffers. (If PRFRM is specified, blocked input to and output from the compiler is allowed.)

Note: If PRFRM is specified, 19456 is the smallest value that can be specified in the SIZE keyword.

SCAN
: specifies the default option at compilation time that indicates whether the source program to be compiled is to be prescanned. The source program would be prescanned for meaningful blanks, embedded blanks, and reserved words and put into a form acceptable to the compiler. NOADJUST specifies that the source program is not to be scanned. ADJUST specifies that the source program is to be scanned.

SIZE=size
: specifies (For FORTRAN E) the default option at compilation time

for the maximum number of bytes of main storage available to the FORTRAN E compiler. The value specified must be an integer of from 15360 to 9999999, or it may be of the form nnnnK where nnnn is an integer of from 15 to 9999 and K represents 1024 bytes. If 9999999 or 9999K is specified, all available main storage is to be used by the FORTRAN compiler. If this parameter is omitted, a value of 15360 is assumed. For further information on this parameter, refer to the publication <u>IBM System/360 Operating System: FORTRAN (E) Programmer's Guide</u>, Form C28-6603.

SIZE=size (For FORTRAN H)
specifies the amount of main storage available to the FORTRAN H compiler. The value specified must be an integer of from 204800 to 9999999. If this parameter is omitted, a value of 204800 is assumed.

<u>Note:</u> The SIZE parameter for FORTRAN H does not specify the setting of a default option at compilation time.

OBJLIST
specifies the default option at compilation time for the production of pseudo-assembly listing of the direct program. LIST specifies that the listing is to be produced; NOLIST that the listing is not to be produced.

LINECNT=lines
specifies the default option at compilation time for the number of lines to be printed on each page of the FORTRAN G or H output listing. The value specified must be a two-digit integer of from 01 to 99. If this parameter is omitted, a value of 50 is assumed.

OPT
specifies the default option at compilation time to optimize the execution time of the object modules produced by the FORTRAN H compiler. 0 specifies that the object module is not to be optimized. 1 specifies that it is to receive full register assignment and basic program optimization; 2 that it is to receive full register assignment and complete program optimization.

<u>Example:</u> The following example illustrates the use of the FORTRAN macro-instruction to specify an E-design-level FORTRAN compiler that operates in 20480 bytes of main storage. Any storage in excess of 15360 bytes but less than 20480 bytes is used for the dictionary, overflow table, and I/O buffers. The BCD character set is to be the default character set option at compilation time. Unless otherwise specified at compilation time, a FORTRAN source program listing is to be produced, and compiled source programs are to be processed by the linkage editor. Also, punched decks and source program maps of variables, constants, etc., are not to be produced. The source program is not to be prescanned. A default option of 132 is assumed as the maximum line length.

```
 FORTRAN DESIGN=E,SIZE=20480,STORAGE=SPACE,SORCODE=BCD
```

FORTLIB

The FORTLIB macro-instruction is used to specify the inclusion of the FORTRAN subroutine library (SYS1.FORTLIB) in the new operating system. SYS1.FORTLIB must be allocated space and, if desired, cataloged in the new system during the preparation for system generation. This library must also exist as a cataloged partitioned data set (SYS1.FORTLIB) in the operating system that is generating the new system. This macro-instruction is optional.

All library members of SYS1.FORTLIB that have a non-IBM name, are copied intact into the SYS1.FORTLIB of the new system. All IBM supplied members are copied from SYS1.MODLIB.

Only one design level of SYS1.FORTLIB may be generated during a system generation process. The E-design-level library contains subroutines for programs compiled on the FORTRAN E compiler. The G- and H-degisn-level libraries support programs compiled on any design level of the FORTRAN compiler. (The subroutines in SYS1.FORTLIB may also be used by any operating system program.)

Name	Operation	Operand
	FORTLIB	DESIGN=$\left\{\begin{matrix}E\\G\\H\end{matrix}\right\}$ [UNIT=name] [VOLNO=serial] [UNTABLE=number] [OBJERR=unit] For G and H Libraries only: [ONLNRD=unit] [ONLNPCH=unit]

Operand Field:

DESIGN
specifies the design level of the FORTRAN subroutine library as E, G, or H.

UNIT=name
specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the FORTRAN subroutine library being generated.

VOLNO=serial
specifies the serial number of the volume that is to contain the FORTRAN subroutine library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.FORTLIB during the preparation for system generation.

Note: If the UNIT and VOLNO parameters are omitted, the FORTRAN subroutine library is placed on the new system residence volume. (If one of these parameters is specified, both must be specified.)

UNTABLE=number
specifies the number of FORTRAN logical I/O units to be used at object time. This number does not necessarily correspond to the

number of I/O devices in the installation's System/360 Computing System. The value specified must be a two-digit integer of from 08 to 99. If this parameter is omitted, a value of 08 is assumed.

OBJERR=unit
specifies which FORTRAN logical I/O unit is to be used for object time error messages and FORTRAN dumps. The value specified must be a two-digit integer that does not exceed the value given to UNTABLE, and cannot be the same as the value given to ONLNRD or to ONLNPCH. If the OBJERR parameter is omitted, a value of 06 is assumed.

Note: The cataloged procedures for FORTRAN E define logical unit 01 as SYSIN, 02 as SYSCP, and 03 as SYSOUT. If the value of the OBJERR parameter is not 03, either a DD statement must be added to the cataloged procedure, or the cataloged procedure must be modified. It is recommended that when using FORTRAN E a value of 03 be given to the OBJERR parameter in order to avoid the allocation of an additional output device for SYSOUT.

ONLNRD=unit
specifies which FORTRAN logical I/O unit is to be used when the READ (on-line) statement is encountered in a FORTRAN G or H source program. The value specified must be a two-digit integer that does not exceed the value given to UNTABLE, and cannot be the same as the value given to OBJERR or to ONLNPCH. If the ONLNRD parameter is omitted, a value of 05 is assumed.

ONLNPCH=unit
specifies which FORTRAN logical I/O unit is to be used when the PUNCH (on-line) statement is encountered in a FORTRAN G or H source program. The value specified must be a two-digit integer that does not exceed the value given to UNTABLE, and cannot be the same as the value given to OBJERR or to ONLNRD. If the ONLNPCH parameter is omitted, a value of 07 is assumed.

Note: FORTRAN G and H cataloged procedures assume logical I/O units 06, 05, and 07 as the OBJERR, ONLNRD, and ONLNPCH units, respectively. If a different value is given to any of those parameters, the new unit must be specified on the appropriate DD statement at object time. The cataloged procedures must be temporarily modified by the override technique, or they must be permanently modified to reflect the new unit.

Example: The following example illustrates the use of the FORTLIB macro-instruction to specify the inclusion of the FORTRAN E subroutine library in the operating system to be generated. The unit name is 2301. The volume serial number is 333555. Thirty-two logical units are to be used by the object time load modules. The third unit is to be used for error messages and FORTRAN dumps.

```
   FORTLIB UNIT=2301,VOLNO=333555,UNTABLE=32,OBJERR=03,DESIGN=E
```

PL1

The PL1 macro-instruction is used to specify the inclusion of the PL/I compiler. This macro-instruction is optional. If it is used, UNIV must be specified as the value of the INSTSET keyword in the CENPROCS macro-instruction. (The universal instruction set is required for PL/I compilations and executions.) If this macro-instruction is included, the PL1LIB macro-instruction must also be included. If the PL1 macro-instruction is used during a Processor/Library generation, the same SUPRVSOR macro-instruction specified for the operating system being modified must also be included.

Name	Operation	Operand
	PL1	DESIGN=F $\left[\text{PUNCH=}\begin{Bmatrix}\underline{\text{NODECK}}\\ \text{DECK}\end{Bmatrix}\right]$ $\left[\text{TYPERUN=}\begin{Bmatrix}\underline{\text{LOAD}}\\ \text{NOLOAD}\end{Bmatrix}\right]$ $\left[\text{SORCODE=}\begin{Bmatrix}\underline{\text{EBCDIC}}\\ \text{BCD}\end{Bmatrix}\right]$ [SIZE=size] $\left[\text{OBJLIST=}\begin{Bmatrix}\underline{\text{NOLIST}}\\ \text{LIST}\end{Bmatrix}\right]$ $\left[\text{MSGLEV=}\begin{Bmatrix}\underline{\text{FLAGW}}\\ \text{FLAGE}\\ \text{FLAGS}\end{Bmatrix}\right]$ $\left[\text{OPT=}\begin{Bmatrix}\underline{0}\\ 1\end{Bmatrix}\right]$ $\left[\text{SORLIST=}\begin{Bmatrix}\underline{\text{SOURCE}}\\ \text{NOSOURCE}\end{Bmatrix}\right]$ $\left[\text{CHARSET=}\begin{Bmatrix}\underline{\text{CHAR60}}\\ \text{CHAR48}\end{Bmatrix}\right]$ $\left[\text{EXTLIST=}\begin{Bmatrix}\underline{\text{NOEXTREF}}\\ \text{EXTREF}\end{Bmatrix}\right]$ $\left[\text{ATRLIST=}\begin{Bmatrix}\underline{\text{NOATR}}\\ \text{ATR}\end{Bmatrix}\right]$ $\left[\text{REFLIST=}\begin{Bmatrix}\underline{\text{NOXREF}}\\ \text{XREF}\end{Bmatrix}\right]$ [SORMGIN=(m,n)] [LINECNT=number] $\left[\text{CMPTIME=}\begin{Bmatrix}\underline{\text{NOMACRO}}\\ \text{MACRO}\end{Bmatrix}\right]$ $\left[\text{MACLIST=}\begin{Bmatrix}\underline{\text{SOURCE2}}\\ \text{NOSOURCE2}\end{Bmatrix}\right]$ $\left[\text{COMPILE=}\begin{Bmatrix}\underline{\text{COMP}}\\ \text{NOCOMP}\end{Bmatrix}\right]$ $\left[\text{STMDIAG=}\begin{Bmatrix}\underline{\text{NOSTMT}}\\ \text{STMT}\end{Bmatrix}\right]$ [DELETE=(item[,item]...)]

Operand Field: Many of the parameters of the PL1 macro-instruction specify the setting of default options at compilation time. Default options are the options that are assumed if the corresponding values of the PARM keyword are omitted from an EXEC statement in a PL/I compilation. The DELETE parameter specifies a list of values that cannot be used as values of the PARM keyword.

DESIGN
    specifies the F design level of the PL/I compiler.

PUNCH

specifies the default option at compilation time for the production of a punched deck of the object program. DECK specifies that a punched deck is to be produced. NODECK specifies that a punched deck is not to be produced.

TYPERUN
    specifies the default option at compilation time for the production of input to the linkage editor from the program being compiled. LOAD specifies that the program is to be processed by the linkage editor after compilation. NOLOAD specifies that the source program is only to be compiled.

SORCODE
    specifies the default option at compilation time that indicates the character set used to keypunch the source programs to be compiled. BCD specifies the BCD character set. EBCDIC specifies the EBCDIC character set.

SIZE=size
    specifies the maximum number of bytes of main storage available to the PL/I compiler at compilation time. The value specified must be an integer of from 45056 to 999999. If this parameter is omitted, a value of 45056 is assumed.

OBJLIST
    specifies the default option at compilation time for the production of a listing of the object program. LIST specifies that a listing is to be produced; NOLIST, that a listing is not to be produced.

MSGLEV
    specifies the default option at compilation time for the type of compilation error messages to be printed. FLAGW specifies that warning messages, error messages, and severe error messages are to be printed. FLAGE specifies that only error messages and severe error messages are to be printed. FLAGS specifies that only severe error messages are to be printed.

OPT
    specifies the default option at compilation time to optimize the execution time of the object program produced by the compiler. 0 specifies that it is not to be optimized; 1 that it is to be optimized.

SORLIST
    specifies the default option at compilation time for the production of a printed listing of the PL/I source program. SOURCE specifies that a listing of the source text is to be produced. NOSOURCE specifies that a listing is not to be produced.

CHARSET
    specifies the number of characters in the character set used to write the source program to be compiled. CHAR60 specifies a character set with 60 characters. CHAR48 specifies a character set with 48 characters.

EXTLIST
    specifies the default option at compilation time for the production of a listing of all external data, external entries, and files. EXTREF specifies that a listing is to be produced. NOEXTREF specifies that a listing is not to be produced.

ATRLIST
    specifies the default option at compilation time for the production of a listing for each identifier, the identifier with full qualification, the statement number declaring the identifier, and a list of attributes pertaining to the identifier. ATR specifies that a listing is to be produced. NOATR specifies that a listing is not to be produced.

REFLIST
: specifies the default option at compilation time for the production of a listing for each identifier, the identifier with full qualification, the statement number declaring the identifier and a list of all statements in which a reference is made to the identifier. XREF specifies that a listing is to be produced. NOXREF specifies that a listing is not to be produced.

SORMGIN=(m,n)
: specifies the default option at compilation time for the margins for scanning the source statements. The value m specifies the beginning margin and n specifies the end margin. If the source statement input to the compiler is from the system input stream (i.e., following a DD* statement), the condition $2 \leq m \leq n \leq 100$ must be valid. If the input is not from the system input stream, the condition $1 \leq m \leq n \leq 100$ must be valid. If this operand is omitted, a value of 2 is assumed for m, and a value of 72 is assumed for n.

LINECNT=number
: specifies the default option at compilation time for the maximum number of lines to be printed in each page of a PL/I compiler output listing. The value specified must be an integer of from 10 to 99. If this parameter is omitted, a value of 50 is assumed.

CMPTIME
: specifies the default option at compilation time for the compile-time processor. MACRO specifies that compile-time processing is required; NOMACRO, that compile-time processing is not required.

MACLIST
: specifies the default option at compilation time for the listing of the input to the compile-time processor. SOURCE2 specifies that a listing is to be produced; NOSOURCE2 that the listing is not to be produced.

COMPILE
: specifies the default option at compilation time for compilation to proceed after the compile-time processor has been used. COMP specifies that compilation is required; NOCOMP, that compilation is not required.

STMDIAG
: specifies the default option at compilation time for the contents of diagnostic messages printed during execution of the compiled source program. STMT specifies that the messages are to contain source program statement numbers; NOSTMT specifies that the messages are not to contain source program statement numbers. Offsets from PL/I entry points are included in the messages in both cases.

DELETE=item_n
: specifies that the keyword values or keywords in the value list cannot be used at compilation time in the PARM field of the EXEC statement. The following values can be specified; each has been described above.

DECK	FLAGS	NOXREF
NODECK	OPT	SORMGIN
LOAD	SOURCE	LINECNT
NOLOAD	NONSOURCE	MACRO
EBCDIC	CHAR60	NOMACRO
BCD	CHAR48	SOURCE2
SIZE	EXTREF	NOSOURCE2
LIST	NOEXTREF	COMP
NOLIST	ATR	NOCOMP
FLAGW	NOATR	STMT
FLAGE	XREF	NOSTMT

Example: The following example illustrates the use of the PL1 macro-instruction to specify an F design-level PL/I compiler that operates in 56320 bytes of main storage. The compiled source programs are to be processed by the linkage editor unless otherwise specified at compilation time. It is assumed that all warning and error messages are to be printed, that the EBCDIC character set is used to punch the source programs, that the character set used to write the source programs has 60 characters, and that the execution time of the object programs is not to be optimized. The default options for scanning the source statements are 2 and 72, and for the number of lines in each printed page is 50. The following keywords, and keyword values cannot be used at compilation time in the PARM field of the EXEC statement: CHAR48, BCD, FLAGE, FLAGS, OPT, LINECNT, and SORMGIN. It is assumed that, unless otherwise specified at compilation time, listings of the source text are to be produced; and that a punched deck of the object program, listings of the object program, a listing of external data, entries, and files, listings of identifiers and their attributes, numbers, qualifications, and references are not to be produced. It is also assumed that, unless specified at compilation time, compile-time processing is not required, and that the diagnostic messages are to contain source program statement numbers.

```
 PL1 DESIGN=F,TYPERUN=LOAD,SIZE=56320,
       DELETE=(CHAR48,BCD,FLAGE,FLAGS,OPT,LINECNT,SORMGIN)
```

PL1LIB

The PL1LIB macro-instruction is used to specify the inclusion of the PL/I subroutine library (SYS1.PL1LIB) in the new operating system. SYS1.PL1LIB must be allocated space and, if desired, cataloged in the new system during the preparation for system generation. This library must exist as a cataloged partitioned data set (SYS1.PL1LIB) in the operating system that is generating the new system. This macro-instruction is optional.

Name	Operation	Operand

	PL1LIB	UNIT=name
		VOLNO=serial
		$\left[\text{LIBFCNS=}\left\{\begin{array}{l}\underline{\text{REAL}}\\\text{COMPLEX}\end{array}\right\}\right]$

Operand Field:

UNIT=name
specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the PL/I subroutine library being generated.

VOLNO=serial
specifies the serial number of the volume that is to contain the PL/I subroutine library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.PL1LIB during the preparation for system generation.

Note: If the UNIT and VOLNO parameters are omitted, the PL/I subroutine library is placed on the new system residence volume. If one of these parameters is specified, both must be specified.)

LIBFCNS
specifies the inclusion of complex object time functions in SYS1.PL1LIB. REAL specifies that the complex functions are not to be included. COMPLEX specifies that the complex object time functions are to be included.

Example: The following example illustrates the use of the PL1LIB macro-instruction to specify the inclusion of the PL/I subroutine library in the operating system to be generated. Complex object time functions are not to be included. This library is to be placed on the new system residence volume.

```
   PL1LIB
```

RPG

The RPG macro-instruction is used to specify the inclusion of the report program generator (RPG) language processor. This macro-instruction is optional. IESRPG and RPG are the name and alias, respectively, of the RPG language processor.

Name	Operation	Operand
	RPG	

Operand Field: The operand field must be left blank.

Example: The following example illustrates the use of the RPG macro-instruction to specify the RPG language processor.

```
   RPG
```

GENERATE

The GENERATE macro-instruction is used to specify the data sets, volumes, and I/O devices required for the system generation process, the system generation output options, and the type of generation being done.

The GENERATE macro-instruction must be the last system generation macro-instruction in the user's input deck. The GENERATE macro-instruction must immediately be followed by an assembler END statement. This macro-instruction is required.

Name	Operation	Operand
	GENERATE	UT1SDS=(SYS1.name$\left\{\begin{matrix}\underline{,SL}\\ ,NL\end{matrix}\right\}$)
		UT2SDS=(SYS1.name$\left\{\begin{matrix}\underline{,SL}\\ ,NL\end{matrix}\right\}$)
		UT3SDS=SYS1.name
		OBJPDS=SYS1.name
		RESNAME=name
		RESVOL=serial
		$\left[\text{RESTYPE=}\left\{\begin{matrix}\underline{2311}\\ 2301\end{matrix}\right\}\right]$
		[LNKNAME=name]
		[LNKVOL=serial]
		$\left[\text{LBMAINT=}\left\{\begin{matrix}\underline{F}\\ E\end{matrix}\right\}\right]$
		$\left[\text{ASMPRT=}\left\{\begin{matrix}\underline{OFF}\\ ON\end{matrix}\right\}\right]$
		[LEPRT=(option[,option]...)]
		$\left[\text{DIRDATA=}\left\{\begin{matrix}\text{CATALOG}\\ \text{VTOC}\\ \text{PDS}\end{matrix}\right\}\right]$
		$\left[\text{GENTYPE=}\left\{\begin{matrix}\underline{\text{ALL}}\\ \text{(NUCLEUS,n)}\\ \text{PROCESSOR}\end{matrix}\right\}\right]$

Operand Field:

UT1SDS
UT2SDS
UT3SDS
specify the names of the sequential data sets to be used during system generation by the assembler, linkage editor and utilities. These data sets must exist as cataloged data sets in the operating system which is generating the new system. The data set specified by UT3SDS is used by the linkage editor and must reside on a direct-access volume. If the data sets specified by UT1SDS or UT2SDS reside on magnetic tape, either standard labels (SL) or no labels (NL) must be specified. (See "Input Deck For System Generation" in the section "Specifying the System.")

OBJPDS=SYS1.name
specifies the name of the partitioned data set to be used for the storage of object modules assembled during system generation. This data set must exist as a cataloged partitioned data set in the operating system that is generating the new system. (See "Input Deck Organization" in the section "Specifying the System.")

RESNAME=name
specifies the unit name of a direct-access device in the generating system. The volume on this device is to become the new system residence volume.

RESVOL=serial
specifies the serial number of the new system residence volume. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.NUCLEUS during the preparation for system generation.

RESTYPE
specifies the unit number of the new system residence device as 2311 or 2301.

LNKNAME=name
specifies the unit name of a direct-access device in the generating system. The volume on this device is to contain the link library being generated.

LNKVOL=serial
specifies the serial number of the volume that is to contain the link library. The value specified must be the same as the value given to the VOLUME keyword of the DD statement used to allocate space for SYS1.LINKLIB during the preparation for system generation.

Note: If LNKNAME and LNKVOL are omitted, the link library is placed on the new system residence volume. (If one of these parameters is specified, both must be specified.)

LBMAINT
specifies the size of the load modules in the SYS1.LINKLIB and SYS1.FORTLIB being generated. E specifies 1024-byte load modules. F specifies larger load modules. (If F is specified, the F-design-level linkage editor must be used to maintain the generated system. If E is specified, either the E-design-level or the F-design-level linkage editor can be used.)

Note: If the F-design-level linkage editor is used in the generating system, and the E-design-level linkage editor is to be used for system maintenance in the new system, LBMAINT=E should be specified.

ASMPRT
specifies whether assembly listings are to be produced for the modules assembled during system generation. ON specifies that assembly listings are to be generated; OFF that assembly listings are not to be generated.

LEPRT=option$_n$
specifies linkage editor print options as one or more of the following:

Value	Print Option
LIST	List of control statements in card-image format
MAP	Module map
XREF	Cross-reference table (XREF includes the MAP option)

If this parameter is omitted, only linkage editor error messages, if any, are printed. For a more detailed description of these options see the publication IBM System/360 Operating System: Linkage Editor, Form C28-6538.

DIRDATA

specifies the system directory data to be printed during system generation as one of the following:

Value	System Directory Data
CATALOG	The catalog of the new system is to be printed.
VTOC	The volume table of contents (VTOC) of each volume in the new system is to be printed. The catalog is also to be printed.
PDS	The directories of all partitioned data sets in the new system are to be printed. The VTOCs and the catalog are also to be printed.

If the DIRDATA parameter is omitted, no system directory data is printed.

GENTYPE

specifies the type of system generation. (See Table 2 in the section "Specifying the System.") This may be one of the following:

Value	Generation Type
ALL	An operating system is to be generated. This is the standard type of generation and is assumed if the GENTYPE keyword is omitted.
NUCLEUS	Only a nucleus is to be generated.
PROCESSOR	Only language processors and libraries are to be generated.

When NUCLEUS is specified, n, a decimal integer of from 1 to 9, must also be specified. This identifies the member of the nucleus library (SYS1.NUCLEUS) that is to contain the nucleus to be generated. (The value 1 specifies the member IEANUC01, which is the primary nucleus loaded by the normal IPL procedure.) For further information, refer to the publication <u>IBM System/360 Operating System: Operator's Guide</u>, Form C28-6540.

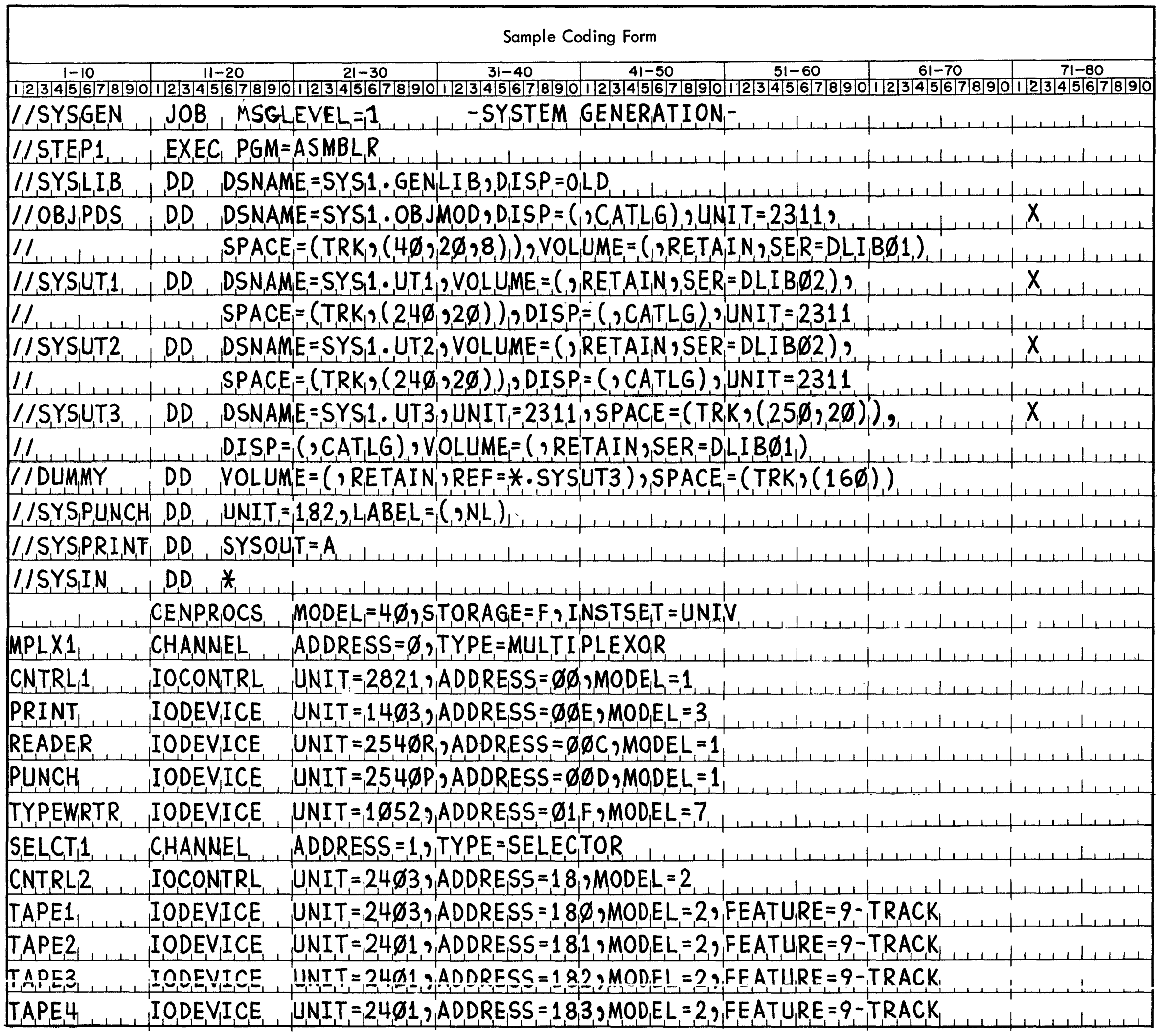
If GENTYPE=ALL, the RESVOL value may not be equal to the serial number of the system residence volume of the generating system; and the LNKVOL value may not be equal to the serial number of the volume that contains the SYS1.LINKLIB of the generating system. If sufficient space is available, and if GENTYPE equals NUCLEUS or PROCESSOR, members may be added to the libraries of the generating system.

<u>Note:</u> If a NUCLEUS generation is specified, the same machine configuration specified for the operating system to be modified must be specified in this generation. Also, a previously resident function cannot be made transient, because no libraries except SYS1.NUCLEUS are affected by this type of generation. However, a previously transient function can be made resident.

Example: The following example illustrates the use of the GENERATE macro-instruction to specify that the sequential data sets named SYS1.UT1, SYS1.UT2, and SYS1.UT3 be used during system generation by the assembler, linkage editor, and utilities. SYS1.UT3 resides on a direct-access volume. SYS1.OBJMOD is the name of the partitioned data set to be used for the storage of load modules assembled during system generation. The unit name of the new system residence device is 190, the device type is 2311, and the serial number of the system residence volume is SYSTEM. Assembly listings, linkage editor printed output, and system directory data are not to be produced. The link library is to be placed on the system residence volume. The size of the load modules in the new SYS1.LINKLIB is to be 1024 bytes. An operating system is to be generated.

```
    GENERATE UT1SDS=SYS1.UT1,UT2SDS=SYS1.UT2,UT3SDS=SYS1.UT3,
             OBJPDS=SYS.OBJMOD,RESNAME=190,RESTYPE=2311,RESVOL=SYSTEM,
             LBMAINT=E,GENTYPE=ALL
```

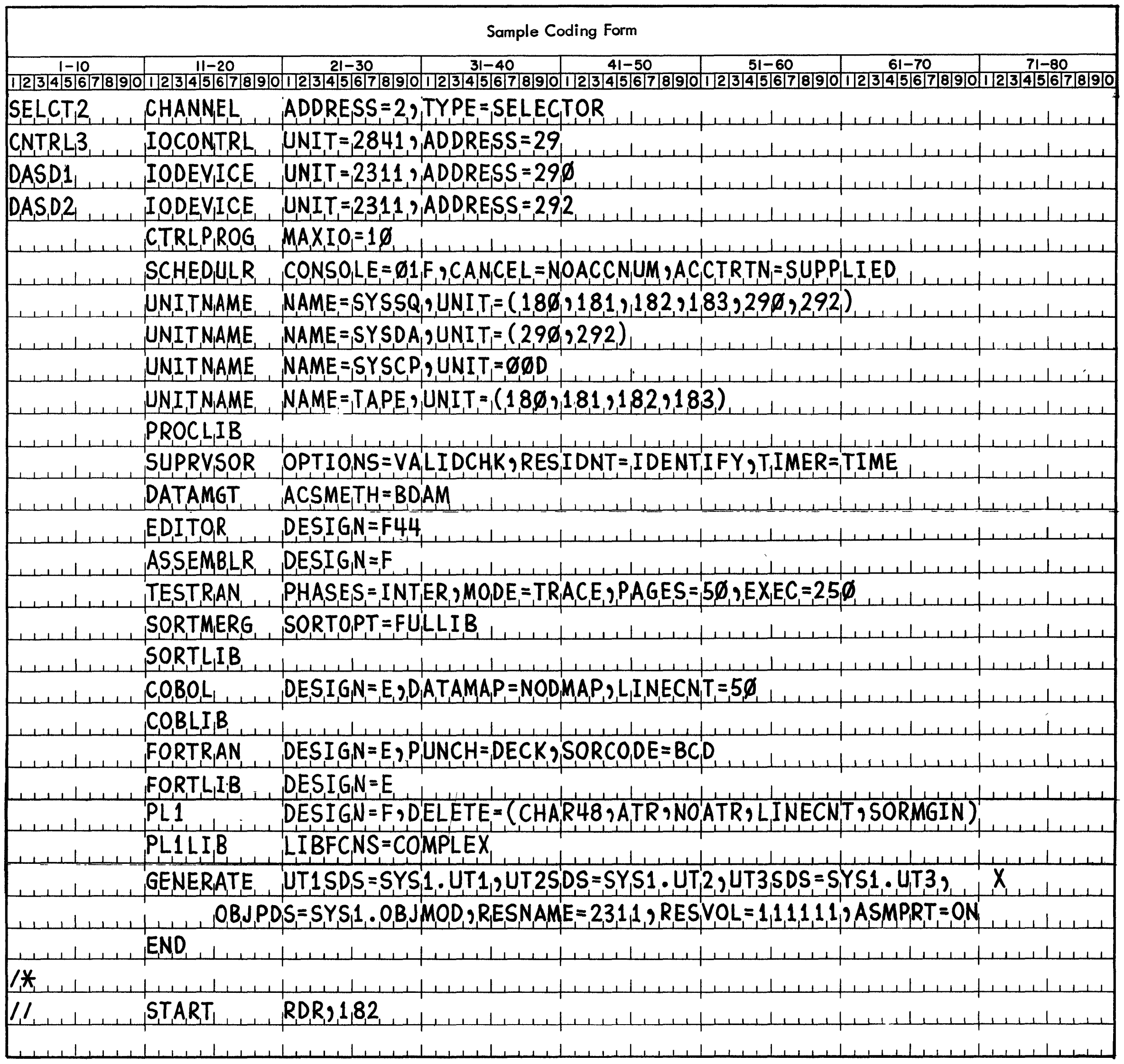
Figure 11 is an example of an input deck for the system generation process. It is assumed, in this example, that the system residence volume and the system data sets were initialized as shown in Figures 2 and 8 in the section entitled "Preparation for System Generation."



Sample Coding Form

```
//SYSGEN   JOB  MSGLEVEL=1      -SYSTEM GENERATION-
//STEP1    EXEC PGM=ASMBLR
//SYSLIB   DD   DSNAME=SYS1.GENLIB,DISP=OLD
//OBJPDS   DD   DSNAME=SYS1.OBJMOD,DISP=(,CATLG),UNIT=2311,             X
//              SPACE=(TRK,(4Ø,2Ø,8)),VOLUME=(,RETAIN,SER=DLIBØ1)
//SYSUT1   DD   DSNAME=SYS1.UT1,VOLUME=(,RETAIN,SER=DLIBØ2),            X
//              SPACE=(TRK,(24Ø,2Ø)),DISP=(,CATLG),UNIT=2311
//SYSUT2   DD   DSNAME=SYS1.UT2,VOLUME=(,RETAIN,SER=DLIBØ2),            X
//              SPACE=(TRK,(24Ø,2Ø)),DISP=(,CATLG),UNIT=2311
//SYSUT3   DD   DSNAME=SYS1.UT3,UNIT=2311,SPACE=(TRK,(25Ø,2Ø)),         X
//              DISP=(,CATLG),VOLUME=(,RETAIN,SER=DLIBØ1)
//DUMMY    DD   VOLUME=(,RETAIN,REF=*.SYSUT3),SPACE=(TRK,(16Ø))
//SYSPUNCH DD   UNIT=182,LABEL=(,NL)
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
          CENPROCS  MODEL=4Ø,STORAGE=F,INSTSET=UNIV
MPLX1     CHANNEL   ADDRESS=Ø,TYPE=MULTIPLEXOR
CNTRL1    IOCONTRL  UNIT=2821,ADDRESS=ØØ,MODEL=1
PRINT     IODEVICE  UNIT=14Ø3,ADDRESS=ØØE,MODEL=3
READER    IODEVICE  UNIT=254ØR,ADDRESS=ØØC,MODEL=1
PUNCH     IODEVICE  UNIT=254ØP,ADDRESS=ØØD,MODEL=1
TYPEWRTR  IODEVICE  UNIT=1Ø52,ADDRESS=Ø1F,MODEL=7
SELCT1    CHANNEL   ADDRESS=1,TYPE=SELECTOR
CNTRL2    IOCONTRL  UNIT=24Ø3,ADDRESS=18,MODEL=2
TAPE1     IODEVICE  UNIT=24Ø3,ADDRESS=18Ø,MODEL=2,FEATURE=9-TRACK
TAPE2     IODEVICE  UNIT=24Ø1,ADDRESS=181,MODEL=2,FEATURE=9-TRACK
TAPE3     IODEVICE  UNIT=24Ø1,ADDRESS=182,MODEL=2,FEATURE=9-TRACK
TAPE4     IODEVICE  UNIT=24Ø1,ADDRESS=183,MODEL=2,FEATURE=9-TRACK
```

Figure 11. Generating an Operating System (Part 1 of 2)



Sample Coding Form

```
SELCT2    CHANNEL   ADDRESS=2,TYPE=SELECTOR
CNTRL3    IOCONTRL  UNIT=2841,ADDRESS=29
DASD1     IODEVICE  UNIT=2311,ADDRESS=29Ø
DASD2     IODEVICE  UNIT=2311,ADDRESS=292
          CTRLPROG  MAXIO=1Ø
          SCHEDULR  CONSOLE=Ø1F,CANCEL=NOACCNUM,ACCTRTN=SUPPLIED
          UNITNAME  NAME=SYSSQ,UNIT=(18Ø,181,182,183,29Ø,292)
          UNITNAME  NAME=SYSDA,UNIT=(29Ø,292)
          UNITNAME  NAME=SYSCP,UNIT=ØØD
          UNITNAME  NAME=TAPE,UNIT=(18Ø,181,182,183)
          PROCLIB
          SUPRVSOR  OPTIONS=VALIDCHK,RESIDNT=IDENTIFY,TIMER=TIME
          DATAMGT   ACSMETH=BDAM
          EDITOR    DESIGN=F44
          ASSEMBLR  DESIGN=F
          TESTRAN   PHASES=INTER,MODE=TRACE,PAGES=5Ø,EXEC=25Ø
          SORTMERG  SORTOPT=FULLIB
          SORTLIB
          COBOL     DESIGN=E,DATAMAP=NODMAP,LINECNT=5Ø
          COBLIB
          FORTRAN   DESIGN=E,PUNCH=DECK,SORCODE=BCD
          FORTLIB   DESIGN=E
          PL1       DESIGN=F,DELETE=(CHAR48,ATR,NOATR,LINECNT,SORMGIN)
          PL1LIB    LIBFCNS=COMPLEX
          GENERATE  UT1SDS=SYS1.UT1,UT2SDS=SYS1.UT2,UT3SDS=SYS1.UT3,    X
               OBJPDS=SYS1.OBJMOD,RESNAME=2311,RESVOL=111111,ASMPRT=ON
          END
/*
//        START     RDR,182
```

Figure 11. Generating an Operating System (Part 2 of 2)

System generation messages are produced by the assembler program during the expansion of system generation macro-instructions. These messages are printed in the assembler listing in the SYSPRINT data set. Two types of messages are produced: error messages and informative messages.

ERROR MESSAGES

Table 6 shows the message code and format of system generation error messages. The messages follow.

IEIaaannn text

Explanation: The error indicated by the message text is a coding error in the system generation macro-instruction, aaa. The message serial number, nnn, identifies the message.

For the CHANNEL, IOCONTRL, and IODEVICE macro-instructions, the message text begins with either the name field of the macro-instruction or, if the name field was omitted, the sequential identification number provided by the system.

Examples of these messages are:

5,* * * IEICEN104 INSTSET VALUE NOT SPECIFIED

5,* * * IEICHA102 CHANNEL2-ADDRESS VALUE NOT SPECIFIED

5,* * * IEICHA102 CHAN#2-ADDRESS VALUE NOT SPECIFIED

The second example illustrates a message for a CHANNEL macro-instruction. "CHANNEL2" is the name field of the macro-instruction. The third example illustrates the same message, but in this case the name field of the macro-instruction was omitted and "CHAN#2" was supplied by the macro-instruction.

System Action: The assembler program did not produce a job stream in the SYSPUNCH data set. The program analyzed all remaining system generation macro-instructions and printed any other required messages. Either message IEIGEN113 or IEIGEN116 was printed, followed by the message: GENERATION TERMINATED. Then the system generation process was abnormally terminated.

Severity Code: 5

User Response: Correct the error or errors indicated and begin the system generation process from the start of Stage I.

IEIGEN113 QUIT SWITCH ON BEFORE GENERATE MACRO

Explanation: One or more errors, indicated by messages, were detected before the GENERATE macro-instruction was expanded.

Severity Code: 7

IEIGEN116 QUIT SWITCH SET IN GENERATE MACRO

Explanation: One or more errors were detected during expansion of the GENERATE macro-instruction.

Severity Code: 7

7, * * * GENERATION TERMINATED * * *

Explanation: The system generation process was abnormally terminated.

Severity Code: 7

INFORMATIVE MESSAGES

*, message text

Explanation: This type of message documents the options selected for the new system through the system generation macro-instructions. All options are described, whether the selection was explicit or implicit.

Table 6. System Generation Error Messages

Message Code	Code
IEI	s,* * * IEIaaannn s = Severity code: 5 Error message; error in coding of a system generation macro-instruction. 7 Error message; message is produced by GENERATE macro-instruction. aaa = Indication of system generation macro-instruction at which error was detected: (see list below) nnn = Message serial number text= Message text

aaa	Macro-instruction	aaa	Macro-instruction
ASM	ASSEMBLR	MAL	MACLIB
CEN	CENPROCS	PLL	PL1LIB
CHA	CHANNEL	PL1	PL1
COB	COBOL	PRL	PROCLIB
COL	COBLIB	RES	RESMODS
CTR	CTRLPROG	RPG	RPG
DAT	DATAMGT	SCH	SCHEDULR
EDI	EDITOR	SOL	SORTLIB
FOL	FORTLIB	SOR	SORTMERG
FTC	FORTRAN	SUP	SUPRVSOR
GEN	GENERATE	SVC	SVCTABLE
IEH	SYSUTILS	SVL	SVCLIB
IOC	IOCONTRL	TES	TESTRAN
IOD	IODEVICE	UNI	UNITNAME
LNK	LINKLIB		

Unit names are automatically assigned during system generation to collections of devices for each type of device specified by the UNIT parameter of an IODEVICE macro-instruction. The names and the devices to which they apply follow.

Magnetic Tape Drives

Unit Name	Device Type
2400	2400 series 9-track Magnetic Tape Drive
2400-1	2400 Magnetic Tape Drive with Seven Track Compatibility and without Data conversion
2400-2	2400 series Magnetic Tape Drive with Seven Track Compatibility and Data conversion

Direct-Access Devices

Unit Name	Device Type
2301	2301 Drum Storage
2311	2311 Disk Storage Drive

Unit Record Equipment

Unit Name	Device Type
1052	1052 Keyboard
1403	1403 Printer or 1404 Printer (continuous form only)
1442	1442 Serial Reader Punch
1443	any 1443 Printer
2501	any 2501 Reader
2520	2520 Reader Punch
2540	2540 Reader Punch (read feed)
2540-2	2540 Reader Punch (punch feed)
2671	2671 Paper Tape Reader

Graphics Devices

Unit Name	Device Type
1053	1053 Display Unit
2250	2250 Display Unit, Model 1
2260	2260 Display Station (local attachment)

Operator intervention may be required during the system generation process. If the output of Stage I is punched cards, and Stage I has been successfully completed, the operator is required to place those cards on an input device and to issue a START RDR command for that device.

Operator intervention is also required if system generation is performed with only two direct-access devices. During Stage II of a two-drive generation, a message is issued to the operator requesting him to remove the volume that contains SYS1.GENLIB and to mount the volume that is to contain the new system. (The MOUNT command must not be issued by the operator during a two-drive generation, because it would prevent this demounting and mounting of volumes.)

At start of Stage II the date in the SET DATE command should be higher than any expiration date for the system data sets being generated. Otherwise the operator must type in "REPLY 00,'U'" every time a data set with a higher expiration date is to be modified.

The completion of Stage II is indicated by a reader closed message. (After Stage II is terminated, system utility programs may be used to scratch the data sets indicated by UT1SDS, UT2SDS, and UT3SDS.)

If the new system is to be used in the future as a generating system, SYS1.GENLIB and SYS1.MODLIB should be saved.

It is recommended that a back-up copy of the generated operating system be made using the DUMP/RESTORE utility program. A detailed description of this program can be found in IBM System/360 Operating System: Utilities.

For maintenance purposes the job stream (SYSPUNCH) and the object modules assembled during Stage II (OBJPDS) should be saved after the completion of system generation.

The console sheets produced during system generation should also be saved. The IFC001I message contains information required to reinitialize the SYS1.LOGREC data set (see Appendix G).

This appendix lists sample console sheets for a two-drive and a three-drive generation. The lines prefixed with an asterisk indicate commands typed by the operator. The asterisks and the comments (preceded by ..) do not appear on the console sheets. These messages are explained in detail in the publication IBM System/360 Operating System: Operator's Guide.

Two-Drive Generation

```
  IEE007A READY
* SET DATE=99.360
  START RDR,00C
  START WTR,00E
* START
  IEF236I ALLOCATION FOR SYSGEN STEP0 ............ initialize new SYSRES
  IEF237I SYSIN  ON 00C
  IEF280I K 191,111111,SYSGEN
  IEF233A M 191,DLIB02,SYSGEN .................. mount for GENLIB+MACLIB
  IEF233A M 182,SCRTCH,SYSGEN ....................... mount for SYSPUNCH
  IEF236I ALLOCATION FOR SYSGEN STEP1 .................... start Stage I
  IEF237I SYSIN  ON  00C
  IEF280I  K 191,DLIB02,SYSGEN
  // START RDR,182
  IEF101I RDR CLOSED ......................................... end Stage I
  IEF223A M 191,DLIB02,SYSGEN ........................... start Stage II
  IEF234A R 191,DLIB02 ........................... remove GENLIB+MACLIB
  IEF233A M 191,111111,SYSGEN ......................... mount new SYSRES
  IFC001I D=2311 N=021 F=00070001 L=00070005 S=0007000200 .. SYS1.LOGREC
  IEF280I K 191,1111111,SYSGEN ......................... save new SYSRES
  IEC202I K 182,1G1001 ................................. save job stream
  IEF101I RDR CLOSED ........................................ end Stage II
  IEE007A READY
```

Three-Drive Generation

```
  IEE007A READY
* SET DATE=99.360
  START RDR,00C
  START WTR,00E
* START
  IEF233A M 192,111111,SYSGEN
  IEF236I ALLOCATION FOR SYSGEN STEP0 ............ initialize new SYSRES
  IEF237I SYSIN  ON 00C
  IEF280I K 192,111111,SYSGEN
  IEF233A M 192,111111,SYSGEN ......................... mount new SYSRES
  IEF233A M 191,DLIB02,SYSGEN ...................... mount GENLIB+MACLIB
  IEF233A M 182,SCRTCH,SYSGEN ....................... mount for SYSPUNCH
  IEF236I ALLOCATION FOR SYSGEN STEP1 .................... start Stage I
  IEF237I SYSIN  ON 00C
  IEF280I K 192,111111,SYSGEN
  IEF280I K 191,DLIB02,SYSGEN
  // START RDR,182
  IEF101I RDR CLOSED ......................................... end Stage I
  IEF233A M 191,DLIB02,SYSGEN ........................... start Stage II
  IEF233A M 192,111111,SYSGEN
  IFC001I D=2311 N=021 F=00070001 I=00070005 S=0007000200 .. SYS1.LOGREC
  IEF280I K 192,111111,SYSGEN .......................... save new SYSRES
  IEF280I K 191,DLIB02,SYSGEN ....................... save GENLIB+MACLIB
  IEC202I K 182,LGL001 ................................. save job stream
  IEF101I RDR CLOSED ........................................ end Stage II
  IEE007A READY
```

APPENDIX F: JOB CONTROL STATEMENT INPUT FOR STAGE II

Four different sets of job control language statements are described in this appendix. These sets are for the assembler, linkage editor, IEHMOVE utility, and IEHLIST utility. The assembler, linkage editor, and IEHMOVE utility may be executed more than once during Stage II. The values selected for the parameters result from the specification in the macro-instructions supplied as input to Stage I. In the following examples the underlined macro-instruction keywords are used to show where the value indicated by those keywords is placed. Comments (preceded by ..) do not appear in the statements. XX is the step number.

Assembler

```
//SGXX EXEC PGM=ASMBLR,COND=(4,LT)
//SYSLIB DD DSNAME=SYS1.GENLIB,DISP=(OLD,PASS)
//       DD DSNAME=SYS1.MACLIB,DISP=OLD,VOLUME=(,RETAIN)
//SYSUT1 DD DISP=OLD,VOLUME=(,RETAIN),LABEL=(,UT1SDS),DSNAME=UT1SDS
//SYSUT2 DD DISP=OLD,VOLUME=(,RETAIN),LABEL=(,UT2SDS),DSNAME=UT2SDS
//SYSUT3 DD DISP=OLD,VOLUME=(,RETAIN),DSNAME=UT3SDS
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD DISP=OLD,VOLUME=(,RETAIN),DSNAME=OBJPDS(member)
//SYSIN DD *
       PRINT ON,NODATA
```

Linkage Editor

```
//SGXX EXEC PGM=IEWL,PARM='NCAL,XREF,LIST,SCTR,LET',COND=(8,LT)
//SYSUT1 DD DISP=OLD,VOLUME=(,RETAIN),DSNAME=UT3SDS
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD  DISP=OLD,UNIT=RESNAME,VOLUME=SER=RESVOL,                X
//            DSNAME=SYS1.name(member)
//MODLIB DD DISP=OLD,DSNAME=SYS1.MODLIB,VOLUME=(,RETAIN)
//SYSPUNCH DD DISP=OLD,VOLUME=(,RETAIN),DCB=(,RECFM=F,BLKSIZE=80),    X
//             DSNAME=OBJPDS
//RESLIB DD DISP=OLD,VOLUME=(,RETAIN),DSNAME=PDS ......... RESMODS macro
//SYSLIN DD *
```

IEHMOVE

```
//SGXX EXEC PGM=IEHMOVE,PARM='POWER=2',COND=(8,LT)
//SYSUT1 DD DISP=OLD,VOLUME=(,RETAIN),DSNAME=UT3SDS
//SYSPRINT DD SYSOUT=A
//FROMLIB DD DISP=OLD,DSNAME=SYS1.MODLIB
//TOLIB1 DD DISP=OLD,VOLUME=(,RETAIN,SER=RESVOL),                     X
//             UNIT=RESNAME
//TOLIB2 DD DISP=OLD,VOLUME=(,RETAIN,SER=LNKVOL),                     X
//             UNIT=LNKNAME
//FRLNK DD DISP=OLD,VOLUME=RETAIN,DSNAME=PDS ............. LINKLIB macro
//FRSVC DD DISP=OLD,VOLUME=(,RETAIN),DSNAME=PDS ........... SVCLIB macro
//SYSIN DD *
```

IEHLIST

```
//SGXX EXEC PGM=IEHLIST
//LINK DD DISP=OLD,VOLUME=(,RETAIN,SER=LNKVOL),UNIT=LNKNAME
//SYSRES DD DISP=OLD,VOLUME=(,RETAIN,SER=RESVOL),UNIT=RESNAME
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
```

The SYS1.LOGREC data set is automatically initialized during the system generation process. If it should become necessary to reinitialize this data set after the operating system has been generated, because the data set has been destroyed (indicated by an IFB004I or IFB001I message to the operator), the disk-drum initialization program IFCDIP00 will reinitialize it. IFCDIP00 is executed using the following job control language statements:

```
//ONLYJOB  JOB
//         EXEC  PGM=IFCDIP00,PARM=nnnxx
//SERERDS  DD    DSNAME=SYS1.LOGREC,UNIT=name,DISP=(OLD,KEEP),    X
//               VOLUME=SER=serial
//
```

In the preceding statements:

nnn
: is the number of uniquely addressable I/O devices in the system.

xx
: is a hexadecimal code number for the system residence device type.

<u>Code</u>	<u>Device Type</u>
01	IBM 2311 Disk Storage Drive
02	IBM 2301 Drum Storage

name
: is the unit name of the system residence device.

serial
: is the serial number of the system residence volume.

<u>Note:</u> Information on these values is provided in the publication <u>IBM System/360 Operating System: Operator's Guide</u>, in the description of the IFC001I message. For an example of this message see Appendix E.

APPENDIX H: UPDATING THE TTRS IN THE OPEN, CLOSE, EOV, AND FEOV MODULES

The TTRs of the OPEN, CLOSE, EOV, and FEOV modules are automatically updated during the system generation process. The IEHIOSUP program must be executed if, at any time after the system is generated, the modules that form the OPEN, CLOSE, EOV, or FEOV routines are changed or replaced, or whenever the SYS1.SVCLIB is moved. This program is executed using the following job control language statements:

```
//ONLYJOB   JOB
//          EXEC  PGM=IEHIOSUP
//SYSUT1    DD    DSNAME=SYS1.SVCLIB,DISP=(OLD,KEEP),              X
//                UNIT=name,VOLUME=SER=serial
//SYSUT2    DD    DSNAME=SYS1.SVCLIB,DISP=(OLD,KEEP),              X
//                UNIT=name,VOLUME=SER=serial
//SYSPRINT  DD    SYSOUT=A
//
```

In the preceding statements:

name
    is the unit name of the device that contains the volume on which the SVC library being updated resides.

serial
    is the serial number of the volume that contains the SVC library being updated.

Note: Both SYSUT1 and SYSUT2 refer to the SVC library being updated.

C28-6554-1

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM / Technical Newsletter

File Number	S360-31
Re: Form No.	C28-6554-1
This Newsletter No.	N28-2206
Date	December 7, 1966
Previous Newsletter Nos.	None

IBM SYSTEM/360 OPERATING SYSTEM
SYSTEM GENERATION

This technical newsletter amends the publication IBM System/360 Operaing System: System Generation, Form C28-6554-1. On replacement pages, each change or addition to the original text is indicated by a vertical bar in the left margin.

Pages to be Inserted	Pages to be Removed
25, 26, 26A, 27, 28	25-28
69-70	69-70
71-72	71-72
73-74	73-74
75, 75A, 76	75-76
77-78	77-78

Summary of Amendment

Changes to the FORTRAN and PL1 macro-instructions

Note: Please file this cover letter at the back of the publication. Cover letters provide a quick reference to changes and a means of checking recept of all amendments.

IBM Technical Newsletter

File Number	S360-31
Re: Form No.	C28-6554-1
This Newsletter No.	N28-2213
Date	January 13, 1967
Previous Newsletter Nos.	N28-2206

IBM SYSTEM/360 OPERATING SYSTEM
SYSTEM GENERATION

This technical newsletter amends the publication IBM System/360 Operaing System: System Generation, Form C28-6554-1. On replacement pages, each change or addition to the original text is indicated by a vertical bar in the left margin. The attached pages replace pages 65-66.

Summary of Amendment

This amendment corrects the format of the COBOL macro-instruction.

Note: Please file this cover letter at the back of the publication. Cover letters provide a quick reference to changes and a means of checking recept of all amendments.

IBM Corporation, Programming Systems Publications, P.O. Box 390, Poughkeepsie, N.Y. 12602

PRINTED IN U.S.A.

READER'S COMMENTS

Title: IBM System/360 Operating System System Generation

Form: C28-6554-1

Is the material:

	Yes	No
Easy to Read?	___	___
Well organized?	___	___
Complete?	___	___
Well illustrated?	___	___
Accurate?	___	___
Suitable for its intended audience?	___	___

How did you use this publication?

___ As an introduction to the subject ___ For additional knowledge

Other ___________________________

fold

Please check the items that describe your position:

___ Customer personnel	___ Operator	___ Sales Representative
___ IBM personnel	___ Programmer	___ Systems Engineer
___ Manager	___ Customer Engineer	___ Trainee
___ Systems Analyst	___ Instructor	Other_____________

Please check specific criticism(s), give page number(s),and explain below:

___ Clarification on page(s)
___ Addition on page(s)
___ Deletion on page(s)
___ Error on page(s)

Explanation:

fold

FOLD ON TWO LINES,STAPLE AND MAIL
No Postage Necessary if Mailed in U.S.A.

staple s

fold

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N. Y. 12602

ATTN: PROGRAMMING SYSTEMS PUBLICATIONS
DEPT. D58

Printed in U.S.A.

---

fold

C28-6554-1

IBM

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

st