File No. S360-36
Form No. C28-6814-0

IBM Systems Reference Library

IBM System/360 Model 44

Programming System

Systems Programmer's Guide

This publication describes how to construct an IBM System/360 Model 44 Programming System, and how to modify and extend its capabilities.

Among the subjects discussed in this publication are:

- How to construct and edit a Model 44 Programming System.
- How to write an accounting routine and incorporate it into the system.
- How to define the input/output configuration at IPL time.
- How to write routines at the Execute Channel Program (EXCP) level of the input/output facilities.
- How to expand the user communication region.

PREFACE

This publication provides information for programmers responsible for constructing an IBM System/360 Model 44 Programming System, and modifying and extending its capabilities. It is directed to experienced programmers who have a detailed knowledge of the components, functions, and structure of the Model 44 Programming and Computing Systems.

Certain information and procedures necessary for the complete understanding of procedures explained in this publication are given in other publications and are not duplicated here. Therefore, the following publications are prerequisite to this one:

IBM System/360 Model 44 Programming System: Concepts and Facilities, Form C28-6810, describes the functions and capabilities of the programming system.

IBM System/360 Model 44 Programming System: Assembler Language, Form C28-6811, contains the information necessary to prepare code in the assembler language. For system editing, it is essential for the user to be familiar with the sections entitled "Assembler Instruction Statements" and "Conditional Assembly Instructions."

IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812, contains detailed information about preparing programs to be executed under system control.

For system editing, in addition to this publication, the system programmer should obtain a symbolic listing of the various system components. The text makes frequent reference to labels, variables, etc., that actually appear in the listing.

First Edition

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

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

A form is provided at the back of this publication for readers' comments. If the form has been removed, comments concerning this publication may be addressed to IBM Corporation, Programming Publications, 1271 Avenue of the Americas., New York, N.Y., 10020.

ILLUSTRATIONS

TABLES

This chapter provides information about how to originally construct an operative IBM System/360 Model 44 Programming System, and how to edit such a system to reflect unique installation requirements. It describes space allocation and IPL procedures, system residence volume initialization, and the use of the system construction program. Input/output device assignment procedures, and information about supervisor reassembly are also included.

The IBM System/360 Model 44 Programming System is a collection of absolute phases and relocatable modules that can be combined to meet the specific programming needs of a given installation. The process used to produce an initial operative programming system is called system construction.

IBM provides the phases and modules from which system libraries (the phase library and, optionally, the module library) are constructed. Distributed with the system are six stand-alone programs, two of which must be used to construct an operative system.

System components to be placed into the phase library are the supervisor, the FORTRAN IV compiler, the assembler, and three system support programs: the job control processor, the linkage editor, and the utilities processor. (Each of these components is distributed in absolute form.)

System components that are placed into the module library are service and mathematical subroutines. (Each of these components is distributed in relocatable form.)

The six stand-alone programs (programs that do not operate under system control) provided with the system are:

1. Two disk initialization programs
2. System construction program
3. Absolute loader
4. Print/punch program
5. Save/restore program

An installation may delete or replace any of the system components provided, except the supervisor and the job control processor.

To obtain an initial operative programming system containing all distributed system elements, an installation must:

- Initialize an IBM 2315 Disk Cartridge as the system residence volume.
- Allocate space for all data sets required by the system.
- Use the system construction stand-alone program to transfer the desired components of the distributed system onto the system residence volume.
- Perform an initial program loading (IPL) procedure, assigning input/output devices (system units) for use by the system.

No reassembly of system components is necessary in this process. However, the system must be reassembled to make such modifications as incorporating a new installation function, permanently changing a machine configuration, or altering default conditions for system options. This process of reassembling--of creating in effect a new system residence volume--is called system editing.

FORM OF DISTRIBUTION

The programming system is distributed by IBM on one reel of tape from which the card decks required for system construction can be punched; if the installation does not have a tape drive, the programming system can be obtained in card form. The tape consists of a series of files: the first contains a print/punch program, the second contains a table of contents, and each remaining file contains a card deck.

The print/punch program is used to print or punch the contents of the tape; it is a special stand-alone program to be used with this tape only. The table of contents is an alphabetical list (according to card identification) of the card decks on the tape together with the file numbers that contain them.

Complete instructions for the use of the print/punch program are given in the publication IBM System/360 Model 44: Operator's Guide, Form C28-6815. The following two examples show typical input parameters that can be used to control the print/punch routine:

1. To print the table of contents only:

   xxxx,cuu,1,1

   where:

   xxxx is the device type (1403 or 1443)
   cuu is the device address

2. To punch all of the card decks contained on the tape:

   xxxx,cuu,EOV,2

   where:

   xxxx is the device type (2520, 2540, or 1442)
   cuu is the device address

Details regarding the use of input parameters, which enable the operator to select a file or group of files to be printed or punched, are included in the instructions mentioned above.

Optional Tape for System Editing

If the system programmer's modifications require that system editing be done, he must obtain a copy of the system in symbolic card form. Tapes from which these symbolic card decks can be punched (or the system assembled) are available through the local IBM branch office.

The components of the system (in symbolic form) occupy two reels of magnetic tape. These tapes contain blocked records consisting of 20 card images per record. The arrangement of the system components is given below.

Reel 1: The order of the components on this reel is:

Save/Restore Program
Print/Punch Program
Absolute Loader
System Construction Program
IPL Program
Supervisor
Job Control
Linkage Editor
Assembler Program
Utility Programs

Reel 2: The order of the components on this reel is:

FORTRAN Compiler
FORTRAN Object Fix
FORTRAN Expander
FORTRAN Library

PREPARATION FOR SYSTEM CONSTRUCTION

Before a system can be constructed, an installation must calculate both the amount of space to be allocated for required system data sets, and the number of entries to be allotted to the directories of the phase and module libraries. After these values are calculated, they must be specified on ALLOC control cards provided as input, along with the IBM-distributed programming system to the stand-alone system construction program. (Each ALLOC card must be followed by a LABEL control card. The LABEL card defines the characteristics of the data set named on the immediately preceding ALLOC card.)

SPACE ALLOCATION CONSIDERATIONS

Space for the following system data sets must be allocated on the system residence volume.

Note: In the text that follows, only the data sets SDSIPL, SDSABS, and SDSUAS must be so named by the user. All other data set names used here, SDSCAT, SDS000, SDSPSD, SDS001, and SDSREL are for convenience only; an installation can choose different names for them.

- SDSIPL (IPL record). The IPL record functions as the initializing routine for the IPL procedures.
- SDSABS (Phase library). The phase library, a directoried data set, contains program phases ready for execution.
- SDSUAS (Job control table). The job control table is used by job control to store system-unit assignment information.
- SDSCAT (System catalog). The catalog data set contains the names and volume identifications of cataloged data sets. It is used by the system to locate data sets specified by name alone. (Note that space need not be allocated to this data set unless the catalog function is to be used by the installation.)

Space for the following system data sets may be allocated on the system residence volume. If, however, these data sets are not to be stored on the system residence volume, space on a different volume must be allocated for them after an initial system has been constructed.

- SDSREL (Module library). The module library, a directoried data set, contains selected modules and serves as an automatic source of input to the linkage editor.

- SDS000 (Compiler data set). The compiler data set is used to collect output from the assembler and the FORTRAN compiler; it is the input to the linkage editor.

- SDSPSD (Pseudo-directory). The pseudo-directory is used by the linkage editor as a directory to data set SDS000.

- SDS001 (System work data set). The system work data set is used as a general system work area, and may be used by any processing program.

SPACE ALLOCATION PLANNING

This section provides guidelines for allocating space to all data sets required by the system.

Space for two data sets only, SDSIPL and SDSABS, must be allocated before an installation's first initial program loading (IPL) procedure is performed. Space for the other required data sets may be allocated during the IPL procedure itself. (See the section "Initial Program Loading (IPL) Procedure.")

SDSIPL (IPL Record)

The IPL record is allocated one 2880-byte block occupying one track. The formats of the ALLOC and LABEL control cards required to reserve this space are:

```
// ALLOC SDSIPL,devadr='volidx',1
// LABEL 2880
```

devadr
: This field specifies the device address of the system residence volume.

  The address is specified in hexadecimal form as cuu, where c is the channel address and uu is the address of a device attached to that channel.
  c = 0 for the standard multiplex channel.
  = 1 or 2 for one of the optional high speed multiplex channels.
  uu = 00 to FE (0 to 254 in hexadecimal)
  volidx = the volume identification of the disk mounted on the device specified by devadr.

Note: SDSIPL must be the first data set for which space is allocated. The order in which space is allocated for other data sets is at the user's discretion and determines the order of data sets on the system residence volume. To reduce access time, space should be allocated to data sets in frequent use before those less frequently used.

SDSABS (Phase Library)

Before coding an ALLOC control card for the phase library, the user must calculate both the size of the library, expressed as the total number of 720-byte blocks to be allocated for the data set, and the number of entries the library's directory is to contain (one per phase). For assistance in making these calculations, refer to Table 1. This table shows the size of each of the system components in the phase library as initially distributed by IBM, and the number of directory entries allocated for them.

In making a final calculation of the space to be allocated for the phase library, however, the user must consider not only system components, but also any programs that he plans to permanently incorporate into the library. In addition, space must be allocated to allow for temporary entries made by the linkage editor during job execution.

If the user is uncertain about how much space to allocate, it should be sufficient to allow a total of 120 directory entries (specify 119, one entry is added by the system) and 500 blocks, resulting in an allocation of 101 tracks to the phase library. This is 75 entries and 215 blocks more than required by the distributed components.

The space necessary to accommodate any given phase is calculated as follows:

- Number of blocks. Divide the size of the phase (in bytes) by 720; if there is a remainder, add 1.

- Number of directory entries. One per phase.

Table 1. Phase Library System Component Sizes

System Component	No. of Blocks (1 block = 720 bytes)	No. of Directory Entries (1 entry = 24 bytes)
Supervisor (resident)	29	1
Transient routines[1]		
OPEN	4	4
CLOSE	2	2
CANCEL	2	2
DUMP	2	2
2311ERP (error recovery procedure)	2	2
2400ERP (error recovery procedure)	2	2
Card read-punch ERP (error recovery procedure)	1	1
Printer ERP (error recovery procedure)	1	1
Error message writer	1	1
Job Control Processor	51	6
Linkage Editor	21	3
Utilities	55	8
FORTRAN IV Compiler	70	6
Assembler	40	2
System Level Directory	2	1

[1]A transient routine is one that is brought into the supervisor transient area of main storage as required.

The formats of the ALLOC and LABEL control cards required to reserve space for the phase library are:

```
// ALLOC SDSABS,devadr='volidx',datlen,
   dirlen ,FMT
// LABEL 720
```

devadr='volidx'
This field must be the same as the corresponding field specified in the ALLOC control card as described under "SDSIPL (IPL Record)."

datlen
This field specifies the total number of blocks to be allocated for the data set, exclusive of its directory length.

dirlen
This field specifies the total number of entries to be allotted to the data set's directory. Each entry is 24 bytes long; the system will automatically allocate the total number of 720-byte blocks needed to accommodate the directory at the beginning of the data set.

FMT
The FMT keyword causes the system to write sequential blocks containing zeros throughout the area reserved for a direct access data set.

The blocks are the size specified in the LABEL statement for the data set. If the data set is directoried, both the directory and the data area are formatted.

This facility enables a program to write or read any data block within the data set at any time, thereby making non-sequential processing possible.

Assembler language programmers may use the POINT supervisor call to go directly to the proper position for writing or reading any block.

SDSUAS (Job Control Table)

The SDSUAS data set is used by the job control processor to record system and symbolic unit assignments specified by the user. SDSUAS must be allocated at least three 720-byte blocks, but not more than six blocks for a 64K system. (Each unit assignment entry occupies 40 bytes.)

The first block, the fixed area, records system unit assignments specified by the FIX operand of the SET command issued at IPL time. Subsequently, at each IPL, the information contained in this block is read into main storage for use by the system. This IPL-time transfer of data saves setup time; system unit assignments need not be redefined each time the operator performs an IPL procedure.

The operator can, however, change system unit assignments by issuing appropriate ACCESS (or ALLOC) statements during IPL or between jobs. A change to the fixed set of system units might be made if, for example, the printer normally used for system messages were inoperative. Changes made by the operator overlay, and override, portions of the fixed area brought into main storage. Fixed system unit assignments modified by the operator are called permanent assignments.

When the job control processor must relinquish its use of main storage, it uses the second block of the SDSUAS data set, the permanent area, as an auxiliary storage area in which to keep track of these permanent assignments.

Note: Although called "permanent," permanent assignments remain in effect only until the next IPL is performed.)

The third block (see note) of SDSUAS, the temporary area, is also used by the job control processor as an auxiliary storage area. The temporary area keeps track of symbolic unit assignments made by the user, via ACCESS or ALLOC statements, within a job. Temporary assignments remain in effect only for the duration of a job. After each job, the system assumes the use of the permanent unit assignments.

Note: More than one block can be allocated for the temporary area. More than one block should be allocated if the installation plans to design a job that refers to more than about ten sequential data sets or more than six directoried data sets; the actual number depends upon the types of references within a specific job and the amount of main storage available. The following warning message will be printed when the capacity of the temporary area is exceeded:

IA86I - CAUTION JOB TBL FULL

(The message is explained in detail in the publication IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812.)

The formats of the ALLOC and LABEL control cards required to allocate space for the SDSUAS data set are:

```
// ALLOC SDSUAS,devadr='volidx',datlen
// LABEL 720
```

devadr='volidx'
: This field must be the same as the corresponding field specified in the ALLOC control card as described under "SDSIPL (IPL Record)."

datlen
: This field specifies the total number of blocks to be allocated for the data set. The value specified must be three or more.

SDSCAT (System Catalog)

The system catalog may be considered as a form of directoried data set containing no data. Because catalog entries are 24 bytes long, the minimum block length that can be allocated for the catalog is 24 bytes; the maximum length, as for any directoried data set, is 720 bytes. Since all data sets are allocated no less than one full track of space, and since the system always adds one control entry to the number of entries specified by the user, maximum use of a 1-track catalog can be achieved by allocating a block length of 720, and specifying 119 entries. The for-

mats of the ALLOC and LABEL control cards required to reserve space for the catalog are shown below. (Note that the data length field of the ALLOC statement is specified as zero.)

```
// ALLOC SDSCAT,devadr='volidx',0,dirlen
// LABEL block-length
```

devadr='volidx'
: This field must be the same as the corresponding field psecified in the ALLOC control card as described under "SDSIPL (IPL Record)."

dirlen
: This field specifies the total number of catalog entries to be allotted to the data set's directory.

block-length
: This field specifies the block length in bytes. The value specified must be no less than 24, no greater than 720.

SDSREL (Module Library)

Before coding an ALLOC control card for the module library, the user must calculate both the size of the library (expressed as the total number of 360-byte blocks to be allocated for the data set), and the number of entries the library's directory is to contain (one per module name). For assistance in making these calculations, refer to Table 2. This table shows the size of each of the system components in the module library as initially distributed by IBM, and the number of directory entries to be allocated for them.

In making a final calculation of the space to be allocated for the module library, however, the user must consider not only system components, but also any programs that he plans to permanently incorporate into the library. If the user is uncertain about how much space to allocate, it should be sufficient to allow a total of 180 directory entries (specify 179, one entry is added by the system) and 500 blocks, resulting in an allocation of 64 tracks to the module library. This is 85 entries and 292 blocks more than required by the distributed components.

The space necessary to accommodate any given module is calculated as follows:

- Number of blocks. Divide the size of the module (in bytes) by 360; if there is a remainder, add 1. (Note that the modules are blocked at five records per block.)
- Number of directory entries. One for each module name.

The formats of the ALLOC and LABEL control cards required to reserve space, on the system residence volume, for the module library are:

```
// ALLOC SDSREL,devadr='volidx',datlen,
   dirlen
// LABEL 360,RECLEN=72
```

devadr='volidx'
: This field must be the same as the corresponding field psecified in the ALLOC control card as described under "SDSIPL (IPL Record)."

datlen
: This field specifies the total number of blocks to be allocated for the data set.

dirlen
: This field specifies the total number of entries to be allotted to the data set's directory.

RECLEN=72
: This field specifies the record length to be used when creating the module library. This field must be present as shown whenever it is desired to create a module library having a format acceptable to the linkage editor.

Table 2. Module Library System Component Sizes

System Component	No. of Blocks (1 block = 360 bytes)	No. of Directory Entries (1 entry = 24 bytes)
Service and IBCOM routines[1]	89	22
FORTRAN mathematical subroutines	119	73

[1]These routines are used for specialized FORTRAN input/output operations.

SDS000 (Complier Data Set) and SDS001 (System Work Data Set)

For most efficient processing the SDS000 and SDS001 data sets should be stored on magnetic tape. If, however, these data sets must be stored on the system residence volume, the following considerations determine the amount of space to be allocated for them.

- The amount of space remaining after other system data sets and the volume table of contents (VTOC) have been placed on the volume.
- The amount of space to be allocated for user data sets on the volume.
- The relative amounts of space to be allocated for SDS000 and SDS001. (The contents of SDS000 accumulate from assembly to assembly and from compilation to compilation. The assembler uses one block of SDS001 for approximately every three assembler language statements, and the FORTRAN compiler uses one block for approximately every six FORTRAN source statements.)

If the user is uncertain about how much space to allocate, it should be sufficient to allocate SDS000 a total of 200 360-byte blocks, resulting in an allocation of 25 tracks, and to allocate SDS001 a total of 1000 360-byte blocks, resulting in an allocation of 125 tracks. These allocations should provide enough space on SDS001 for the assembly of a 2700-card assembler language program or for the compilation of a 1200-statement FORTRAN program (based on a 64K system) and enough space on SDS000 for a link edit resulting in one or more phases totaling approximately 52,000 bytes. Note that one track accommodates 40 card images. The formats of the ALLCC and LABEL control cards required to reserve space on the system residence volume are:

```
// ALLCC SDS000,devadr='volidx',datlen
// LABEL 360
```

devadr='volidx'
: This field must be the same as the corresponding field specified in the ALLOC control card as described under "SDSIPL (IPL Record)."

datlen
: This field specifies the total number of blocks to be allocated for the data set.

```
// ALLOC SDS001,devadr='volidx',datlen
// LABEL 360
```

devadr='volidx'
: This field must be the same as the corresponding field specified in the ALLOC control card as described under "SDSIPL (IPL Record)."

datlen
: This field specifies the total number of blocks to be allocated for the data set.

SDSPSD (Pseudo-directory)

SDSPSD, which serves as the directory of SDS000, is allocated one track of space. (119 directory entries, each 24 bytes long, plus one system-generated 24-byte control entry, occupy one track.) The ALLOC and LABEL control cards required to reserve this space, on the system residence volume, are shown below. Note that the data length field of the ALLOC card is specified as zero.

```
// ALLOC SDSPSD,devadr='volidx',0,119
// LABEL 360
```

devadr='volidx'
This field must be the same as the corresponding field specified in the ALLOC control card as described under "SDSIPL (IPL Record)."

Summary of Space Allocation

In the preceding discussions of the allocation of space for the various data sets, recommendations were made to assist system programmers in determing a reasonable starting point for space allocation. The recommendations, which are summarized in Table 3, will be suitable for a wide variety of installations. Once an installation determines its own requirements from experience, space can be reallocated to suit those requirements.

Note that in addition to space required for data sets, Table 3 includes space (described later under "Initializing the System Residence Volume") required for the VTOC and space used for the IPL bootstrap routine on track 0.

Table 3. Space Allocation Recommendations

Data Set Name	Directory Entries Allocated	Blocks Allocated	Tracks
SDSIPL	0	1	1
SDSABS	119	500	126
SDSUAS	0	4	1
SDSCAT	119	0	1
SDSREL	179	500	64
SDS000	0	200	25
SDS001	0	1000	125
SDSPSD	119	0	1
Track 0	-	-	1
VTOC	36	0	2
Tracks remaining for the user			53

SYSTEM CONSTRUCTION PROCEDURE

This section describes the procedures to be followed to obtain an initial operative programming system. It explains the procedure for initializing the system residence volume and the use of the system construction program.

INITIALIZING THE SYSTEM RESIDENCE VOLUME

After the amount of space to be allocated for system data sets has been calculated, the user must initialize an IBM 2315 Disk Cartridge as the installation's system residence volume. Initialization is the process of writing sector identification, a volume label, and a volume table of contents (VTOC) on a direct-access volume. These functions are performed by the 2315 stand-alone disk initialization program provided with the distributed system. To use this program, the user must prepare an INITIAL control card. This card is placed behind the disk initialization program, as the last card in the deck. The format of the INITIAL contrcl card required for system residence volume initialization is:

```
INITIAL TYPE=xxxx,DVADR=cuu,
    VOLID='volidx',VTOC=e,EDATE=yyddd,
    SYSLOG=cuu
```

TYPE=xxxx
This field identifies the type of device on which the volume is mounted. The value of this field must be SDSD.

DVADR=cuu
This field specifies the device address (physical location) of the volume to be initialized. The DVADR value, cuu, denotes its channel and unit address.
c = 0 for the standard multiplex channel.
= 1 or 2 for one of the optional high speed multiplex channels.
uu = 00 to FE -- the unit address in hexadecimal.

VOLID='volidx'
This field specifies the 6-byte identification number to be assigned to the volume tc be initialized. The six characters of this field must be enclosed in single quotation marks.

VTOC=e
This field specifies the number of entries, e, that will be in the volume's VTOC. (Entries are placed 20

to a track.) The system uses this information to determine how much space must be reserved for the table of contents. The count includes format 4 and format 5 labels, as well as all format 1 labels. The minimum possible value for the system residence volume is 5. Note that the system reserves entire tracks for the volume table of contents. The system adds 4 (for entries created by the system) to the value of e specified by the system programmer and then reserves enough tracks to contain that number of entries. For example, if e is 50, the system adds 4 giving 54; therefore, three tracks (space for 60 entries) is reserved. The space for six additional VTOC entries that is added to form a complete track is available for the volume table of contents.

EDATE=yyddd
This field specifies the date on which the volume is initialized. The date is in the form yyddd, where yy is the year, and ddd is the day of the year (001-366).

SYSLOG=cuu
This field specifies the device address of the console printer keyboard. The meaning of cuu is the same as described above for DVADR=cuu.

The following is an example of an INITIAL control card that might be prepared to govern initialization of a system residence volume:

```
INITIAL TYPE=SDSD,DVADR=0C0,
   VOLID='SYSRES',VTOC=20,
   EDATE=67033,SYSLOG=009
```

SYSTEM CONSTRUCTION PROGRAM

The system construction program provided with each distributed Model 44 Programming System is a stand-alone program executed without system control. The program constructs an operative system from absolute and relocatable decks containing the executable phases and relocatable modules the installation chooses to include in its system. The following procedure must be followed to execute the system construction program:

1. Initialize the system residence volume. (See the preceding section, "Initializing the System Residence Volume.")

2. Place the system construction program deck into the card reader.

3. Place the ALLOC control card shown below into the card reader. This card defines the system unit that the installation will use for the printing of messages.

```
//SYSLOG ALLOC SDSLOG,devadr=
```

devadr
This field specifies the address of the device to be used for the printing of system messages. The address, in the hexadecimal form cuu, must be immediately followed by an equal sign and a blank. Note that cuu is described under "SDSIPL (IPL Record)."

4. Place the following SET card into the card reader:

```
SET date
```

date
This operand is in the form yyddd, where yy consists of the last two digits of the current year, and ddd represents the day of the year (001-366).

5. Place the ALLOC and LABEL control cards prepared for SDSIPL (the IPL record) into the card reader.

6. Place the SDSIPL deck (provided as part of the distributed programming system) into the card reader, followed by a /* end-of-data control card. This deck is the first module (BDA00000) of the input/output supervisor deck (360-IO-613).

7. Place the ALLOC and LABEL control cards prepared for SDSABS (the phase library) into the card reader.

8. Place the SDSABS deck (provided as part of the distributed programming system) into the card reader, followed by a /* end-of-data control card. This deck, which begins with module BEA00000, consists of the remainder of the input/output supervisor deck (360-IO-613); see step 6 above.

9. Place any additional decks to be stored on the system residence volume into the card reader, as described above, each preceded by a set of related ALLOC and LABEL control cards; each should be followed by a /* end-of-data control card. (The order in which these decks are placed into the card reader determines the order in which the data sets appear on the volume.)

10. Place a /& end-of-job control card into the card reader.

11. Dial the console load-unit switches to the address of the card reader.

12. Press the console Load button.

In summary, the deck sequence for system construction is as follows:

1. 2315 disk initialization program.

2. INITIAL control statement.

3. Stand-alone system construction program.

4. ALLOC control card defining SYSLOG.

5. SET control card.

6. SDSIPL deck, preceded by related ALLOC and LABEL cards.

7. SDSABS deck, preceded by related ALLOC and LABEL cards.

8. Additional decks to be stored on the system residence volume, each preceded by ALLOC and LABEL cards.

9. /& control card.

INITIAL PROGRAM LOADING (IPL) PROCEDURE

Operation of the constructed system is initiated by the initial program loading (IPL) procedure. The IPL procedure concludes with a SET command, which results in the job control processor being fetched to begin processing.

The first IPL performed on the programming system differs somewhat from subsequent IPLs in that the user must specify the FIX option of the SET command. This option signals the system that the user is about to define the data set-symbolic unit relationships known as system units. System unit definitions are made via ALLOC, ACCESS, and LABEL control cards that immediately follow the SET command. Note that all control cards used during the IPL procedure are operator commands; i.e., they do not begin with the // identifiers and column 1 is blank when no system unit is specified.

A maximum of 14 system units may be defined.[1] Figure 1 is an example of an input deck that might be prepared to define these units. For this example, it is assumed that all system disk data set allocation was done during system construction. Note that space for SYSAB1 must always be allocated before the IPL procedure is performed. In this example, ALLOC statements are used to define the printer, punch, card reader, and console typewriter; ACCESS statements are used for the other units.

The distributed system supports a maximum of 17 symbolic units: SYSAB1 through SYS005. During an IPL procedure, only 14 of the units can be allocated or accessed. (Note that "SYSCAT" is a naming convention and is not counted as a system unit.) An attempt to define SYS004, SYS005, or the fifteenth unit will result in an error message from the job control processor.

```
        SET     67090,FIX
SYSAB1  ACCESS  SDSABS,0C0='SYSRES'
SYSREL  ACCESS  SDSREL,SAME=SDSABS
SYSLOG  ALLOC   SDSLOG,1052='LOGOUT
SYSRDR  ACCESS  SDSRDR,2540='INPUT'
SYSIPT  ACCESS  SDSIPT,SAME=SDSRDR
SYSLST  ALLOC   SDSLST,1403='OUTPUT'
        LABEL   ,CTLASA
SYSOPT  ALLOC   SDSOPT,SAME=SDSLST
        LABEL   ,CTLASA
SYSPCH  ALLOC   SDSPCH,2540P='PUNCH'
SYSPSD  ACCESS  SDSPSD,SAME=SDSABS
SYSUAS  ACCESS  SDSUAS,SAME=SDSABS
SYS000  ACCESS  SDS000,SAME=SDSABS
SYS001  ACCESS  SDS001,SAME=SDSABS
SYSCAT[1] ACCESS SDSCAT,SAME=SDSABS
        LISTIO
/&
```

[1]The system unit SYSCAT may be used only during the IPL procedure. The unit does not actually exist in the system; "SYSCAT" is a name which allows the user to identify the catalog to the system.

Figure 1. Sample Input Deck for System Unit Definition

There is no need to define all of the system units. If SYSAB2 is not defined at

---

[1]The job control processor requires that five system units be defined: SYSAB1, SYSRDR, SYSLOG, SYSLST, and SYSUAS. Fourteen system units are required if full use is to be made of the system.

IPL time, the job control processor will automatically create an entry for it using the same definition as for SYSAB1. If SYSAB2 is defined at IPL time, the job control processor will use the definition given.

After the user has defined all required system units, "normal" use can be made of the system, but the first step the user may want to take is to assemble or linkage edit any programs to be added to the phase library. The publication IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812, explains these procedures.

PREPARATION FOR SYSTEM EDITING

System editing is the process of reassembling components of a current programming system to make such modifications as incorporating a new installation function, or altering a default condition for a system option. (Note, however, that reassembly is not always necessary for modification. Some minor changes can be made by means of REPLACE (REP) statements by producing a new system using the system construction program. The REP statement is explained in the publication IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812.)

System editing is most easily performed using two IBM 2315 Disk Cartridges (one of which is the current system residence volume) and a symbolic deck of the system. Changes can be made by means of the job control processor, the assembler language compiler, and the linkage editor. The changes that can be made by means of the job control processor are performed as described under "System Construction" (i.e., the size of new data sets, and the number of directory entries for the phase and module libraries must be specified in ALLOC and LABEL statements).

REASSEMBLY OF THE SUPERVISOR

The update facilities of the assembler are used to specify changes to be made during reassembly of the supervisor. However, before any modifications are made, the system programmer should obtain an assembly listing of the system. The listing can be obtained by assembling the system as provided on the optional tape (see "Optional Tape for System Editing"); the assembler's update feature should be used to produce the listing by specifying the UPDASMB3 option. The assembly listing will provide the user with the locations of control blocks and tables that can be altered to meet installation requirements.

The following sections describe the changes that should be made to tailor a system to an installation's particular machine configuration and supervisor feature options. To summarize most of the information presented in these sections, a description of the changes necessary to add a new device to the system is also provided.

Conditional Assembly Instructions in the Supervisor

The symbolic source deck of the supervisor contains conditional assembly instructions that have operand values which can be changed to specify conditions that apply at a particular installation. Samples of SETA instructions in the supervisor are listed in Table 4. Many of the operands of these instructions specify default conditions. Default conditions are the conditions that are assumed if the corresponding values are omitted from a control statement. The operand values shown are the settings in the distributed version of the system. It is usually to the user's advantage to indicate deletion of features that are not to be used at his installation. By deleting unneeded features, storage space is saved.

A complete list of conditional assembly instructions that may be changed is contained in the component comments module, which can be obtained as explained under "Component Comments Modules."

Assembler Instructions in the Supervisor

The symbolic source deck of the supervisor contains EQU assembler instructions that have operand values which can be changed to specify conditions that apply at a particular installation. Samples of EQU instructions in the supervisor are listed in Table 5; the operand values shown are the settings in the distributed system.

A complete list of assembler instructions that may be changed is contained in the component comments module, which can be obtained as explained under "Component Comments Modules."

Table 4. Samples of Conditional Assembly Instructions in the Supervisor

Name	Operation	Operand	Default Condition	Description of the Operand Value
&ACCNT	SETA	0		If 0, the installation will not provide an accounting routine; if 1, it will provide an accounting routine.
&DUMP	SETA	0	DUMP/NODUMP field of the JOB statement.	0 specifies that NODUMP is assumed; 1 specifies that DUMP is assumed.
&FLPT	SETA	0		If 0, floating-point registers will not be used. If 1, floating-point registers will be used.
&HGHCT	SETA	2		Specifies the highest channel number in the installation.
&LABEL	SETA	0		If 0, the system treats all magnetic tapes as unlabeled. If 1, the system will handle the standard tape labels.
&STAPE[1]	SETA	1		If 1, the system assembles routines for support of magnetic tape. If 0, the routines are not assembled.
&TCON	SETA	1	Convert feature in the volume field of ACCESS and ALLOC statements.	1 specifies that ON is assumed for 7-track 2400-series tape drives; 0 specifies that OFF is assumed.
&TDEN	SETA	2	Density for 7-track 2400-series tape drives. This option appears in the volume field of ACCESS and ALLOC statements.	2 specifies that a density of 800 b.p.i. is assumed. 1 specifies that 556 b.p.i. is assumed. 0 specifies that 200 b.p.i. is assumed.
&TIMER	SETA	1		If 1, the timer feature is to be used; if 0, it will not be used.
&TPAR	SETA	1	Parity specification in the volume field of ACCESS and ALLOC statements.	1 specifies that odd parity is assumed for 7-track 2400-series tape drives; 0 specifies that even parity is assumed.
&TTRN	SETA	0	Translate feature in the volume field of ACCESS and ALLOC statements.	0 specifies that OFF is assumed for 7-track 2400-series tape drives; 1 specifies that ON is assumed.
&T9ND	SETA	0	Density for 9-track dual density IBM 2400 Model 4, 5, and 6 Tape Drives. This option appears in the volume field of ACCESS and ALLOC statements.	0 specifies that a density of 1600 b.p.i. is assumed; 1 specifies that 800 b.p.i. is assumed.
&WCHK	SETA	0	WRCHK/NORCHK specification of LABEL,ACCESS and ALLOC statements.	0 specifies that NOWRCHK is assumed; 1 specifies that WRCHK is assumed.

[1]Changing the instruction's operand value to 0 does not result in the deletion of associated transient error recovery routines. They must be explicitly deleted from the library if they are not wanted.

Table 5. Samples of Assembler Instructions in the Supervisor Source Deck

Name	Operation	Operand	Description of the Operand Value
NCCW	EQU	60	Specifies the maximum number of double words available for channel command words (CCWs).
NCHQ	EQU	20	Specifies the maximum number of channel requests that can be placed into the channel queue.
NDEV	EQU	5	Specifies the number of device routines available for use by the system.
NFCB	EQU	18	Specifies the maximum number of file control blocks (FCBs) that can be incorporated into the system.
NSUT	EQU	21	Specifies the maximum number of system units that can be defined for the system.
NUCB	EQU	13	Specifies the maximum number of unit control blocks (UCBs) that can be incorporated into the system.
NXCA	EQU	4	Specifies the length (in words) of the extended save area of the user communication region. The extended save area is usually used to store accounting routine output. Its maximum length is dependent only upon installation requirements.

SYSTEM CONTROL BLOCKS AND TABLES

The sections that follow describe the content and formats of system control blocks and tables which can be modified during system editing.

Unit Control Block

The Unit Control Block (UCB) provides information about the characteristics of a specific input/output device. There must be one UCB for each uniquely addressable input/output device in the installation's machine configuration. The format of the UCB is shown in Figure 8 in Appendix A.

UCB Table: Unit control blocks are stored, contiguously, in the UCB Table, an area of the resident supervisor. The size of the table can be extended, without defining any UCB fields, by increasing the operand value of the supervisor EQU instruction labeled NUCB. (This instruction is described in the section "Assembler Instructions in the Supervisor.") Extending the table in this way provides an installation with greater flexibility in its use of ADD and SUB commands. Additional information related to this subject can be found in the chapter entitled "ADD and SUB Command Usage."

To add a UCB to the table, which may be necessary when, for example, a new input/output device is permanently attached to the system, the user must insert the following statements.

```
1. UCBnnn   EQU *
2.          DC X'0cuu'
3.          DC X'mm'
4.          DC X'tt'
5.          DC (LUCB-4)X'00'
```

UCBnnn
: is a symbolic label; nnn represents the relative location of the UCB within the table. For example, 018 would be the eighteenth UCB in the table.

cuu
: is the unit address, in hexadecimal form, of the device described by the UCB.

mm
: is device mode (see Figure 11 in Appendix C).

tt
: is the device type (see Figure 11 in Appendix C).

LUCB
: is a symbol representing the length in bytes of the UCB.

To replace a UCB in the table, which may be necessary when, for example, an input/output device is permanently substituted for another device in the system, the user must replace three DC statements, statements 2 through 4 above.

To delete a UCB, which may be necessary when, for example, an input/output device is permanently removed from the machine configuration, the user must delete all five statements.

File Control Block

The File Control Block (FCB) provides information about the characteristics of a specific data set and the volume on which it resides. There must be one FCB for each symbolic unit defined for the system. The format of the FCB differs by device type; the FCBs for each type are shown in Figures 9 and 10 in Appendix B.

FCB Table: File control blocks are stored, contiguously, in the FCB table, an area of the resident supervisor.

To provide space for a new FCB in the table, which may be necessary when, for example, the installation designs a program containing a job step referring concurrently to more symbolic units than there are FCBs provided for in the table[1], the user must insert the following DS statement.

```
FCBnnn  DS  (LFCB/4)F
```

FCBnnn
: is a symbolic label; nnn represents the relative location of the FCB within the table. For example, 022 would be the twenty-second FCB in the table.

LFCB
: is a symbol representing the length in bytes of the FCB.

System Unit Table

The system unit table is a group of contiguous 2-byte blocks in the system communication region. Each block contains information relating a symbolic unit to its associated UCB and FCB. For each entry in the table, there are two address pointers: a 1-byte UCB pointer pointing to a UCB in the UCB table, and a 1-byte FCB pointer, pointing to an FCB in the FCB table.

In the distributed version of the system, 21 symbolic units are provided for in the system unit table. The symbols of these units can be determined from the first 21 entries in Table 7. If additional units are required the user must enter an appropriate FCB pointer into the system unit table for each additional unit. (The UCB pointer is placed into the system unit table by the system.) The following DC statements can be used to enter an FCB pointer:

```
SYSnnn  DC  X'00'
        DC  X'pp'
```

SYSnnn
: is a symbolic label; nnn represents a number which when added to 16 specifies the relative location of the entry within the table. For example, 006 specifies the twenty-second (006+16) entry in the table, i.e., the entry for the first additional unit.

pp
: is the relative FCB number (in hexadecimal).

Channel Queue

The channel queue is used, during processing, to keep track of all input/output requests not yet executed by the system. Each entry in the queue occupies four bytes. The distributed version of the system provides for 20 channel queue entries. If the user wants to change this number, he must reassemble the supervisor and change the operand value of the EQU instruction labeled NCHQ. (See the section "Assembler Instructions in the Supervisor.") Changing the operand value effectively changes the size of the queue. The maximum size of the queue is dependent only upon installation requirements and storage availability; the minimum size is one entry (four bytes).

Channel Command Word Area

The channel command word (CCW) area is used for the storage of channel command words during IPL processing. Each channel command word occupies eight bytes. In the

---

[1] See Table 5 for the number of FCBs provided for in the distributed version of the system.

distributed version of the system, 60 entries are provided for the area. If the user wants to change this number, he must change the operand value of the EQU instruction labeled NCCW. (See the section "Assembler Instructions in the Supervisor.") Changing the operand value effectively changes the size of the storage area.

As a general rule, the area should be large enough only to accommodate the CCWs for all devices specified in unit control blocks either at assembly or IPL time. The maximum size of the area, however, is dependent upon installation requirements and storage availability.

Job Control Device Table

The job control device table, shown in Figure 12 in Appendix C, is used by the job control processor to convert to an internal code a device type (e.g., 1052) specified by the user in a control statement. The code is used to find the UCB for the device type specified.

Channel Command Word Table

The channel command word (CCW) table, shown in Figure 13 in Appendix C, contains the channel command words included in the distributed version of the system. The number of channel command words required for an input/output operation varies according to the device; the number and contents of these words are given in the publication that describe each device.

Initialization Device Table

The initialization device table is used by the system to obtain device-related information during the initial program loading (IPL) procedure. The format of this table as it appears in the distributed version of the system is shown in Figure 11 in Appendix C.

Adding a New Input/Output Device: The following text indicates the changes that would be made during system editing to add a new input/output device to the system. Additional information related to this subject can be found in the chapter "Execute Channel Program."

To add a new device to the system, the user must either provide new entries or make changes to existing entries in the initialization device table, the job control device table, and the CCW table. If an entry is added to the job control table, the user must also change the count of the number of bytes alloted to the table. The count field is maintained in the convert type routine, which is labeled CTYP.

Depending upon installation requirements, it may also be necessary when adding a new device, to modify the supervisor assembler instructions labeled NCCW, NFCB, NSUT, and NUCB. A description of these instructions can be found in the section "Assembler Instructions in the Supervisor."

REASSEMBLY OF THE ASSEMBLER PROGRAM

The symbolic source deck of the assembler program contains assembler and conditional assembly instructions that have operand values which can be changed to specify conditions that apply at a particular installation. The DC instruction that specifies the default conditions for the EXEC statement will be used to illustrate an assembler instruction in the assembler program.

To change the default conditions for the EXEC statement, the system programmer must change the operand of the DC instruction labeled SWSPH1, part of the assembler communication region. The operand of the DC instruction, which defines the contents of two bytes, is specified as X'8070' in the distributed system and, therefore, bits 0, 9, 10, and 11 are set to 1. The default condition associated with each bit defined by SWSPH1 is as follows:

Bit Number	Default Condition	Setting in Distributed System
0	Assembler	1
1	UPDASMB1	0
2	UPDASMB2	0
3	UPDASMB3	0
4	UPDATE1	0
5	UPDATE2	0
6	not used	0
7	SYMBMAX	0 (see note)
8	LINK	0
9	DECK	1
10	LIST	1
11	XREF	1
12-16	not used	0

Note: When this bit contains a 0, it is equivalent to specifying the SYMBMIN option. If the user specifies the SYMBnnn option, this bit is ignored.

A conditional assembly instruction is included in the assembler program to provide a means of eliminating large blocks of comments from the listing of the assembler program, thereby reducing the time required to print the listing. The conditional assembly instruction in the distributed version is:

```
&REMARK   SETA   1
```

The 1 in the operand indicates that all comments are to be listed. By using a SETA instruction with a 0 in the operand, the large blocks of comments can be omitted from the listing.

A complete list of assembler and conditional assembly instructions that may be changed is contained in the component comments module, which can be obtained as explained under "Component Comments Modules."

REASSEMBLY OF FORTRAN COMPONENTS

The symbolic source deck of FORTRAN contains assembler and conditional assembly instructions that have operand values which can be changed to specify conditions that apply at a particular installation. Examples of assembler and conditional assembly instructions that can be changed are given below.

Assembler Instructions in the FORTRAN Compiler

To change the default conditions for the EXEC statement, the system programmer must change the operand of the DC instruction labeled FORTOPT. The operand of the DC instruction, which defines the contents of one byte, is specified as X'48' in the distributed system and, therefore, bits 1 and 4 are set to 1. The default condition associated with each bit defined by FORTOPT is as follows:

Bit Number	Default Condition	Setting in Distributed System
0	MAP	0
1	LINK	1
2	DECK	0
3	not used	0
4	SOURCE	1
5	BCD	0
6-7	not used	0

A complete list of assembler instructions that may be changed is contained in the component comments module, which can be obtained as explained under "Component Comments Modules."

Conditional Assembly Instructions in the FORTRAN Library

The following conditional assembly instruction, which is included in the BOAUOPT routine in the distributed system, specifies that the boundary adjustment routine (BNAADJST) is to be used by the system:

```
&FIX  SETA  1
```

If the BNAADJST routine is not to be used, the system programmer must change the operand of this instruction to a 0.

The following conditional assembly instruction, which is included in the BOAUOPT routine in the distributed system, specifies that error messages indicating that the boundary adjustment routine has corrected an error are not to be printed:

```
&PRNTMES  SETA  0
```

If error messages are desired, the system programmer must change the operand to a number from 1 through 254; the number indicates the maximum number of times the error message is to be printed. Note that, although the printing of the message stops when the specified number is reached, the boundary adjustment routine continues to operate normally.

The number of units available to the FORTRAN compiler is specified by the following conditional assembly instruction, which is included in the UNITAB routine in the distributed system:

```
&UNITS  SETA  8
```

The system programmer can increase (8 is the minimum) the number in the operand to 15 without additional changes to the system. However, if more than 15 units are to be used, the system programmer must expand the system unit table (see "System Unit Table") to include the additional units and must associate the new data set reference numbers with the new entries in the system unit table. This association is done by inserting a group of five DC instructions for each new unit into the UNITAB routine ahead of the EQU statement labeled

ENDTABLE. The groups of DC instructions that must be inserted for each data set reference number over 15 is as follows:

Instructions	Notes
DC X'aa'	See "Status 1"
DC X'bb'	See "Status 2"
DC AL1(NULL)	
DC X'cc'	See "Index Number"
DC A(1)	

Status 1: The value of aa is determined by the settings of the following bits:

Bit Number	Bit Setting	Meaning
0	0	Data set closed
	1	Data set open
1	0	Null condition
	1	Sequential input/output operation performed
2	0	Last operation was READ or WRITE
	1	Last operation was a control operation
3	0	Last input/output operation did not refer to a FORMAT statement
	1	Last input/output operation referred to a FORMAT statement
4	0	Last input/output operation was an input operation
	1	Last input/output operation was an output operation
5	0	Last control operation was not a BSR operation
	1	Last control operation was a BSR operation
6	0	Last control operation was not an REW operation
	1	Last control operation was an REW operation
7	0	Last control operation was not a WEF operation
	1	Last control operation was a WEF operation

Status 2: The value of bb is determined by the settings of the following bits:

Bit Number	Bit Setting	Meaning
0	0	The unit is not a SYSname type unit
	1	The unit is a SYSname type unit
1	0	Input operations are allowed on this unit
	1	No input operations are allowed on this unit
2	0	Output operations are allowed on this unit
	2	No output operations are allowed on this unit
3	0	Control operations are allowed on this unit
	1	No control operations are allowed on this unit
4-7	0	Not used

Index Number: The value of cc specifies the position of a unit in the system unit table to be associated the data set reference number represented by this group of DC instructions. The data set reference number is determined by the position of a group of DC instructions in the sequence of groups that define all of the units.

Reassigning Units: A data set reference number can be associated with a different unit in the system unit table by changing the DC instruction that specifies the index number. However, the system programmer must make sure that the DC instructions defining status 1 and status 2 allow the operations that he wants and/or provide the protection that he wants.

A complete list of conditional assembly instructions that may be changed is contained in the component comments module, which can be obtained as explained under "Component Comments Module."

SYSTEM EDITING PROCEDURE

This section describes the procedures to be followed when modification of a system component becomes necessary.

COMPONENT COMMENTS MODULES

After the system programmer has determined that modifications to a system component are necessary for his purposes, he should obtain the component comments module for that component. A component comments module contains all pertinent facts regarding all assembly parameters or default settings for a component; it also contains the linkage editing procedures for using the relocatable modules produced by the assembler program.

A component comments module is the first assembly module of each component. A listing is obtained by using the assembler update feature to print the contents of the desired module (see Figure 2). The following lists show the identifications of the component comments modules for each component:

Components on Reel 1	Module Identification
Save/Restore	BACA0000
Print/Punch	BADA0000
Absolute Loader	BBAA0000
System Construction	BCAA0000
IPL Program	BDAA0000
Supervisor	BFAA0000
Job Control	BIAA0000
Linkage Editor	BKAA0000
Assembler	BLAA0000
Utility Programs	BMAA0000

Components on Reel 2	Module Identification
FORTRAN Compiler	BNAA0000
FORTRAN Object Fix	BNXA0000
FORTRAN Expander	BNZA0000
FORTRAN Library	BOAA0000

The job shown in Figure 2 illustrates the method of obtaining a listing of the component comments module. In this job, the contents of the module for the linkage editor are obtained.

EDITING USING TWO DISKS

After the appropriate symbolic modifications have been prepared using the information given in the component comments module (see also "Preparation for System Editing"), the system programmer must reassemble and link edit the modified component to replace the existing one. The reassembling and link editing is most safely done using two single disk storage drives, one magnetic tape unit and, for system data set SDS001, either an additional magnetic tape unit or the equivalent storage space on an IBM 1316 Disk Pack. However, another additional magnetic tape unit is required for system data set SDS003 whenever the system programmer desires to modify the FORTRAN compiler or to retain the updated symbolic component.

```
//LIST    JOB
//SYS002  ACCESS  SYSSYM,2400='REEL1'
//        LABEL   1600
//ASM     EXEC    ASSEMBLE(UPDASMB3,NOLINK,NODECK)
        SKPTO   BKAA0000
/*                                                        [1]BKAA9990
/&
```

[1]This symbol must begin in column 73.

Figure 2. Listing a Component Comments Module

The procedure for editing using two disks is as follows:

1. Initialize the second disk using the utility programs of the existing system.

2. Allocate space for the various system data sets as explained under "Preparation for System Construction."

3. Copy the IPL data set from the original system (i.e., the system being modified) to the new system using the COPY function of the utility programs.

If members of the module library are being replaced, the remaining steps are replaced by the procedure given under "Replacement of Module Library Members." Note that if the supervisor is the member being replaced, the system programmer must know the ending location (SVAREA+80) of the new supervisor; if the ending location is greater than that of the original supervisor, all components must be reassembled and replaced. In the distributed version of the system, the ending location (SVAREA+80) of the supervisor is 4200 (hexadecimal).

4. Copy the phase library, excluding those members that are to be replaced, from the original system to the new system using the CPYMEM function of the utility programs.

5. Reassemble and link edit the replacement members for the phase library using the phase library data set of the new system as SYSAB2.

6. Copy the module library from the original system to the new system using the CPYMEM function of the utility programs.

7. Execute the IPL procedure and define the system units; specify the FIX option in the SET card (see "Initial Program Loading (IPL) Procedure").

Replacement of Module Library Members

The following procedure, which replaces steps 4 through 7 above, is used when a member of the module library is to be replaced:

4. Copy the phase library from the original system to a new system using the CPYMEM function of the utility programs.

5. Copy the module library, excluding those members that are to be replaced, from the original system to the new system using the CPYMEM function of the utility programs.

6. Reassemble and place the new members into the module library using the CPYMEM* function of the utility programs.

7. Execute the IPL procedure and define the system units; specify the FIX option in the SET card (see "Initial Program Loading (IPL) Procedure").

Example of Editing Using Two Disks

To illustrate system editing, an example showing the replacement of the linkage editor is shown in Figure 3. For this example, it is assumed that system data set SDS000 is large enough to contain the relocatable output and that system data set SDS001 is on magnetic tape.

For modifications to some sections of the FORTRAN compiler, it is necessary to make one pass using the UPDATE1 option to insert changes into the original FORTRAN symbolic language program. The output of that pass is then read from SYS003 as input[1] to the FORTRAN expander program, which creates input that is acceptable to the assembler program and writes the created input[1] on SYS002. Finally, the output from the expander program is used as input to an assembly pass using the UPDASMB3 option to obtain the modified module in relocatable form. When a section of the FORTRAN compiler is modified in this way, two magnetic tape units are used as SYS002 and SYS003 until the final pass.

---

[1]Blocked 20 cards to a record.

```
//INIT    JOB
//       EXEC    UTILS
 INITIAL TYPE=SDSD,DVADR=0C1,VOLID='SYSRES',VTOC=50
/*
//       ALLOC   SDSIPL,0C1='SYSRES',1
//       LABEL   2880,RECLEN=2880
//       ALLOC   SDSABS,SAME=SDSIPL,500,119,FMT
//       LABEL   720,RECLEN=720
//       ALLOC   SDSREL,SAME=SDSIPL,500,179
//       LABEL   360,RECLEN=72
//       ALLOC   SDSPSD,SAME=SDSIPL,0,119
//       LABEL   360,RECLEN=360
//       ALLOC   SDSUAS,SAME=SDSIPL,4
//       LABEL   720,RECLEN=720
//       ALLOC   SDS000,SAME=SDSIPL,1000
//       LABEL   360,RECLEN=360
/*
/&
//REPLAC  JOB
//SYS003  ACCESS  SDSIPL,SAME=SDSIPL
//SYS002  ACCESS  SDSIPL,SAME=SYSAB1
//       EXEC    UTILS
         COPY    SIZIN=2880,SIZOUT=2880
/*
//SYS002  ACCESS  SDSABS,SAME=SYSAB1
//SYS003  ACCESS  SDSABS,SAME=SDSIPL
//       EXEC    UTILS
         CPYMEM  EXCL=(BKLNKEDT,BKLNKED1,BKLNKED2),SIZIN=(720,720)
/*
//SYS002  ACCESS  SYSSYM,080='REEL1'
//       LABEL   1600,RECLEN=80
//SYS001  ACCESS  SDS001,081=FRESH
//       LABEL   360
//A       EXEC    ASSEMBLE(LINK,UPDASMB3)
         SKPTO   BKA00000
*        CHANGE CARDS
*
*
/*                                                              1BKA99990
//SYSAB2  ACCESS  SDSABS,SAME=SDSIPL
//MODULE  EXEC    LNKEDT(KEEP,NOAUTO)
*        LINKEDIT DECK FROM COMMENTS MODULE OTHER THAN MODULE CARDS
/*
//SYS002  ACCESS  SDSREL,SAME=SYSAB1
//SYS003  ACCESS  SDSREL,SAME=SDSIPL
//       EXEC    SIZIN=(360,360) UTILS
         CPYMEM
/*
/&
//       STOP    IPL & FIX 0C1
```

1This symbol must begin in column 73.

Figure 3. Editing Using Two Disks

EDITING USING ONE DISK

If two single disk storage drives are not available, system editing can be done using only one drive. The procedure is illustrated by the example that follows.

Example of Editing Using One Disk

An example of a method for system editing that can be used if only one disk is available is shown in Figure 4. For this example, it is assumed that system data

sets SDS000, SDS001, and SDS002 are on magnetic tape. If system data set SDS003 is needed, it is assumed that it also is on magnetic tape.

Note that when using this alternate method, if an input/output error occurs between the DELETE and RENAME statements (second and third statements from end of Figure 4), the process will not be completed.

The above method can also be used if there is enough space on the system residence volume to contain both SDS000 and SDS001. If such space is available, only SDS002 need be on magnetic tape. To increase the amount of available space on the system residence volume, the system programmer can delete all data sets (e.g., the module library) that are not required for system editing; later, after system editing has been completed, the deleted data sets can be added to the new system residence volume.

If only one magnetic tape drive is available and it is impossible to allocate space in disk storage for either SDS000 or SDS001, the optional symbolics may be punched by assigning SDS003 to the punch unit and using the UPDASMB1 option of the assembler program. The magnetic tape drive is then available to contain SDS000 or SDS001 (whichever cannot be allocated to disk storage). The procedure is then the same as shown in Figure 4, except that the ASSEMBLE option of the assembler program is used instead of the UPDASMB3 OPTION.

SYSTEM MODULES

Table 9 of Appendix D lists all modules of the distributed version of the system, including FORTRAN IV library subprograms. It is intended as reference material, to assist the user during system editing in identifying the various segments of the system.

```
//INIT    JOB
//        ALLOC    SDSAB2,SAME=SYSAB1,500,119,FMT
//        LABEL    720,RECLEN=720
/*
/&
//REPLAC  JOB
//SYS002  ACCESS   SDSABS,SAME=SYSAB1
//SYS003  ACCESS   SDSAB2,SAME=SYSAB1
//CPY     EXEC     UTILS
        CPYMEM   EXCL=(BKLNKEDT,BKLNKED1,BKLNKED2),SIZIN=(720,720)
/*
//SYS002  ACCESS   SYSSYM,080='REEL1'
//        LABEL    1600,RECLEN=80
//SYS000  ALLOC    SDS000,081=FRESH
//SYS001  ALLOC    SDS001,082=FRESH
//MODULE  EXEC     ASSEMBLE(LINK,UPDASMB3)
        SKPTO    BKA00000
*       CHANGE CARDS
*
*
/*                                                              [1]BKA99990
//SYSAB2  ACCESS   SDSAB2,SAME=SYSAB1
//LNK     EXEC     LNKEDT(KEEP,NOAUTO)
*       LINKEDIT DECK FROM COMMENTS MODULE OTHER THAN MODULE CARDS
/*
//      ACCESS   SDSABS,SAME=SYSAB1
//      DELETE   SDSABS
//      RENAME   SDSAB2,SDSABS
//      PAUSE    RE-IPL RESIDENCE VOLUME
```

[1]This symbol must begin in column 73.

Figure 4. Editing Using One Disk

This chapter provides the information required to write an installation accounting routine and add the routine to the Model 44 Programming System. The routine resides in the phase library and will be loaded automatically by the job control processor at those points when accounting statistics (i.e., the data supplied by JOB and EXEC statements) are available.

Specification of the Accounting Routine: To specify the use of an installation accounting routine, the user must change the operand value of the supervisor SETA instruction labeled &ACCNT from 0 to 1. (See the section "Conditional Assembler Instructions in the Supervisor" in the chapter entitled "System Construction and Editing.")

Entry to the Accounting Routine: The accounting routine receives control when the job control processor completes the reading of a STOP statement or of a JOB or EXEC statement that specifies the optional "accounting information" field. The statement that has caused the entry is identified by a hexadecimal code the system places in register 0. These codes are listed below:

Code	Meaning
00	Identifies a JOB statement.
04	Identifies an EXEC statement.
08	Identifies a STOP statement.

Size of the Routine: The size of an accounting routine must not exceed 4096 bytes.

Addressing within the Accounting Routine: Because the actual main storage address in which the accounting routine will be loaded is not known to the user, the routine must be "address-free." This means that the routine may not use address constants that refer to addresses within the routine. Such addresses must be dynamically generated, e.g., by the Load Address (LA) instruction.

Register Usage: Registers 0, 1, and 15 can be used in an accounting routine without saving their contents. However, if registers 2 through 12 are to be used, their contents must first be saved; register 13 contains the address of an 11-word register-save area. Register 14 contains the address of the location in the job control processor to which control can be returned after execution of the routine. Register 1 contains the address of the entry point of the accounting routine and, therefore, may be used for initial base addressing.

Input to the Accounting Routine: The system provides accounting information in the user communication region, an area within the system supervisor. SVC 18(EXTRACT) is used to obtain the location of the communication region.

EXTRACT causes the system to place the address of byte 0 of the user communication region into register 1 and to return control to the calling program. The address of any particular word or byte within the region is obtained by adding the byte count to this value.

The communication region contains the accounting information listed below. A complete description of the contents of the communication region can be found in the chapter "User Communication Region".

Word	Byte	Description
6,7	24-31	Job name, in EBCDIC form, specified by the user in a JOB statement.
8,9	32-39	Job step name, in EBCDIC form, specified by the user in an EXEC statement.
32-35	128-143	Accounting information. This information, in EBCDIC form, is a copy of the accounting information field specified by the user in either a JOB or EXEC statement.

Additional data for the accounting routine, such as timing information, can be obtained through the use of an SVC. A full description of all applicable SVCs is contained in the publication IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812.

Output from the Accounting Routine: Data generated by installation accounting routines can be stored, for later analysis, in the extended save area of the user communication region, bytes 144 and up. The size of this area is determined by the installation during system editing. Data stored in this area is never altered by the system. Additional information about the save area

can be found in the chapter "User Communication Region."

Exit from the Accounting Routine: The accounting routine can return control to the system in either of two ways: normally or abnormally. The normal return is to restore the contents of registers 2 through 12 and issue a BCR 15,14 instruction. The abnormal return, which might be resorted to if an unacceptable account number were read, is to issue an SVC 15 to cancel the job. CANCEL causes the system to terminate the current job immediately. A message for the operator is written, and a dump is taken if a dump was requested in the job's control statements. The system then loads the job control processor, which reads the system input unit, ignoring all statements until a JOB statement is detected.

Adding the Accounting Routine to the System: The accounting routine must be added to the system as a permanent member of the phase library. To add the routine, provide an EXEC statement to execute the linkage editor, and specify KEEP in the parameter list field; follow the EXEC statement with a PHASE statement specifying BACCOUNT as the phase name. The value specified for the origin parameter of the PHASE statement; must be either * or S. The PHASE statement must be followed by an INCLUDE statement and, if execution of the accounting routine is to begin with other than its first instruction or one indicated by an END card (assembly end), an ENTRY statement. (These statements are described in the publication IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812.)

ADD AND SUB COMMAND USAGE

This chapter provides information directed specifically to the system programmer about the use of the ADD and SUB commands. Both of these commands provide the user with a means of changing an installation's input/output device configuration during the initial program loading (IPL) procedure.

ADD AND SUB COMMANDS

During the IPL procedure the operator, by issuing ADD and SUB commands, can specify changes that have been made to the installation's input/output device configuration. The ADD command informs the system of the availability of a new device[1]; the SUB command removes a previously available device from the system. There is a direct relationship between the use of ADD and SUB and several system control blocks and tables. The system programmer should understand this relationship to avoid the misuse of these commands at his installation.

ADD causes the system to enter an additional unit control block (UCB), for the specified device, into the UCB table. SUB causes the system to delete the UCB previously associated with the device.

Consequently, the maximum number of valid ADD commands that can be issued is dependent upon two variables: the number of SUB commands previously issued, and the size of the UCB table. As a precautionary measure, then, the operator should issue any required SUB commands before issuing ADD commands. It is the installation's responsibility, however, to increase the size of the UCB table, when necessary.

Note: The use of SUB commands may make it necessary for the operator to redefine units that follow the one that was deleted.

The size of the UCB table can be increased by reassembling the supervisor and changing the operand value of the supervisor EQU instruction labeled NUCB. For a description of supervisor EQU instructions, see the section "Conditional Assembler Instructions in the Supervisor" in the chapter "System Construction and Editing."

The formats of the ADD and SUB commands are shown below. Additional information about these commands can be found in the publication IBM System/360 Model 44 Programming System: Operator's Guide, Form C28-6815.

The ADD Command

The ADD command indicates to the system the availability of an additional input/output device.

```
┌──────────────────────────────────────┐
│ ADD type,devadr                      │
└──────────────────────────────────────┘
```

type
: Specifies the type of device to be added. Valid entries are shown in Table 6.

devadr
: Specifies the address of the device to be added, expressed in hexadecimal form as cuu, where cuu is as described under "SDSIPL (IPL Record)."

The SUB Command

The SUB command informs the system that a specified device is no longer available, and should, therefore, be subtracted from the list of available devices.

```
┌──────────────────────────────────────┐
│ SUB devadr                           │
└──────────────────────────────────────┘
```

devadr
: Specifies the address of the device to be subtracted, expressed in hexadecimal form as cuu, where cuu is as described under "SDSIPL (IPL Record)."

---

[1]Additional information relating to the addition of a new device can be found in the chapter entitled "Execute Channel Program", under the heading "Modifying the Device Table."

Table 6. Device Types

Type Field Entry	Explanation
SDSD	Single Disk Storage Drive
1316	IBM 2311 Disk Storage Drive
2400	IBM 2400 Magnetic Tape Unit with a 9-track read/write head--800 bytes/inch only
2400II	IBM 2400 Magnetic Tape Unit with a 9-track read/write head--1600 bytes/inch only
2400D	IBM 2400 Magnetic Tape Unit with a 9-track read/write head--dual density
2400T7	IBM 2400 Magnetic Tape Unit with a 7-track read/write head
2400T7C	IBM 2400 Magnetic Tape Unit with a 7-track read/write head and the convert feature
1052	IBM 1052 Console Printer Keyboard
1442	IBM 1442-N1 Card Read-Punch
1442P	IBM 1442-N2 Card Punch
2520	IBM 2520 Card Read Punch
2520P	IBM 2520-B2, B1 Card Punch
2501	IBM 2501 Card Reader
2540	IBM 2540 Card Read-Punch (reader side)
2540P	IBM 2540 Card Read-Punch (punch side)
1403	IBM 1403 Printer-Model 2, 3 or N1 (132 characters)
1403M7	IBM 1403 Print-Model 7 (120 characters)
1443	IBM 1443 Printer-Model N1 (120 characters)
1443S	IBM 1443 Printers-Model N1 (144 characters) special feature

USER COMMUNICATION REGION

This chapter contains information about the user communication region, its format and use. It is included in this publication to reduce the user's need to refer to other publications. The chapter should be especially helpful to readers who are designing an installation accounting routine.

The user communication region is an area within the system supervisor that may be used by both system programs, such as the assembler, and problem programs. Programs may read information from this area but, to avoid accidental destruction of system data, should use supervisor calls to insert or alter information.

Its contents are as follows:

Word	Byte	Description
0,1	0-4	Date, set by the operator, in the form yyddd, where yy is the year and ddd is the day of the year (001-366).
2	8-11	Address of the first byte of the problem program area.
3	12-15	Address of the last byte available for use by the problem program.
4	16-19	Address of the highest byte in the problem program area filled by a phase loaded by means of any FETCH or LOAD supervisor call.
5	20-23	Address of the last byte in the problem program area filled by the most recent FETCH or LOAD supervisor call.
6,7	24-31	Job name in EBCDIC characters.
8,9	32-39	Job step name in EBCDIC characters.
10	40	User program switch byte. This byte is set to zeros whenever the system reads a JOB control statement.
10	41	Highest assembly error severity. Reset to zero by JOB statement. 00 - Normal. No errors. 04 - Warning messages listed. Execution should be successful. 08 - Error messages listed. Execution may fail. 0C - Severe errors. Execution impossible. 10 - Terminal errors. Job has been cancelled.
10	42,43	Not used.
11	44-47	User interprogram communicatcns area. This area may be used by one job step to preserve information for use by a later job step. This area is set to zeros whenever the system reads a JOB control statement.
12,13	48-55	User intraprogram communicatcns area. This area may be used by one job step phase to preserve information for use by ancther phase within the same job step. This area is set to zeros whenever the system reads an EXEC control statement initiating a new job step.

Word	Byte	Description
14-31	56-127	Up to six 8-byte EBCDIC option parameters from the job step EXEC statement are stored here by the job control processor. If less than six parameters are stored, the area is padded on the right with blanks. The area is reset to blanks before the beginning of the next job step. The full 72 bytes of this area and the 8-byte intraprogram communication area also are used by the system and processor programs between job steps for temporary storage of certain control statements.
32-35	128-143	Up to 16 bytes of accounting information are stored here for use by installation routines. This information, in EBCDIC form, is obtained from JOB and/or EXEC statements.
36-	144-	Data generated by installation accounting routines may be stored here. This field is not included in the distributed version of the system. The field can be included, however, by reassembling the supervisor and changing the operand value of the supervisor EQU instruction labeled NXCA. The operand value of this instruction specifies the size of the field. For additional information about this instruction, refer to the section "Assembler Instructions in the Supervisor" in the chapter "System Construction and Editing."

COMMUNICATION REGION SUPERVISOR CALLS

The communication region supervisor calls are INSERT, EXTRACT, UPSAND, and UPSOR. They are used for communication between a problem program and the system's user communication region.

The applicable codes are:

INSERT - SVC 17

EXTRACT - SVC18

UPSAND - SVC 19

UPSOR - SVC 20

INSERT - SVC 17

The INSERT supervisor call is used to store information in the user communication region.

The system does not require a problem program to provide any information in this area other than that contained in control statements. If a program does use the area, however, it should use the INSERT supervisor call to reduce the chances of accidental destruction of data needed by the system.

It is not necessary to know the location of the region to use INSERT. To refer to information already stored, the location can be determined by use of the EXTRACT supervisor call.

INSERT cannot be used to alter the contents of words 0 through 10, bytes 0 through 43 of the user communication region.

INSERT cannot be used to modify the user communication region permanently. The region is reinitialized when the initial program load procedure is executed.

When the INSERT supervisor call is executed, register 1 must contain the address of a parameter list. This list consists of two words aligned on full-word boundaries. The first word contains the address in the problem program area of the information to be stored in the user communication region. The second word contains the address of a 4-byte area containing control information.

The first byte of control information must be hexadecimal 00. The second byte gives the number of 4-byte words to be stored in the region. The last two bytes indicate where in the region the data is to be stored. This last location is expressed in terms of the word where the system is to start storing the information, the first word in the region being word 0.

For example, to store eight bytes of data in the intraprogram communications

area, words 12 and 13, bytes 48 through 55, the following coding could be used:

```
INSERT   EQU  17
              .
              .
              .
         LA   1,PARAM
         SVC  INSERT
              .
              .
              .
PARAM    DC   A(LOC)
         DC   A(CONTRL)
              .
              .
              .
LOC      (data to be stored)
         DS   0F
CONTRL   DC   X'00'
         DC   AL1(2)
         DC   AL2(12)
```

The system stores the eight bytes of information at LOC in the intraprogram communications area and returns control to the instruction following the SVC. The low order byte of register 15 contains the hexadecimal code 00, indicating no errors.

If an attempt is made to use INSERT to store information outside the communications region or in words 0 through 10, nothing is stored. Register 15, on return, contains the hexadecimal code 04.

Other registers are unchanged.

EXTRACT - SVC 18

The EXTRACT supervisor call is used to obtain the location of the communication region.

EXTRACT causes the system to put the address of byte 0 of the communication region in register 1 and return control to the calling program. The address of any particular word or byte within the region is obtained by adding the byte count to this value.

UPSAND - SVC 19 and UPSOR - SVC 20

The UPSAND and UPSOR supervisor calls are used to set or alter the contents of the user program switch byte in the user communications region. This byte is used for communication between job steps as well as within a job step.

When UPSAND is used, the logical product (AND) of the user program switch byte and the low-order byte of register 1 is stored in the user program switch byte.

When UPSOR is used, the logical sum (OR) of the user program switch byte and the low-order byte of register 1 is stored in the user program switch byte.

When the two bytes are combined by either UPSAND or UPSOR, they are matched bit for bit.

With UPSAND, if each of the corresponding bits is a 1, the result is a 1. If either is 0, the result is 0.

With UPSOR, if either of the corresponding bits is a 1, the result is a 1. If both are 0, the result is 0.

These combinations are listed below:

A	B	UPSAND	UPSOR
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

When an UPSAND or UPSOR supervisor call is executed, register 1 must contain a comparison byte in its low-order positions, bits 24 through 31. The system alters the user program switch byte accordingly and returns control to the instruction following the SVC. Error codes do not apply.

For example, to set the user program switch byte to all 1's, the following coding could be used:

```
UPSOR    EQU  20
         IC   1,MASK
         SVC  UPSOR
              .
              .
              .
MASK     DC   X'FF'
```

As a result of this UPSOR supervisor call, byte 40 of the communications region is set to 11111111. This byte is reset to all 0's when the system reads a JOB control statement initiating another job.

This chapter explains the use of the execute channel program (EXCP) supervisor call, SVC 0, and is accompanied by descriptions of specific control blocks used with SVC 0. Factors that affect the operation of EXCP, such as device variations, program modifications, and the use of the WAIT supervisor call, SVC 1, are also discussed. It is recommended that the user obtain a symbolic listing of the supervisor and the channel scheduler before reading this chapter. The text makes frequent reference to labels, variables, etc., that actually appear in the listing.

EXCP LEVEL PROGRAMMING

At the EXCP level of programming the user can work with devices that are not supported by the system and can manipulate devices in ways not provided by the read/write supervisor call routines. For a list of supported devices, see IBM System/360 Model 44 Programming System: Concepts and Facilities, Form C28-6810. Routines written and tested at the EXCP level may subsequently be incorporated into the system's read/write level through reassembly of the supervisor.

EXCP level operations make use of both system and user routines. System routines schedule the requested operation and execute the privileged instructions required for input/output. They also handle some of the input/output interruption conditions that do not require special attention for a particular device.

Supervisor Call EXCP

The EXCP supervisor call initiates the execution of routines written at the EXCP level. Before an EXCP supervisor call is issued, the address of an input/output block must be placed into register 1.

After the EXCP supervisor call has been processed, control is returned to the instruction following the EXCP statement. At that time, register 15 contains a code that indicates the conditions encountered by the statement; the following return codes may appear in register 15:

Hexadecimal Code	Meaning
00	The EXCP supervisor call was accepted by the system.
04	No unit control block exists for the requested system unit.
08	The device requested is not available for use.

Supervisor Call WAIT

The WAIT supervisor call causes the execution of a problem program to be suspended until the operation initiated by a previous EXCP statement has been completed. Before a WAIT supervisor call is issued, the address of an input/output block must be placed into register 1.

After the WAIT supervisor call has been processed, control is returned to the instruction following the WAIT statement, and the completion of the operation is indicated by a 00 hexadecimal code in byte 0 of word 1 of the input/output block.

The WAIT supervisor call can be used either immediately following an EXCP supervisor call or at a later point in the problem program. For most efficient program execution, the WAIT statement should be placed immediately preceding the portion of the program that uses the results of the EXCP operation associated with the specified input/output block. This placement of the WAIT statement allows processing, not dependent upon that EXCP operation, to be done while the EXCP operation is in progress.

REQUIREMENTS

To execute an operation at the EXCP level, the user should provide:

- Channel command words or device-dependent routines to initialize these words.
- Device-dependent interruption and error recovery routines.
- One or more input/output blocks (IOBs).

Channel Command Words

A channel command word (CCW) specifies a command to be executed and, for commands initiating data transfer, the area to or from which data is to be transferred.

Channel command words executed by means of EXCP must be on double-word boundaries. CCWs may be constructed before the EXCP supervisor call is issued and a pointer to the word list must be present in the input/output block (ICB), or the words may be constructed or updated by including an initialization section in the device routine.

Descriptions of the CCWs required for the various input/output devices can be found in publications that describe device functions. These publications are listed in the IBM System/360: Bibliography, Form A22-6822 (subject codes 03 through 09). Additional information regarding the format of CCWs is contained in the publication IBM System/360: Principles of Operation, Form A22-6821.

Device Routine

A device routine provides control over input/output operations during channel program execution. Device routines can be used to examine the status of input/output operations and determine the actions to be taken for various conditions.

If bytes 5 through 7 of the input/output block, the CCW list pointer, contain zero, the system will first enter the device routine to set up the CCW list. The routine is entered again for interruption analysis and error recovery, after execution of the channel commands.

Device routines are executed in the supervisor state, but must not contain any supervisor calls or privileged instructions. Furthermore, a device routine cannot be interrupted except for machine check and certain types of program check. Device routines also must not alter the contents of registers 1 and 14. These registers are used for communication between the device routine and the system. Detailed information about device routines is contained in the section "Device Routine Requirements."

Input/Output Block

The input/output block (IOB), which consists of the first six words of a request control block (RCB), is used for communication between the user's program and the system. At the EXCP level, the IOB is usually used in place of an RCB. However, although the system requires only the information contained in the IOB, the user can write a program that uses all of the fields contained in an RCB.

When an IOB or RCB is created, it must be aligned on a full word boundary. All fields not supplied with information by the user's program must be defined as hexadecimal zeros.

The format of the IOB is shown in Figure 5. Its fields are explained below. The format of the RCB is shown later in the chapter.

Word	Byte	Description
0	0	System unit index, supplied by programmer. (See the system unit index table, Table 7.)
0	1-3	Device routine address, supplied by programmer.
1	4	Postrequest flag indicating in hexadecimal code whether the block currently is active, supplied by the system. 00 - no operation pending 01 - operation in progress
1	5-7	Address of channel command list, supplied by the programmer.
2	8	Reserved for system use.
2,3	9-15	Last seven bytes of channel status word, supplied by the system at interruption time.
4	16-19	Sense information, supplied by the system when a unit check condition occurs.
5	20-23	Four EBCDIC characters to be used by the system to locate an error recovery program in the phase library. This is an optional field supplied by the programmer.

Table 7. System Unit Index Values

System Unit	Hexadecimal Code	Decimal Code
SYSAB1	01	1
SYSAB2	02	2
SYSREL	03	3
SYSLOG	04	4
SYSRDR	05	5
SYSIPT	06	6
SYSLST	07	7
SYSOPT	08	8
SYSPCH	09	9
SYSPSD	0A	10
SYSDMY	0B	11
SYSUAS	0C	12
	0D	13
reserved	0E	14
	0F	15
SYS000	10	16
SYS001	11	17
SYS002	12	18
.		
.		
.		
SYS009	19	25
SYS010	1A	26
.		
.		
.		
SYS015	1F	31
SYS016	20	32
SYS017	21	33
.		
.		
.		
SYS200	D8	216

EXECUTION

When the EXCP supervisor call is issued, register 1 must contain the address of an input/output block. The system examines the block to determine whether it contains a valid system unit index value and the addresses of a device routine and channel program.

If the index value specifies a system unit for which no unit control block (UCB) exists, control returns with a 04 hexadecimal code in register 15. If the index refers to a device that is not operational, control returns with a 08 hexadecimal code in register 15. In either case, the system treats the requested operation as completed.

If the index value is valid but the requested channel facilities are busy, the system queues the request and returns control to the user. If the EXCP is followed by a WAIT supervisor call, SVC 1, the system delays further execution until the EXCP operation is completed.

When the required channel facilities are available, the system examines the IOB for a channel program address.

If an address is present, the system starts execution of the channel program.

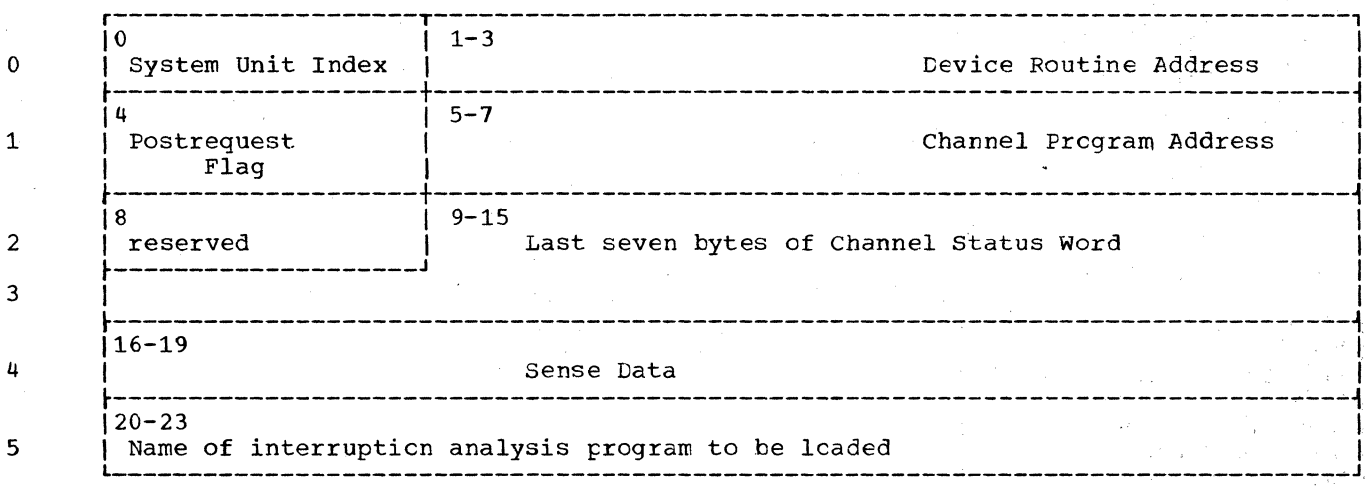

Figure 5. Input/Output Block (IOB) Format

If there is no channel program address, the system places the address of the UCB associated with system unit in register 1, the system return address in register 14, and enters the device routine. When the device routine is ready to return control to the system, it branches to the address in register 14. The addresses in registers 1 and 14 must be unchanged when the system regains control. The device routine can determine cause of entry by examining byte 0, word 3 of the UCB. When the routine is entered to prepare a channel program, the byte contains 00.

The device routine prepares the channel program and places its address into the IOB. It also must identify the type of operation being requested by putting a return code in register 15. Operation/return codes are explained below.

Code	Meaning
00	The operation is completed and no input/output operation is required. For example, a repositioning request may have been given which the device routine has analyzed and determined to be unnecessary, i.e., the device was already in the requested position.
04	An event type request (non-data transmitting which does not make the subchannel busy).
08	An activity type request (makes the subchannel busy).
0C	A transient routine is requested.

After the device routine enters the operation/return code, it returns control to the system by branching to the address in register 14.

Execution now proceeds as it would if the channel program had been prepared in advance. The system commences execution.

If a channel status word is stored as a result of the system's execution of the Start Input/Output (SIO) command, the device routine is entered exactly as on an input/output interruption indicating an error condition or a device end (result of an immediate instruction). Refer to Table 8 for the condition that will cause this.

If no CSW was stored as a result of the SIO, control returns, if WAIT is not in effect, to the user's main program (not to the device routine) until there is an input/output interruption.

INTERRUPTION PROCESSING

Execution of an EXCP request may generate one or more input/output interruptions. When an interruption occurs, the system scans the channel status word to determine the cause. Some conditions, such as channel end not accompanied by device end, are handled entirely by the system.

The system reenters the device routine if the interruption is one of the following types:

Type	UCB Request Flag
Device End	04
Program Controlled	08
Attention	0C

As before, on entry to the routine, register 1 contains the address of the UCB, and register 14 contains the system return address. The device routine examines the UCB request flag, byte 0, word 3, of the

Table 8. Conditions Causing a Device Routine to be Entered

Condition	When CSW is stored at SIO	At Input/Output Interruption[1]
Device End	Yes	Yes
Unit Check	Yes[2]	Yes
Unit Exception	No	Yes
Incorrect Length	No	Yes
Channel Data Check	No	Yes
Chaining Check	No	Yes
Busy and Status Modifier	No	Yes[3]

[1]All conditions at input/output interruption are accompanied by device end.
[2]When the command address is nonzero (broken chain).
[3]When the sense data contains '40' in byte 0 (intervention required), the device routine is not entered.

UCB to determine cause of entry. Bytes 1 through 3 of word 3 contain the address of the IOB associated with the operation that caused the interruption. The system stores the channel status word in words 2 and 3 of the IOB.

If the UCB request flag contains 04, then byte 0, word 2, of the IOB will contain one of the following codes:

Code	Meaning
00	Device end and no error
04	Unit exception
08	Condition other than unit exception

If a unit check condition is present in the CSW, the system stores sense data from the channel and unit into word 4 of the IOB.

A device routine initiates error recovery operations, when necessary, in the same manner as an initial request. It places a 04 or 08 return code into register 15 and branches to the address in register 14. As before, a 00 return code tells the system the operation is finished. The system does not permit other requests to disturb a volume while error recovery procedures are in progress.

The device routine may keep a specialized interruption analysis and error recovery program in the phase library for use when needed. The device routine instructs the system to load and enter such a program by putting a 0C return code into register 15 and branching to the address in register 14. Word 5 of the IOB must contain the name of the desired program.

The program name is specified in the IOB as four EBCDIC characters. The name must be entered in the directory of the phase library in the form xxxx , representing four EBCDIC characters followed by four blanks.

The program is loaded into the transient area of main storage. The system treats it as a logical extension of the device routine. The program operates in the supervisor state, but can issue neither supervisor calls nor privileged instructions. It uses the same return codes as a device routine and returns to the system through the address in register 14. This address and the unit control block address in register 1 must be unchanged.

A program-controlled interruption differs from a device-end interruption in its use of return codes. If, after a program controlled interruption, the device routine returns to the system with a nonzero return code in register 15, the system assumes that the operation is still in progress. If the register contains a 00 code, the system treats the operation as completed. Because the operation may be completed only through use of a Halt I/O instruction, and because a PCI interruption is not flagged if PCI occurs with unit check or device end, it is essential that a 00 return code be used only when the operation is actually completed. When the system finds a 00 return code it resets the postrequest flag in the IOB to 00, returns control to the user's main program (not the device routine), and treats the channel and device as free.

An attention interruption is handled in the same manner as a device-end interruption.

INCORPORATING DEVICE ROUTINES INTO THE SYSTEM

To incorporate a device routine for a new input/output device the user must change several items in the system as initially distributed. In addition, the system requires that each new device routine provide control information that will allow the system to operate as it does when it uses the device routines included as part of the system. These requirements are described separately in the following text.

The text also describes, in general terms, the device routines provided as part of the system. This description illustrates which type of processing can or should be done by a device routine. Device routines use information provided by the system in various control blocks and registers; this information is also explained in the text.

Changes for New Devices

To allow for input/output devices not provided for in the system, the user must make changes to various parts of the system and may be required to add a unit control block for each new device to be used.

Modifying the System Communication Region: To accommodate the routines, control blocks, and various other entries required for a new input/output device, several entries in the system communication region (SCOMRG) must be changed to indicate the additions to the system.

Modifying the Initialization Device Table: To indicate that a new input/output device has been made available to the system, the system programmer must change the initialization device table to include the device specification of the new device and some additional control information. The format of this table, as it appears in the distributed version of the system, is shown in Figure 11 (Appendix C).

Number of Device Routines: The total number of device routines in the system must be increased to include the device routine(s) added to handle the new input/output device. To indicate this increase, the user must change the value of the supervisor EQU instruction labeled NDEV. Supervisor EQU instructions are described in the chapter entitled "System Construction and Editing", under the heading "Assembler Instructions in the Supervisor."

Changes to Assembler Instructions: Depending upon installation requirements, the supervisor assembler instructions labeled NCCW, NFCB, NSUT and NUCB may require modification when a new device type is added to the system. A description of these instructions can be found in the chapter entitled "System Construction and Editing" under the heading "Assembler Instructions in the Supervisor."

DEVICE ROUTINE - CONTROL BLOCK RELATIONSHIPS

To permit device routines written by the user to operate under system control, the device routines must place certain information into control blocks so that it is available for use by the system. Some of the information is always required, while other information is required only when the user plans to design problem programs that will request input/output operations on these devices at the read/write level. For example, the current block count field of the file control block (FCB) must be maintained if a NOTE supervisor call (SVC 7) is to be used in the user's problem program.

Unit Control Block: Depending upon the type of device and the processing to be done by the device routine, it may be necessary to place the current position (UCPPT) of the device into the UCB. The current position of a device is defined for disk storage devices as the cylinder and head numbers; for card and tape devices as the block count. The following statement, which is used in a device routine included in the system, illustrates how to place the current position of the device into the UCB:

```
ST   RG0,UCPPT(RUCB)
```

Before this instruction is issued, the current position (e.g., block count) is placed into register RG0, and an indexing value to specify which unit control block is to receive the code is placed into register RUCB.

If the user wants to maintain error counts, as is done by device routines provided by the system, he must place the error counts into the UCB. The following list shows the symbolic references used for the error counts that have been provided for in the UCB.

Symbolic Reference	Type of Error Count
UCPND	Permanent no-data-transmitted errors
UCPRE	Permanent read errors
UCPWE	Permanent write errors
UCRND	Recovered no-data-transmitted errors
UCRRE	Read errors that have been corrected by recovery procedures
UCRWE	Write errors that have been corrected by recovery procedures

File Control Blocks: Depending upon whether the NOTE statement is to be used, the device routine may have to place the current block count (FCBCT) into the file control block. The current block count must be in the file control block if the NOTE statement is to be used. The following statement, which is used in a device routine included in the system, illustrates the placement of the block count into the file control block:

```
ST   RG6,FCBCT(RFCB)
```

Before this instruction is issued, the count is placed into register RG6 and an indexing value, to specify which file control block is to receive the value, is placed into register RFCB.

Input/Output Blocks: If the device routine requires that a channel command word list be executed by the system, the routine must place the address of that list (IOCCW) into the input/output block. The list address can be placed into the input/output block as described below for the current block count in the request control block (RCB).

Request Control Blocks: If the user elects to provide an entire request control block, rather than an input/output block, the

device routine he writes must handle all fields that follow the input/output block. The descriptions of fields given below refer only to the fields that are used by the device routines provided as part of the system. The manner in which those fields are used by device routines is given to illustrate which extra fields might be desirable for use with a new type of input/output device.

The device routine places the return code (IORCD) into the request control block. The hexadecimal codes that the device routine places into the request control block are:

Hexadecimal Code	Meaning
00	The operation was completed successfully.
04	An end-of-file or end-of-volume condition has occurred.
08	A permanent read error has been detected, or a permanent write error has been detected.
0C	A permanent no-data-transmitted error has been detected.
10	An illegal request has been made.
14	An incorrect-length error has occurred.

The following statement is used to place the decimal code into the request control block:

```
STC   RG6,IORCD(RIOB)
```

Before this instruction is issued, the code is placed into register RG6, and an indexing value to specify which request control block is to receive the return code is placed into register RICB.

After a read or write operation has been completed, the device routine places the current block count (IOBCT) into the request control block. The following statement is used to place the current block count into the request control block:

```
ST    RG0,IOBCT(RIOB)
```

Before this instruction is issued, the count is placed into RG0,and an indexing value to specify which request control block is to receive the value is placed into register RIOB.

If the device routine controls certain multiple-step operations, the routine maintains the operation counter (IOOPC) in the request control block. To place a value into the operation counter, the following statement is used:

```
STC   RG7,IOOPC(RIOB)
```

Before this instruction is issued, the number to be placed into the operation counter is placed into register RG7, and an indexing value to specify which request control block is to receive the number is placed into register RIOB.

Two counters (IOCNT1 and IOCNT2) are used to maintain counts of the number of retries that are made during recovery operations. If the user wishes, he may use these counters for similar purposes or for some other purpose required by his device routine.

The device routines included in the system use another byte (IOECD) in the request control block to contain an error recovery code. This code is used only by the device routine as a means of identifying the action required when the device routine is entered again. The system programmer may use this byte for any codes that he requires in his device routine.

DEVICE ROUTINE DESIGN

The design of a device routine for an input/output device not provided for in the system is determined by the user. When designing a device routine, the user need only be sure that it satisfies the requirements stated earlier.

Although the user is free to write his device routine as he wishes, a general diagram of the actions normally done in a device routine is shown in Figure 6. This diagram is not intended for any specific device, but it shows the general approach used in the device routines provided as part of the programming system. In the description of Figure 6 that follows, the methods used in device routines provided in the system are mentioned to suggest a possible method to the system programmer. (Numbered blocks in the figure are explained in the list following the chart.) The information used by the device routines is described under "Information Available to Device Routines."

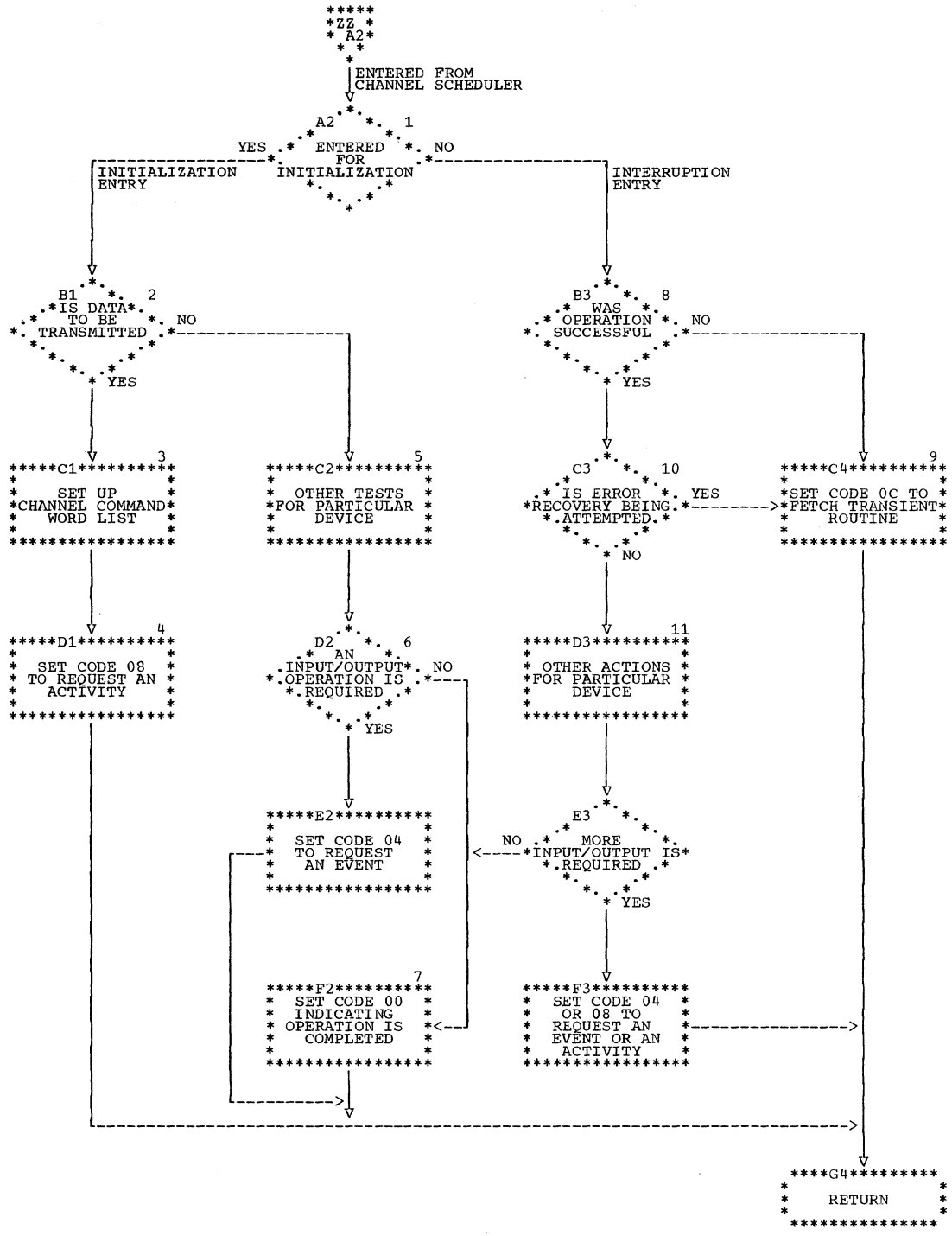

Figure 6. General Flow of a Device Routine

Explanations of Logic Blocks:

1. Cause-of-entry can be determined by checking the code in byte 12 of the UCB.

2. If the device is being addressed at the read/write level, the device routine can determine which operation has been requested by checking the code in byte 32 of the RCB.

3. If data is to be transmitted, the device routine must set up a channel command word list. Depending upon the device, the channel command word list may consist of one or more CCWs.

   The number of channel command word lists for any device depends upon the operations that the device routine must initiate. Descriptions of the CCWs required for the various operations of input/output devices are given in publications that describe device functions. These publications are listed in the IBM System/360 Bibliography, Form A22-6822 (subject codes 03-09). Additional information regarding the format of channel command words is contained in the publication IBM System/360 Principles of Operation, Form A22-6821.

4. The device routine places a code into register 15 if the chain of commands contains either a data transmission request or an event request requiring the positioning of a device before a read or write operation, e.g., a seek request on a disk. (The actual read/write CCW will be issued through an activity exit upon successful completion of the event.)

5. Byte 32 of the RCB is tested.

6. If the check of byte 32 indicates that no data is to be transmitted but an input/output operation is required (e.g., a rewind operation), the device routine must set up a channel command word list. After preparation of the list, the routine places an event code into register 15 and returns control to the system.

7. If the check of byte 32 indicates that no input/output operation is required, the device routine places a completion code into register 15. If the original request was invalid, the device routine must, additionally, set up an error code in the RCB.

8. When byte 12 of the UCB indicates that the device routine was entered for interruption processing, the device routine must examine byte 8 of the IOB to determine whether a previously initiated operation was performed without error.

9. Entered via block (8):
   If an error has occurred and if the error recovery routine is transient, the device routine places a code into register 15 requesting the loading of the error recovery routine. If the recovery routine gets control when loaded, it attempts to recover from the error by retrying the operation.

   Entered via block (10):
   If the device routine detects that the interruption resulted from a previously initiated attempt to recover from an error, the recovery routine is entered to analyze whether the error has been corrected. The device routine must again place a code into register 15 requesting the loading of the error recovery routine.

   As implied above, the device routine must provide an error recovery procedure. The error recovery procedure must determine which type of error has occurred and then attempt to correct it, if possible. A procedure for handling errors that cannot be corrected, or that recur when an attempt to correct them is made, must also be provided. The action taken for the various types of errors is determined by the user. For example, he may decide that a job should be canceled if a channel data check has been detected, or he may write the procedure so that a return code is issued providing the option of canceling or continuing -- the action taken depending upon other variables. If an error can be corrected by the computer operator, the user may elect to send a message to the operator and specify what is required to correct the error. To issue such a message, the facility of the error message system routine called by a FETCH exit from the device routine should be used.

   The operations required to retry an operation are dictated by the type of error and the type of device. In general, the procedure is to reestablish the conditions which existed before the action was taken that resulted in an error. For example, a tape must be repositioned to the beginning of a record that could not be read correctly; to do this, a backspace operation is required. However, an error occurring while transmitting the read command to the device does not require any repositioning.

The error recovery procedure must also place an appropriate return code in register 15.

10. Test to determine whether the interruption resulted from a previously initiated error recovery attempt.

11. If no error was detected and if no recovery procedure was in progress, the device routine checks for other conditions that may exist for the particular device. Based on these checks, the device routine may have to set up a channel command word list to complete an input/output request.

    Other actions that the device routine would perform at this point would include updating pointers, updating block counts, and setting program flags. After all actions have been completed, the device routine places a code into register 15 indicating that the operation is completed.

INFORMATION AVAILABLE TO DEVICE ROUTINES

Before the channel scheduler gives control to a device routine, the system provides information for use by the device routine. Such information is placed into registers or the various control blocks.

The contents and format of the unit control block are shown in Appendix A; those for the file control block are shown in Appendix B.

In the descriptions of information available to device routines, all fields are described, including those which the device routine must handle as described previously under "Device Routine Requirements."

Information in the Registers

The system places the following information into the indicated registers before control is given to the device routine:

Register Number	Contents
1	The address of the unit control block.
2	The address of the input/output block (or of the request control block if one is used).
10	The address of the first byte in the constant pool.

Information in the Request Control Block

In the description of the information that the system places into the request control block, those entries marked with one asterisk must be provided by the device routine, except for words 6 through 9. The contents of these words must be provided only if read/write level operations are to be performed. These words are used primarily for communication between device routines and read/write routines.

Those entries marked with two asterisks are not available until the routine has been entered for interruption processing. Entries not marked are available when the device routine is entered for initialization processing. They remain available when the device routine is entered later for interruption processing. The information in the request control block (see Figure 7) is as follows:

Word	Byte	Description
0	0	System unit index number of the system unit to be used in the input/output operation. This value can be determined from Table 7.
0	1-3	Address of the device dependent routine to be used to set up the channel commands and analyze interruptions.
1	4	Postrequest flag indicating whether the block currently is active: 00 = No operation pending 01 = Operation in progress
*1	5-7	Address of the first channel command required to execute the operation.
2	8	Reserved for system use.
**2,3	9-15	Last seven bytes of channel status word, stored when an operation is started and when an interruption occurs.

Word	Byte	Description
**4	16-19	Sense information, stored when unit check condition occurs.
*5	20-23	Name of program to be loaded from the phase library when necessary for interruption analysis.
*6	24	Error recovery code identifying a type of error.
*6	25-27	Counters used to keep track of number of attempts made to recover an error.
*7	28	Return code 00 = Operation completed 04 = End of file or end of volume 08 = Permanent transmission error 0C = No data transmitted 10 = Invalid request 14 = Incorrect length
*7	29-31	Address of file control block being used for this operation.
*8	32	Request code identifying the type of operation to be set up by a device routine. 01 = Write 02 = Read 07 = Rewind 0F = Rewind and Unload 1F = Write End of File 3F = Point
*8	33-35	Address of buffer to be used for transmission.
*9	36	Incorrect length control byte 20 = Suppress incorrect length indication 00 = Check for incorrect length
*9	37	Reserved
*9	38-39	Number of bytes to be transmitted (see note).

Note: Following completion of an input/output operation, the contents of word 9 are replaced, by the device routine, with the updated data set position block count.

Word			
0	0 System Unit Index Number	1-3 Address of Device Dependent Routine	
1	4 Postrequest Flag	5-7* Address of First Channel Command	
2	8 Reserved	9-15** Last Seven Bytes of Channel Status Word	
3			
4	16-19** Sense Information		
5	20-23* Name of Program to be Loaded from Phase Library		
6	24* Error Recovery Code	25-27 Counters	
7	28* Return Code	29-31* Address of File Control Block	
8	32* Request Code	33-35* Address of Buffer for Transmission	
9	36* Incorrect Length Control Byte	37* Reserved	38-39 Number of Bytes to be Transmitted
	*Must be provided by device routine. **Not available until after entry for interruption processing.		

Figure 7. Request Control Block

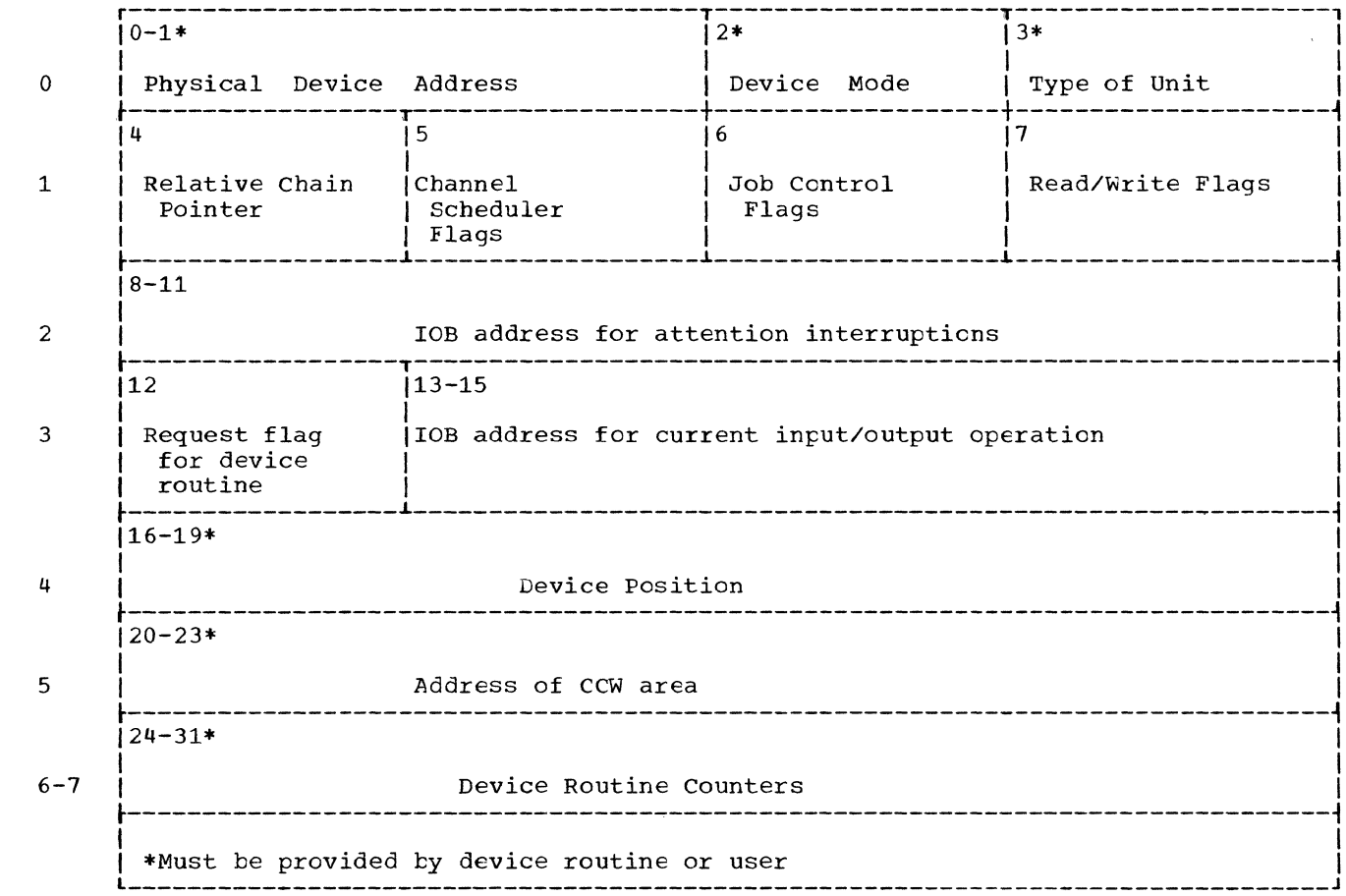

Figure 8. Unit Control Block (UCB) Format

Unit Control Block Fields

Fields in the unit control block are defined below. Entries marked with an asterisk must be provided by a device routine or the user.

Word	Byte	Description
*0	0,1	Physical device address
*0	2	Device mode 01 = Burst mode 02 = Overrunable byte mode 03 = Non-overrunable byte mode
*0	3	Type of unit 10 = 1052 Console Printer-Keyboard 20 = 2501 Card Reader 21 = 2540 Card Reader 22 = 2520 Read-Punch 23 = 1442 Read-Punch 28 = 2520P Card Punch 29 = 2540P Card Punch 2A = 1442P Card Punch 30 = 1403 Printer 31 = 1403M7 Printer 32 = 1443 Printer 33 = 1443S Printer 40 = 2400 Magnetic Tape Unit 41 = 2400H Magnetic Tape Unit 42 = 2400D Magnetic Tape Unit 48 = 2400T7 Magnetic Tape Unit 49 = 2400T7C Magnetic Tape Unit 50 = Single Disk Storage Drive 51 = 2311 Disk Stcrage Drive
1	4	Relative chain pointer, pointing to the next UCB in either an acitvity, event, or transient routine chain.

Word	Byte	Description
1	5	Channel scheduler flags 80 = Intervention request processed by device routine 40 = Channel end expected 20 = Device end received 10 = Retry operation 08 = Device not ready 04 = Attention waiting 02 = Event in progress 01 = Device busy
1	6	Job control flags 80 = Device down 40 = System standard assignment 20 = Job control assigned 10 = Programmer assigned 08 = Assigned this step 04 = The volume cannot be dismounted
1	7	Read/Write flags 80 = System multiple operation 40 = Error message to be requested 20 = End of volume (EOV) 10 = System request 08 = Multiple operation 02 = Volume label present 01 = Volume has been mounted
2	8-11	Address of input/output block used when an attention interruption occurs. (Applicable only to those devices that can signal attention.)
3	12	Request flag for device routine 00 = Setup 04 = Device-end interruption 08 = Attention interruption 0C = Program controlled interruption
3	13-15	Input/output block address associated with this operation
*4	16-19	Current physical position of the device in terms of the cylinder and head positions for direct access devices and block count for sequential devices.
*5	20-23	Address of channel command word area associated with the device.
*6,7	24-31	Permanent and temporary error counters set and used by the system's device routines.

Word	0	1	2	3
0	Flag Byte	Flag Byte	Reserved	Number of blocks per track
1	4-7* Block Count			
2	8-9 Logical Record Length		10-11 Reserved	
3	12-15 Seek and search address			
4	16 Block Number	17 Length of key area (2311 only)	18-19 Maximum number of bytes per block	
5	20-23 Displacement of first block of a directoried data set member			
6	24-27 Device address of first block of the data set			
7	28-31 Number of last block written			
8	32-35 Number of blocks reserved for the data set			

*Must be provided by device routine

Figure 9. File Control Block Format (Disk)

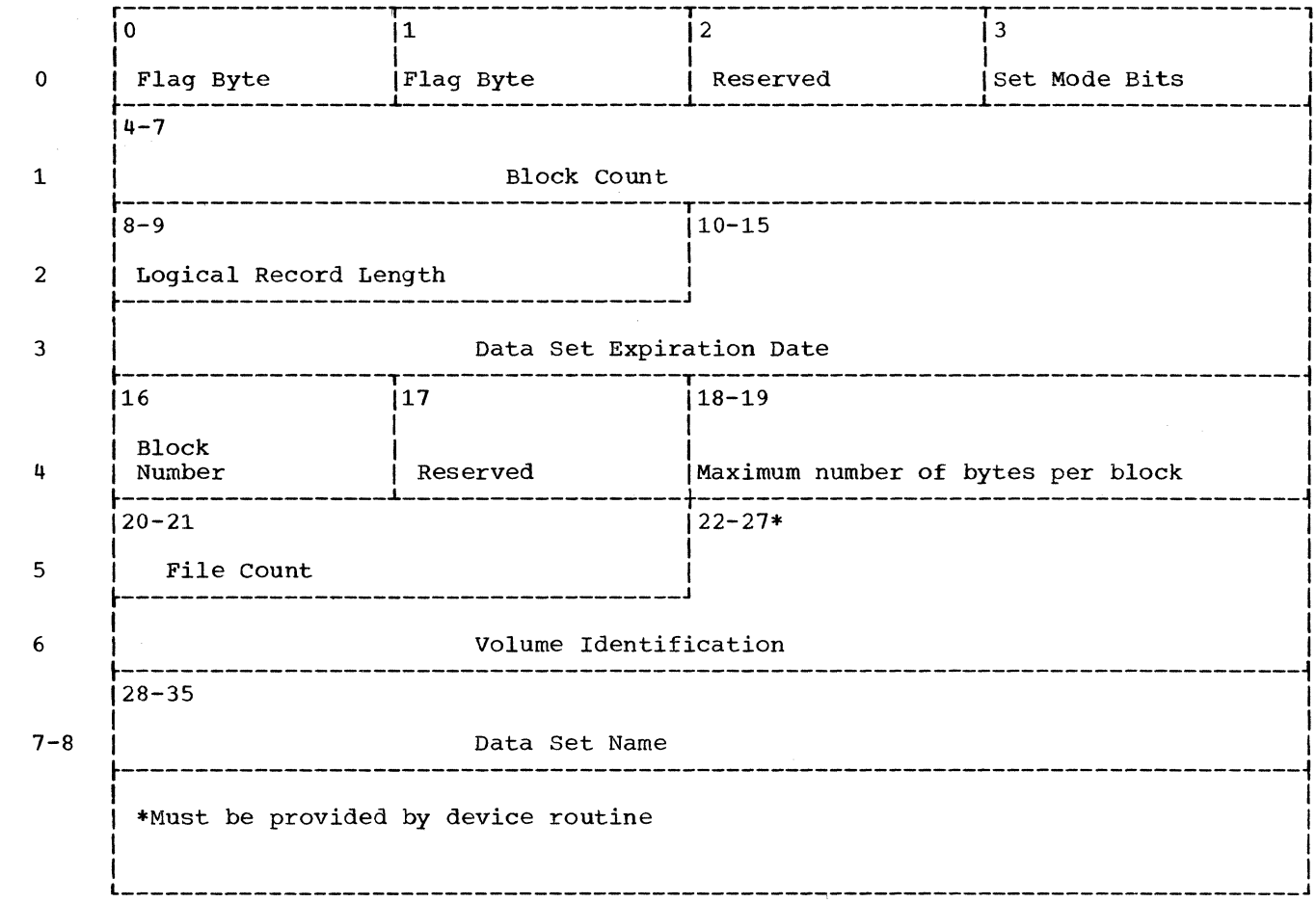

Figure 10. File Control Block Format (Tape and Card)

File Control Block Fields

Fields in the file control block for disk volumes are defined below. The entry marked with an asterisk must be provided by a device routine.

Word	Byte	Description
0	0	Flag Byte 80 = Standard unit 40 = Fresh data set 20 = Update VTOC 10 = Labeled 08 = Header checked 04 = Modification 02 = Disconnected 01 = Open
0	1	Flag Byte 80 = Control character 40 = ASA control characters 20 = Write check 10 = Ignore (dummy data set) 08 = Deleted 04 = Output data set 02 = Data set formatted
0	2	Reserved
0	3	Number of blocks per track
*1	4-7	Current block count
2	8,9	Logical record length
2	10,11	Reserved
3	12-15	Seek and search address for the current operation
4	16	Number of block being processed
4	17	Number of bytes in key area (2311 only)

Word	Byte	Description
4	18-19	Maximum number of bytes per block
5	20-23	Displacement (number of blocks) of first block of member in a directoried data set. If the data set associated with the RCB is not directoried, these bytes contain zeros.
6	24-27	Address of the first block in the data set
7	28-31	Number of the last block written
8	32-35	Number of blocks reserved for this data set

Fields in a file control block for tape and card units are the same as for direct access devices, except as follows:

Word	Byte	Description
0	1	Flag byte 80 = Control character 40 = ASA control characters 10 = Ignore (dummy data set) 08 = Deleted 04 = Output data set
0	3	Mode of tape operation (not used for card data sets).
2,3	10-15	Expiration date of data set
4	17	Reserved
5	20,21	Current file count, including trailer label file marks
*5,6	22-27	Volume identification in EBCDIC
7,8	28-35	Data set name in EBCDIC (not used for card data sets).

Device Specification	Device Type Code	No. of CCWs	Device Mode	CCW Relocation	Pointer to CCW List
1052[1]	10	06	33	01	CONCCW
2501	20	02	02	00	RD1CCW
2540	21	02	03	00	RC1CCW
2520	22	02	02	00	RP1CCW
1442	23	02	02	00	RP2CCW
2520P	28	02	02	00	RD2CCW
2540P	29	13	03	00	PC2CCW
1442P	2A	02	02	00	RP3CCW
1403	30	02	03	00	PT1CCW
1403M7	31	02	03	00	PT2CCW
1443	32	02	03	00	PT3CCW
1443S	33	02	03	00	PT4CCW
2400	40	02	01	00	TP1CCW
2400H	41	02	01	00	TP2CCW
2400D	42	02	01	00	TP3CCW
2400T7	48	02	01	00	TP4CCW
2400T7C	49	02	01	00	TP5CCW
SDSD	50	05	01	01	DK1CCW
1316[2]	51	06	01	01	DK2CCW
[1]Location TBLORG [2]Location TBLEND					

Figure 11. Initialization Device Table

Initialization Device Table Fields

Device Specification
Contains an abbreviated device name. The name can consist of a maximum of eight characters.

Device Type Code
A 2-byte code designating a device type. In selecting a code, the user must be sure that the first digit of the code is different from the first digit of existing device type codes. The system uses the first digit of the code as a pointer to the device routine to be used for a particular type of device.

Number of Channel Command Words
The number of CCWs required varies according to device.

Device Mode
Specifies the mode of the device as either burst mode (01), overrunnable byte mode (02), or nonoverrunable byte mode (03). A nonzero in the four high-order bits (e.g., 33 for the 1052) indicates an attention type device.

CCW Relocation
a 01 indicates relocatability. The entry for this field, either 00 or 01, varies according to the channel command list required for the specific device.

Pointer to the CCW List
Symbolic pointer to the device's CCW list.

```
                                        DEVICE TYPE      DEVICE TYPE CODE
CTYPTB   DC     C'1052   '      1052 CONSOLE     10
         DC     X'10'
         DC     C'2501   '      2501 READER      20
         DC     X'20'
         DC     C'2540   '      2540 READER      21
         DC     X'21'
         DC     C'2520   '      2520 READ-PUNCH  22
         DC     X'22'
         DC     C'1442   '      1442 READ-PUNCH  23
         DC     X'23'
         DC     C'2523P  '      2520 PUNCH       28
         DC     X'28'
         DC     C'2540P  '      2540 PUNCH       29
         DC     X'29'
         DC     C'1442P  '      1442 PUNCH       2A
         DC     X'2A'
         DC     C'1403   '      1403 PRINTER     30
         DC     X'30'
         DC     C'1403M7 '      1403 PRINTER     31
         DC     X'31'
         DC     C'1443   '      1443 PRINTER     32
         DC     X'32'
         DC     C'1443S  '      1443 PRINTER     33
         DC     X'33'
         DC     C'2400   '      2400 TAPE        40
         DC     X'40'
         DC     C'2400H  '      2400 TAPE        40
         DC     X'41'
         DC     C'2400D  '      2400 TAPE        42
         DC     X'42'
         DC     C'2400T7 '      2400 TAPE        48
         DC     X'48'
         DC     C'SDSD   '      SINGLE DISK STORAGE DRIVE 50
         DC     X'50'
         DC     C'1316   '      1316 DISK PACK   51
         DC     X'51'
```

Note: In this table, all character constants consist of 7 characters; blank characters are used following the device type to fill out the constant.

Figure 12. Job Control Device Table

```
CONCCW   CCW    0,0,X'00',0     1052 CONSOLE PRINTER
         DC     X'0A000008'          ATTENTION CCWS
         DC     X'20000008'
         CCW    0,0,X'00',0
         DC     X'04000000'          ATTENTION IOB
         DC     X'00FFFFEC'          COMPLEMENT FOR CCW
         DC     4C'0'
RD1CCW   CCW    0,0,X'60'1      2501 CARD READER
         CCW    0,0,X'00',0
PC1CCW   CCW    0,0,X'60',1     2540 CARD READER
         CCW    0,0,X'00',0
PC2CCW   CCW    0,0,X'60',1     2540 CARD PUNCH
         DC     24F'0'
RD2CCW   CCW    0,0,X'60',1     2520 CARD PUNCH
         CCW    0,0,X'00',0
RP1CCW   CCW    0,0,X'60',1     2520 CARD READ PUNCH
         CCW    0,0,X'00',0
RP2CCW   CCW    0,0,X'60',1     1442-N1 CARD READ PUNCH
         CCW    0,0,X'00',0
RP3CCW   CCW    0,0,X'60',1     1442-N2 CARD PUNCH
         CCW    0,0,X'00',0
PT1CCW   CCW    0,0,X'60',1     1403 PRINTER
         CCW    3,0,X'00',1
PT2CCW   CCW    0,0,X'60',1     1407 PRINTER
         CCW    3,0,X'00',1
PT3CCW   CCW    0,0,X'60',1     1443 PRINTER 120 POSITIONS
         CCW    3,0,X'00',1
PT4CCW   CCW    0,0,X'60',1     1443 PRINTER 144 POSITIONS
         CCW    3,0,X'00',1
TP1CCW   CCW    0,0,X'60',1     2400 TAPE 800 BPI
         CCW    0,0,X'00',0
TP2CCW   CCW    0,0,X'60',1     2400 TAPE 1600 BPI
         CCW    0,0,X'00',0
TP3CCW   CCW    0,0,X'60',1     2400 TAPE 9 TRK AND DUAL DENSITY
         CCW    0,0,X'00',0
TP4CCW   CCW    0,0,X'60',1     2400 TAPE 7 TRACK
         CCW    0,0,X'00',0
TP5CCW   CCW    0,0,X'60',1     2400 TAPE 7 TRACK AND CONVERT
         CCW    0,0,X'00',0
DK1CCW   CCW    0,0,X'20',1     SINGLE DISK STORAGE DRIVE
         DC     F'24'           RELOCATION FOR *+24
         DC     F'5'
         CCW    0,0,X'20',0
         CCW    0,0,X'00',0
         CCW    0,0,X'00',0
DK2CCW   CCW    0,0,X'20',6     2311 DISK STORAGE DRIVE
         CCW    0,0,X'60',5
         DC     X'08FFFFF8'     RELOCATION FOR *-8
         DC     F'0'
         CCW    0,0,X'00',0
         CCW    0,0,X'00',0
         CCW    0,0,X'00',0
```

Figure 13. CCW Table

Table 9 is a listing of all modules of the IBM System/360 Model 44 Programming System as initially distributed. The list also provides symbolic identifications and phase names for each component, where applicable.

Symbolic identifications allow the user to associate any card image with the module to which it belongs. The 3-character identification code -- four characters for FORTRAN modules -- is found at the beginning of the serialization field, columns 73 through 80 of each card image.

Phase names are applicable to all modules that are members of the phase library. Phase names do not apply to FORTRAN programs that are members of the module library.

System Mod Level Directory

The system mod level directory is a non-executable phase (BESYSLEV) that contains the version and modification level of each component in the user's system. The following statements can be used to obtain a listing of the contents of the system mod level directory:

```
//MODLEV  JOB
//SYS002  ACCESS  SDSABS,SAME=SYSAB1
// EXEC   UTILS
 PRTMEM   INCL=BESYSLEV,SIZIN=(720,60)
/*
```

Table 9. System Components

Module	Symbolic Card ID[1]	Phase Name
2315 Disk Initialization	BMG (BAA)	IN15
2311 Disk Initialization	BMF (BAB)	IN11
Save/Restore	BAC	BACDPRS
Save/Restore	BIB (BAC)	BACDPRS
Print/Punch	BAD	BADPRPU
Absolute Loader	BBA	BBLDR1
Absolute Loader	BBA	BBLDR2
System Construction	BCA	BCASC15
IPL Phase I	BDA	BDAIP15
System Mod Level Directory	BEA	BESYSLEV
System Communication Region	BFA	BFSUPVSR
SVC Handler	BFB	BFSUPVSR
Channel Scheduler	BFC	BFSUPVSR
Device Routine Utilities	BFD	BFSURVSR
Disk Initiator	BFE	BFSUPVSR
SDSD ERP	BFF	BFSUPVSR

Table 9. System Components (continued)

Module	Symbolic Card ID[1]	Phase Name
Tape Initiator	BFG	BFSUPVSR
Card Initiator	BFH	BFSUPVSR
Printer Initiator	BFI	BFSUPVSR
Console Typewriter	BFJ	BFSUPVSR
I/O Routines	BFK	BFSUPVSR
IPL Phase II Add/Subtract	BFL	BFSUPVSR
Disk ERP Phase 1	BGA	BGD1
Disk ERP Phase 2	BGB	BGD2
Tape ERP Phase 1	BGC	BGT1
Tape ERP Phase 2	BGD	BGT2
Card ERP	BGE	BGCD
Printer ERP	BGF	BPR
Error Message Writer	BGG	BGMG
OPEN Phase 1	BHA	BHOPEN1
OPEN Phase 2	BHB	BHOPEN2
OPEN Phase 3	BHC	BHOPEN3
OPEN Phase 4	BHD	BHOPEN4
CLOSE Phase 1	BHE	BHCLOSE1
CLOSE Phase 2	BHF	BHCLOSE2
Cancel Phase 1	BHH	BHCNCEL1
Cancel Phase 2	BHI	BHCNCEL2
Dump Phase 1	BHJ	BHDUMP1
Dump Phase 2	BHK	BHDUMP2
Job Control Basic Phase	BIA	BIAJBBAS
Job Control Phase 1	BIB	BIBJBPH1
Job Control Phase 2	BIC	BICJBPH2
Job Control Phase 3	BID	BIDJBPH3
Job Control Phase 4	BIE	BIEJBPH4
Job Control Phase 5	BIF	BIFJBPH5

Table 9. System Components (continued)

Module	Symbolic Card ID[1]	Phase Name
Linkage Editor	BKA	BKLNKEDT
Linkage Editor	BKA	BKLNKED1
Linkage Editor	BKA	BKLNKED2
Assembler Phase 1	BLA	BLAST
Assembler Phase 2	BLB	BLAZE
Utilities Root Phase	BMA	BMUTILS
Utilities Data Set Copy	BMB	BMUTCOPY
Utilities Data Set Copy	BMB	BMUTPCHA
Utilities Squeeze/Map	BMC	BMUTSQMP
Utilities Initialization Basic	BMD	BMUTINIT
Utilities Tape Initialization	BME	BMUTINTP
Utilities 2311 Initialization	EMF	BMUTIN11
Utilities 2315 Initialization	BMG	BMUTIN15
FORTRAN Compiler		
FORTRAN Phase 1	BNA	BNAFORT
FORTRAN Phase 1	BNB	BNAFORT
FORTRAN Phase 1	BNC	BNAFORT
FORTRAN Phase 1	BND	BNAFORT
FORTRAN Phase 2	BNE	BNAALL
FORTRAN Phase 2	BNF	BNAALL
FORTRAN Phase 3	BNG	BNAGEN
FORTRAN Phase 4	BNH	BNAEXT
FORTRAN Phase 4	BNI	BNAEXT
FORTRAN Object Fix Up	BNX	BNAADJST
FORTRAN Expander	BNZ	BNAXPND
FORTRAN Library		Not Applicable
Unit Table	BOA	
IBCOM	BOB	
FIOCS	BOC	
DIOCS	BOD	

Table 9. System Components (continued)

Module	Symbolic Card ID[1]	Phase Name
Namelist	BOE	
Object Fix Option	BOF	
LINK/OVERLAY	BOH	
Square Root	BOMA	
Single Real Exponentiation	BOMB	
Single Real Log	BOMC	
Single Real SIN/COS	BOMD	
Single Real Arctan	BOME	
Single Real Hyperbalic Tangent	BOMF	
Double Square Root	BOMG	
Double Exponentiation	BOMH	
Double Log	BOMI	
Double SIN/COS	BOMJ	
Double Arctan	BOMK	
Double Hyperbalic Tangent	BOML	
Complex Mult./Div.	BOMM	
Complex Absolute Value	BOMN	
Complex Square Root	BOMP	
Complex Exp.	BOMQ	
Complex Log	BOMR	
Complex SIN/COS	BOMS	
Complex Double Mult./Div.	BOMT	
Complex Double Absolute Value	BOMU	
Complex Double Square Root	BOMV	
Complex Double Exponentiation	BOMW	
Complex Double Log	BOMX	
Complex Double SIN/COS	BOMY	
Single Real Hyperbolic SIN/COS	BOMZ	
Single Real Arc SIN/COS	BONA	
Single Real TAN/COT	BONB	
Double Hyperbolic SIN/COS	BONC	
Double Arc SIN/COS	BOND	

Table 9. System Components (continued)

Module	Symbolic Card ID[1]	Phase Name
Double TAN/COT	BONE	
Single Error Function	BONF	
Single GAMMA/LOG-GAMMA	BONG	
Double Error Function	BONH	
Double GAMMA/LOG-GAMMA	BONI	
Exp. Integer to Integer	BONJ	
Exp. Real to Integer	BONK	
Exp. Double to Integer	BONL	
Exp. Real to Real	BONM	
Exp. Double to Double	BONN	
Exp. Complex to Integer	BONP	
Exp. Double Complex to Integer	BONQ	
Integer MAX Function	BONR	
Real MAX Function	BONS	
Double MAX Function	BONT	
EXIT	BONU	
Sense Light Simulation	BONV	
Overflow	BONW	
Divide Check	BONX	
DUMP/PDUMP	BONY	

[1]Object decks are identified using the same symbols as for the corresponding symbolic decks, except where noted otherwise in parentheses.

APPENDIX E: SAMPLE PROGRAM

This appendix contains a description of the sample program provided by IBM to test the functioning of the various components of the programming system after system construction. Included are a description of the sample program, operating instructions, and a description of the program execution results. More detailed operating procedures can be found in IBM System/360 Model 44 Programming System: Operator's Guide, Form C28-6815.

DESCRIPTION

The sample program, which computes the coefficients of Chebyshev polynomials, consists of a FORTRAN main program and two subprograms (one SUBROUTINE subprogram and one FUNCTION subprogram). The sample program is designed to operate with SYSRDR assigned to a card reader and SYSLST assigned to a printer. Statements within the sample program card deck (BAZSAMPL) ensure that SYSIPT is assigned to the same device as SYSRDR and that SYSOPT is assigned to the same device as SYSLST.

The BAZSAMPL card deck consists of:

1. Job control statements that assign SYSIPT and SYSOPT to the same devices as SYSRDR and SYSLST, respectively.
2. Job control statements for three FORTRAN compilations.
3. Three sets of FORTRAN source statements (one for the main program and one for each subprogram).
4. Job control statements for a linkage edit and for the execution of the sample program.
5. Data deck for sample program execution.

The program processes two data cards as its input. Figure 14 shows a complete list of the expected output from the execution of the program. This list may be used for checking output.

OPERATING INSTRUCTIONS

1. Mount the system residence disk.
2. Perform the initial programming loading procedure for the system.
3. Place the BAZSAMPL card deck in the card reader, ready the reader, and press the End of File key.
4. Respond to messages written by the system on SYSLOG (the console printer-keyboard) as required.

OUTPUT

1. The job control processor will read, process, and print all job control statements.
2. For each set of source statements, the FORTRAN IV compiler will list the following:
   a. Heading including date and level.
   b. The set of source statements.
   c. Storage map including size of each COMMON block and size of program.
3. The linkage editor will prepare and list a storage map including the relative address of each external reference.
4. The executed sample program will list the results of execution, which should correspond to the list shown in Figure 14.

```
**SAMPLE PROGRAM EXECUTION HAS BEGUN.

                COEFFICIENTS OF CHEBYSHEV POLYNOMIALS   C N,I,A,B

                                    N
T N,PHI,A,B     SUMMATION C N,I,A,B *COS I*PHI
                    I 0

        C   4,  0,  1.2700,  0.2700        4.63396

        C   4,  1,  1.2700,  0.2700        8.58701

        C   4,  2,  1.2700,  0.2700        6.77608

        C   4,  3,  1.2700,  0.2700        4.42450

        C   4,  4,  1.2700,  0.2700        2.60144

      THE RESULTS AS SHOWN ABOVE ARE CORRECT FOR THE PURPOSE OF THIS DEMONSTRATION.

                COEFFICIENTS OF CHEBYSHEV POLYNOMIALS   C N,I,A,B

                                    N
T N,PHI,A,B     SUMMATION C N,I,A,B *COS I*PHI
                    I 0

        C   6,  0,  1.2700,  0.2700       28.97987

        C   6,  1,  1.2700,  0.2700       54.65051

        C   6,  2,  1.2700,  0.2700       45.65968

        C   6,  3,  1.2700,  0.2700       33.42590

        C   6,  4,  1.2700,  0.2700       20.94514

        C   6,  5,  1.2700,  0.2700       10.70439

        C   6,  6,  1.2700,  0.2700        4.19585

      THE RESULTS AS SHOWN ABOVE ARE CORRECT FOR THE PURPOSE OF THIS DEMONSTRATION.

*SAMPLE PROGRAM EXECUTION HAS COMPLETED.
```

Figure 14. Output of Sample Program

Printed in U.S.A. C28-6814-0

IBM®

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

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

READER'S COMMENTS

Title: IBM System/360 Model 44 Programming System
Systems Programmer's Guide

Form: C28-6814-0

Your comments assist us in improving the usefulness of our publications; they are a major part of the input used for technical newsletters and revisions.

Please do not use this form for technical questions about the system; it only delays the response. Instead, direct your technical questions to your local IBM representative.

Corrections or clarifications needed:

Page	Comment

If you wish a reply, please include your name and address below:

fold fold

FIRST CLASS
PERMIT NO. 33504
NEW YORK, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

1271 AVENUE OF THE AMERICAS
NEW YORK, N.Y. 10020

Attention: PUBLICATIONS

fold fold

Printed in U.S.A. C28-6814-0

IBM®

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

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