File No. 1410/7010-36
Form C28-0319-4

IBM Systems Reference Library

IBM 1410/7010 Operating System (1410-PR-155)

System Monitor

This publication provides programmers and systems analysts with the basic principles and programming requirements for the System Monitor. The programming information given includes the control cards required, the linkage sequences to Monitor routines and data fields, the messages produced by the Monitor, and the console inquiries accepted.

The System Monitor provides control for the IBM 1410/7010 Operating System by furnishing programmed transition between runs and between jobs, relocating programs, and assigning input/output units. The System Monitor consists of the Resident and Transitional Monitors, 1410-SV-962, and the Linkage Loader, 1410-UT-963.

The reader is assumed to have a basic knowledge of the IBM 1410 and/or 7010 Data Processing Systems and to have read the publication *IBM 1410/7010 Operating System; Basic Concepts*, Form C28-0318.

NOTE: The IBM 1302 Disk Storage Unit is now designated the IBM 2302 Disk Storage Unit; there has been no change in the unit itself, in the applications for which the unit may be used, or in the programming parameters used to specify those applications. The IBM 2302 Disk Storage Unit designation has been used in the text of this publication; programming parameters remain unchanged and refer to 1302.

MAJOR REVISION (September 1965)

This publication is a major revision of *IBM 1410/7010 Operating System; System Monitor,* Form C28-0319-3, and makes that publication and its associated Technical Newsletters (N27-1216 and N27-1234) obsolete. The changes reflect the removal of restrictions for chaining and make minor corrections to text and figures. Text changes are indicated by a vertical line at the left of the text; figure changes are indicated by a bullet (•) at the left of the figure title.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the contents of this publication to:
IBM Corporation, Programming Systems Publications, Department 637, Neighborhood Road, Kingston, New York 12401

Contents

Purpose of this Publication

This publication describes use of the System Monitor to control the 1410/7010 Operating System. Information is included about the functions and components of the System Monitor and its relationship to the Operating System. The publication also gives instructions for writing programs to run under control of the System Monitor.

Purpose of the System Monitor

The System Monitor controls the 1410/7010 Operating System. Some of the functions that it performs for programs within the Operating System are:

1. Assignment of input/output units
2. Program loading, including relocation of programs, and linkage between programs and sub-routines that were independently written and compiled
3. Programmed transition from run to run and job to job

The Operating System can provide, at the option of each installation, control facilities for IBM Tele-processing Systems. Detailed information concerning this type of application is contained in the publication *IBM 1410/7010 Operating System; Tele-processing Supervisor,* Form C28-0321.

Prerequisite and Related Literature

The publication *IBM 14010/7010 Operating System; Basic Concepts,* Form C28-0318, is a prerequisite. The reader is assumed to be familiar with both the terminology and concepts defined by that publication.

For a disk-oriented 1410 or 7010 Data Processing System, the publication *IBM 1301, Models 1 and 2, Disk Storage and IBM 2302, Models 1 and 2, Disk Storage with IBM 1410 and 7010 Data Processing Systems,* Form A22-6788, is a prerequisite.

Related literature includes the following publications, which contain detailed information about the other elements and functions of the 1410/7010 Operating System:

System Generation, Form C28-0352
Basic Input/Output Control System, Form C28-0322
Tele-processing Supervisor, Form C28-0321
Utility Programs, Form C28-0353
Generalized Tape Sorting Program, Form C28-0354
Random-Processing Scheduler, Form C28-0323
Autocoder, Form C28-0326
FORTRAN, Form C28-0328
COBOL, Form C28-0327
Operator's Guide, Form C28-0351
Generalized Sorting Program using IBM 1301/2302 Disk Storage, Form C28-0404
File Organization System for IBM 1301/2302 Disk Storage, Form C28-0405
Support of IBM 1311 Disk Storage Drives Under the Operating System, Form C28-0402

System Monitor Concepts

The System Monitor coordinates the functions of all Operating System components, including user-written programs operating under it. In this coordination, the System Monitor does the following:

1. It reads and analyzes control cards that describe functions to be performed.
2. It assigns input/output units.
3. It reads programs from the card reader, tapes, or disk. These programs may be requested by control cards or by imbedded calls within programs.
4. It relocates these programs to unused areas of storage, provides communication among program elements, and loads the programs into storage.
5. It turns processing control over to the programs in storage and receives control from them when they are completed. These dependent programs may be user-written programs or IBM-provided components of the Operating System.
6. It handles all input/output operations for dependent programs.

To perform these control functions, a part of the System Monitor, the *Resident Monitor,* stays in lower core storage at all times. The Resident Monitor loads programs into core storage, handles input/output, and controls end-of-program actions.

The rest of the System Monitor and all other components of the Operating System are kept on tape or disk and are brought into core storage as required. The other parts of the System Monitor are:

1. *Transitional Monitor,* which analyzes control cards and assigns input/output units
2. *Linkage Loader,* which relocates programs and provides communication among independently compiled program elements

Programs compiled by the language processors are in relocatable format. Relocatable programs have been arbitrarily assigned locations by the language processors or the programmer. The Linkage Loader (1) relocates the addresses in the relocatable programs to core locations above the Resident Monitor, and above other programs that will be in storage below the program; or (2) relocates the addresses to installation-specified locations. After relocation, programs are in absolute format. The absolute addresses in these programs are the addresses the programs will use when they are executed.

The over-all operation of the System Monitor is shown in Figure 1.

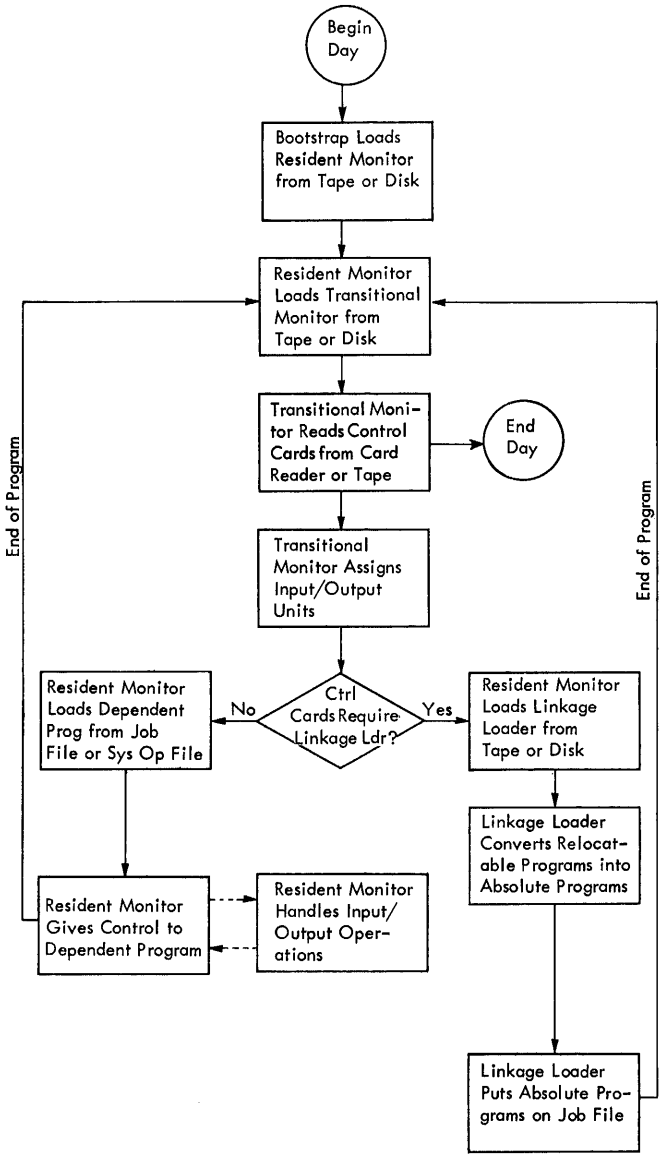


Figure 1. Operation of the System Monitor

Programs to be tested can be run under control of the System Monitor in the same way as fully tested programs. One or more Utility programs of the Operating System can be run at selected points during program execution and/or at the end of program to dump core storage, tapes, or disk.

Use of the System Monitor

Five types of control information are supplied by the installation to the System Monitor:

1. *Monitor control cards.* Monitor control cards are identified by MONSS in columns 6 through 10. These cards initiate program execution, assign input/output units to programs, and supply certain information to the System Monitor. They are described in the section "Use of Transitional Monitor."

2. *Linkage Loader control cards.* These cards select the elements composing a program and control program relocation. They are described in the section "Use of Linkage Loader."

3. *Imbedded calls.* These are requests coded in a dependent program to incorporate elements into the program. They are described in the section "Use of Linkage Loader."

4. *Linkage sequences.* These are coding sequences within a dependent program that enable the program to use facilities of the Resident Monitor. Each sequence contains a branch instruction and may also contain one or more fields of control information. The branch instruction gives control to a routine in the Resident Monitor. These sequences are described in the section "Use of Resident Monitor Functions."

5. *Console inquiries.* These are control messages typed on the console printer by the operator. They are described in the section "Messages and Inquiries."

Batch Processing

A major advantage of the System Monitor is that it permits batching programs, that is, running programs in groups, with minimum operator intervention. Program batches are subdivided into jobs and runs.

The day's work is divided into *jobs.* Each job consists of one or more program executions, called *runs,* plus control information. All programs in a job must have compatible input/output assignments.

The usual job is a request for a program in absolute format, residing on tape or disk, plus the necessary control information and input data.

However, almost limitless combinations of runs can be performed in a single job. For example, a job can include a compilation, relocation of all programs in the job, execution of programs to test them, use of one or more of the Utility Programs, and execution of one or more fully tested production programs.

Programs in source language may just be compiled, or may be compiled and executed (compile-and-go operation) within one job. However, if errors are found in compilation, the request for execution of the compiled program will be automatically canceled.

Program Construction

For the Operating System, a program is constructed of one or more phases. Many programs will consist of only one phase. If a program is too large for storage, it must consist of several phases; the phases are executed in succession, with each phase partly or completely overlaying the preceding phase.

Each phase consists of one or more subprograms. A subprogram is the basic program unit with which the Linkage Loader performs its processing.

For the purposes of the Linkage Loader, a subprogram begins with a TITLE card and ends with a Termination card. The cards between the TITLE and Termination cards contain the relocatable statements produced by a language processor. (See the section "Use of Linkage Loader" for a description of the TITLE card. Termination cards are described in the section "Program Patching.")

A subprogram can be an entire, self-contained program, or it can be a closed subroutine to be executed in conjunction with other subprograms to form a phase.

The subprograms in a job may be in source language, requiring compilation, or in relocatable form. If a job contains more than one subprogram, it may contain a mixture of source and relocatable decks. In a job, a relocatable subprogram may accompany the control cards, or it may reside in a program library on tape or disk. In the latter case, the control cards must include a call for the subprogram, or another subprogram in the job must contain an imbedded call.

Subprograms are divided into two types: primary and secondary. A primary subprogram has an initial entry point specified by its Termination card. A primary subprogram may be a phase in itself or may form the nucleus for a group of subprograms to be made into a single, executable phase. A secondary subprogram does not have an initial entry point on its Termination card. Secondary subprograms may compose a complete phase or may be combined with a primary subprogram to form a phase.

In Autocoder, for example, the distinction between primary and secondary subprograms is based on the END or SPEND statements. If the programmer specifies an address in one of these statements, that address is used for the initial entry point of the subprogram, and the language processor informs the Linkage Loader that this is a primary subprogram. If the operand of the END or SPEND statement is blank, the language processor informs the Linkage Loader that this is a secondary subprogram.

A program must contain at least one primary subprogram to establish the initial entry point for exe-

cution of the program. For a multiphase program, the first phase must include at least one primary subprogram. Succeeding phases may or may not contain primary subprograms. If the succeeding phase is of an executable nature, the preceding phase must contain the linkage to initiate its execution.

The initial entry point of a program phase is established by the Termination card of the first primary subprogram processed for that phase. The Linkage Loader ignores entry points specified in other Termination cards processed within that phase.

Communication Symbols

The System Monitor enables use of symbolic references, called *communication symbols,* so that a subprogram can refer to a routine in another separately compiled subprogram and to parts of the Resident Monitor.

The two types of communication symbols are:

1. *Linkage symbols,* which are used for communication between subprograms
2. *System symbols,* which are used for communication from a subprogram to routines or data fields of the Resident Monitor

Autocoder includes language statements that directly relate to the use of communication symbols. For the most part, COBOL and FORTRAN automatically generate these symbols as they are needed in a program.

Linkage Symbols

Linkage symbols are established by two types of cards: the TITLE card at the beginning of a subprogram, and Definition cards included in a subprogram. (Autocoder includes the DEFIN statement for the creation of Definition cards.)

Each time the Linkage Loader encounters one of these cards, it places into its symbol table the characters that have been declared as the linkage symbol and the relocated address assigned by the Linkage Loader to the symbol. For example, a TITLE card declares a subprogram name as a linkage symbol, and the origin point of the subprogram (after relocation) becomes the address assigned to that linkage symbol. This procedure enables a subsequent subprogram to refer to a previously processed subprogram by the linkage symbol that is the previous subprogram's name.

A linkage symbol can appear in a subprogram in either of two formats:

1. A symbol consisting of one to ten characters, such as a subprogram name. The first character must be alphabetic; the remaining characters can be alphabetic or numeric. No special characters can be used. If the symbol is less than ten characters, it is left-justified, with trailing blanks in a ten-position field.
2. A five-character symbol consisting of four characters followed by a slash (ABCD/). The first character must be alphabetic; the next three characters can be alphabetic or numeric.

Reference to linkage symbols of the first format are made by DCWS and DCWF cards. COBOL and FORTRAN automatically generate these cards as needed. Autocoder includes language statements for the generation of these cards. When the Linkage Loader encounters a DCWS card, it puts into the subprogram a branch instruction containing the address assigned to the linkage symbol in the DCWS card. A DCWF card causes the Linkage Loader to put into the subprogram a five-character constant that is the address equated to the linkage symbol.

NOTE: DCWS and DCWF cards can contain only one- to ten-character linkage symbols.

DCWS and DCWF cards are termed *imbedded* calls because they are requests for subprograms coded into a subprogram.

References to linkage symbols of the ABCD/ format can be made by using the symbol in the A-field or B-field of an instruction or by using the symbol as an address constant. For example, in the Autocoder language, the programmer can use the DEFIN statement to declare WORK/ as a linkage symbol that represents the address of a work area. The programmer can then use WORK/ in the operand of an instruction that refers to that work area, such as S +10, WORK/. Such an instruction can be included in any other subprogram processed by the Linkage Loader during the same run as the subprogram that declared the linkage symbol.

Note that linkage symbols are designed for use as references between separately compiled subprograms. Each of the language processors provides facilities for references within a subprogram.

System Symbols

System symbols are defined by the System Monitor and require no declaration by subprograms. Subprograms need only refer to them. These symbols, given in the section "Use of Resident Monitor Functions," refer to routines and data in the Resident Monitor.

All system symbols have the format /ABC/. These symbols can appear in the operand of any instruction or as an address constant. The Linkage Loader converts all system symbols to actual addresses when it relocates the dependent programs.

Listing of Communication Symbols

The Linkage Loader produces a listing of all linkage symbols used during the processing of subprograms. Each symbol is listed with the core-storage address it represents. Thus, the listing shows the location of program elements after they have been relocated.

For all system and linkage symbols that are not defined, the Linkage Loader will:

1. Print all unresolved symbols. This list includes undefined communication symbols and all unfilled calls for subprograms.
2. Leave undefined communication symbols in the instruction in which they appeared.

The Linkage Loader will permit programs with undefined communication symbols or unresolved calls to be executed; however, these programs will fail if execution is attempted of an instruction containing an unresolved call or undefined communication symbol.

Structure and Functions of the System Monitor

Structure of the System Monitor

The System Monitor consists of three major elements:

1. *Resident Monitor.* This element contains the control routines required while running dependent programs and, therefore, remains in core storage at all times.

2. *Transitional Monitor.* This element performs functions related to the transition from run to run and job to job. It is brought into storage by the Resident Monitor when transitional functions are needed. One of its major functions is the analysis and processing of Monitor control cards.

3. *Linkage Loader.* This element performs the functions required to prepare relocatable programs for execution. It is brought into storage by the Resident Monitor when requested by Monitor control cards.

Functions of Resident Monitor

Input/Output Control System

The Input/Output Control System (IOCS) is created during System Generation to meet the requirements of the installation. This IOCS, termed the *Resident IOCS,* is in the Resident Monitor. It contains the basic routines, such as those for error checking, that perform functions common to all input/output operations. (For detailed information concerning the Resident IOCS, see the publication *Basic Input/Output Control System.*)

SPOOL Routines

The Resident Monitor contains facilities for running card-to-tape, tape-to-card, and tape-to-printer operations concurrently with dependent programs. This type of operation is called Simultaneous Peripheral Operation On Line (SPOOL).

Unit-Record Routines

The Resident Monitor contains three routines to handle input/output operations for unit-record equipment. One of these routines is for card input (Read routine), another for card output (Punch routine), and the third for printer output (Print routine). Each of these routines has an alternate routine that performs the unit-record functions with magnetic tape. For each unit-record function, the installation can specify whether the function is to be performed with magnetic tape or unit-record equipment.

Assignment Routine

Facilities for assigning input/output units for program use are included in the Resident Monitor and the Transitional Monitor. The Resident Monitor contains tables of information about the installation's input/output equipment, and facilities for providing this information as required during program execution. The Transitional Monitor analyzes control card information related to input/output assignment and coordinates this information with the Resident Monitor's Assignment routine.

End-of-Program Routine

The End-of-Program routine analyzes each end of program to determine the function to be performed next. If it is a normal end of program (successful completion of the program being executed), the Transitional Monitor is brought into storage to process the Monitor control cards that define the next run.

If the end of program is caused by program failure (unusual end of program), the End-of-Program routine preserves the contents of core storage on tape, if a tape was provided for that purpose. The Transitional Monitor then processes the Monitor control cards for the next run, if the job or program is being tested, or skips to the control cards for the next job.

A third type of end program, special end of program, is used by the Linkage Loader and the language processors to indicate that the contents of their output files are not valid.

Load Routine

The Load routine brings programs, which have been relocated and are ready for execution, into storage. These absolute-format programs are on the Job file produced by the Linkage Loader, or on the System Operating File. The Load routine loads from the file named in the Monitor control card that executes the program. (The files are discussed later in this section.)

The Load routine also searches for phases required during the execution of a job. For example, this routine is used by one phase of a multiphase program to locate the next phase and bring it into storage. Each program phase, at its completion, specifies to the Load routine which phase is to be located next. Note that the *next program* phase need *not* be the *next in numerical sequence.*

For example, Phase 1 of a program could issue a request for Phase 2 or any other phase, basing its choice on the result of the processing in Phase 1. Dependent programs may also use the subroutines of the Load routine to read, search, backspace, and rewind the file from which the program was loaded.

NOTE: The Load routine cannot load the two highest core-storage positions. The Operating System places group marks with word marks in the two highest core-storage positions; these group marks with word marks must not be destroyed by dependent programs.

Communication Region

The Communication Region is a collection of fields that contain control information. Some fields contain information used for communication among various IBM programs within the Operating System; others contain information of value to the user's programs. Some fields contain data; other fields contain addresses.

For example, one Communication Region field contains the current date. This field can be addressed by any program. The Resident IOCS uses this field to determine the date for writing and checking tape labels. The user's programs can address this field for such purposes as creating a header record for printer output.

Communication Region fields can be modified only through the Resident Monitor's Modification routine, which is described in the section "Use of Resident Monitor Functions."

Wait-Loop Routine

The Wait-Loop routine allows programs to suspend processing until the operator performs a specified function, such as changing forms on the printer. A program can write a message on the console printer, branch to the Wait-Loop routine, and resume processing after the operator signals the Resident Monitor that the function specified by the message has been completed.

Console Inquiry Routine

Facilities for responding to console inquiries are included in both the Resident Monitor and the Transitional Monitor. The Resident Monitor accepts console inquiries during a run, such as a signal from the operator to exit from the Wait-Loop routine. The Transitional Monitor accepts console inquiries between jobs, such as a signal to begin reading from the Alternate Input Unit.

The Console Inquiry routines (in both the Resident and Transitional Monitors) serve to communicate control information to the System Monitor. Programs operating under control of the System Monitor can also use the console printer for communication with the operator.

Initialization Routine

The Initialization routine performs housekeeping to prepare the System Monitor for work. This routine is at the top of the Resident Monitor area and is overlaid by the first dependent program brought into storage. The functions and use of the Initialization routine are explained in the publication *Operator's Guide.*

Functions of the Transitional Monitor

Control Card Interpretation Routine

When the Transitional Monitor is brought into storage, one of its major functions is the interpretation of the Monitor control cards that define the next run or job. The Control Card Interpretation routine analyzes the Monitor control cards and, in accordance with their specifications, gives control to the System Monitor elements required to prepare and execute the next run.

Job Routine

The Job routine prepares the Resident Monitor for the next job. This preparation includes resetting various program switches, clearing certain areas in the Communication Region, and canceling certain input/output unit assignments.

Functions of the Linkage Loader

The Linkage Loader converts relocatable subprograms into absolute format and, in the process, resolves communication symbols into machine addresses.

Each Linkage Loader run produces a Job file, which contains program(s) in absolute format. Each program consists of all the subprograms to be executed as one program. A program on the Job file is brought into core storage by the Resident Monitor's Load routine when a Monitor control card requests execution of that program.

The Linkage Loader CHAIN feature permits the Linkage Loader to construct multiphase programs from programs compiled by the FORTRAN and COBOL language processors, as well as by the Autocoder processor.

Input to the Linkage Loader is from the Standard Input Unit, the Go file, and/or the System Library file. (These input/output files are discussed in the following topic, "Input/Output Files.") When several subprograms are to be combined into a single program, the subprograms can be supplied to the Linkage Loader from any combination of the three input files, or they can all be from the same file.

Selection of the subprograms to be converted into a single program is determined by control cards or by imbedded calls in the subprograms being processed.

The Linkage Loader has the ability to incorporate patches or Snapshot requests into a previously compiled subprogram.

Input/Output Files

The input/output files for the Operating System and the dependent programs running under it are divided into five categories:

1. *System Files.* These input/output files, described below, are used by the System Monitor.
2. *Work Files.* These tape, disk, or unit-record files are used by Operating System programs and can be used by installation programs. The symbolic names for these files are:
   MW0 (MW-ZERO) through MW9
   MWA through MWZ
3. *Reserve Files.* These tape, disk, or unit-record files are reserved for use by installation programs. The symbolic names for these files are:
   MR0 through MR9
   MRA through MRZ
4. *Tele-processing System Files.* These tape, disk, or unit-record files are used by dependent programs that operate under control of the Tele-processing Supervisor. The symbolic names for these files are:
   MT0 through MT9
   MTA through MTZ
5. *SPOOL Files.* These unit-record and tape files are used by card-to-tape, tape-to-card, and tape-to-printer programs that run concurrently with, but independently of, other programs. The symbolic names for these files are:
   SU1 and SU2 (unit-record files)
   ST1 and ST2 (tape files)

System Files

System Operating File (SOF)

The System Operating File contains Operating System programs in absolute format. This file contains the System Monitor and can contain the language processors, the Utility programs, and the Sort Definition program.

The System Operating File is created at System Generation and is constructed according to the program requirements of each installation. When the file is created, the user may add user-written programs, such as an inventory or payroll program. The System Operating File may be on tape or disk.

The first element of the System Operating File is the Bootstrap routine, which initially loads the Resident Monitor into core storage. The following elements are IBM-supplied and installation components of the Operating System.

System Library File (LIB)

An installation can have any number of libraries of relocatable programs, but only one is designated as the System Library file for a particular run of the Linkage Loader.

A library may be on tape or disk. If it is on tape, it can be on the same tape as the System Operating File or it can be on a separate reel. The System Operating File tape can contain several libraries. If a library is on a separate tape, this tape can contain only one library.

Libraries are created through use of the System Generation maintenance programs. (See the *System Generation* publication.)

Standard Input Unit (SIU)

The Standard Input Unit contains the file of control information for the System Monitor. Source programs for the language processors, relocatable programs to be processed by the Linkage Loader, and data and control cards for dependent programs may also be placed in the Standard Input Unit. The Standard Input Unit can be a card reader or a magnetic tape unit. If it is a magnetic tape unit, the tape must be written in odd parity.

NOTE: The term *card* means one 80-character input record. This record can be read either from an actual card, in installations that use a card reader for the Standard Input Unit, or it can be a tape record of 80 characters. Tape files consisting of this type of record are said to be in *card-image format.*

Alternate Input Unit (AIU)

At the option of each installation, capability for using the Alternate Input Unit may be specified at System Generation. Control information can then be submitted to the System Monitor through either the Alternate or the Standard Input Unit. This configuration permits a high-priority job in the Alternate Input Unit to break into the previously established sequence of jobs in the Standard Input Unit.

The Alternate Input Unit can be a card reader or a magnetic tape unit; it may be the same type of unit as the Standard Input Unit or it may be different. The type of unit for the Alternate Input Unit can be changed by a Monitor control card.

Requests for changing from the Standard Input Unit to the Alternate Input Unit can be made between jobs through a console inquiry.

Standard Print Unit (SPR)

The Standard Print Unit is used for all printer output from IBM programs within the Operating System. Diagnostic messages for the programmer are written

on this unit. It can also be used by the installation's programs.

The Standard Print Unit can be either the IBM 1403 Printer or a magnetic tape unit. If it is a tape unit, it can be the same unit as the Standard Punch Unit, which is described below. If the user specifies the variable print and punch modules at System Generation, the Standard Print Unit can be on the same tape unit as the Core Image file, the Temporary Storage file, and the Standard Punch Unit. (See "System File Sharing and Device Switching Feature.")

The Standard Print Unit can be omitted from a system; however, omission means that the programmer will not receive diagnostic messages or language processor listings.

Standard Punch Unit (SPU)

The Standard Punch Unit is used for all punched-card output from IBM programs within the Operating System. It is also available for use by the installation's programs.

The Standard Punch Unit may be either the IBM 1402 Card Read Punch or a magnetic tape unit. If it is a tape unit, it can be the same unit as the Standard Print Unit. If the variable print and punch modules are selected, the Standard Punch Unit can be on the same tape unit as the Core Image file, the Temporary Storage file, and the Standard Print Unit. This unit can be omitted from a system.

When the Standard Punch and/or Standard Print Units are tape, the POW program can be used on an IBM 1401 to print and/or punch the tape records. When use of the POW program is specified at System Generation, the Transitional Monitor writes the POW program as the first record on the tape when the tape is opened, either during initialization of the system or at the direction of the operator. (The POW program is described in the *Operator's Guide* publication.)

Job File (MJB)

The Job file is the output file of the Linkage Loader. It contains programs in absolute format. Job file programs are loaded into storage by the Resident Monitor's Load routine.

The Job file must be assigned to the same type of device (tape or disk) as the System Operating File.

Go File (MGO)

The Go file contains relocatable programs compiled by the language processors. The Go file may contain one or more relocatable subprograms that were output from one or more language processor runs in the same job. The Go file serves as one of the input sources for the Linkage Loader.

The Go file may be on tape or disk, as specified for each language processor during System Generation.

Core Image File (MDM)

The Core Image file is used by the Resident Monitor to record the status of core storage at particular times during processing. This file is used by the Resident Monitor's End-of-Program routine each time an unusual end of program occurs. (The records written by the End-of-Program routine can later be used as input to the Utility program's Storage Print program, which edits and writes them on the Standard Print Unit.)

The Core Image file serves programs that use the IOCS macro-instruction for writing checkpoint records. Checkpoint records can be used to restart a program that was temporarily interrupted.

The Core Image file can also be used by dependent programs to log read errors. (See the *Basic Input/Output Control System* publication.)

Use of the Core Image file is optional, but if it is included in an installation, it must be assigned to a tape unit. If the variable print and punch modules are selected, the Core Image file can be on the same tape unit as the Standard Print Unit, Standard Punch Unit, and the Temporary Storage file.

TP Library (MLT)

The TP Library file consists of the TP programs and a TP Directory; the latter speeds the locating and loading of the TP programs.

The TP Library file may be on either disk or tape. However, if the file is on tape, it must be the only file on that tape.

Temporary Storage File (MDT)

The Temporary Storage file is used if there is insufficient core storage available for a TP program. If a TP program extends beyond the area reserved for it, the Tele-processing Supervisor (provided that requisite modules have been included at System Generation time) unloads the main-line program onto the Temporary Storage file, to make the required area available. When it is time to return control to the main-line program, the Supervisor reloads the program from the Temporary Storage file.

If the variable print and punch modules are selected, the Temporary Storage file can be on the same tape unit as the Standard Print Unit, the Standard Punch Unit, and the Core Image file; however, if the variable modules are not selected, the Temporary Storage file cannot share a tape unit.

Use of Linkage Loader

The output from each of the language processors is in relocatable format to permit subprograms to be assigned to any available area of core storage. The Linkage Loader assigns a subprogram to a particular area of storage in accordance with three factors:

1. The size of the installation's Resident Monitor
2. The size and relative location of any other subprograms to be loaded at the same time
3. Control information specifying predetermined storage assignments

Subprograms are *compiled* with address assignments relative to *Absolute Zero* (machine location 00000). That is, program instructions are compiled by the Autocoder and COBOL language processors, from location 00000 upward or, by the FORTRAN processor, from location 00001 upward. The Autocoder processor can also compile from the origin point specified by an ORG card.

Subprograms are *relocated* by the Linkage Loader relative to an address in core storage called *Base Zero.* Base Zero is the address of the location immediately above the last location occupied by the installation's Resident Monitor minus its Initialization routine. (For installations that permanently reserve an area above the Resident Monitor for programs within a Teleprocessing system, Base Zero is the location immediately above that reserved area.) Base Zero is, therefore, the lowest location available to dependent programs. Figure 2 illustrates the relationship of Absolute Zero and Base Zero.

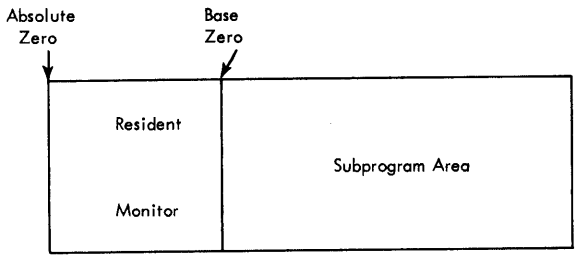


Figure 2. Relationship of Base Zero to Absolute Zero

When a subprogram is processed by the Linkage Loader, the subprogram's origin point is relocated in accordance with the installation's Base Zero. If, for example, the installation's Resident Monitor ended at location 10000, the Linkage Loader assigns an actual origin point of 10001 to the first subprogram it processes. Other addresses contained in the subprogram are relocated accordingly.

This type of relocation is termed *upward relocation.* Upward relocation is applied to the load addresses assigned by the language processor to locate instructions and areas in core storage and to the reference addresses in the A-fields and B-fields of the subprogram's instructions. Thus, the subprogram can be assigned to any area of core storage, yet interrelationship of the subprogram's elements is maintained.

NOTE: The Linkage Loader will not permit a dependent program to overlay the Resident Monitor. If a load address, after relocation, is below Base Zero, the Linkage Loader will stop processing the program and will return control to the Resident Monitor. The Resident Monitor will bring in the Transitional Monitor, which will resume processing with the next job.

Upward relocation is the most common type of relocation used for a subprogram. However, the Linkage Loader performs two other types of relocation within subprograms.

One of these is *downward relocation.* Where separately compiled subprograms are to share data areas, common data areas are used. These areas are located at the top of storage or at addresses specified by the programmer. Normally the specified address should be near the top of storage. The common data areas are addressed at the high-core-storage position. When assigning addresses to common data areas, the language processors arbitrarily designate 99999 as the top of storage. The Linkage Loader then relocates the addresses downward to the actual top of storage or to the specified location.

The third type of relocation is *NO relocation.* The Linkage Loader does not relocate certain fixed addresses, such as index register addresses.

The value used by the Linkage Loader to perform upward relocation is called the upward relocation factor. For the first subprogram in core storage, the relocation factor is equal to Base Zero; for succeeding subprograms in the same phase, the relocation factor is equal to Base Zero plus the size of all preceding subprograms in that phase.

The subprograms that form a given phase can be in any order in core storage; a primary subprogram need not be the first.

NOTE: During compilation, a character must be placed in the highest location of a subprogram. (That is, a subprogram cannot end with an area into which nothing is loaded, such as an area defined by an Autocoder DA statement that does not set a group mark with work mark in its highest position.) Some character is necessary in the subprogram's highest location to enable the Linkage Loader to properly relocate any following subprograms.

This requirement is automatically satisfied by the COBOL and FORTRAN language processors. The requirement can be met for an Autocoder compilation by a statement that defines a one-character constant in the subprogram's highest location.

Control Cards for the Linkage Loader

Control cards for the Linkage Loader direct processing during one execution of the Linkage Loader, rather than directing the general flow of processing during a batch. Linkage Loader control cards are not Monitor control cards; that is, they do not contain the identification MON$$.

The Monitor control card that initiates execution of the Linkage Loader must immediately precede the control cards for the Linkage Loader. The Linkage Loader determines that it has come to the end of its control cards when it finds a card that is not in a format it recognizes. This card must be a Monitor control card.

Figure 3 lists the Linkage Loader control cards, summarizes the important information about them, and shows their formats.

CALL Card

The operand of the CALL card is the name of the subprogram to be processed by the Linkage Loader. This card is used to request a subprogram from the System Library file or the Go file. A subprogram in the Standard Input Unit does not require a CALL card; its TITLE card serves the same function. When the Linkage Loader encounters a CALL card, it adds the specified name to a list of requests. When the Linkage Loader is ready to begin processing the subprograms on this request list, it searches for them first in the Go file and then in the System Library file.

This search is performed by reading the names of the programs on the Go file and the System Library file and comparing them with the names on the request list. This technique minimizes the time required for searching.

Because the Linkage Loader uses this technique to minimize search time, the final sequence of subprograms in core storage is not necessarily the same as the CALL cards that requested the subprograms. (The CALLN and CALLP cards, described later, can be used to control the relative positioning of subprograms in core storage.)

For example, assume that the following subprograms are arranged sequentially on the System Library file: SP1, SP2, SP3, SP4, and SP5. Assume also that CALL cards were read from the Standard Input Unit in the order: CALL SP3, CALL SP1, CALL SP4. The Linkage Loader begins its search of the System Library file and locates the TITLE card for SP1. It checks this name against the names on the request list and finds that there is a request for SP1; therefore, it processes SP1 and places it on the Job file. The Linkage Loader then searches forward in the System Library file and locates the TITLE card for SP2. It checks the request list, finds no request for SP2, and continues the search. The TITLE card for SP3 is found in the Library and checked against the list. The Linkage Loader finds a request for SP3 and, therefore, processes it and places it on the Job file behind SP1. This procedure continues until all the subprograms on the request list have been located, processed, and placed on the Job file.

The final sequence of subprograms would be SP1, SP3, SP4 — the order in which they were located on the Library. (The relative location of subprograms in storage is referred to as the *memory map.*) The relocation factor for the three subprograms was adjusted as follows: SP1 was given an origin point of Base Zero, and all upward relocation for SP1 was performed with a relocation factor equal to Base Zero; SP3 was given an origin point of Base Zero plus the size of SP1, and all upward relocation for SP3 was performed with a relocation factor equal to Base Zero plus the size of SP1; SP4 was given an origin point of Base Zero plus the combined size of SP1 and SP3, and all upward relocation for SP4 was performed with a relocation factor equal to Base Zero plus the combined size of SP1 and SP3.

Imbedded Calls

In addition to requests for subprograms specified by CALL cards, Autocoder language DCWS, DCWF, and CALL cards are requests for subprograms. Because these cards are within subprograms, this type of request is termed an *imbedded call.*

When the Linkage Loader encounters an imbedded call, it checks its symbol table for the name of the called subprogram. If it finds the name, the subprogram has already been processed, since its TITLE card established the name as a linkage symbol. Then the Linkage Loader replaces the imbedded call, for a DCWS card, with a branch to the address assigned to

Location in Control Card Deck	Format			Functions	Remarks
	6 15	16 Card Type 20	21 Operand(s)		
Beginning of first phase in program		PHASE	PROGMNAME1	Terminates the previous phase, begins a phase, causes all previous calls to be processed. Defines PROGMNAME1 as the name of the program.	Erases linkage symbols from Linkage Loader symbol table. Resets the downward relocation factor (see BASE2 card) to the top of core storage minus two. Columns 60 through 70 must be blank.
Beginning of succeeding phase		PHASE			
Beginning of succeeding phase (user-specified phase number)	005	PHASE			User specification of succeeding phase number, which must be in ascending order, need not be sequential, and may not be blanks or 999. Columns 60 through 70 must be blank.
Immediately before BASE1 or PHASE card; anywhere within a subprogram except between TITLE and BASE1 cards		PRTCT PRTCT PRTCT PRTCT	LINKSYMBOL ABCD/ 15000	Keeps Linkage Loader from erasing linkage symbols with addresses equal to or higher than operand. Blank-operand card removes the protection.	Protection is changed by another PRTCT card and is effective only for the current Linkage Loader run.
Immediately after TITLE card (if subprogram is on System Library file and is to be relocated to same point each time) or immediately after PHASE card or PHASE and PRTCT cards		BASE1 BASE1 BASE1	LINKSYMBOL ABCD/ 15000	Replaces current relocation factor with address for the linkage symbol or with absolute address in operand.	Erases from Linkage Loader symbol table linkage symbols with a value equal to or higher than the BASE1 value but less than PRTCT value.
		BASE1	*+X00	Sets current relocation factor to next higher address that is a multiple of 100.	
		BASE1		Sets relocation factor to Base Zero.	Cannot be used in System Generation mode of operation.
Within control cards for a phase or within a subprogram		CALL	SUBPGNAME2	Calls subprogram from System Library or Go file when all calls are collected.	
Within control cards for a phase		CALLN	SUBPGNAME2	Causes all previous calls to be processed, then brings in called subprogram from System Library or Go file.	
Within control cards for a phase		CALLP	SUBPGNAME2	Causes all previous calls to be processed, then brings in called subprogram from System Library or Go file and adds patches to called subprogram.	Patch cards must immediately follow CALLP card. (See "Program Patching" section for patch card formats.)
Before subprogram affected is processed or, within a subprogram, after TITLE card, but before any cards referring to common data area		BASE2 BASE2 BASE2	LINKSYMBOL ABCD/ 15000	Sets upper limit of common data area to address for the linkage symbol or to absolute address in operand.	This downward relocation factor applies to all succeeding subprograms during the current Linkage Loader run or until the next BASE2 card. A PHASE card with a name in the operand resets the downward relocation factor to the top of core storage minus two.
		BASE2		Sets upper limit of common data area to top of core.	
First card of subprogram		TITLE	SUBPRGNAME xxxxx yyyyy NOTE: Leave five blanks between xxxxx and yyyyy.	Begins a subprogram. Defines SUBPRGNAME as linkage symbol. xxxxx=Origin for subprogram expressed as an absolute address. yyyyy=Size of common data area Autocoder subprogram will use.	NOTE: This TITLE card format is in the relocatable object deck produced by the language processors and may contain additional operands not used by the Linkage Loader; it is not the same format as the Autocoder symbolic TITLE card.

Figure 3. Control Cards for Linkage Loader

Location in Control Card Deck	Format			Functions	Remarks
	6 15	16 Card 20 Type	21 Operand(s)		
Within a subprogram	ABCD/ LINKSYMBOL LINKSYMBOL LINKSYMBL1 ABCD/	DEFIN DEFIN DEFIN DEFIN DEFIN	LINKSYMBOL 15000 ABCD/ LINKSYMBL2 WXYZ/	Defines the linkage symbol in columns 6 through 15 for use by the Linkage Loader as an entry point or data field within a subprogram.	Can be coded directly in the Autocoder language. See the Autocoder publication for full details.
After subprogram requested but before the Linkage Loader symbol table is erased and before the end of the phase	IDENT	SNAP	SUBPGNAME2 xxxxxyyyyy zzzzzyyyyy zzzzzss NOTE 1: Do not put commas between operands. NOTE 2: ss must be in columns 56 and 57. Therefore, if only one area is to be snapped, leave blanks for the second yyyyyzzzzz.	Requests Snapshots of storage during execution of the subprogram. Causes all previous calls to be processed. IDENT is any one to five characters to identify the Snapshot listing. xxxxx = Relocatable address of instruction before which the Snapshot is to be taken. yyyyy = Relocatable address of lower limit of area to be snapped. zzzzz = Relocatable address of upper limit of area to be snapped. ss = Length of instruction at xxxxx; if the instruction length is less than ten, it should be written with a zero in column 56.	Cannot be used in a compile-and-go operation. Instruction before which Snapshot is to be taken must be at least seven characters long, cannot be chained, and cannot be in an IOCS macro-instruction. NOTE: When relocating the addresses on the SNAP card, the Linkage Loader will add the value for the subprogram name linkage symbol to the relocatable addresses in the SNAP card. Therefore, if the subprogram was origined at other than 00000, subtract the origin value from the compiled addresses to get the addresses to be used on the SNAP card.
Anywhere within control cards for Linkage Loader		INPUT INPUT	Mxx SIU	Directs Linkage Loader to read control cards from a unit other than the one being read.	Mxx is the name of a currently assigned symbolic unit, such as MR1. End of file on the substitute unit causes a return to the Standard Input Unit, as does an INPUT card with the operand SIU. (See the section "Assignment of Input/Output Units.")
Before CALLN, CALLP, PHASE, SNAP, or Monitor control cards or end of file on Standard or Alternate Input Unit		DISGO		Go file will not be searched for subprograms.	Effective during current Linkage Loader run or until next CONGO card.
Anywhere after DISGO card		CONGO		Cancels DISGO card function.	

PROGMNAME1 is the linkage symbol that is the program name defined by the PHASE card.
SUBPGNAME2 is the linkage symbol that is the subprogram name defined by the TITLE card.
LINKSYMBOL is any linkage symbol.

All three of these linkage symbols consist of one to ten characters that are left-justified, with trailing blanks in a ten-position field.

ABCD/ is any five-character linkage symbol.

Figure 3. Control Cards for Linkage Loader (Continued)

the linkage symbol or, for a DCWF card, with a five-character constant of that address.

If the called subprogram has not yet been processed, the Linkage Loader places the subprogram name on its request list. (The same request list is used for CALL card requests.) When the Linkage Loader later locates and processes the subprograms on this list, it supplies the previous imbedded calls with the branch addresses or constants.

NOTE 1: Autocoder provides the DEFIN statement for establishing linkage symbols to represent any point in a subprogram. (The linkage symbol established by the TITLE card of a subprogram represents the origin point of that subprogram.) Linkage symbols established by the DEFIN statement can also appear in DCWF and DCWS cards; these cards are treated by the Linkage Loader in the same manner as DCWF and DCWS cards that contain a subprogram name. However, note that a subprogram is located and processed only if it has been called by the name contained in its TITLE card. Therefore, if subprogram ANYNAME contains an entry point represented by the linkage symbol ABCDEFGHIJ, the card DCWS ABCDEFGHIJ cannot be completely processed unless a call has been made for ANYNAME, either by a CALL card or an imbedded call.

NOTE 2: The linkage symbols established within a particular phase of a multiphase program can be referred to by subprograms within the same phase and succeeding phases, but they cannot be referred to in preceding phases. For example, Phase 1 cannot contain a DCWS card with the linkage symbol WORK/ if that symbol is not established by a TITLE card or Definition card until Phase 2.

Organization of programs should be planned with special care regarding imbedded calls. Imbedded calls, if not ordered properly, could cause additional searches of the System Library file and, therefore, be time consuming.

The following rules apply to calling programs:

1. Rules of precedence for memory maps are:
   a. The first subprograms in core storage will be those read from the Standard Input Unit (in the order read).
   b. The second group of subprograms in core storage will normally be those read from the Go file (in the order read).
   c. The third group of subprograms in core storage will normally be those read from the System Library file (in the order read).
   d. If the subprograms in items b and/or c contain imbedded calls, then item c will be repeated until all calls are completed. The Go file is not searched again.
2. Imbedded calls are considered to be for subprograms in the System Library file. If the subprogram requested by an imbedded call is on the Go file, it must also be requested by a CALL card that is not imbedded in a subprogram.

   NOTE: The CALL card may be omitted if the requested subprogram follows the requesting subprogram on the Go file.
3. The Go file is reset to its beginning and searched if any subprogram names are on the request list when the Linkage Loader reads a CALLN, CALLP, PHASE, SNAP, or Monitor control card, or finds the end of file on the Standard or Alternate Input Unit.

   NOTE: The DISCO card can be used to prevent this search of the Go file.
4. Subprograms placed in the Standard or Alternate Input Unit do not require a CALL card; their TITLE cards serve this function.
5. If a subprogram on the Go file is to replace a subprogram with the same name on the System Library file and the subprogram is requested by an imbedded call, a CALL card should be used to guarantee loading the subprogram from the Go file.

Example 1

This example illustrates the procedure for directing the Linkage Loader to process a subprogram from the System Library file. The subprogram (SP1) constitutes a self-contained, single-phase program that has no imbedded calls for other subprograms.

Only one control card is necessary: CALL SP1. The Linkage Loader will locate SP1 on the Library, process it, and place it on the Job file.

Example 2

This example illustrates the procedure for directing the Linkage Loader to construct a single-phase program from three subprograms: SP1, SP2, and SP3. (One of the three must be a primary subprogram; the other two can be primary or secondary.) In this example, SP1 is on the System Library file, SP2 is in the Standard Input Unit, and SP3 is on the Go file. None of the subprograms contain imbedded calls.

The sequence of cards in the Standard Input Unit is as follows:

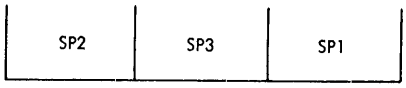
```
CALL    SP1
TITLE card and relocatable object deck for SP2
CALL    SP3
```

The Linkage Loader proceeds as follows:

1. CALL SP1: The name of SP1 is placed on the request list.
2. TITLE card and relocatable object deck for SP2: SP2 is processed and the name SP2 is added to the request list, with a flag indicating that SP2 has been processed.

3. CALL SP3: The name SP3 is added to the request list. The Linkage Loader resumes reading from the Standard Input Unit and finds no more Linkage Loader control cards. It begins searching the Go file, locates SP3, and processes it. Next, the Library is searched. SP1 is located and processed.

The memory map is shown below. The area between the outer vertical lines represents the area of core storage used for the subprograms processed.



Example 3

This example illustrates the construction of a single-phase program from three subprograms, one of which is processed because of an imbedded call. SP1 is on the System Library file and contains an imbedded call for SP3; SP2 is on the Go file.

The sequence of cards in the Standard Input Unit is as follows:

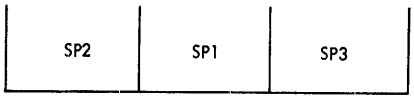
```
CALL    SP1
CALL    SP2
```

The Linkage Loader proceeds as follows:

1. CALL SP1: The name SP1 is placed on the request list.

2. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the Standard Input Unit and finds no more Linkage Loader control cards. It begins searching the Go file, locates SP2, and processes it. Next, the Library is searched. SP1 is located and processed. During the processing of SP1, the Linkage Loader finds the imbedded call for SP3. At that time, the name SP3 is placed on the request list. When the processing of SP1 is completed, the Linkage Loader checks the request list, finds the unsatisfied request, and resumes the search of the Library. SP3 is located and processed.

The memory map is as follows:



CALLN Card

The function of the CALLN (Call Now) card is essentially the same as the CALL card, except that the CALLN card directs the Linkage Loader to locate and process the specified subprogram immediately. This card can be used to control the memory map.

When the Linkage Loader reads a CALLN card, it does the following:

1. All subprograms specified by previous CALL cards are located and processed.

2. Any imbedded calls resulting from processing these subprograms are also processed (except imbedded calls for the subprogram named in the CALLN card).

3. Then the subprogram specified by the CALLN card is processed.

4. Any imbedded calls resulting from processing the subprogram specified by the CALLN card are added to the request list for subprograms (if the imbedded calls have not been satisfied by processing already performed).

The following rules apply to the use of the CALLN card:

1. Rules of precedence for memory maps:
   a. All subprograms called before the subprogram named by the CALLN card will be the first subprograms in core storage. (The relative order of those subprograms is determined in accordance with the rules of precedence for the CALLN card.)
   b. The subprogram named by the CALLN card will be placed in core storage following the subprograms specified in item a, above.

2. The name of a subprogram on the Standard Input Unit should never be the operand of a CALLN card. If the preceding calls should be processed before a subprogram on the Standard Input Unit, the CALLN card should be used to call the last preceding subprogram.

Example 1

This example illustrates the construction of a single-phase program from four subprograms that must be positioned in a certain order. Three subprograms are on the System Library file and the fourth is in the Standard Input Unit. None of them contain imbedded calls. The sequence of subprograms on the Library is SP1, SP2, SP3; SP3 must be positioned in storage after SP1 and before SP2 and SP4.

The sequence of cards in the Standard Input Unit is as follows:

```
CALL     SP1
CALLN    SP3
CALL     SP2
```

TITLE card and relocatable object deck for SP4

The Linkage Loader proceeds as follows:

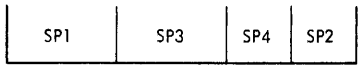
1. CALL SP1: The name SP1 is placed on the request list.

2. CALLN SP3: The name SP3 is stored, but not placed on the request list. The Linkage Loader begins searching for previously called subprograms. It locates SP1 on the Library and processes it. The Linkage Loader then determines that the request list has been satisfied. (SP1 was the only name on the list.) The name SP3 is moved to the request list. SP3 is then located and processed.

3. CALL SP2: The name SP2 is placed on the request list.

4. TITLE card and relocatable object deck for SP4: The Linkage Loader processes SP4. It resumes reading from the Standard Input Unit and finds no further Linkage Loader control cards. SP2 is then located and processed.

The memory map is as follows:



Example 2

This example illustrates the construction of a single-phase program from three subprograms, one of which contains an imbedded call for a subprogram that must be placed between the other two subprograms.

The three subprograms are on the System Library file in the order SP1, SP2, SP3. SP3 must be positioned after SP1 and before SP2; SP1 contains an imbedded call for SP3.

The sequence of cards in the Standard Input Unit is as follows:

```
CALL    SP1
CALLN   SP3
CALL    SP2
```

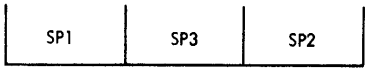
The Linkage Loader proceeds as follows:

1. CALL SP1: The name SP1 is placed on the request list.

2. CALLN SP3: The name SP3 is stored but not placed on the request list. The Linkage Loader begins searching for previously called subprograms. It locates SP1 on the Library and processes it. During the processing of SP1, the Linkage Loader finds the imbedded call for SP3. The Linkage Loader will supply the imbedded call in SP1 with the address of SP3 when the TITLE card of SP3 is processed. The Linkage Loader then determines that there are no more names on the request list, moves SP3 to the request list, locates SP3, and processes it.

3. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the Standard Input Unit and finds no more Linkage Loader control cards. SP2 is then located and processed.

The memory map is as follows:



CALLP Card

The CALLP (Call and Patch) card functions in the same manner as the CALLN card. The CALLP card also directs the Linkage Loader to incorporate patches into the specified subprogram. The patches, in relocatable machine-language form, must immediately follow the CALLP card in the Standard Input Unit. (See the section "Program Patching" for patch card formats.)

PHASE Card

PHASE cards are used to create a multiphase program and to assign a name to that program. Each time the Linkage Loader encounters a PHASE card it performs the following functions:

1. All previous calls (from both control cards and imbedded calls) are processed. The Linkage Loader gives the resulting phase a termination record, which indicates the entry point of that phase if it included a primary subprogram. (This information is used by the Load routine of the Resident Monitor to initiate execution of the phase after it has been loaded into core storage.)

2. The Linkage Loader creates a header record for the next phase, consisting of the name and number of the phase.

The first PHASE card for a multiphase program must specify the name of that program. Succeeding PHASE cards for the same program must have blank operand fields. For the first PHASE card, the Linkage Loader generates a header record, containing the name specified in the operand, and a phase number of 001. For succeeding PHASE cards of the same multiphase program, the Linkage Loader generates a header record, containing the name specified by the first PHASE card, and a phase number determined by incrementing a counter each time a PHASE card with a blank operand is encountered. Therefore, the first PHASE card with a blank operand indicates the beginning of the program's second phase, and the header record for this phase contains the phase number 002.

The programmer can specify the phase number, for any phases after the first phase, by putting a three-digit phase number (other than 999) in columns 6 through 8 of a blank-operand phase card. This phase number must be higher than the last phase number assigned; otherwise the Linkage Loader will ignore the user-assigned number and will assign the next sequential number to the phase. Following a user-assigned phase number, the next blank-operand phase card without a user-assigned number will be assigned a phase number one higher than the user-assigned phase number.

The following rules apply to the use of the PHASE card:

1. A PHASE card immediately precedes each group

of Call cards (CALL, CALLN, CALLP) that specify the subprograms to be included in that phase.

2. If the first PHASE card is omitted, the name of the first subprogram that is processed is used by the Linkage Loader as the name of the entire multiphase program. The header record of each phase will have this name. A single-phase program that consists of more than one subprogram should have a PHASE card if the program is relocated and executed in the same job. Because the subprograms are not necessarily *processed* in the order they are *called,* the PHASE card is needed to set the program name for the Monitor control card that executes the program.

3. A PHASE card with a program name in the operand field causes the Linkage Loader to reset the relocation factor to Base Zero and to erase from the Linkage Loader's symbol table any linkage symbols left from processing previous subprograms. (This erasure of symbols can be controlled by use of the PRTCT card, which is described later.)

4. A PHASE card with a blank operand field does not reset the relocation factor, nor does it cause the erasure of any symbols from the symbol table.

5. Columns 60 through 70 of the PHASE card must be left blank. (These columns are used by the Linkage Loader. Their contents are defined in the *System Generation* publication.)

Example

Because the PHASE card is most commonly used with the BASE1 card, examples illustrating PHASE card use follow the BASE1 card description.

BASE1 Card

The BASE1 card is used to control the relocation factor. This control of the relocation factor enables the programmer to direct the Linkage Loader to relocate a phase in such a manner that it will overlay the preceding phase at a predetermined point.

Four types of operands can be used in a BASE1 card:

1. *Linkage Symbol:* If a linkage symbol appears in the operand of a BASE1 card, the Linkage Loader sets the relocation factor to the address equated with that symbol in the Linkage Loader's symbol table. For this reason, the linkage symbol must be defined by a subprogram processed before the one that includes the BASE1 card. In most cases, the linkage symbol is declared by the TITLE card of a previous subprogram. Using a subprogram name as a BASE1 operand enables one phase of a program to be relocated to the origin point of a subprogram in a previous phase even though, at the time the BASE1 card was coded, the actual origin point of the previous subprogram had not yet been determined. When the phase that included the BASE1 card is loaded into core storage, it will overlay the previous phase, beginning at the origin point of the subprogram named in the operand of the BASE1 card.

NOTE: Because Autocoder provides the DEFIN statement defining linkage symbols other than the origin point of a subprogram, Autocoder users can construct phases that overlay preceding phases at locations other than the beginning of a subprogram.

2. *Actual Address:* If the operand of a BASE1 card contains an actual address, the Linkage Loader sets the relocation factor to that address. Care must be exercised in the use of this operand, because the final location of program elements that preceded a BASE1 card containing an actual address cannot always be predicted.

3. **+X00:* If the operand of a BASE1 card contains *+X00 (asterisk, plus sign, X, zero, zero), the Linkage Loader sets the current relocation factor to the next highest address that is a mutliple of 100. (If the relocation factor is already a multiple of 100, no adjustment is performed.) This operand is not necessarily related to a multiphase program. It is provided to control the relocation of subprograms that depend on certain areas being located at an even-hundred address. (For example, use of the Autocoder Clear Storage instruction can result in this requirement.)

4. *Blanks:* If the operand field of a BASE1 card is blank, the Linkage Loader sets the relocation factor to Base Zero *unless the preceding* MODE *control card specified* SG *mode.* In SG mode, a blank BASE1 operand causes relocation at *absolute* zero. This form of relocation should be used with extreme caution.

Each time the Linkage Loader encounters a BASE1 card, it performs the following functions:

1. It sets the relocation factor to the address value specified by the operand of the BASE1 card.

2. It erases from the symbol table all linkage symbols equated to addresses higher than the address value specified by the operand of the BASE1 card. (This erasure occurs only for BASE1 cards with a symbol or actual address in the operand. The *+X00 operand does not cause symbols to be erased.) The erasure of symbols can be controlled by the PRTCT card, which is described later.

The following rules apply to the use of the BASE1 card:

1. A BASE1 card, when placed in the Standard or Alternate Input Unit, should immediately follow a PHASE card. A BASE1 card will affect all subprograms that have not been processed, even if the calls for those subprograms preceded the BASE1 card. (A PRTCT

card can be inserted between the PHASE card and BASE1 card.)

2. If a BASE1 card appears in a subprogram, it must appear immediately after the TITLE card. This location is required for subprograms that are on the System Library file and are to be relocated to the same point each time.

3. The address value of a linkage symbol used as the operand of a BASE1 card must have been defined during processing of a previous subprogram.

Example 1

This example illustrates the construction of a two-phase program from two subprograms. SP1 constitutes the first phase of the multiphase program, and SP2 constitutes the second phase, which is to overlay the first completely. Both subprograms are on the System Library file. Neither of the subprograms contains imbedded calls.

The sequence of cards in the Standard Input Unit is as follows:

```
PHASE     SAM
CALL      SP1
PHASE
BASE1     SP1
CALL      SP2
```

The Linkage Loader proceeds as follows:

1. PHASE SAM: A header record is created. It contains the name SAM and the phase number 001.

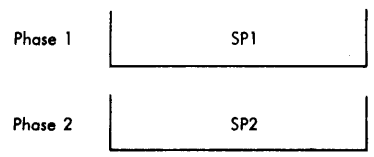
2. CALL SP1: The name SP1 is placed on the request list.

3. PHASE (blank operand): All previous calls are processed. (In this example, SP1 is the only previous call.) A termination record is generated for the first phase. A header record is generated for the second phase; it contains the name SAM and the phase number 002.

4. BASE1 SP1: The relocation factor, which is not affected by a blank-operand PHASE card, is set to the address value established by the TITLE card of SP1. (All linkage symbols with an address equal to or higher than the TITLE card address are erased from the symbol table.)

5. CALL SP2: The name SP2 is placed on the request list. The Linkage Loader resumes reading from the Standard Input Unit and finds no more Linkage Loader control cards. SP2 is located and processed.

The memory maps are as follows:



Example 2

This example illustrates the construction of a two-phase program from five subprograms. SP1 and SP2 constitute the first phase and SP3, SP4, and SP5 constitute the second phase, which is to overlay only SP2 of the first phase (leaving SP1 in storage). All the subprograms are on the System Library file. None of them contain imbedded calls.

The sequence of cards in the Standard Input Unit is as follows:

```
PHASE     MATILDA
CALL      SP1
CALLN     SP2
PHASE
BASE1     SP2
CALL      SP3
CALL      SP4
CALL      SP5
```

The Linkage Loader proceeds as follows:

1. PHASE MATILDA: A header record is created. It contains the name MATILDA and the phase number 001.

2. CALL SP1: The name SP1 is placed on the request list.

3. CALLN SP2: In accordance with the procedure for CALLN cards, SP1, then SP2, is located and processed.

4. PHASE (blank operand): A termination record is generated for the first phase. It contains the entry point specified by the Termination card of whichever subprogram was the primary one. A header record is generated for the second phase. It contains the name MATILDA and the phase number 002.

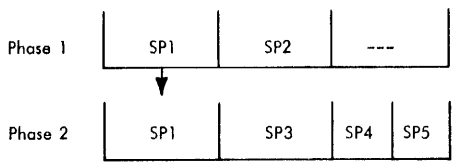
5. BASE1 SP2: The relocation factor is set to the address value established by the TITLE card of SP2. (All linkage symbols with an address value equal to or higher than the TITLE card address are erased from the symbol table.)

6. CALL SP3: The name SP3 is placed on the request list.

7. CALL SP4: The name SP4 is placed on the request list.

8. CALL SP5: The name SP5 is placed on the request list. The Linkage Loader resumes reading from the Standard Input Unit and finds no further Linkage Loader control cards. SP3, SP4, and SP5 are located on the Library and processed.

The memory maps are as follows:



Example 3

This example illustrates the construction of two separate multiphase programs during a single run of the Linkage Loader. All subprograms are on the System Library file, and none of them contain imbedded calls.

The sequence of cards in the Standard Input Unit is as follows:

```
PHASE     OLSEN
CALL      SP1
PHASE
BASE1     SP1
CALL      SP2
PHASE     JOHNSON
CALL      SP3
PHASE
BASE1     SP3
CALL      SP4
```

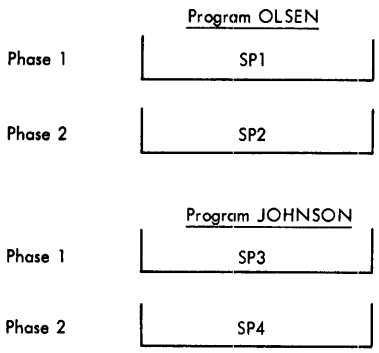
The Linkage Loader proceeds as follows:

1. PHASE OLSEN through CALL SP2: The Linkage Loader begins construction of a two-phase program in the manner illustrated by the preceding examples.

2. PHASE JOHNSON: The request list is checked for outstanding calls. The Linkage Loader finds the name SP2 on that list (from the CALL SP2 card), locates and processes SP2, and generates a termination record for the preceding phase. The relocation factor is then reset to Base Zero, and linkage symbols are erased from the symbol table. A header record is generated for the next phase. It contains the name JOHNSON and the phase number 001.

3. CALL SP3 through CALL SP4: The Linkage Loader constructs the second multiphase program.

The memory maps are as follows:



NOTE: The two programs are not necessarily related in any way. This example is designed to illustrate a means of directing the Linkage Loader to produce a Job file that is, in effect, a library of programs in absolute format.

PRTCT Card

The PRTCT card is used to limit the erasure of linkage symbols from the symbol table. The Linkage Loader will retain in its symbol table all linkage symbols with addresses equal to or higher than the address value specified by the operand of the PRTCT card. The operand of a PRTCT card can be either a linkage symbol or an absolute address. If System Generation (SG) mode has been specified, this protection will be retained until it is changed by another PRTCT card or by the initialization performed each time the Linkage Loader is brought into core storage for execution. If, however, System Generation mode has not been specified, the next PHASE card with a symbolic name in the operand will reset the area of protection to /AMS/; this has the effect of cancelling all protection.

The following rules apply to the use of the PRTCT card:

1. The PRTCT card must precede any control card that would cause erasure of symbols higher than the address value specified by the operand of the PRTCT card.

2. The address value of a linkage symbol used as the operand of a PRTCT card must have been defined during the processing of a previous subprogram in the same execution of the Linkage Loader.

Example

This example illustrates construction of a two-phase program in which one subprogram must be placed in upper storage and be used by other subprograms in both phases. All of the subprograms are on the System Library file, and none of them contain imbedded calls.

The sequence of cards in the Standard Input Unit is as follows:

```
PHASE     OTHELLO
CALL      SP1
CALLN     SP2
BASE1     30000
CALL      SP3
PHASE
PRTCT     30000
BASE1     SP2
CALL      SP4
```

The Linkage Loader proceeds as follows:

1. PHASE OTHELLO through CALLN SP2: SP1 and SP2 are processed as explained in previous examples.

2. BASE1 30000: The relocation factor is set to 30000. (Note that all previous calls have been satisfied because of the CALLN card. Note also that if SP2 had contained an imbedded call for a subprogram that

had not yet been processed, the new relocation factor would be applied to that subprogram.)

3. CALL SP3: The name SP3 is placed on the request list.

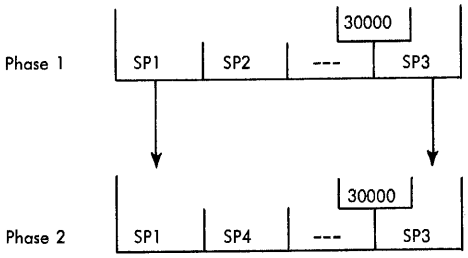
4. PHASE (blank operand): SP3 is located and processed. A termination record is generated for the first phase. A header record is generated for the second phase. It contains the name OTHELLO and the phase number 002.

5. PTRCT 30000: The Linkage Loader notes that linkage symbols equated to an address value of 30000 and higher are not to be erased from the symbol table. (The operand of the PRTCT card could also be SP3.)

6. BASE1 SP2: The relocation factor is reset to the address value established by the TITLE card of SP2. Linkage symbols equated to an address value higher than SP2 and lower than 30000 are erased from the symbol table.

7. CALL SP4: SP4 is placed on the request list. The Linkage Loader resumes reading from the Standard Input Unit and finds no more Linkage Loader control cards. SP4 is located and processed.

The memory maps are as follows:



BASE2 Card

The operand of the BASE2 card, which can be either a linkage symbol or an absolute address, sets the upper limit of a common data area.

Each time the Linkage Loader encounters a BASE2 card, it sets the downward relocation factor. This factor is then used for the adjustment of addresses that refer to a common data area.

The following rules apply to use of the BASE2 card:

1. The BASE2 card must be positioned in the Standard Input Unit so that it is read before the processing of any subprograms affected by it. The BASE2 card can also be placed within a subprogram, but before any references to a common data area.

2. More than one BASE2 card can be used, but rule 1 also applies to the additional cards.

3. If a linkage symbol is used for the operand of a BASE2 card, the address value of that symbol must have been defined during the processing of a previous subprogram.

4. The BASE2 limit applies to all succeeding subprograms processed in the Linkage Loader run, unless changed by another BASE2 card or by a PHASE card with a name in the operand. A PHASE card with a name in the operand resets the downward relocation factor to the top of core storage minus two.

Example

This example illustrates the construction of a single-phase program from three subprograms, each of which refers to a common data area. All three subprograms are on the System Library file, and none of them contain imbedded calls.

The sequence of cards in the Standard Input Unit is as follows:

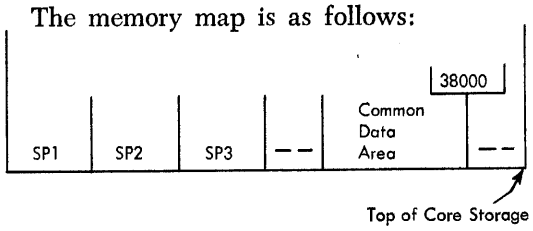
```
BASE2   38000
CALL    SP1
CALL    SP2
CALL    SP3
```

The Linkage Loader proceeds as follows:

1. BASE2 38000: The factor for downward relocation is set to 38000.

2. CALL SP1 through CALL SP3: These three subprograms are located and processed. During processing, all references to the common data area are adjusted in accordance with the factor set by the BASE2 card.

The memory map is as follows:



SNAP Card

The SNAP card directs the Linkage Loader to include the Snapshot program in a specified subprogram. (The Snapshot program is one of the Utility programs provided in the Operating System. See the *Utility Programs* publication for a description of the Snapshot program.)

The operands used in this card are as follows:

```
SUBPRGNAMExxxxxyyyyyzzzzzyyyyyzzzzzss
```

where:

SUBPRGNAME is the one- to ten-character subprogram name, left-justified, with trailing blanks in a ten-position field.

xxxxx is the address of the instruction immediately before which a Snapshot is to be taken.

yyyyy is the address of the lower limit of an area for which a Snapshot is to be taken. This address, if not an even hundred, will be changed to an even

hundred by the Snapshot program, which replaces the units and tens positions with zeros.

zzzzz is the address of the upper limit of an area for which a Snapshot is to be taken. This address, if not an address ending in 99, will be changed by the Snapshot program, which replaces the units and tens positions with 99.

ss is the length of the instruction at location xxxxx. If the instruction length is less than 10, it should be written as a two-digit number with a leading zero, e.g., 07.

NOTE 1: ss must be in card columns 56 and 57. Therefore, if only one area is to be snapped, leave blanks for the second yyyyyzzzzz.

NOTE 2: Do not put commas between operands.

The addresses used as operands are the relocatable addresses for a program compiled from 00000 upward. Normally these addresses can be obtained from the listing of the compilation. The Linkage Loader will add the value equated to the subprogram name linkage symbol to the addresses specified in the operand field of the SNAP card. However, if the origin of the subprogram was not at 00000, the user must subtract the origin value, as specified in the Autocoder ORG card, from the compiled addresses to get the addresses to be used on the SNAP card. If FORTRAN is used, the user must subtract 1 from the compiled addresses.

Columns 6 through 10 of the SNAP card can be used to establish a one- to five-character identification for the Snapshot(s) specified by the card. Any characters in those columns will be printed on the Snapshot listing.

The instruction at the point where the Snapshot is to be taken must be at least seven characters long, cannot be chained, cannot be part of an IOCS macroinstruction, and cannot contain any linkage symbols.

The following rules apply to the use of the SNAP card:

1. A SNAP card must be positioned in the Standard Input Unit after the subprogram that it affects, or the call for that subprogram, but before the subprogram's name is erased from the Linkage Loader symbol table and before the end of the phase.

2. Any number of SNAP cards can be submitted between PHASE cards.

3. A SNAP card can contain either one or two sets of upper and lower limits for Snapshot areas.

4. A SNAP card forces all pending calls for subprograms.

5. The Snapshot program must be in the Resident Monitor or in the library designated as the System Library file for this Linkage Loader run.

The SNAP card can be used only with a program that has been compiled. For a compile-and-go operation requiring Snapshot listings, use a Snapshot calling sequence in the source program. This sequence is given in the *Utility Programs* publication.

The Linkage Loader will replace the instruction before which the Snapshot is to be taken with a branch to a sequence that links to the Snapshot routine. The replaced instruction will appear in the linkage sequence, which is placed after the named subprogram.

INPUT Card

The INPUT card can be used to direct the Linkage Loader to read its control cards and relocatable object decks from a source other than the Standard Input Unit. The operand of the INPUT card is the name of the symbolic unit to be substituted for the Standard Input Unit. However, if System Generation (SG) mode has been specified, the INPUT card can only contain the operand MW2 or SIU.

The following rules apply to use of the INPUT card:

1. The name of the substituted symbolic unit must be one that is currently assigned to a physical unit. (See the section "Assignment of Input/Output Units.")

2. End of file at the substitute unit causes the Linkage Loader to resume reading from the Standard Input Unit.

3. INPUT cards can be placed at any point in the control card deck for the Linkage Loader.

4. The INPUT card can contain the operand SIU. In this case, the Linkage Loader resumes reading from the Standard Input Unit. Such a card would be used in a substitute unit, after an INPUT card in the Standard Input Unit had directed the Linkage Loader to begin reading from that substitute unit.

NOTE: The term *Standard Input Unit*, as used in the above description, refers to the unit currently used by the Monitor as its source of control information. This source could be either the Standard or Alternate Input Unit.

TITLE Card

The TITLE card is the first card of each subprogram. It defines the name of the subprogram as a linkage symbol. It is compiled by the Autocoder, FORTRAN, and COBOL language processors. The format for the TITLE card is:

```
TITLE    SUBPRGNAMEXXXXX    yyyyy
```

where:

SUBPRGNAME is the one- to ten-character subprogram name. (If fewer than ten characters, it is left-justified with trailing blanks in a ten-position field.)

xxxxx is the base to which the subprogram was compiled. This is normally 00000 or blank for Autocoder. (If blank, 00000 is assumed by the Linkage Loader.) It is 00001 for FORTRAN programs; and it is the size of IBCOBOL for COBOL programs.

yyyyy is the total number of core-storage locations in the common data area required by Autocoder subprograms. This field is blank for FORTRAN and COBOL subprograms.

NOTE: Five blanks separate xxxxx and yyyyy.

The value expressed by xxxxx, which is the low origin point for the subprogram, is subtracted from the current upward relocation value to produce the relocation factor for this subprogram.

The value expressed by yyyyy, the common data area size for Autocoder programs, is added to each relocated load address to determine if core storage has been exceeded. If core storage is exceeded while in the System Generation (SG) mode of operation, the Linkage Loader prints the message CORE EXCEEDED on the Standard Print Unit. This is only a warning message; the Linkage Loader puts the subprogram that exceeded core storage on the Job file and, if requested by an EXEQ card, the subprogram is executed. However, if core storage is exceeded while not in the System Generation mode of operation, the run is terminated.

NOTE: The TITLE cards generated by the COBOL and FORTRAN processors do not contain the size of the common data area. Therefore, the CORE EXCEEDED message will be produced only when the relocated load address exceeds the upper core-storage limit of the object machine.

After the Linkage Loader reads a TITLE card, it reads the next card in order to determine if it is a BASE1 card. If so, the BASE1 card will be processed before the TITLE card.

The TITLE cards generated by the language processors contain additional operands which are not used by the Linkage Loader; for instance, card columns 6-10 contain the date, and card columns 73-80 contain the card sequence number and the subprogram identification, in that order.

DISGO Card

The DISGO card specifies that the Go file is not to be searched for called subprograms. This card should be used to save search time when all called subprograms are on the System Library file.

After the Linkage Loader reads the DISGO card, it will not search the Go file (1) for the rest of the current Linkage Loader run, or (2) until a CONGO card is read.

The DISGO card is needed because the CALLN, CALLP, PHASE, SNAP, and Monitor control cards, and the end of file on the Standard or Alternate Input Unit, cause the Linkage Loader to process all calls on its request list. For this processing, the Linkage Loader rewinds the Go file and searches it from the beginning.

When a DISGO card is in effect, the Go file must remain mounted, ready, and assigned by an ASGN card. (For details about the ASGN card, see the "Assignment of Input/Output Units" section.)

CONGO Card

The CONGO card cancels the effect of the DISGO card. Thus, in the next search after the CONGO card, the Go file will be searched.

The CONGO card is required only if a DISGO card has been used because, if a DISGO card is not used, the Go file is searched once in each search.

Use of the CHAIN Feature

The CHAIN feature is a process by which multiphase programs can be created from modules that have been written in one or more of the Operating System languages. The programs are divided into portions, called *links,* when they are processed by the Linkage Loader. These links are similar to phases that are created by the use of PHASE cards. However, PHASE cards can be used easily only with programs written in Autocoder.

Because there is already a method within the Operating System of segmenting Autocoder programs by using PHASE cards, the CHAIN feature is of most advantage to the FORTRAN and COBOL programmer. Chaining provides a convenient way of dividing a program that exceeds the available core storage into portions that are successively executed. For example, if a 41,000-character FORTRAN program is to run on a computer that has 40,000 positions of available core storage, a low-activity portion of the program can be segmented as a separate link.

Chaining can also be used to combine subprograms that are written in different languages into one program. The CHAIN feature enables the programmer to link into one job several programs that must operate on the same data and that otherwise would have to be executed in several consecutive jobs. Moreover, chaining permits different programmers to write segments of a job. Because of the modularity of the finished job, changes to a portion of the job are more easily effected than if the job were all one program.

Refer to the publication *System Generation,* Form C28-0352, under the topic "Creation Charts" for the specific Linkage Loader needed for chained programs.

General Considerations for Chained Programs

The first link of the program is called the *main link.* Normally, the main link controls execution of all other links, which are called *dependent links.*

The following rules apply to the main link:

1. It can be written in Autocoder or COBOL with the dependent links written in any language. It can be written in FORTRAN only if dependent links are written in FORTRAN or Autocoder.
2. It can do any processing desired within the limits of the specified language.
3. It must contain the definition of the entire Autocoder or FORTRAN common data area used during execution of the main link and all dependent links.
4. It should contain any secondary subprograms that are used by all dependent links (see "COMN Card").
5. It can make reference to linkage symbols defined only within itself.
6. It, or one of the dependent links, must give control to the Resident Monitor at the /EOP/ or /UEP/ entry points when processing is completed (see "Use of Resident Monitor Functions").

A dependent link is any link of the program except the first one. A *previous dependent link* is any link defined before another dependent link. A *subsequent dependent link* is any link defined after another dependent link. The order of definition is the order in which the link control cards appear in the Standard Input Unit; it is not necessarily the order in which the links are executed. The following rules must be used when writing dependent links:

1. A dependent link may have reference to any linkage symbol defined in the main link or within itself.
2. A dependent link may have reference to any linkage symbol defined in a previous dependent link only if that portion of the dependent link containing the linkage symbol has not been overlaid. To prevent the symbol table of a dependent link from being overlaid, a PRTCT card must be used.
3. A dependent link cannot have a reference to a linkage symbol defined in a subsequent link.
4. A dependent link can do any processing desired within the limits of the specified language.
5. A dependent link can, but usually does not, overlay any part of the main link (see "LINK Card").

Because the programmer may specify the point to which a dependent link is to be relocated (see "LINK Card"), he must make certain that none of the subprograms required in future processing are overlaid. The programmer must also ensure that the links containing linkage symbols with a reference from a dependent link are in core storage at the time needed.

The Linkage Loader assigns a number to each link. The number assigned to the main link is 001; the first dependent link is 002, the next 003, etc.

Each link is composed of one or more subprograms. When it is constructing the chained program, the Linkage Loader can obtain the subprograms from the Go file, the System Library file, or the Standard Input Unit.

Special Considerations for COBOL Programs

The Procedure Division of all COBOL programs uses a small section of upper core storage. In this discussion of the CHAIN feature, this section of core storage is called the COBOL COMMON area. When the Linkage Loader is loading a main link written in COBOL, there should be a BASE2 card with /AMS/ as the operand. This forces COBOL COMMON to reside as high as possible in core storage. If there is a FORTRAN dependent link of a program whose main link is written in COBOL, BASE2 must be set at /AMS/ minus the number of characters in the COBOL COMMON area to avoid overlaying that part of core storage.

The BASE2 card is written, in this case, as:

```
BASE2 user-supplied-actual-address
```

User-supplied-actual-address is /AMS/ less the number of core positions used by the COBOL COMMON area, where /AMS/ is the size of core storage minus 2. The size of the COBOL COMMON area can be determined by referring to the listing which appeared as output on the SPR as a result of the compilation of the program. The number of characters used is specified at the end of the main program before the memory map.

For a COBOL program, the first operand (hereafter called the LINKSYMBOL) of any LINK card contains the first seven characters of the PROGRAM-ID source statement. The remaining three characters of the LINKSYMBOL field contain the serial number of the subprogram. The serial number is that number given at compilation time to the four modules of a COBOL program. Each module has an embedded call for the next higher module.

If the main link of a multiphase program is written in COBOL, its LINK card is written:

```
LINK MAINPGM001
```

This calls the first module of the COBOL main program which, in turn, calls the next module, etc. Dependent links written in COBOL are called using 002 as the last three characters of the LINKSYMBOL. This is done so that the Data Division (module 001) of the main link is not overlaid by the dependent link. The LINK card of a dependent link written in COBOL is written:

```
LINK DEPENDL002
```

In order to link together three COBOL programs, it is necessary to avoid overlaying the Data Division of the main link. For example, if the three COBOL links

(one main and two dependent) are named UPDATEA, UPDATEB, and UPDATEC, respectively, the LINK cards necessary to chain these programs are:

```
LINK UPDATEA001
LINK UPDATEB002
LINK UPDATEC002
```

A chained program consisting of a COBOL main link, two COBOL dependent links, an Autocoder dependent link, and a FORTRAN dependent link requires the following LINK cards:

```
LINK UPDATEA001
LINK UPDATEB002
LINK UPDATEC002
LINK AUTO
LINK FORTRAN
```

More than one multi-phase program may be built during an execution of the Linkage Loader. Each sequence of multi-phase control cards begins with a LINK card and must be terminated by an END card. If the sixth operand of the EXEQ LINKLOAD card specifies COBOL, all the main links in each multi-phase program must be COBOL programs.

The CHAIN feature may be used also for single-phase programs to simplify input to the Linkage Loader. The control cards used with this option are:

```
MON$$       EXEQ    LINKLOAD,,,,X,COBOL
ONEPHASE    LINK    ABCOBOL001
            END
```

The above cards have the same effect as the following:

```
MON$$       EXEQ    LINKLOAD
            PHASE   ONEPHASE
            DISCO
            CALL    IBCBLADOVR
            CALL    IBCBLDSPLY
            CALL    IBCBLCMPAR
            CALL    IBCBLSUBSC
            CALLN   IBCOBOL
            CONGO
            CALL    ABCOBOL001
            PHASE
```

Control Cards for Chaining

Five control cards describe the use of the CHAIN feature in the Operating System. Two of these cards are optional; they are the COMN card and the ENTRY card. Three of these cards are required; they are the EXEQ LINKLOAD card, the LINK card, and the END card.

Control cards for chained programs are summarized in Figure 4.

EXEQ LINKLOAD Card

The fifth and sixth parameters of the MON$$ EXEQ LINKLOAD card (used to execute the Linkage Loader) specify that the program is to be chained, define the name of the program, and indicate whether or not any of the subprograms were written in COBOL (see "Use of Transitional Monitor" for details on the EXEQ LINKLOAD card).

The fifth parameter must be specified; however, it may be a dummy name if the true name of the multi-phase program will appear on the main LINK card.

LINK Card

The LINK card defines a main or dependent link by specifying the main subprogram of the link. The LINK card for the main link may also indicate the multi-phase program name, if desired. The format of this LINK card is:

```
            LINK        LINKSYMBOL
                  or
PHASENAME   LINK        LINKSYMBOL
```

The LINK card for a dependent link may indicate, if desired, the point (BASE) to which the link is to be relocated. The format of this LINK card is:

```
            LINK        LINKSYMBOL
                  or
            LINK        LINKSYMBOL, BASE
```

LINKSYMBOL is the linkage symbol for the name of the main subprogram of the link. For a COBOL program, LINKSYMBOL is the first seven characters of the eight-character IDENT field of the PROGRAM-ID source statement. (For a more detailed discussion of LINK cards for COBOL links, see "Special Considerations for COBOL Programs.")

BASE is the linkage symbol or absolute address of the origin point. If BASE is omitted for a dependent link, Linkage Loader assigns the origin of the dependent link at the end of the main link. The BASE operand must not be used for dependent links in a program written in COBOL.

The main subprogram for a link would normally be on the Go file or the System Library file. If the subprogram is in the form of a relocatable deck, however, the deck must be placed on the Standard Input Unit following its LINK card.

COMN Card

The COMN card specifies additional subprograms to be included with the link. This card calls additional subprograms from the Go file or System Library file. These subprograms are loaded before the subprogram named in the LINK card.

The COMN card must follow the LINK card defining the link.

The formats of the COMN card are:

```
COMN    SUBPGNAME1,SUBPGNAME2,SUBPGNAME3,
        . . . , SUBPGNAMEX
```

Location in Control Card Deck	Format			Functions	Remarks
	6 15	16 Card Type 20	21 Operand(s)		
At the beginning of each main and dependent link; first LINK card follows EXEQ LINKLOAD card		LINK	LINKSYMBOL LINKSYMBOL = Linkage symbol for name of main subprogram in link.	Specifies a main or dependent link. Card for main link may indicate program name. If BASE is omitted for a dependent link, a Linkage Loader sets the origin of the dependent link at the end of the main link. Main link ignores BASE parameter.	For a COBOL program, LINKSYMBOL is the following: COBOL dependent link: the 7-character IDENT field of the PROGRAM-ID record. Autocoder dependent link: the 10-character operand of the TITLE statement. FORTRAN dependent link: the 10-character operand of the MON$$ EXEQ card. In Autocoder and FORTRAN languages, the LINKSYMBOL for COBOL programs must end in 002.
	PHASENAME	LINK	LINKSYMBOL		
		LINK	LINKSYMBOL, BASE BASE = Linkage symbol or absolute address assigned origin of dependent link.		
Following the LINK card		COMN	SUBPGNAME1, SUBPGNAME2,... SUBPGNAMEX	Specifies subprograms (in addition to subprogram named in LINK card) to be included in the link.	Two consecutive commas specify that all subprograms before the two commas are to be loaded before any other subprograms named in the card. All subprograms named in COMN card are loaded before subprogram named in LINK card. COMN card can be used only for main link of program written in COBOL.
		COMN	SUBPGNAME1,, SUBPGNAME2,... SUBPGNAMEX SUBPGNAMEX= Name of subprogram to be included in link.		
After subprogram containing entry point specified in ENTRY card		ENTRY	LINKSYMBOL or xxxxx LINKSYMBOL = Linkage symbol entry point. xxxxx = Absolute address.	Specifies the entry point in a link that does not contain a primary subprogram.	ENTRY card cannot be used for links written in COBOL.
At the end of the chained program		END		Marks the end of the cards defining the chained program.	This card must be present.
NOTE: If a comment is punched in the LINK, COMN, or ENTRY cards, one blank must be left between the last operand and the comment.					

● Figure 4. Linkage Loader Control Cards for Chained Programs.

```
COMN   SUBPGNAME2,,SUBPGNAME1,SUBPGNAME3,
       . . . , SUBPGNAMEX
```

In the first example, all the subprograms named in the COMN card are loaded before the main subprogram specified in the LINK card. The order in which the subprograms named in the COMN card are loaded depends on the order in which the Linkage Loader finds them. The finding order is the same as for the CALL card.

In the second example, the subprogram SUBPGNAME2 is loaded first. All other subprograms named in the COMN card are loaded next, in the order in which the Linkage Loader finds them. Then, the main subprogram named in the LINK card is loaded.

The two consecutive commas, shown in the second example, specify that all subprograms named before the two commas are to be loaded (in the order in which the Linkage Loader finds them) before any other subprograms specified in the card.

Any relocatable decks between the LINK card and the COMN card in the Standard Input Unit are loaded before the subprograms named in the COMN card; any relocatable decks between the COMN card and the next LINK card or the END card (see "ENTRY Card") are loaded after the subprograms in the COMN card.

NOTE: The COMN card may be used in either main or dependent links of either Autocoder or FORTRAN programs; however, it may be used only in the main link of a program written in COBOL.

ENTRY Card

The ENTRY card is used to specify the entry point of a link that does not contain a primary subprogram. The entry point is specified by the operand of the ENTRY card; this operand can be a linkage symbol or an absolute address. If the operand is a linkage symbol, it can be a subprogram name, or it can designate any point within a subprogram. An absolute address should be used only with extreme caution. The ENTRY card must have a 4 punched in card column 72 unless the fifth parameter is used in the EXEQ LINKLOAD card.

The ENTRY card should be placed after the LINK card that specifies the entry point in the subprogram.

NOTE: The ENTRY card cannot be used for links written in COBOL.

END Card

The END card marks the end of the chained program. This card is required.

Execution of Chained Programs

The main link normally controls the execution of a chained program and calls each dependent link at the appropriate point. Each dependent link normally returns control to the main link (see "Returning to the Main Link") or to the Resident Monitor at the /EOP/ or /UEP/ entry points. If a dependent link calls another dependent link instead of returning to the main link, the calling dependent link becomes, in effect, a new main link.

The Chain subroutine is used to control the calling and loading of dependent links by the Absolute Loader. The Return subroutine is used to return control to the calling link.

Calling a Dependent Link

In an Autocoder program, the calling sequence is the following:

```
DCWS CHAIN
DCW  LINKNUM
NOP
```

LINKNUM is defined in the calling link by:

```
LINKNUM DCW +nnn
```

where:

nnn is the three-digit number assigned to the dependent link by the Linkage Loader.

In a FORTRAN program, the calling method is:

```
CALL CHAIN (LINKNO)
```

LINKNO is an integer (variable or constant) and designates the number assigned to the dependent link being called.

In a COBOL program, the calling method is as follows:

CALL CHAIN USING { *data-name-1* / *literal-1* }

The literal designates the number assigned to the dependent link being called.

NOTE: For more detailed explanations of the use of the CHAIN feature with individual languages, see the discussions in the IBM 1410/7010 Operating System publications: *Autocoder,* Form C28-0326; *COBOL,* Form C28-0327; and *FORTRAN,* Form C28-0328.

Returning to the Main Link

In an Autocoder program, a dependent link can return control to the main link, call another dependent link, or return control to the Resident Monitor at the /EOP/ or /UEP/ entry point. Control can also be returned to the main link from dependent links written in Autocoder by using an Autocoder branch instruction. To return to the next sequential instruction following the dependent link calling sequence, the user may give the following instructions:

```
DCWS RETURN
NOP
```

In a FORTRAN program, returns are made by:

```
CALL RETURN
  or
RETURN
```

The CALL RETURN statement must be distinguished from the RETURN statement. The CALL RETURN statement is used only in links (main or dependent) that are FORTRAN main programs. The RETURN statement is used only in links that are FORTRAN SUBROUTINES.

In a dependent link, the CALL RETURN statement causes a return to the main link at the statement immediately following the CALL CHAIN statement that called the dependent link.

In a main link, the CALL RETURN statement causes either (1) a return to the Monitor (if no dependent links have been called) or (2) a return to the statement in the main link immediately following the last CALL CHAIN statement executed (if any dependent links have been called).

If another dependent link is called by a dependent link, control cannot be returned to the old main link written in FORTRAN.

For an example of a FORTRAN program using the CHAIN feature, see *FORTRAN,* Form C28-0328.

In a COBOL program, a dependent link written in COBOL may call another dependent link or return control to the Resident Monitor at the /EOP/ or /UEP/ entry point. Ordinarily, control cannot be returned to the main link in COBOL programs. The reason for this is that a dependent link written in COBOL overlays the Procedure Division of the main link. Segments of the Procedure Division may be loaded many times without disturbing the Data Division. This enables files to be opened in an initialization phase, processed in a series of main phases, and closed in a termination phase.

It is possible, however, to run a COBOL multiphase program so that the main COBOL link (containing both Data and Procedure Divisions) remains in core storage and calls several COBOL dependent Procedure Divisions to use the Data Division of the main link. In order to do this, three things must be done in the second, dependent program.

1. The Data Division of the main link must be duplicated.
2. There must be a dummy entry the size of the main link's Procedure Division.
3. There must be a dummy entry the size of the main link's COBOL COMMON area.

As stated above, the dependent link written in COBOL overlays the Procedure Division of the main link. The procedure outlined below is designed to avoid overlaying that section of core storage. The main link is executed with a PHASE card (containing no operand) placed at the end of the Linkage Loader control cards. The PHASE card gives the ending address of the main link. This address must be used as the BASE1 value when the dependent link is compiled. The BASE1 value *minus* the address of the first literal defined in the second module of the COBOL main link gives a value which is the size of the dummy entry needed to avoid overlaying the main link's Procedure Division.

A dummy entry the size of the main link's COBOL COMMON area is needed to protect that part of core storage. This entry forces the dependent link's Procedure Division COMMON area to load below the main link's COBOL COMMON area in core storage. For an example of a COBOL program thus chained together, see *COBOL,* Form C28-0327.

Resident Monitor functions can be used through linkage sequences and system symbol references coded into a source program.

The linkage sequences are branch instructions and, usually, one or more fields of control information. The branch instruction gives control to a routine in the Resident Monitor. The linkage sequences are listed in Figure 5.

Macro-instructions to generate most of the linkage sequences are included in the Autocoder Macro Library. These macro-instructions are discussed in the section following Figure 5.

The system symbols refer to areas of the Communication Region, so that all dependent programs can use certain information stored in the Resident Monitor. System symbols referring to areas in the Communication Region are listed in Figure 6.

Use of Communication Region

Contents of the Communication Region

Information in the Communication Region can be used by any dependent program at any time. However, the information can be modified only through the linkage sequence described below.

Some Communication Region fields contain data; others contain an address constant that gives the location of an area or data.

Each area of the Communication Region contains a word mark in its high-order position only.

Modification of the Communication Region

To modify a field of the Communication Region, the dependent program must contain a linkage sequence constructed as follows:

BXPA	/MCR/
DCW	/XXX/
DCW	(low-order address of information to be placed in Communication Region field /XXX/)
DCW	(address of dependent program routine to be given control if field cannot be modified)
(Next sequential instruction in dependent program)	

An example of the need for a dependent program to modify a Communication Region field is modification of the /IPI/ field during one program to set a switch for a later program in the same job. The following Autocoder macro-instruction can be used to generate the linkage sequence for modification of a Communication Region field:

```
ANYLABEL    MONOP    MODIFY,XXX,YYYYY,ZZZZZ
```

where:

ANYLABEL is any symbolic label.

MONOP must appear as shown.

MODIFY must appear as shown.

xxx is the system symbol, without slashes, of the Communication Region field to be modified.

yyyyy is the low-order address of the information to be placed in the Communication Region field.

zzzzz (optional) is the address to which control is to be given if the Communication Region field cannot be modified. If this operand is omitted, control will be returned to the next sequential instruction after the macro-instruction.

Use of Unit-Record Routines

The three unit-record routines in the Resident Monitor can be used by dependent programs. One of these routines reads from the Standard or Alternate Input Unit, another writes on the Standard Print Unit, and the third writes on the Standard Punch Unit.

All three routines perform their functions with unblocked records, in Move mode. If tape is used for any of the routines, the unit-record functions are performed in odd parity, and the records are unblocked.

Read Routine

The Read routine reads 80-character records from the Standard or Alternate Input Unit. (In constructing the linkage sequence to the Read routine, the programmer makes no distinction between reading from the Standard or Alternate Input Unit.)

The high-order address of the input area last filled by the Read routine is placed in the /CRD/ field in the Communication Region. /CRD/ must be used to retrieve information from an input area. The input areas are in the Resident Monitor. Word marks must not be set or cleared in the input areas.

Resident Monitor Functions	Linkage Sequence in Autocoder Language	Remarks
Read Routine: Reads 80-character records from Standard Input Unit and Alternate Input Unit	B /RSI/ B (address of end-of-file routine in dependent program) BBE USERERROR,/MCS/,n NSI Where the contents of n must be selected by the user.	/MCS/ contains the Channel Status Character constructed by the Resident IOCS from any machine indicators turned on by read errors. The Channel Status Character is described in the Basic Input/Output Control System publication.
Print Routine: Writes 132-character records on Standard Print Unit	One output area: BXPA /PRT/ DCW #5 DCW (high-order address of output area) NSI Two output areas: BXPA /PRT/ DCW #5 DCW (high-order address of output area) DC (high-order address of other output area) NSI	Output areas defined by dependent program, 134 positions long 1. First position. Carriage-control character. (See Figure 7.) 2. Positions 2 through 133. Record to be written on Standard Print Unit. (If tape, first character is also written.) 3. Position 134. Group mark with word mark.
Punch Routine: Punches 80-character records on Standard Punch Unit	One output area: BXPA /PCH/ DCW #5 DCW (high-order address of output area) NSI Two output areas: BXPA /PCH/ DCW #5 DCW (high-order address of one output area) DC (high-order address of other output area) NSI	Output areas defined by dependent program, 81 positions long: 1. Positions 1 through 80. Record to be punched by Standard Punch Unit. 2. Position 81. Group mark with word mark.
Console Inquiry Routine: Clears /MCI/ to blank, informs operator of need for information, enters waiting loop until operator has typed input, and returns control to NSI.	BXPA /MCR/ DCW /MCI/ DCW (address of a blank) DCW (address of dependent program's error routine) IOCTL TYPE,MESSAGE BCE *-11,/MCI/ NSI	High-order position of input message area is designated by /RIQ/. Operator identifies input message with $3x. x is placed in /MCI/. MESSAGE is the label for the console printer message, coded in the dependent program, that informs the operator of the need for the $3x message.
Wait-Loop Routine	B /WAT/ NSI	Console inquiry message, $50, causes the Resident Monitor to exit from the Wait-Loop routine and return control to NSI. NOTE: If the HALT option was specified in the Wait Loop routine, press INQUIRY REQUEST, press START, and enter console inquiry message $50.
Load Routine: Locates and loads a phase on the Job file or System Operating File (as specified on the EXEQ card for the program).	Locate, load, and execute phase: B /LOD/ DCW +nnn Locate and load phase: B /LOD/ DCW -nnn NSI nnn = phase number of a phase of the program being executed	
Subroutines of Load Routine: 1. Backspace System Tape: Backspaces records on the Job file or System Operating File (as specified on the EXEQ card for the program). 2. Read System Tape: Reads one record from the Job file or System Operating File (as specified on the EXEQ card for the program).	ZA (number of records to be backspaced, minus one),X13 B /ZBS/ NSI BXPA /ZRR/ DCW (@L@ for Load mode or @M@ for Move mode) DCW (input area address) BBE USERERROR. /ZRS/,n NSI	The fourth instruction is optional. It will give control to a dependent program error routine (USERERROR). /ZRS/ contains the Channel Status Character constructed by the Resident IOCS from any machine indicators turned on by read errors.

Figure 5. Linkage Sequences for Use of Resident Monitor Functions

Resident Monitor Functions	Linkage Sequence in Autocoder Language	Remarks
Subroutines of Load Routine (Continued): 3. Rewind System Tape: Closes and opens, with rewind and label checking if applicable, the Job file or System Operating File (as specified on the EXEQ card for the program). 4. Simple Search of System Tape: Locates the requested phase of the program currently being executed. 5. Read System Disk: Reads one record in load mode from the Job file or System Operating File (as specified on the EXEQ card for the program) into an input area.	Where the contents of n must be selected by the user. B /ZRW/ NSI B /ZSS/ DCW (three-position unsigned phase number) DCW (address of dependent program routine to be given control if phase is not found). NSI Read Specific Track: BPXA /ZRR/ DCW @N@ BBE USERERROR,/ZRS/,n NSI Read Next Track: BXPA /ZRR/ DCW (four-character track address) BBE USERERROR,/ZRS/,n NSI Where the contents of n must be selected by the user.	Input area defined by Resident Monitor: High-order position is /MBF/. Low-order data position (one position left of terminating group mark with word mark) is /MBX/. USERERROR is the label of an error routine in the dependent program. /ZRS/ contains the Channel Status Character constructed by the Resident IOCS from any machine indicators turned on by read errors.
End-of-Program Routine	Normal End of Program: B /EOP/ Unusual End of Program: B /UEP/ Special End of Program: BXPA /MCR/ DCW /CGO/ DCW (address of an A bit) DCW (address of dependent program routine to be given control if field cannot be modified) NSI NOTE: NSI is usually B /EOP/.	
Modification of Communication Region Routine	BXPA /MCR/ DCW /XXX/ DCW (low-order address of information to be placed in field /XXX/) DCW (address of dependent program routine to be given control if field cannot be modified)	

NSI is the next sequential instruction in the dependent program.

Figure 5. Linkage Sequences for Use of Resident Monitor Functions (Continued)

System Symbol	Contents of Field	Contents Type	Length	Modifiable
/AMS/	Actual machine size minus two (highest location that can be loaded by the Load routine). For example, for a machine with 80,000 positions of core storage, this field would contain 79997. This field is filled in at System Generation.	Address Constant	5	No
/CRD/	Contains the high-order address of the input area last filled by the Resident Monitor's Read routine.	Address Constant	5	No
/DAT/	Date (first two positions for the year, followed by three positions for the day). This area is set by a Monitor control card during initialization of the Resident Monitor. (See the section, "Use of Transitional Monitor.")	Data	5	No
/IPI/	Interprogram information. (This area can be used to store control information between runs within a job. For example, a program can set switches in this area that can be tested by the next program executed.) This area is set to zeros between each job, but certain systems programs (e.g., Sort Definition, Linkage Loader) alter its contents.	Data	5	Yes
/LIN/	Number of lines per page for printer output. This field is set at System Generation. (This field can be reset by any program that wishes to establish a constant that can be compared to a program counter for its printer output. However, any program that resets this field must set it back to the original value before end of program.)	Data	2	Yes
/ORG/	Contains lowest location of storage area available to dependent programs. (This field, which is set at System Generation, is used by the Linkage Loader to determine the value of Base Zero.) This location is at the top of the Resident Monitor minus the Initialization routine or at the top of the area reserved for programs under control of the Tele-processing Supervisor, if such an area exists.	Address Constant	5	No
/PHN/	Phase number of program phase currently being executed. (This area is set to 001 at each Monitor EXEQ card.)	Data	3	No
/PNM/	Name of program currently being executed. (The name is left-justified in the area.)	Data	10	Yes
/SIZ/	Highest location loaded for the current dependent program (including any previous phases of it). This field is set by the Resident Monitor's Load routine.	Address Constant	5	Yes
/TPB/	Lowest location of the area reserved for programs under control of the Tele-processing Supervisor. (Tele-processing Systems are described in the Tele-processing Supervisor publication.) This location is also the top of the Resident Monitor minus the initialization routine.	Address Constant	5	No
/MCI/	Set by console inquiry $3x, where x is placed in /MCI/. It is set to blank by End-of-Program routine.	Data	1	Yes
/MGA/	Contains track address of last track on disk Go file.	Disk Address	4	Yes
/CGO/	Go and other indicators. The A bit ON indicates that the contents of the Go and Job files are not valid. The B bit ON indicates that a language processor should put its output on the Go file. The B bit OFF indicates that the Go file contents are not valid. These indicators are set by the language processors, Linkage Loader, Monitor MODE control card, and dependent programs. Both bits are turned OFF between jobs.	Data	1	Yes

Figure 6. System Symbols for Communication Region Fields

The Read routine checks each input record to determine whether it is a Monitor control card. If it is, the routine branches to the dependent program's end-of-file address specified in the linkage sequence. Therefore, input records for a dependent program must not contain MON$$ in columns 6 through 10.

The Resident IOCS checks for all read errors.

The linkage sequence for use of the Read routine is as follows:

```
B       /RSI/
B       (address of end-of-file routine in dependent program)
BBE     USERERROR,/MCS/,n
        (Next sequential instruction in the dependent program)
```

If the Read routine signals an end-of-file condition, it returns control to the branch instruction containing the address of the dependent program's end-of-file routine. Otherwise, control is returned to the third instruction in the linkage sequence. (The third instruction, as shown, is optional.)

If the Standard Input Unit is tape, a program using the Read routine can also check for read errors by including the third instruction, as shown. This instruction interrogates /MCS/. /MCS/ contains the Channel Status Character constructed by the Resident IOCS from any machine indicators turned on by read errors. The Channel Status Character is described in the *Basic Input/Output Control System* publication.

USERERROR in the third instruction is the address to which control is to be given if an error occurred. The contents of n, which must be selected by the user, will be compared to /MCS/ to determine if control should be given to USERERROR.

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Resident Monitor's Read routine:

```
ANYLABEL  STDIO  READ,AREA,EOFADDR,USERERROR,n
```

where:

ANYLABEL is any symbolic label.

STDIO must appear as shown.

READ must appear as shown.

AREA is the high-order address of an 81-character field (including a group mark with word mark at the end) into which the record is moved. This area must be defined by the dependent program.

EOFADDR is the address in the dependent program to which control passes when end of file occurs.

USERERROR (optional) is the address in the dependent program to which control is to be given if an error occurs.

n (optional) is the character to which /MCS/ will be compared to determine if control should be given to USERERROR. The user must specify the contents of n.

Print Routine

The Print routine writes 132-character records on the Standard Print Unit. The Print routine can be addressed by two types of linkage sequences. One requests a print operation using one-area logic; the other requests a print operation using two-area logic. Both linkage sequences include the address of the output area(s) containing the record(s) to be written on the Standard Print Unit.

NOTE: Only one record will be written per entry to this routine.

Each output area must be defined by the dependent program, and each must be 134 positions long. The first position of the area is used to specify the carriage-control character. (See Figure 7 for the carriage-control characters.) Positions 2 through 133 of the output area contain the record to be written on the Standard Print Unit. (If the Standard Print Unit is tape, the first position of the output area is also written.) Position 134 of the output area must contain a group mark with word mark.

The two types of linkage sequences for using the Print routine are as follows.

One Output Area:

```
BXPA    /PRT/
DCW     #5
DCW     (high-order address of the output area)
(Next sequential instruction in the dependent
program)
```

Two Output Areas:

```
BXPA    /PRT/
DCW     #5
DCW     (high-order address of one output area)
DC      (high-order address of the other output
        area)
(Next sequential instruction in the dependent
program)
```

NOTE: The DCW of five blanks (immediately below the branch instruction) is used for internal functions of the Print routine. This field must be included in every linkage sequence to the Print routine.

d	IMMEDIATE SKIP TO	d	SKIP AFTER PRINT TO
1	Channel 1	A	Channel 1
2	Channel 2	B	Channel 2
3	Channel 3	C	Channel 3
4	Channel 4	D	Channel 4
5	Channel 5	E	Channel 5
6	Channel 6	F	Channel 6
7	Channel 7	G	Channel 7
8	Channel 8	H	Channel 8
9	Channel 9	I	Channel 9
0	Channel 10	?	Channel 10
#	Channel 11	.	Channel 11
@	Channel 12	□	Channel 12

d	IMMEDIATE SPACE
blank	1 Space
K	2 Spaces
L	3 Spaces

d	SPACE AFTER PRINT
/	1 Space
S	2 Spaces
T	3 Spaces

Figure 7. Carriage-Control Characters for Print Routine

By testing for a word mark in the position immediately following the DCW that contains the address of the output area, the Print routine determines which type of linkage sequence is being used. If a word mark is present, it is assumed to be the word mark over the operation code of the next instruction in the dependent program. If no word mark is present in that position, as with the DC, the Print routine assumes the presence of an address for a second output area.

NOTE: If only one output area is specified, the position following the address of the output area must contain a word mark.

If two output areas are specified, the Print routine first uses the output area originally specified by the DCW. After completion of each print operation, the routine rotates the two addresses in the linkage sequence. Before the first print operation, the linkage sequence would appear as follows:

```
BXPA    /PRT/
DCW     #5
DCW     XXXXX
DC      yyyyy
```

After the first print operation, the linkage sequence would appear as follows:

```
BXPA    /PRT/
DCW     #5
DCW     yyyyy
DC      XXXXX
```

The next time the Print routine is addressed by this linkage sequence, it will use the output area represented by yyyyy. After that print operation is completed, the two addresses are rotated again.

If the printed output is to start on a new page, the first data record must have a "1" as the carriage-control character. If the last data record of the printed output is to be the last line printed on a page, one additional record of blanks, with a "1" as the carriage-control character, should follow the last data record.

The Resident IOCS checks for all print errors (or write-tape errors, if tape is used), except for wrong-length records written on tape. The POW program prints only 133-character tape records.

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Print routine:

```
ANYLABEL    STDIO    PRINT,AREA1,AREA2
```

where:

ANYLABEL is any symbolic label.

STDIO must appear as shown.

PRINT must appear as shown.

AREA1 is the high-order address of a 134-position field from which the record is to be printed. The first position of the field specifies the carriage-control character (see Figure 7). Positions 2 through 133 contain the record to be printed. Position 134 contains a group mark with word mark.

AREA2 (optional) specifies a second output area with the same format as AREA1.

Punch Routine

The information concerning the Print routine also applies to the Punch routine with the following exceptions:

1. The Punch routine writes 80-character records on the Standard Punch Unit. The output area(s) for these records must be 81 positions long. (Position 81 must contain a group mark with word mark.)
2. The system symbol used in the BXPA instruction is /PCH/.

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Punch routine:

```
ANYLABEL    STDIO    PUNCH,AREA1,AREA2
```

where:

ANYLABEL is any symbolic label.

STDIO must appear as shown.

PUNCH must appear as shown.

AREA1 is the high-order address of an 81-position field from which the record is to be punched. Position 81 must contain a group mark with word mark.

AREA2 (optional) specifies a second output area with the same format as AREA1.

The Resident IOCS checks for all punch errors (or write-tape errors, if tape is used), except wrong-length record written on tape. The POW program punches only 80-character tape records.

Use of Console Inquiry Routine

Through the Console Inquiry routine, a dependent program can request input from the console printer, enter a waiting loop while the operator types in the information, and then retrieve the input.

The linkage sequence for console input is:

```
BXPA    /MCR/
DCW     /MCI/
DCW     (address of a blank)
DCW     (address of dependent program's error
        routine)
IOCTL   TYPE,MESSAGE
BCE     *-11,/MCI/
        (Next sequential instruction in the dependent
        program. This instruction should move the con-
        sole input from /RIQ/ to a field in the dependent
        program.)
```

MESSAGE is the label for a console printer message coded in the dependent program. This message is typed to request that the operator enter a $3x message. Then the sequence enters a waiting loop that is broken when the $3x message is entered.

The typed-in message begins with $3x. The x is moved to a one-character field designated /MCI/. /MCI/ is cleared to blank by the Console Inquiry routine, before it requests the input, and by the End-of-Program routine.

The high-order position of the console input area is designated /RIQ/. Thus, the character "$" is at the address designated by /RIQ/, "3" is at the address designated by /RIQ/+1, etc. The input area length is defined at System Generation; it can be five positions minimum to 20 positions maximum. In a Tele-processing system, the input message area must be 20 positions long. Wrong-length-record errors are ignored by the Console Inquiry routine.

The input area is not cleared by the Console Inquiry routine. Because a following console message might overlay the $3x message, the dependent program should immediately move the message from the input message area and compare the $3x in the message to the x in /MCI/ to verify that the message retrieved is the correct one.

Use of Wait-Loop Routine

When a dependent program requires that processing be suspended until the machine operator performs a function (such as changing forms in the printer), the program can write a message on the console printer and then enter the Wait-Loop routine. When the operator has performed the function specified by the program's message, he breaks the waiting loop by means of console inquiry $50. The Resident Monitor then

returns control to the dependent program at the instruction immediately following the branch to the Wait-Loop routine.

The entry point to the Wait-Loop routine is represented by the system symbol /WAT/. In Autocoder, the instruction B /WAT/ is used to enter the routine. COBOL generates this branch from the STOP (literal) statement. FORTRAN generates this branch from the PAUSE statement.

In a Tele-processing system, the system can receive interrupts during the waiting loop.

An option in the Wait-Loop routine can cause a halt rather than execute a waiting loop every time a branch is made to the /WAT/ entry point. If this option is chosen, it is specified at system generation time. The HALT is broken by the operator by:

1. Pressing INQUIRY REQUEST
2. Pressing START
3. Entering the class A message $50

Use of Load Routine

The Load routine in the Resident Monitor can be used by dependent programs to locate and load phases of a multiphase program.

The linkage sequence to request locating, loading, and execution of a specific phase is as follows:

```
B       /LOD/
DCW     +nnn
```

When the phase number, nnn, contains a plus sign over the units position, the Load routine initiates execution of the phase at its entry point.

The linkage sequence to request only locating and loading of a specific phase is as follows:

```
B       /LOD/
DCW     -nnn
        (Next sequential instruction in dependent program)
```

When the phase number, nnn, contains a minus sign over the units position, the Load routine returns control to the next sequential instruction following the linkage sequence that issued the request for the phase.

The following considerations should be observed in using linkage sequences to request program phases:

1. Requests can be issued only for phases of the program being executed. (The name of the program being executed is contained in the /PNM/ field of the Communication Region.)

2. The number of the phase being executed is contained in the /PHN/ field of the Communication Region. Using this field, the dependent program can construct a request for a specific phase in relation to the phase being executed.

3. If the requested phase cannot be located, control is given to the Resident Monitor's End-of-Program routine at the unusual-end-of-program entry point. The requested phase must be on the System Operating File or Job file, as specified in the EXEQ card for the program. (Details concerning end-of-program procedures are in the topic "Use of End-of-Program Routine." Details on the EXEQ card are in the section "Use of Transitional Monitor.")

4. As a phase is brought into storage, it overlays all information in the previous phase that falls inside the new phase's lowest and highest loading addresses. Therefore, information to be communicated from one phase to the next must reside below the lowest address into which information is loaded for a succeeding phase, or above the highest address.

For a tape-oriented Resident Monitor, dependent programs can use subroutines in the Load routine to read, backspace, rewind, and search the Job file or System Operating File, whichever was specified in the Monitor EXEQ control card for the program. For a disk-oriented Resident Monitor, dependent programs can use a subroutine in the Load routine to read the Job file or System Operating File, whichever was specified in the EXEQ card for the program.

Read System Tape Subroutine

This subroutine reads the next record from the Job file or System Operating File into the area designated in the linkage sequence. The input area must be defined by the dependent program. Wrong-length-record errors are not checked. If the subroutine detects the end of file, the file will be rewound and opened with label checking, if labels exist, and the first data record after the label will be read into the designated area.

The linkage sequence for the Read System Tape subroutine is:

```
BXPA    /ZRR/
DCW     (@L@ for Load mode or @M@ for Move mode)
DCW     (input area address)
BBE     USERERROR,/ZRS/,n
        (Next sequential instruction in the dependent program)
```

The fourth instruction is optional. It will give control to a dependent program error routine labeled USERERROR if an error occurred. The user must select the contents of n to match the error conditions to be checked. /ZRS/ contains the Channel Status Character constructed by the Resident IOCS from any machine indicators turned on by read errors.

The following Autocoder macro-instruction can be

used to generate the linkage sequence to the Read System Tape subroutine:

```
ANYLABEL  SYSIO  READ,X,AREA,USERERROR,n
```

where:

ANYLABEL is any symbolic label.
SYSIO must appear as shown.
READ must appear as shown.
x must be an L for Load mode or an M for Move mode.
AREA is the high-order address of the area into which the record is to be read. This area must be defined by the dependent program.
USERERROR (optional) is the address in the dependent program to which control is to be given if an error occurs.
n (optional) is the character used for comparison to determine if an error occurs. The user must select the contents of n.

Backspace System Tape Subroutine

This subroutine backspaces one or more records, as specified in the linkage sequence, on the Job file or System Operating File. The subroutine will backspace over tape labels, treating them as data records.

The linkage sequence for the Backspace System Tape subroutine is:

```
ZA     (number of records to be backspaced, minus one), X13
B      /ZBS/
(Next sequential instruction in the dependent program)
```

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Backspace System Tape subroutine:

```
ANYLABEL  SYSIO  BSP,X
```

where:

ANYLABEL is any symbolic label.
SYSIO must appear as shown.
BSP must appear as shown.
x is the number of records to be backspaced.

Rewind System Tape Subroutine

This subroutine closes and opens, with rewind and label checking if applicable, the Job file or System Operating File, whichever was specified in the EXEQ card for the program.

The linkage sequence for the Rewind System Tape subroutine is:

```
B    /ZRW/
(Next sequential instruction in the dependent program)
```

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Rewind System Tape subroutine:

```
ANYLABEL  SYSIO  RWD
```

where:

ANYLABEL is any symbolic label.
SYSIO must appear as shown.
RWD must appear as shown.

Simple Search Subroutine (Tape or Disk)

This subroutine locates but does not load the requested phase of the program being executed. If the phase is not located, control is transferred to the not-found address specified in the linkage sequence.

The linkage sequence for the Simple Search subroutine is:

```
B      /ZSS/
DCW    (three-position unsigned phase number)
DCW    (address of dependent program routine to be given control if phase is not found)
(Next sequential instruction in dependent program)
```

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Simple Search subroutine:

```
ANYLABEL  SYSIO  SEARCH,nnn,NOTFNADDR
```

where:

ANYLABEL is any symbolic label.
SYSIO must appear as shown.
SEARCH must appear as shown.
nnn is the unsigned, three-digit number of the phase to be found.
NOTFNADDR (optional) is the address in the dependent program to which control is to be given if the phase is not found. If omitted, control will be returned to the instruction after the macro-instruction.

Read System Disk Subroutine

This subroutine reads a record from the System Operating File or Job file in a disk-oriented Operating System. The subroutine reads in Load mode. The input area is defined by the Resident Monitor; the high-order position is /MBF/ and the low-order position (one position to the left of the terminating group mark with word mark) is /MBX/.

The linkage sequence for the Read System Disk subroutine for a specific track (to be read from the System Operating File or Job file, whichever is specified in the EXEQ card) is:

```
BXPA   /ZRR/
DCW    (four-character track address)
BBE    USERERROR,/ZRS/,n
(Next sequential instruction in dependent program)
```

The linkage sequence to read the next track from the System Operating File or Job file, whichever is specified in the EXEQ card, is:

```
BXPA    /ZRR/
DCW     @N@
BBE     USERERROR,/ZRS/,n
        (Next sequential instruction in dependent program)
```

USERERROR is the label of an error routine in the dependent program. /ZRS/ contains the Channel Status Character constructed by the Resident IOCS from any machine indicators turned on by read errors. The user must select the contents of n to match the error conditions to be checked.

The following Autocoder macro-instruction can be used to generate the linkage sequence to the Read System Disk subroutine. The forms of the macro-instruction to read a specific or the next track are:

```
ANYLABEL    SYSIO    READ,,nnnn,USERERROR,n
ANYLABEL    SYSIO    READ,,,USERERROR,n
```

where:

ANYLABEL is any symbolic label.

SYSIO must appear as shown.

READ must appear as shown.

nnnn specifies the four-digit track address of the record to be read. If omitted, the next sequential track will be read. (Omission must be indicated by a comma, as shown in the second form above.)

USERERROR (optional) is the address in the dependent program to which control is to be given if an error occurred.

n (optional) is the character to be used for comparison to determine if an error occurred. The user must select the contents of n. If a comma is used for this operand, write it as: @, @. If a commercial *at* (@) sign is used for this operand, write it as: @@.

NOTE: Commas must appear as shown.

Use of End-of-Program Routine

Dependent programs can end execution in one of three ways: normal end of program, unusual end of program, or special end of program. The results of the three end-of-program types are summarized below.

END OF PROGRAM	NOT TEST MODE	TEST MODE
Normal entry point: /EOP/	Continue processing with the next Monitor control card.	Continue processing with the next Monitor control card.
Special entry point: /EOP/; A-bit of /CGO/ ON	Skip to next JOB card.	Continue processing with the next Monitor control card. However, Linkage Loader will not be loaded and executed and programs on the Job file will not be executed, if requested.
Unusual entry point: /UEP/	Write contents of storage on Core Image file, if this file was specified at System Generation and currently has a unit assigned to it. Skip to next JOB card.	Write contents of storage on Core Image file, if this file was specified at System Generation and currently has a unit assigned to it. Continue processing with next Monitor control card.

The mode is determined by the MODE card (described in the section "Use of Transitional Monitor").

NOTE: As the Transitional Monitor skips to the next JOB card, it will process Monitor control cards directed to the Tele-processing system and will process a Monitor END card.

NOTE: If the program Snapshot is included at System Generation, a Snapshot of all core storage is taken at /UEP/, regardless of mode.

Normal End of Program

When a dependent program completes its required functions, it signals end of program by branching to the Resident Monitor at the entry point represented by the system symbol /EOP/. This linkage sequence in Autocoder language is B /EOP/. The Resident Monitor brings the Transitional Monitor into storage to read the Standard or Alternate Input Unit for the next Monitor control card.

NOTE: The next Monitor control card can be a control card indicating the beginning of the next job or requesting the performance of another run within the current job (such as the execution of a Utility program). Therefore, within one job, the Resident Monitor can receive any number of end-of-program indications.

Unusual End of Program

An unusual end of program can be signaled to the Resident Monitor in one of three ways: by the dependent program itself; by the console inquiry $10, from the operator; or by having the operator depress the PROGRAM RESET key and then the START key. The linkage sequence to the unusual-end-of-program portion of the End-of-Program routine is:

```
B /UEP/.
```

When the Resident Monitor receives an unusual-end-of-program indication, it preserves the contents of storage on the Core Image file (if the installation includes such a file). The Resident Monitor then checks to determine if the dependent program is being tested, as indicated by a MODE card. If the TEST mode is not ON, the Resident Monitor cancels any subsequent portions of the current job by bringing the Transitional Monitor into storage to read the Standard or Alternate Input Unit until it finds a Monitor control card that indicates the beginning of the next job (JOB CARD), the end of the current batch (Monitor END card), or a card directed to the Tele-processing system. Thus, if the program is not being tested, execution of remaining programs within this job will not be performed. As an example, the Core Image file will not be printed, even though a storage print was requested by a Monitor control card requesting execution of the Utility programs.

If the dependent program is being tested, the Transitional Monitor is brought into storage to read the Standard or Alternate Input Unit for the next Monitor control card. If, for example, a storage print is to be taken for the program, a Monitor control card that requests execution of the Utility programs would be placed at this point in the deck.

NOTE: Because the contents of core storage is written on the Core Image file only for an unusual end of program, and because the Core Image file serves as input to the Storage Print program of the Utility programs, a dependent program being tested should end with a branch to /UEP/, rather than /EOP/. This ensures that a Storage Print can always be obtained for a program being tested, whether it comes to a successful conclusion or not.

Special End of Program

The special end of program is the same as unusual end of program, except that the Go and Job file contents are indicated as invalid and the contents of storage are not written on the Core Image file. Because the Go and Job file contents are indicated as invalid, the Linkage Loader and any program to be loaded from the Job file will not be executed if requested, whether the TEST mode is ON or OFF. (See the "Use of Transitional Monitor" section for a description of the MODE card.)

To obtain a special end of program, the dependent program must have the following coding sequence to set the A-bit of the /CGO/ field ON. This sequence must be in the coding before the program branches to /EOP/.

```
BXPA    /MCR/
DCW     /CGO/
DCW     (address of an A-bit)
DCW     (address of dependent program routine
        to be given control if field cannot be
        modified)
(Next sequential instruction in the dependent
program. Usually this instruction is B /EOP/.)
```

The following rules are programming requirements that must be met by all dependent programs.

Use of the Resident IOCS

All programs within the Operating System — those supplied by IBM and those that are user-written — use the Resident IOCS. The Autocoder, COBOL, and FORTRAN processors automatically generate the necessary linkages to the Resident IOCS, in accordance with the input/output statements of each source language. (The Autocoder language statements for input/output operations are contained in the *Basic Input/Output Control System* publication.)

No Halts

Dependent programs must not use programmed halts. At end of program, the programs must branch to the Resident Monitor's End-of-Program routine.

Index Registers

The System Monitor sets a word mark in the high-order position of each index register. Group marks must not be placed in that position. Dependent programs must not clear or set word marks in the index registers.

Each time the Transitional Monitor is brought into storage by the Resident Monitor to process Monitor control cards, it clears all index registers, but leaves the word mark in the high-order position of each register. Transitional Monitor then uses the index registers and does not clear them prior to loading a dependent program. The signs of index registers are never minus after processing of the Monitor control cards.

Various elements of the System Monitor require index registers 14 and 15. Therefore, X14 and X15 must not be used by any interruptable instruction sequence in a dependent program. If used by a non-interruptable sequence, X14 and X15 must not be left negative.

Index register 13 is used for subroutine linkage by all programs within the Operating System. (For example, the first instruction in the Resident Monitor's Print routine is Store B-Address Register into X13. The Print routine then uses the contents of index register 13 for reference to the control information in the linkage sequence and for returning to the dependent program at the instruction following the linkage sequence.)

Index register 13 should be used for linkage to all subroutines introduced into the Operating System by the installation. Because of the convention of using index register 13 for subroutine linkage, IBM subroutines do not save and restore the contents of this register. Subroutines introduced by the installation also do not need to save and restore its contents. However, if a dependent program uses index register 13 for purposes other than subroutine linkage, it should save the contents before issuing a linkage sequence to a subroutine and should restore the contents of the register after control is returned from the subroutine.

If index registers 13, 14, and 15 are used by installation programs, they must never be left negative.

Word Separator Characters

A word mark should never be set over a word separator character. (A word mark over a word separator character would be placed on the Core Image file as three word separator characters, but would be read from the file into core storage as a word separator character with a word mark over the following character.)

Priority Alert Mode

Dependent programs should not exit from Priority Alert mode, except as specified in the section "Use of Resident Monitor Functions," when addressing the Resident Monitor (such as in a branch to the Print routine).

Unit-Record Interrupts

A dependent program cannot be initiated by means of the Priority Select and PRIORITY ON switches on the console except as noted in the section "Simultaneous Peripheral Operation On Line."

/SIZ/ Field

A dependent program should have a character in the highest core-storage position it will use. This character enables the Load routine to properly set the /SIZ/ field of the Communication Region.

Reading from the Standard Input Unit

Dependent programs must use the Read routine of the Resident Monitor to perform any read operations for the Standard (or Alternate) Input Unit.

Divide Overflow and Arithmetic Overflow Indicators

Each time the Transitional Monitor is brought into storage by the Resident Monitor to process Monitor control cards, it turns off the divide overflow and arithmetic overflow indicators.

Two Highest Positions of Core Storage

The Operating System places group marks with word marks in the two highest positions of core storage. These group marks with word marks must not be destroyed by dependent programs.

System Regeneration

If the installation regenerates the Operating System and if the Resident Monitor in the new System Operating File was changed from that in the previous System Operating File, the Job files produced by the previous Operating System cannot be used with the new system.

Monitor control cards in the Standard (or Alternate) Input Unit direct operation of the System Monitor. These cards are read and analyzed by the Transitional Monitor. The Monitor control cards are listed in Figure 8.

At System Generation, the installation can specify that the System Monitor write each Monitor control card on the console printer and/or the Standard Print Unit. The installation can also elect at System Generation to record all JOB cards on the Standard Punch Unit to help separate punched output.

Figure 8 shows the format for Monitor control cards. The standard Autocoder rules apply to the operands of Monitor control cards:

1. Operands must be separated by a comma.
2. Operands cannot contain blanks.
3. An intentionally omitted operand must be indicated by placing its trailing comma adjacent to the preceding comma, except where the omitted operand is the last one.

JOB Card

The JOB card indicates the beginning of a job. The operand is the name assigned to the job by the programmer.

Upon reading a JOB card, the Resident Monitor closes out the previous job as follows:

1. It completes any remaining IOCS requests.
2. It clears Communication Region fields: size of dependent program field (/SIZ/), interprogram information field (/IPI/), phase number field (/PHN/), program name field (/PNM/), and other fields used internally.
3. It turns off the indicators in the /CGO/ field set for the operands of the MODE card (described below).
4. If requested, it allows the operator to close, rewind, and unload the tapes for the Standard Print Unit, Standard Punch Unit, and Core Image file, to permit off-line processing after the previous job is completed. (These indicators are set by $B2, $B3, and $B4 console inquiries, which are described in the section "Messages and Inquiries.")
5. It cancels all Reserve file assignments and all assignments of *alternate* physical units. (See the "Assignment of Input/Output Units" section.)

The Resident Monitor spaces the Standard Print Unit to the next page and prints on the Standard Print Unit and the console printer one line, which gives:

1. An "S" or "A" to indicate whether the JOB card was read from the Standard or the Alternate Input Unit.
2. A two-character sequence number that indicates the number of JOB cards that have been read from that unit since the beginning of the batch.
3. The contents of the JOB card.

The JOB card contents and information will be written on the Standard Print Unit and the console printer whether or not other Monitor control cards are also printed and/or typed. An additional feature, which can be specified at System Generation, is the punching of JOB cards on the Standard Punch Unit.

The System Monitor does not use the operand(s) of JOB cards. However, an installation-written module to use the operand(s) can replace the Transitional Monitor's IBACCOUNT module.

The IBACCOUNT module, as supplied in the Transitional Monitor, is a dummy module. After the JOB card is printed and all other JOB card functions are completed, the Transitional Monitor branches to this module, which, as supplied, will immediately return control to the Transitional Monitor. The installation can put into this module accounting routines or any other types of routines desired. To replace the dummy module with an installation module, see information concerning generation of the Transitional Monitor in the *System Generation* publication.

MODE Card

This card describes the type of operation wanted in a job. The operands for the card are GO, TEST, and SG.

The GO operand may appear on a card with TEST and/or SG operands. The GO operand sets the B bit of the /CGO/ field on. The B bit is turned OFF by the next JOB card or the next MODE card without a GO operand.

The GO operand is used to specify compile-and-go operation. If it is specified, the output from following compilations will be written on the Go file for subsequent execution within the job.

If the Go file being used was created any time before the last Linkage Loader run, the GO mode must not be in effect. If the GO mode is in effect when a previously created Go file is used, the Linkage Loader will tape-mark the Go file at the point where the tape is when the Linkage Loader execution begins.

The TEST operand specifies that following programs

Location in Control Card Deck	Format: 6 – 15	Format: 16 – 20 Card Type	Format: 21 – 72 Operand(s)	Functions	Remarks
Before first JOB card during initialization	MON$$	DATE	yyddd yy=Last two numbers in year ddd=Three-digit number for day of the year	Sets Communication Region field /DAT/.	Must be given in daily information cards during initialization of the System Monitor.
Before first JOB card during initialization	MON$$	SPOOL	xxyyNNNNNNNNNNb xx, yy=SPOOL units NNNNNNNNNN=Name of edit routine.	Initiate SPOOL (Tape parity is odd)	One unit record device (xx) and one tape unit (yy) must be specified. NNNNNNNNNN will be specified only for Type II and Type III SPOOL.
Same as above	MON$$	SPOOB	Same as above	Same as above.	Same as above
Same as above	MON$$	SPOOU	Same as above.	Initiate SPOOL (Tape parity is even)	Same as above.
First card in a job	MON$$	JOB	ANY NAME AND INFORMATION	Closes out previous job and begins new job.	Clears /SIZ/, /IPI/, /PHN/, /PNM/. Turns off /CGO/ indicators for MODE card operands.
Before EXEQ card for program that will use symbolic unit	MON$$	ASGN	ABC,XX,YY,ZZ ABC=Symbolic name for input/output file XX,YY,ZZ=Assignment symbols for any number of physical units	Assigns the symbolic unit in the first operand to the physical units represented by the assignment symbols in the second and following operands. The second operand is the base unit and the additional operands are alternate units.	See the section, "Assignment of Input/Output Units," for full details.
	MON$$ MON$$	ASGN ASGN	ABC,XYZ LIB,SOF	Assigns the symbolic unit in the first operand to the physical units for the symbolic unit in the second operand.	Limited to Work files and Reserve files, and to the combination: System Library file and System Operating File.
	MON$$	ASGN	ABC	Cancels the assignment previously made for the symbolic unit in the operand.	
Before EXEQ card for program to which MODE card applies	MON$$ MON$$ MON$$ MON$$ MON$$	MODE MODE MODE MODE MODE	GO,TEST GO,SG GO TEST SG	GO operand specifies that language processors write on Go file for compile-and-go operation. TEST operand specifies that rest of job is to be executed if program comes to an unusual end. SG operand specifies that System Generation functions are to be performed. (See the System Generation publication.)	MODE card without GO turns GO indicator off. NOTE: If a previously created Go file is being used, do not use MODE GO. TEST indicator, once turned on, remains on until next job. SG indicator, once turned on, remains on until System Monitor is initialized
After JOB card	MON$$	EXEQ	NAME1, MJB,XXX	Initiates search for program in first operand on System Operating File (SOF) or Job file (MJB) specified in second operand, then causes it to be loaded and executed.	
	S MON$$	EXEQ	LINKLOAD, SOF, XXX, YYYYYYYYYY, PROGNAME, COBOL		

NAME1 for installation programs is defined by first PHASE card of program.

NAME1 for IBM-provided programs can be:

- LINKLOAD (Linkage Loader)
- AUTOCODER, COBOL, FORTRAN (language processors)
- SORTDEFINE (Sort Definition program)
- UTILITIES (Utility programs)
- SG1, SG2 (System Generation programs)
- TPADLIBGEN, TPATLIBGEN, TPRDLIBGEN, TPRTLIBGEN (Tele-Processing system programs)
- DSKLIBLDR (Disk Library Loader program)

XXX=Symbolic unit where input data, source statements, or the control cards are located.
YYY=Name of the System Library file for this run of the Linkage Loader.
PROGMNAME=Specifies name of program. Inclusion of this parameter tells the Linkage Loader that the program is to be chained.
COBOL=Used only for chained programs containing subprograms written in COBOL.
S (in column 1 of EXEQ LINKLOAD card)=Linkage Loader memory map not printed.

Anywhere	MON$$	COMT	ANY INFORMATION TO BE TYPED AND PRINTED	Information in columns 21 through 72 is typed on console printer and printed on Standard Print Unit.	Information is typed and printed, whether other Monitor control cards are similarly recorded or not.
At end of cards in Alternate Input Unit and Standard Input Unit	MON$$	END		In Alternate Input Unit, causes Resident Monitor to resume reading from Standard Input Unit, after giving operator opportunity to close Standard Print Unit, Standard Punch Unit, and Core Image file. In Standard Input Unit, signifies end of current series of jobs and causes Resident Monitor to enter a waiting loop for further instructions from operator.	If Standard and Alternate Input Units are on tape, END card causes tape to be rewound and unloaded.

Figure 8. Monitor Control Cards

are being tested. At unusual end of program, the Resident Monitor's End-of-Program routine preserves the contents of storage on the Core Image file, if this file was specified at System Generation. Then, if TEST mode was specified, the Transitional Monitor, when brought into storage, reads the Standard or Alternate Input Unit for any Monitor control cards. Thus, a card requesting execution of the Utility programs to print the Core Image file can be read. If TEST mode had not been specified, the Resident Monitor would cancel any remaining portion of the current job.

If a Utility program storage print is wanted when a program under test successfully reaches end of program, the program must branch to /UEP/.

The TEST indicator, when turned on by a MODE card, stays on until the next job or until System Monitor initialization.

The SG operand turns the System Generation indicator on. This operand specifies that a System Generator file or System Operating File is being generated, or that System Generation maintenance procedures are being performed. (See the *System Generation* publication.)

EXEQ Card

The EXEQ card initiates a search for the named program and causes it to be loaded and executed. Before the program is loaded, storage from the locations specified in the /ORG/ field to the top of core minus two (/AMS/) will be cleared.

The program to be executed is named in the first operand. This program may be an installation program, the Linkage Loader, or other IBM-provided programs. The name to be used for an installation program is defined by the first PHASE card of the program.

The second operand specifies the file containing the program. This operand can be either SOF, for the System Operating File, or MJB, for the Job file. If the second operand is omitted, the program is assumed to be on the SOF. (Omission must be indicated by a comma, if additional operands are used.) The Resident Monitor's Load routine loads the named program from the file specified.

The third operand, which is optional, changes the Resident Monitor's Read routine to read from a substitute unit instead of the Standard or Alternate Input Unit. After the program is executed, the Read routine automatically resumes reading from the Standard Input Unit. This operand cannot be used if the EXEQ card is in the Alternate Input Unit, because the Read routine treats the specified unit as a temporary Alternate Input Unit. This operand can be used only if the facility for reading an Alternate Input Unit was incorporated into the Resident Monitor at System Generation.

This operand must be the symbolic name for a Work file, Reserve file, the Standard Input Unit, or the Alternate Input Unit.

For an installation program, the third operand can specify the location of input data. This file must consist of only one reel. (No Monitor control cards should appear in this input data.)

For the language processors, the third operand indicates that the source statements do not immediately follow the EXEQ card. This operand specifies the unit containing the source statements.

For the Linkage Loader, the third operand indicates the location of the Linkage Loader control cards.

The EXEQ LINKLOAD card can have a fourth operand to specify the System Library file for this Linkage Loader run. In a tape-oriented system, the Linkage Loader uses two factors to determine the Library to be used:

1. The fourth operand of the EXEQ card. If blank, the Linkage Loader assumes that the Library is the IBM Library (named IBMLIBR) on the System Operating File.

2. The ASGN card. If absent, the Linkage Loader assumes that the Library named is on the System Operating File.

Installation programs can specify fourth and succeeding operands for use of the program named in the EXEQ card.

When the EXEQ card contains more than three operands, the address of the high-order character of the fourth operand is placed in card columns 6 through 10 in the card input area, overlaying MON$$. No word mark is set in column 6. The /CRD/ field contains the high-order address of the card input area until the next read operation performed by the Read routine.

The EXEQ LINKLOAD card used for construction of chained programs must have a fifth parameter and can have a sixth. The fifth parameter, which tells the Linkage Loader that the program is to be chained, defines the name of the program. The program name can be one to ten characters long. The sixth parameter, which is the word COBOL, is included in the card only if one or more of the subprograms is written in COBOL.

An EXEQ LINKLOAD card may have an S in column 1 to specify that the memory map should not be printed. However, the S in column 1 will not suppress the printing of diagnostic messages. (See the "Messages and Inquiries" section.)

COMT Card

The COMT (Comments) card is used to record information on the console printer and the Standard Print Unit. All information in columns 21 through 72 of the COMT card is typed and printed, whether or not other Monitor control cards are similiarly recorded. The COMT card is used to:

1. Give additional instructions to the operator.
2. Identify a program's output in greater detail than the JOB card permits.

ASGN Card

The ASGN card is used to assign input/output units. The functions of the ASGN card are described in detail in the section "Assignment of Input/Output Units."

END Card

The END card must be the last card in the Standard Input Unit and the Alternate Input Unit.

When this card is read at the Standard or Alternate Input Unit, the Transitional Monitor does the following:

1. Closes the unit with a rewind and unload, if the unit is tape.
2. Types two messages: END SIU (or END AIU) and 20501 ENTER B MESSAGES. All class B console inquiries will be accepted except $B1, which will be ignored if the END card was on the Standard Input Unit. (For an END card on the Alternate Input Unit, see item 4.) Thus, the operator can close the Standard Print Unit, the Standard Punch Unit, and/or the Core Image file. During this time, the System Monitor is in a waiting loop and Tele-processing system interrupts may occur. (See the "Messages and Inquiries" section for details on console inquiries.)
3. Remains in the waiting loop until the $BX inquiry is entered. Then, the Transitional Monitor resumes reading from the Standard Input Unit, which must be on the same physical unit used in the previous batch. If the END card was on the Standard Input Unit, the Transitional Monitor will ignore all cards until it reaches the first JOB card. Then it will begin processing. If the END card was on the Alternate Input Unit, the Transitional Monitor resumes processing with the next Monitor control card on the Standard Input Unit.
4. If the END card was on the Alternate Input Unit, a $B1 inquiry (before the $BX inquiry) will make the Transitional Monitor resume reading from the Alternate Input Unit.

Two counters are used to determine the number of JOB cards read since the beginning of a batch. One counter is associated with JOB cards read from the Standard Input Unit, the other with JOB cards read from the Alternate Input Unit. Each time an END card is read from one of the input units, the counter associated with that unit is set to zero.

DATE Card

The DATE card contains the date in its operand. The first two positions contain the last two numbers in the year, and the next three positions contain the three-digit number for the day of the year.

This card is required in the daily information, which is before the first JOB card, during initialization of the Resident Monitor. (See the *Operator's Guide* publication for details on initialization.)

The operand is placed in the /DAT/ field of the Communication Region.

SPOOL Card

The SPOOL card is used to assign physical units and specify editing routines (if any) for SPOOL operations (see $B7 messages in the section "Simultaneous Peripheral Operation On Line"). When this control card is used, it must precede the first Monitor JOB card at initialization.

PAUSE Card

When the Transitional Monitor recognizes a PAUSE card, processing stops and control passes to the Monitor wait loop. Enter $50 on the console keyboard to continue processing.

Simultaneous Peripheral Operation On Line (SPOOL) is a programming concept dealing with the concurrent execution of card-to-tape, tape-to-card, and tape-to-printer operations and a dependent program. In the 1410/7010 Operating System, SPOOL includes the SPOOL control routines, user-written editing routines (if desired), and SPOOL input/output areas, which are incorporated into Resident Monitor. All tape operations are executed in Move mode. Standard IOCS error correction is provided for all SPOOL operations.

Three types of programmed control for SPOOL are available:

1. *Type I* implements normal operations with no blocking, deblocking, or editing of records. Label processing is that processing specified for system files.
2. *Type II* permits inclusion of user-written editing routines that contain no input/output operations.
3. *Type III* permits inclusion of user-written editing routines containing IOCS macro-instructions.

The selection of one of these three types of SPOOL control is made at System Generation. See the *System Generation* publication for procedures and additional machine requirements needed to accommodate SPOOL.

Initialization

Through Control Cards

The format of the MON$$ SPOOL control card is:

MON$$ SPOOLxxyynnnnnnnnnnb

where:

SPOOL specifies the operation but the type of parity is not specified

xx and yy are two physical units

nnnnnnnnnnb is the name of the editing routine to be loaded from the SOF. The name need not be ten characters, but a blank must be entered after the last character of the name. If no editing routine is specified (that is, if the SPOOL control card is in the format SPOOLxxyyb), only Type I SPOOL will be performed. In order to specify the parity of the tape, it is necessary to substitute SPOOB or SPOOU in the place of SPOOL, where SPOOB specifies odd parity and SPOOU specifies even parity.

Through Console Inquiries

The format of the Class B console inquires is:

$B7bxxyynnnnnnnnnnb

where:

$B7b specifies the operation, but the type of parity is not specified, and xx, yy, and nnnnnnnnnnb have the same meaning as above. However, in order to specify the parity of the tape, it is necessary to substitute $B7B or $B7U in the place of $B7b, where $B7B specifies odd parity and $B7U specifies even parity.

NOTE: If the parity of the tape has not been specified, either through the use of a control card or a console inquiry, odd parity is assumed.

Physical Units

The two physical units designated for SPOOL operations must be a tape unit and a unit-record device. The unit-record device must be designated first. The tape unit and the unit-record device are assigned to symbolic units /STn/ and /SUn/, respectively, n designating the channel on which the unit-record device is located. The two units are disconnected from the system; if any job attempts to assign them, the operator is notified and can enter an alternate unit for the dependent program assignment (see message 20502 in the *Operator's Guide* publication).

Initialization Messages to Operator

Two messages may appear on the console printer during initialization. They are:

10190 xx

Explanation: Unit xx is already assigned.
Action: The operator should re-enter the $B7 inquiry using another unit.

10191 nnnnnnnnnn

Explanation: The editing routine, nnnnnnnnnn, called could not be found.
Action: The operator may re-enter the $B7 inquiry using another routine name.

Initiation of Operation

SPOOL operations can be initiated *any* time after SPOOL initialization. The operator uses the following procedure:

1. Ready the appropriate tape on the SPOOL tape unit.

2. Ready the appropriate unit-record device.

3. Turn the Priority Select Switch on the 1415 console to the appropriate position.

4. Press PRIORITY ON. This provides the initial interrupt needed to start SPOOL.

Operation

Four messages may appear during SPOOL operations. They are:

10101 mms xx

Explanation: An error has occurred on unit xx, a SPOOL unit-record device.

Action: The operator should take any necessary corrective action and press PRIORITY ON twice to resume operation. (See *Operator's Guide.*)

10192 SIP

Explanation: An attempt has just been made to assign units to a SPOOL supervisor which is currently in process.

Action: The operator can discontinue the current functions by making certain the PRIORITY ON switch is off, entering the release message for the current units ($B7Rxxyy), and then pressing the PRIORITY ON switch.

10199 EOSn

Explanation: The SPOOL function has been completed on channel n.

Action: Press PRIORITY ON twice to process another file.

30101 cu

Explanation: Error in one or more fields in the header label of the SPOOL input tape on channel and unit cu.

Action:

1. Press INQUIRY REQUEST.
2. Enter $8R (to retry the label checking) or
   Enter $8A (to accept the label as read).
3. Press INQUIRY RELEASE.

SPOOL Editing Routines

Editing routines written by the user and specified and placed on the System Operating file at System Generation (see the *System Generation* publication) must conform to the following rules:

1. The entire routine must be a one-phase program.

2. Each editing routine must be assembled to load at system symbol /SAn/, where n is the channel on which the SPOOL unit-record device exists. A BASE1 control card is used; i.e., BASE1 /SAn/.

3. Each editing routine *must* begin with a fixed prefix. This is described in the following section. Essentially, the prefix contains the address of entry points in the user's editing routine to which SPOOL control will branch.

4. The size of an editing routine must not exceed the size specified at System Generation.

5. Any IOCS macro-instruction can be used within any editing routine associated with Type III SPOOL.

6. If the execution of the editing routines exceeds certain specified times, SPOOL does not share the time with a batch program. In effect, SPOOL uses the system to the exclusion of any other data processing (except that of Tele-processing). These times are:

Card reader	60 milliseconds
Printer	85 milliseconds
Card punch	235 milliseconds

Fixed Prefix

Each editing routine must begin with the fixed prefix described below. The prefix has the following format:

ˇAAAAAˇBBBBBˇCCCCCˇDDDDDˇEEEEEˇFFFFFˇGGGGG

where:

AAAAA is the address of an area in the editing routine that is used as the input/output area for all tape operations. If this field is blank or zero, the unit-record area in SPOOL control is used.

BBBBB is the address of an entry point in the editing routine to which SPOOL control branches on a unit-record-operation-complete interrupt. When such an interrupt occurs, editing, blocking, and other operations on the data can be performed. If this field is blank, no branch is made.

CCCCC is the address of an entry point in the editing routine to which SPOOL control branches on a tape-operation-complete interrupt. On such an interrupt, editing, blocking, and other operations on the data can be performed. If this field is blank, no branch is made.

DDDDD is the address of a File Table Extension information area (in the editing routine) used for the SPOOL tape unit. If this field is blank, label processing is bypassed.

EEEEE is the address of an entry in the editing routine to which SPOOL control branches immediately before opening the SPOOL tape. If this field is blank, no branch is made.

FFFFF is the address to which SPOOL control branches immediately after closing the SPOOL tape. If this field is blank, no branch is made.

GGGGG is the address of an entry point in the editing routine to which SPOOL control branches immediately upon receiving the input end-of-file interrupt. If this field is blank, no branch is made. This part of the editing routine enables padding and writing partial records.

Returning Control to SPOOL

The editing routine must return control to SPOOL with a BXPA in accordance with the following designations:

RETURN FROM	RETURN POINT
BBBBB	/Sn3/ — to continue with execution of the next tape operation. /Sn4/ — to skip the tape operation and execute another unit-record operation.
CCCCC	/Sn1/ — to continue with execution of the next unit-record operation. /Sn2/ — to skip the unit-record operation and execute another tape operation.
EEEEE	/Sn0/ — return to SPOOL, following OPEN exit.
FFFFF	/Sn9/ — return to SPOOL if no partial records found.
GGGGG	/Sn8/ — return to SPOOL, following CLOSE exit. /Sn3/ — return to SPOOL to write final record.

The following are additional system symbols used in SPOOL control:

SYMBOL	REPRESENTS
/SnE/	The Channel Status Character in the SPOOL IORW. This character reflects the error status of the last SPOOL input/output operation.
/SCn/	The high-order position of the SPOOL card area.
/SPn/	The high-order position of the SPOOL print area.
/SKn/	The carriage control character. This control character is located at /SPn/ minus one and has a word mark.
/SFn/	The low-order position of the file list origin field of the SPOOL file table (the file table address).

Programming Considerations

SPOOL imposes some restrictions on the use of indicators and entry into Operating System reserved areas. These are outlined below.

SUBJECT	TYPE II SPOOL	TYPE III SPOOL
Use of high, low, equal, and zero balance indicators	Allowed	Allowed
Use of divide check and arithmetic overflow indicators	Not Allowed	Not Allowed
Use of index registers X1 through X12	Allowed if user saves and restores	Allowed in Priority Alert mode only if user saves and restores.
Use of index register X13	Not Allowed	Allowed
Use of index registers X14 and X15	Allowed	Not Allowed when in Priority Alert mode. Otherwise allowed.
Entering Priority Alert mode	Not Allowed	Allowed
Entering Resident IOCS	Not Allowed	Allowed

If the user has specified an input/output area (AAAAA) in the editing routine, he is responsible for moving data between his area and the SPOOL input/output areas. There are no word marks in the card or print input/output areas. Both areas are followed by a group mark with word mark.

Extra input/output units may be assigned to SPOOL by use of an ASGN card. It is recommended that Teleprocessing symbolic units be used for these assignments.

If the address at DDDDD is /MLX/, SPOOL will use the standard File Table Extension for system files.

Assignment of Input/Output Units

Input/output units are referenced as:

1. PHYSICAL UNITS. *Physical unit* designates all of, or sections of, a physical device. Each physical unit is assigned a two-character *assignment symbol* at System Generation. The assignment symbols are selected by the installation.

2. SYMBOLIC UNITS. *Symbolic unit* designates a file to be read from or written on a physical unit. Symbolic units are referred to by the *symbolic names* given in this section.

The symbolic unit names are used in the SYMUNIT entry of the IOCS Define the File statement in programs (see the *Basic Input/ Output Control System* publication) and in the first operand of the Monitor ASGN card. The symbolic unit files are assigned to one or more physical units by the ASGN card and IBLOOKM linkage sequence. The assignments are made by assignment routines in the Resident and Transitional Monitors.

To enable an installation to make the most efficient use of input/output units for a series of jobs, program references to input/output units are made symbolically. Also, the programmer is freed from concern as to which specific input/output units will be available when his program is executed

A physical unit is one of the following:

1. A reader, punch, printer, or other unit-record device
2. A tape unit
3. A set of consecutive tracks in a 1301 or 2302 Disk Storage module

NOTE: The 1402 Card Read Punch must be defined for each function. (The read function is specified as one physical unit, the punch function as another.)

The assignment symbols specified for physical units during System Generation are placed into a table in the Resident Monitor. During operation of the system, this table represents the input/output units that are available for use by dependent programs and by the System Monitor.

This table can be changed — in terms of additions and deletions — only by another System Generation. The operator, however, can indicate through $B5 and $B6 inquires that a physical unit is not currently available for use, or that it is again available. (See the "Messages and Inquiries" section for details on console inquiries.)

NOTE: Definition of Tele-processing devices is also performed during System Generation, but in a different manner. They are defined by specifying the related elements of the Tele-processing Supervisor. (See the *Tele-processing Supervisor* publication.) The definitions of Tele-processing devices are not placed in the Resident Monitor's table. However, physical units used by programs operating within the Tele-processing system (other than Tele-processing devices) are given assignment symbols in the same manner as physical units for other programs, and their assignment symbols are also placed in the Resident Monitor's table.

Physical Units

The following example illustrates one possible set of assignment symbols to represent physical units. Since any scheme of two-character symbols can be incorporated into the System Monitor, each installation can designate the set of assignment symbols that best suits its needs.

Channel 1 Tape Units: A0 (A-zero) through A9
Channel 2 Tape Units: B0 through B9
Unit-Record Equipment:
1402 Card Read Punch (read function): C1
1402 Card Read Punch (punch function): C2
1403 Printer: C3
Tracks of Disk Storage (Channel 1, Module 0):
0000-0079: D1
0120-0159: D2
7000-7999: D3
(Note that although the tracks that constitute each physical unit must be consecutive, separate physical units need not be consecutive or contiguous.)
Tracks of Disk Storage (Channel 1, Module 0):
4000-8999: E1
9315-9386: E2
(Note that track assignments need not begin and end at cylinders.)
Tracks of Disk Storage (Channel 2, Module 0):
0000-9999: F1

NOTE: Physical unit track definitions may overlap each other.

Symbolic Units

The symbolic names to be used for symbolic units are defined during System Generation, when they are placed in the Resident Monitor table. During system operation, this table is used in conjunction with the table of physical units to assign input/output units for program use.

Symbolic units are divided into five groups, in accordance with the type of program that uses the symbolic unit:

1. *System files* are used by the System Monitor.
2. *Work files* are used by IBM-provided components of the Operating System.
3. *Reserve files* are used by installation programs.
4. *Tele-processing system files* are used by installation programs that operate under control of the Tele-processing Supervisor.
5. *SPOOL files* are used by the SPOOL Supervisor in performing peripheral operations on-line while dependent programs are being run.

NOTE: Work files can also be used by installation programs, but care must be exercised to ensure that conflicts do not arise between assignments for IBM-provided programs and assignments for installation programs.

The symbolic names for the symbolic units are listed below. For each System file, the symbolic name is predetermined; for the other four groups of symbolic units, the symbolic names are assigned from the specific sets by the Operating System at System Generation.

System files:

SIU — Standard Input Unit
SOF — System Operating File
SPR — Standard Print Unit
SPU — Standard Punch Unit
AIU — Alternate Input Unit
LIB — System Library file
MJB — Job file
MGO — Go file
MDM — Core Image file
MLT — TP Library file (for Tele-processing system)
MDT — Temporary Storage file (for Tele-processing system)

Work files:

MW0 (MW-zero) through MW9
MWA through MWZ

Reserve files:

MR0 through MR9
MRA through MRZ

Tele-processing system files:

MT0 through MT9
MTA through MTZ

SPOOL files:

SU1 — SPOOL Unit-Record Device on Channel 1
SU2 — SPOOL Unit-Record Device on Channel 2
ST1 — SPOOL Tape Unit associated with the SPOOL Unit-Record Device on Channel 1
ST2 — SPOOL Tape Unit associated with the SPOOL Unit-Record Device on Channel 2

Assignment Times for Symbolic Units

The symbolic units are divided into five categories with respect to the times that they can (or must) be assigned to physical units:

1. Symbolic units that are assigned during System Generation, initialization, or reinitialization:

SOF (System Operating File)
SIU (Standard Input Unit)

NOTE: The device type for the System Operating File cannot be changed at initialization or reinitialization.

2. Symbolic units that are assigned during initialization or reinitialization of the System Monitor and cannot be reassigned, except by initialization or reinitialization procedures:

SPR (Standard Print Unit)
SPU (Standard Punch Unit)
MDM (Core Image file)

NOTE: Assignment of these units is part of the daily information supplied to the System Monitor during initialization and reinitialization procedures. These units must be assigned if facilities for them were incorporated into the Resident Monitor at System Generation. However, if the variable print and punch modules are selected, these symbolic units may be assigned at System Generation, initialization, or whenever the Transitional Monitor is called; the units can be reassigned at initialization or between runs in a job. If these units are assigned at System Generation, the assignments cannot be canceled before the first JOB card. Thus, if the variable print and punch modules are chosen, these system files fall into category three. In this case, all three (and the Temporary Storage file) can share the same tape unit.

3. Symbolic units that can be assigned during initialization or reinitialization procedures and can also be assigned or reassigned between each run within a job:

LIB (System Library file)
AIU (Alternate Input Unit)
MWX (All Work files)

NOTE: The Alternate Input Unit cannot be assigned by a control card that is read from the Alternate Input Unit. The System Library file should not be assigned if it is on the System Operating File. Three Work files must be assigned for the Autocoder and COBOL language processors; the physical units, if tape drives, for two of these files should be on one channel and the physical unit for the third file should be on the other channel, for maximum processor efficiency. The physical units, if tape, for the two Work files that must be assigned for the FORTRAN processor should be on dif-

ferent channels. (See the *Operator's Guide* publication for full details.)

4. Symbolic units that are assigned between each job:

MJB (Job file)
MGO (Go file)
MRX (All Reserve files)

NOTE: The System Monitor cancels the assignment of these units at each JOB card. These files can be assigned (or reassigned) between each run within a job. However, care should be taken in reassigning the Job and Go files to ensure that the Linkage Loader and language processors have the proper input/output files.

5. Symbolic units that can be assigned at System Generation or at other times when the Tele-processing system is not open:

MLT (TP Library file)
MDT (Temporary Storage file)
MTX (All Tele-processing system files)

NOTE: If the Tele-processing system is not open, the Temporary Storage file can be shared with the Standard Print Unit, the Standard Punch Unit, and the Core Image file. The assignment of these units is canceled when the total System Monitor is initialized or reinitialized. Between initializations (or reinitializations) of the total System Monitor, these assignments are maintained, regardless of the number of times the Tele-processing system is opened and closed. The assignments can, however, be changed by ASGN cards each time the Tele-processing system is closed.

6. Symbolic units that can be assigned only by the class B console inquiry $B7 (see the "Messages and Inquiries" section):

SU1 (SPOOL Unit-Record Device on Channel 1)
SU2 (SPOOL Unit-Record Device on Channel 2)
ST1 (SPOOL Tape Unit associated with the SPOOL Unit-Record Device on Channel 1)
ST2 (SPOOL Tape Unit associated with the SPOOL Unit-Record Device on Channel 2)

NOTE: The assignment of these units is canceled when the total System Monitor is initialized or reinitialized. Between initializations (or reinitializations) of the total System Monitor, these assignments are maintained. However, the assignments can be changed by the $B7 console inquiry.

Program References to Symbolic Units

After the installation has established the set of symbolic names for the symbolic units, these names are used in installation programs to designate the symbolic units for a program's files. For example, in Autocoder, the symbolic name is used in the SYMUNIT entry of the Define the File statement. COBOL and FORTRAN similarly provide means by which the programmer identifies symbolic units for a program.

The Linkage Loader converts the symbolic names for the symbolic units into addresses that enable the Resident and Transitional Monitors to assign input/output units in accordance with the ASGN cards.

ASGN Card

The ASGN card, which is a Monitor control card, directs the System Monitor to assign one or more physical units to a particular symbolic unit.

As shown in Figure 9, the first operand of an ASGN card is the symbolic name for the symbolic unit. The second operand is the assignment symbol for the first physical unit (the base unit) to be assigned to that symbolic unit. Additional operands can be used to specify alternate units.

In Figure 9, A1 is the base unit; A2 and A3 are the alternate units. When end of unit is reached for A1, the next input/output operation for MR1 will be performed with the physical unit represented by A2. (For tape, end of unit is determined by reaching end of reel; for disk, end of unit is determined by reaching the end of the set of tracks that constitute one physical unit. The use of alternate units does not apply to unit-record equipment.)

At the end of unit for A2, the next input/output operation for MR1 will be performed with the physical unit represented by A3. The end of unit for A3 is considered end of file for the symbolic unit MR1, if MR1 represents a disk file. However, if MR1 represents a tape file, continuous cycling of the physical units is performed. Thus, A1 would be the fourth physical unit used. In such a case, end of file is determined by reel counts, trailer labels, or end-of-reel conditions (for unlabeled tapes).

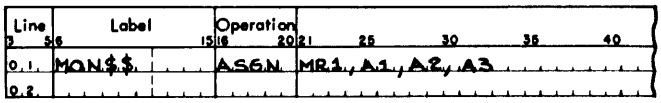


Figure 9. Format of the ASGN Card

A physical unit can be assigned to more than one symbolic unit at the same time. For example, a job involving a compile-and-go operation could use a tape unit as a Work file for the language processor and then use the same unit as a Reserve file for the program that was just compiled. The ASGN cards for the two symbolic units can both be placed immediately behind the JOB card for this job. The System Monitor makes the assignments as the ASGN cards are read, but since assignments are not used until a program initiates an input/output operation through the IOCS, no conflict

occurs because of the dual assignment. Figure 10 illustrates the use of two ASGN cards to assign the same physical unit to different symbolic units.

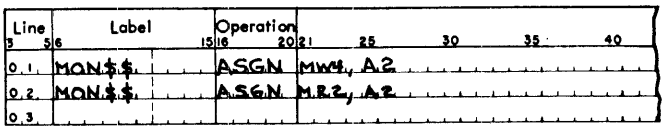


Figure 10. Assignment of Two Symbolic Units to the Same Physical Unit

Note that all symbolic units that share a base unit also share any alternate units specified for that base unit. The physical units are shared in the same order. A base unit assigned to one symbolic unit cannot be assigned to another symbolic unit as an alternate unit. (For example, these two assignments cannot exist simultaneously: MR1,A1,A2 and MR2,A2.) However, if an alternate unit must be assigned as the base unit of another symbolic unit later in the job (or vice versa), the initial assignment must be canceled. If a base unit assigned to one symbolic unit is subsequently assigned to another symbolic unit, all alternate units assigned to the first symbolic unit are also shared by the second. However, if in the second assignment, an alternate unit not named in the first assignment is specified, that alternate will be ignored by the Monitor.

Cancellation of Assignment

Symbolic unit assignments are canceled by an ASGN card containing only the first operand. For example, the ASGN card in Figure 11 cancels the assignment for the Reserve file MR1. This cancellation permits the physical units previously assigned to MR1 to be used for another file.

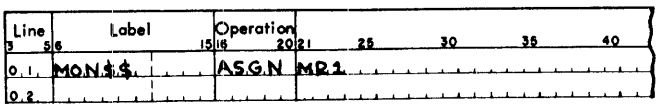


Figure 11. Cancellation of Assignment

The ASGN card in Figure 11 and the ASGN card used to assign MR1's physical units to the next file should both be placed immediately before the EXEQ card for the program that uses the next file. (The card that cancels the assignment to MR1 must precede the card that makes the new assignment of the physical units.)

Placement of the ASGN Card

The ASGN cards are submitted to the System Monitor through the Standard (or Alternate) Input Unit. The position of an ASGN card in the control card deck is determined by the category of symbolic unit named in the ASGN card.

ASGN cards for symbolic units that are assigned during initialization or reinitialization are in the daily information in the Standard Input Unit.

ASGN cards for symbolic units that are assigned only for the duration of a job must follow the related JOB card and precede the EXEQ card for the program that uses the files.

ASGN cards for symbolic units that are used by the Tele-processing system must precede the control card or the console inquiry that opens the Tele-processing system.

Use of the ASGN Card for System Files

The physical units assigned to certain System files cannot be assigned to any other symbolic units during the time they are assigned to the System files. These System files are:

- SOF (System Operating File)
- SIU (Standard Input Unit)
- SPR (Standard Print Unit)
- SPU (Standard Punch Unit)
- LIB (System Library file)
- MDM (Core Image file)
- AIU (Alternate Input Unit)
- MLT (TP Library file)
- MDT (Temporary Storage file)

In addition, physical units assigned to symbolic units used by the Tele-processing system cannot be assigned to any other symbolic units when the Tele-processing system is ready to receive input from Tele-processing devices.

The physical units assigned to the System Operating File and the Standard Input Unit can be reassigned to other symbolic units only through System Generation, initialization, or reinitialization. The physical units assigned to the Standard Print Unit, the Standard Punch Unit, and the Core Image file can be reassigned to other symbolic units only through initialization or reinitialization. If the variable print and punch modules are selected, these system files can also be reassigned by an ASGN card during run time. The physical unit assigned to the System Library file (if this file is not part of the System Operating File) can be reassigned to other symbolic units either through initialization and reinitialization or through cancellation of the Library's assignment by an ASGN card containing LIB as the only operand or containing the operands SOF,LIB. System files must consist of only one reel.

For compile-and-go operations, the Go file must not be assigned to a physical unit used for the language processor's Work files.

System File Sharing and Device Switching Feature

The device switching feature allows the device type of the Standard Print Unit and the Standard Punch Unit to be changed by an ASGN card. This ASGN card can be placed before the first JOB card in the deck or between runs in a job. The device switching feature

for the Standard Print Unit is selected by choosing the Variable Print module. This selection is made by specifying 3 in parameter 1, field P of the GEN08 card (see the *System Generation* publication for an explanation of the GEN08 card). Device switching for the Standard Punch Unit is made by selecting the Variable Punch module. This module is selected by specifying 3 in parameter 1, field R of the GEN08 card. The device switching feature, if desired, must be selected for both the Standard Print Unit and the Standard Punch Unit.

If device switching is requested, one tape unit can be shared by the Standard Print Unit, Standard Punch Unit, Core Image file, and Temporary Storage file. Assignment of these files can be made during System Generation, at initialization, or between each run in a job. These assignments are permanent unless another ASGN card is processed or the system is reinitialized.

Because reassignments are made as they are requested in the program, care should be taken when both the Standard Print Unit and the Standard Punch Unit are reassigned. There are no look-ahead provisions; hence, different logical results may occur. For example, the first assignment for the Standard Print Unit and the Standard Punch Unit might define both files on the same tape unit, A6. In reassigning these files, two of the options are:

```
(1) MON$$    ASGN    SPR,A5
    MON$$    ASGN    SPU,A6

(2) MON$$    ASGN    SPU,A6
    MON$$    ASGN    SPR,A5
```

If the first option is used, the Standard Print Unit is assigned to A5 when the first card is processed. A6 remains open to preserve the original assignment of the Standard Punch Unit. When the second card is processed, A6 is rewound and unloaded. A new Standard Punch Unit is defined on another tape, A6. Three physical tape reels are involved in this reassignment: the original assignment for the Standard Print Unit and the Standard Punch Unit placed these files on A6; the Standard Print Unit was reassigned to A5; and the Standard Punch Unit was reassigned to a new A6.

If the order of the assignment is reversed, as in the second option, only two physical tape reels are used. After the first card is processed, no change occurs to A6. When the second card is processed, a new printer file is defined on A5. A6 still contains the Standard Punch Unit.

When tape-to-tape reassignment of the Standard Print Unit and the Standard Punch Unit is made within a job, three conditions are possible:

1. If the Standard Print Unit and the Standard Punch Unit are assigned to the same output tape, and either the Standard Print Unit or the Standard Punch Unit is later reassigned to the same output tape, this tape will not be rewound.
2. If the Standard Print Unit and the Standard Punch Unit are assigned to the same output tape, and either the Standard Print Unit or the Standard Punch Unit is later reassigned to a different tape unit, the following results occur:
   a. The mixed output tape will not be rewound.
   b. The reel on the physical unit that is called for by the new assignment will be rewound when the unit is opened. A copy of POW will be written if it was specified during System Generation (see the GEN08 card).
3. If the Standard Print Unit and the Standard Punch Unit are not assigned to the same tape unit, and a reassignment of either file is made, the following results occur:
   a. The tape unit to which the file was first assigned will not be rewound and unloaded.
   b. The reel on the physical unit that is called for by the new assignment will be rewound when the unit is opened. A copy of POW will be written if it was specified during System Generation.

NOTE: If reassignment of the Standard Print Unit and the Standard Punch Unit puts both files on the same tape, the resultant mixed tape will not be rewound.

In the event that the system is reinitialized and no assignment is made for the Core Image file, a subsequent checkpoint or dump will cause the checkpoint or dump routine to be entered. However, nothing will be written on the Core Image file.

If the Core Image file is shared with any other system file, either in the current run or in the current batch, a utility storage print cannot be executed from the Core Image file. To obtain a storage print from a shared tape, that tape must be assigned to MW0.

For example, the Standard Print Unit, the Standard Punch Unit, and the Core Image file share A8 and a storage print is requested. The following assignments could be made:

MON$$	ASGN	MDM	the MDM assignment is canceled
MON$$	ASGN	SPR,A7	the SPR is reassigned to A7
MON$$	ASGN	SPU,SPR	the SPU is reassigned to A7
MON$$	ASGN	MW0,A8	the shared tape, A8, is assigned to MW0

```
MON$$   EXEQ    UTILITIES
        DUMP    CORE,MW0,ALL,,,u
```

a storage print is given for all core dump records on the shared tape, i.e., A8

Before A8 is assigned to MW0, it is rewound and unloaded. It must be reloaded before the storage print can be executed.

When restarting from a checkpoint, a function of the restart program is to reposition the system files. If the system files are on a shared tape, the tape might not be repositioned correctly. This will happen if all the files on the shared tape *except the Standard Print Unit or the Standard Punch Unit* have been reassigned. For example, assume that the first assignments for the system files were:

```
MON$$   ASGN    SPR,A7
MON$$   ASGN    SPU,A7
MON$$   ASGN    MDM,A7
MON$$   ASGN    MDT,A7
```

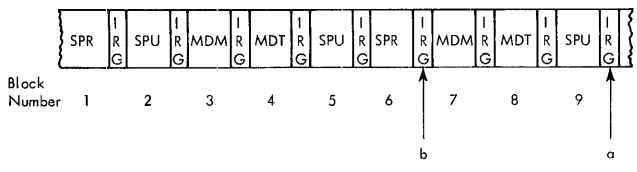
Before a checkpoint is taken, the following reassignments are made:

```
MON$$   ASGN    SPU,A8
MON$$   ASGN    MDT,A8
```

If a restart is called for, A7 will be repositioned correctly because the Standard Print Unit and the Core Image file are still on A7. However, assume that instead of the above reassignments, the following reassignments are made:

```
MON$$   ASGN    SPR,A8
MON$$   ASGN    MDM,A8
MON$$   ASGN    MDT,A8
```

If a restart is called for, A7 will not be repositioned correctly because the block count for repositioning reflects only the Standard Punch Unit records. Since the records of the four system files have been interspersed on A7, and since the tape will be repositioned according to the block count of the Standard Punch Unit only, a number of records (including Standard Punch Unit records) will be lost. For example, A7 has the following format:



where a is the position of the tape before the restart and b is the position of the tape after repositioning. The Standard Punch Unit block count is three. If a restart is called for, A7 will be repositioned (backspaced three blocks to block 7). The record prior to block 7 will be lost. A7 can be repositioned to correct the block count only by reassigning the Standard Punch Unit to A7, i.e.:

```
MON$$   ASGN    SPR,A8
MON$$   ASGN    MDM,A8
MON$$   ASGN    MDT,A8
MON$$   ASGN    SPU,A7
```

Linkage Sequence for Assignment of Symbolic Units

Through the IBLOOKM linkage sequence, a dependent program during its execution can set the symbolic unit selection for any Reserve file, Work file, or Tele-processing system file. Using this sequence, the dependent program is written to operate with any symbolic unit. It is stored in absolute format in the System Operating File. If this sequence were not used, the program would have to be processed by the Linkage Loader each time to resolve the one symbolic unit entry that would change with each execution.

The linkage sequence is:

```
DCWS    IBLOOKM
DCW     @SYM@
DCW     FILE
B       (address of dependent program instruction to which control is to be given if symbolic unit requested is not a Reserve, Work or Tele-processing system file or does not exist)
        (Next sequential instruction in dependent program)
```

where:

SYM = The symbolic name of the symbolic unit to be used by the dependent program.

FILE = The label used in the Define the File Header Line for the corresponding file.

NOTE: The IBLOOKM module must be on the currently assigned System Library file.

To use this sequence, a control card can be read during program execution to supply the information for the SYM operand. Such a program could be a Utility program written to print any tape file.

During operation, the System Monitor types information and diagnostic messages on the console printer and/or prints them on the Standard Print Unit. If an installation does not specify a Standard Print Unit at System Generation, it will not receive certain information and messages that are printed only on the Standard Print Unit.

The System Monitor can also receive inquiries through the console printer.

Memory Map

The Linkage Loader will write information on the Standard Print Unit to provide a memory map of the program being relocated. This memory map information will be interspersed with Linkage Loader diagnostic messages.

The column headings printed at the top of the memory map are:

MESSAGE CARD NAME VALUE SYMBOL VALUE SYMBOL
VALUE SYMBOL VALUE SYMBOL VALUE SYMBOL VALUE

All Linkage Loader error messages will begin in the MESSAGE column.

The CARD column contains the name of the following five key Linkage Loader control cards in the order in which they are processed. All information about each card will be printed on one line.

PHASE Card: The word PHASE will appear in the CARD column. The program name will appear in the NAME column for a PHASE card with an operand; for a blank-operand PHASE card, the NAME column will be blank. The upward relocation factor for the phase will be printed in the leftmost VALUE column. Note that if a BASE1 card is processed after the PHASE card, but before any load cards, the VALUE column of the PHASE card line will not be accurate.

BASE1 Card: The word BASE1 will appear in the CARD column. The leftmost VALUE column will contain the new upward relocation factor.

BASE2 Card: The word BASE2 will appear in the CARD column. The leftmost VALUE column will contain the new downward relocation factor.

PRTCT Card: The word PRTCT will appear in the CARD column. The leftmost VALUE column will contain the new protection value for the Linkage Loader symbol table.

TITLE Card: The word TITLE will appear in the CARD column. The subprogram name will appear in the NAME column. The leftmost VALUE column will contain the absolute address at which the first character in the subprogram will be loaded. To the right of the leftmost VALUE column will be printed REL FACTOR XXXXX, where XXXXX is the relocation factor actually used in relocating the subprogram.

The linkage symbols used, and their corresponding absolute addresses, will be printed in the five sets of SYMBOL and VALUE columns in the right half of the memory map.

Monitor Diagnostic Messages

The following types of messages are typed on the console printer and/or printed on the Standard Print Unit:

1. Diagnostic messages for programmers
2. Messages for operators
3. Initialization messages
4. Bootstrap routine messages
5. Messages during restart

Diagnostic Messages for Programmers

Diagnostic messages for programmers are written on the console printer, on the Standard Print Unit (SPR), or both. The messages that may be issued, their explanations (including an indication of where the messages appear), and their corrective actions, if any, are listed below.

UEPx

Explanation: (console printer) Unusual end of program. Execution of program has been terminated or the Transitional Monitor has determined that the remainder of the job cannot be successfully executed.
If x = 1: Dependent program requested a phase that could not be found.
If x = 2: Dependent program requested a phase to be loaded and executed. Phase was loaded, but has no entry point.
If x = 3: Dependent program attempted input/output with a symbolic unit for which no physical unit was assigned. Or, a file table in the dependent program contains a SYMUNIT field with an address that does not correspond to any entry in the Resident Monitor's table of symbolic units.
If x = 4: Dependent program attempted input/output with a symbolic unit which was assigned to a physical unit of a device type other than that requested by the dependent program.

If x = 5: Dependent program requested a phase that could not be loaded because of a permanent I/O error.
If x = 9: A branch to the /UEP/ entry point of the End-of-Program routine was made from outside the Resident Monitor. When a dependent program branches to /UEP/, the message will contain the contents of the B-Address Register at the time of the branch. If the address is absent, i.e., UEP9 message, the Transitional Monitor entered the End-of-Program routine at the /UEP/ point and the status of core storage was not written on the Core Image file.
NOTE: If x = 3 or 4, the message contains the address of the units position of file table field before the SYMUNIT field.
Action: None by operator.

00501 SOF DIRECTORY NOT FOUND
Explanation: (Console printer and/or Standard Print Unit) Transitional Monitor is unable to find a directory for the System Operating File.
Action: None by operator. (System Operating File must be regenerated.)

00502 MJB DIRECTORY NOT FOUND
Explanation: (Console printer and/or Standard Print Unit) Transitional Monitor is unable to find a directory for the disk Job file. The directory was inadvertently destroyed, an error was made in assignment, or the Linkage Loader is not functioning properly.
Action: Reinitialization is necessary, if the batch is to be continued. The situation should first be analyzed to determine the exact cause of error. Regeneration of the System Operating File may be required.

00503 INVALID DIRECTORY RECORD
Explanation: (Console printer and/or Standard Print Unit) Transitional Monitor has located what should be the directory for the disk System Operating File, but record read was not a valid directory record.
Action: None by operator. (System Operating File must be regenerated.)

00504 SIU/AIU SWITCH FAILED
Explanation: (Console printer) The switch to either the SIU or the AIU has failed.
Action: Check assignment and device type for AIU and SIU. Rerun.

10520 CGO-LL
Explanation: (Console printer and/or Standard Print Unit) A language processor has canceled the Go file and execution of the Linkage Loader is, therefore, being bypassed.
Action: None by operator.

10521 CGO-MJB
Explanation: (Console printer and/or Standard Print Unit) The contents of the Job file are not valid; therefore, the EXEQ card for a program on the Job file is being bypassed.
Action: None by operator.

10522 1ST [2ND, 3RD] OP INV
Explanation: (Console printer and/or Standard Print Unit) An EXEQ card with an invalid operand has just been read and is being bypassed.
Action: None by operator.

10523 NO AIU
Explanation: (Console printer and/or Standard Print Unit) An EXEQ card with a third operand has just been read but cannot be processed because facilities for using an Alternate Input Unit were not specified at System Generation.
Action: None by operator.

10524 AIU ACTIVE
Explanation: (Console printer and/or Standard Print Unit) An EXEQ card with a third operand has just been read from the Alternate Input Unit and is, therefore, being bypassed.
Action: None by operator.

10525 CHG ttttt
Explanation: (Console printer and Standard Print Unit) The symbolic unit in the third operand of the EXEQ card just read cannot be found in the Resident Monitor's assignment table. Therefore, the Transitional Monitor cannot effect a change to that unit. The EXEQ card is bypassed.
ttttt = The address of the Transitional Monitor routine that tried to effect the change.
Action: None by operator.

10526 CHG UNASGN
Explanation: (Console printer and Standard Print Unit) The symbolic unit in the third operand of the EXEQ card just read is not currently assigned to a physical unit. The EXEQ card is bypassed. (This message possibly could occur if parts of the Resident Monitor were destroyed, breaking the assignment for the Standard and/or Alternate Input Unit.)
Action: None by operator.

10527 CHG INV
Explanation: (Console printer and Standard Print Unit) The symbolic unit in the third operand of the EXEQ card just read is currently assigned to a disk physical unit, which is not permissible. The EXEQ card is bypassed.
Action: None by operator.

10529 GO DSK EXC
Explanation: (Console printer) During compilation, the disk physical unit assigned to the Go file has been exceeded. The language processor has canceled the Go file and returned control to the System Monitor.
Action: None by operator.

10530 NOT FOUND
Explanation: (Console printer and/or Standard Print Unit)
Tape System Operating File: The program named in the EXEQ card cannot be found on the System Operating File (if SOF is specified on card) or on the Job file (if MJB is specified on card). The EXEQ card is bypassed.
Disk System Operating File: The program named in the *EXEQ* card cannot be found in the directory specified by the card. The EXEQ card is bypassed.
Action: None by operator. (The System Operating File or Job file should be analyzed to determine the reason for the failure, in a tape system. The directory for the System Operating File or Job file should be analyzed to determine the reason for the failure, in a disk system.)

10531 NOT FOUND IN SOF DIRECTORY
Explanation: (Console printer and/or Standard Print Unit) The program named in the EXEQ card is not listed in the tape System Operating File directory. The EXEQ card is bypassed.
Action: None by operator. (The tape System Operating File directory should be analyzed to determine the reason for the failure.)

10532 INCOMPATIBILITY IN BLOCK COUNTS
Explanation: (Console printer and/or Standard Print Unit) Using block counts to search the System Operating File, the Transitional Monitor is unable to locate the desired program. One correction attempt will be made and, if not successful, the System Operating File will be rewound and sequentially searched.
Action: None by operator.

NOTE: The frequent appearance of this message indicates one or more of the following conditions:

1. Block counts in directory are wrong.
2. Block counts in phase headers are wrong.
3. Backspace often fails on the tape unit.

Correction of the trouble will improve search time for the System Operating File.

10535 MJB INVALID LOC

Explanation: (Console printer and/or Standard Print Unit) In searching the Job file, the Simple Search subroutine has read one record from the file; this record is not a phase header record. The two possible causes are:

1. The physical unit assigned to the Job file does not contain the Job file.
2. The Job file tape has been moved to an incorrect position.

Action: None by operator. The System Monitor will give control to the End-of-Program routine at the /UEP/ entry point.

Messages for Operators

All operator messages, their explanations, and their corrective actions, if any, are listed below. All such messages appear only on the console printer.

ASGN OP INVALID xnn

Explanation: ASGN card operand is invalid. System Monitor rejects assignment and enters End-of-Program routine at /UEP/ entry point.

nn = Number of ASGN card operand that was invalid.

x = (blank): Either the symbolic unit or the physical unit could not be found in the Resident Monitor's assignment table.

x = 4: Sequence rules for sharing physical units were violated. (For example, if "MW2,A1,A2" is a current assignment, then "MW5,A2,A3" cannot be made.)

x = 8: Rules for sharing physical units were violated. (For example, the physical unit used by the Standard Input Unit cannot also be assigned to another symbolic unit.)

x = T: Attempt was made to assign a Tele-processing system file when the Tele-processing system was open.

Action: None by operator unless this message is followed by (ASGN card image) INVALID ASSIGNMENT, in which case the procedure for that message should be followed.

END AIU

Explanation: END card or end of file has occurred on the Alternate Input Unit.

Action: None by operator.

END SIU

Explanation: END card or end of file has occurred on the Standard Input Unit.

Action: None by operator.

ENTER B MESSAGES

Explanation: Transitional Monitor is ready to accept class B messages from operator.

Action:

1. Press INQUIRY REQUEST.
2. If the HALT option was specified for the Wait-Loop routine, press START.
3. Enter message. (Class A messages are also acceptable at this time.)
4. Press INQUIRY RELEASE.

NOTE: This procedure is repeated for each message. The last message must be $BX to cause normal processing to resume.

INV A MSG

Explanation: An invalid console inquiry was entered when only class A messages can be accepted. The message has been ignored and processing resumed.

Action: Repeat the console inquiry if a valid class A message was intended. Otherwise, wait until 20501 ENTER B MESSAGES appears on the console printer before repeating the console inquiry.

INV B INQ

Explanation: Invalid console inquiry has been made while Transitional Monitor is ready to accept class A or B messages. Monitor remains in waiting loop.

Action:

1. Press INQUIRY REQUEST.
2. If the HALT option was specified for the Wait-Loop routine, press START.
3. Enter valid message. (If none, enter $BX.)
4. Press INQUIRY RELEASE.

INV CHAN CHAR IN INIT

Explanation: The channel character entered in the initialization instruction is invalid.

Action: Re-enter the initialization instruction, using a valid channel character.

SWITCH TO NEXT SIU W/O INITIALIZATION

Explanation: The System Monitor has begun reading the new Standard Input Unit batch in accordance with instruction from the console. This message occurs when a new batch has been placed on the Standard Input Unit, and the waiting loop following the end of the previous Standard Input Unit has been broken by the $BX inquiry.)

Action: None by operator.

10101 mms

Explanation: This message indicates a I/O error.

Action: See the *Operator's Guide* publication for corrective actions to be taken. To resume operation, turn the PRIORITY ON switch off and then on.

10103 CPT nnx

Explanation: Records have been written on the Core Image file, because an unusual end of program occurred or the current program took a checkpoint.

nn = Number of such records since the last initialization of the system.

x = C: Checkpoint.

x = D: Contents of core storage written on Core Image file following an unusual end of program.

Action: None by operator.

10104 REP INQ

Explanation: Console inquiry was made while a previous inquiry was being processed; error occurred during a console read; or console inquiry was canceled.

Action: Repeat console inquiry.

10190 xx

Explanation: This message indicates that the operator has tried to initialize SPOOL on a unit already processing SPOOL. xx is the assigned unit.

Action: Re-enter $B7 message on another unit or enter a $B7R release message.

10191 nnnnnnnnn

Explanation: This message indicates that the program is not on SOF.

Action: Re-enter $B7 message with another program name or enter a $B7R release message.

10192 SIP

Explanation: This message indicates that an attempt has been made to initialize a SPOOL Supervisor which is already in process.

Action: If the operator wishes to terminate the function in process he must:

1. Press PRIORITY ON and enter a $B7R release message to cancel current assignment of that unit.
2. Enter correct $B7 message.

10199 EOSn

Explanation: This message indicates that the SPOOL function has been completed on channel n.

Action: No action unless another file is to be processed. If another file is to be processed, turn the PRIORITY ON switch off and then on.

20502 SUB FOR SS

Explanation: An ASGN card has been read for a physical unit that is currently unavailable.

Action:

1. Press INQUIRY REQUEST.
2. Enter class C message: $C1ss.
3. Press INQUIRY RELEASE.

NOTE: The $B6ss message cannot be used at this time to remove the unavailable indication from the unit named in the 20502 SUB FOR SS message.

20503 INV AT END SITUATION

Explanation: END card or end of file has been encountered on the Alternate Input Unit, but a return to the Standard Input Unit cannot be effected. (Possible cause: contents of Standard Input Unit input area have been destroyed.) Transitional Monitor has entered a dead-end waiting loop.

Action: Reinitialization is necessary if batch is to be continued.

30101 cu

Explanation: This message indicates that the header label fields of an input tape do not match the fields in the SPOOL control type.

Action: See the *Operator's Guide* publication for actions to be taken.

Initialization Messages

Messages issued during initialization, along with their explanations and any associated corrective actions, are listed below. For details on initialization, see the *Operator's Guide* publication.

DATE NOT ENTERED

Explanation: First JOB card has been read, but no DATE card has been processed.

Action:

1. Press INQUIRY REQUEST.
2. Enter five-character date.
3. Press INQUIRY RELEASE.

ENTER ASGN FOR MDM

Explanation: Initialization routine is ready to accept physical unit symbols for the MDM.

Action:

1. Press INQUIRY REQUEST.
2. Enter two-position physical unit symbol for MDM.
3. Press INQUIRY RELEASE.

ENTER ASGN FOR MDM AND MDT

Explanation: Initialization routine is ready to accept physical unit symbols for the MDM and MDT.

Action:

1. Press INQUIRY REQUEST.
2. Enter two-position physical unit symbol for MDM and MDT respectively (e.g., A6B5).

ENTER DATE, I/O ASSIGNMENTS, JOB NUMBERS

Explanation: Initialization routine is ready to accept initialization information. (This message is given only during reinitialization without rewind.)

Action:

1. Press INQUIRY REQUEST.
2. Enter five-character date.
3. Press INQUIRY RELEASE.
4. Wait for next message, which should be "IO OR JN".

ENTER JOB NUMBERS

Explanation: Initialization routine is ready to accept job numbers for the Standard or Alternate Input Unit. (This message is given only during reinitialization with rewind.)

Action:

1. Press INQUIRY REQUEST.
2. Enter two-character job numbers (Standard Input Unit first). If the Alternate Input Unit is not included in an installation or was not assigned when the need for reinitialization arose, enter two blank characters for the Alternate Input Unit job count. (A total of four characters must be entered.)
3. Press INQUIRY RELEASE.

INV INITIAL CHAR

Explanation: Character at location 00000 is not a valid initialization status character. (That is, it is not A, B, C, or D.)

Action: Start over with a valid initialization status character.

IO OR JN

Explanation: Initialization routine is ready to accept input/output assignments or job numbers. (This message is given only during reinitialization without rewind.)

The first time this message appears during reinitialization, it indicates the date has just been processed. Each successive time the message appears, it indicates that the assignment just entered through the console printer has been processed; the Initialization routine is ready to accept the next assignment or, if all assignments have been given, the job numbers for the Standard and Alternate Input Units.

Action:

1. Press INQUIRY REQUEST.
2. Enter a five-character assignment (symbolic unit name followed by assignment symbol) or, if all assignments have been entered, enter two-character job numbers (Standard Input Unit first). If the Alternate Input Unit is not included in an installation or was not assigned at the time the need for reinitialization arose, enter two blank characters for the Alternate Input Unit job number.
3. Press INQUIRY RELEASE.

NOTE: An INQUIRY REQUEST and INQUIRY RELEASE must precede and follow each five-character assignment entry and the four-character job numbers entry.

IO OR JN RE ENTER xxx

Explanation: This message is given to permit the operator to correct an invalid assignment or to cancel the assignment by entering the job numbers for the Standard and Alternate Input Units.

Action:

1. Press INQUIRY REQUEST.
2. Enter five-character assignment or two two-character job numbers.
3. Press INQUIRY RELEASE.

MUST ENTER SOF/SIU ASGN SYMB

Explanation: System Operating File is on a unit different from the one specified at System Generation, but operator did not enter a word mark at location 00000 to request a change of assignment.

Action:

1. Press INQUIRY REQUEST.
2. Enter current assignments for both the System Operating File and Standard Input Unit. Enter the two-character assignment symbol for the System Operat-

ing File first, immediately followed by symbol for Standard Input Unit.

3. Press INQUIRY RELEASE.

ENTER RST NUMBER

Explanation: Initialization routine is ready to accept the number of the checkpoint that is to be used for the restart.

Action:

1. Press INQUIRY REQUEST.
2. Enter three-character checkpoint number.
3. Press INQUIRY RELEASE.

ENTER SOF/SIU ASGN SYMB

Explanation: Initialization routine is ready to accept the assignment symbols for the System Operating File and the Standard Input Unit. (This message is given only when a word mark has been entered into location 00000.)

Action:

1. Press INQUIRY REQUEST.
2. Enter assignment symbols for System Operating File and Standard Input Unit. Both must be entered even if only one assignment is being changed. System Operating File symbol must precede Standard Input Unit symbol.
3. Press INQUIRY RELEASE.

EOF SIU/AIU IN INIT

Explanation: Initialization routine has encountered an end-of-file condition on the Standard or Alternate Input Unit.

Action: Start over with the proper input on the unit.

INIT ASGN EXCEED TABLE

Explanation: Initialization routine for disk-oriented system has been given more initial assignments than can be stored in the table it is building for reinitialization. (Because the table will accommodate up to 150 assignments, this message is not normally expected.)

Action: Start over with fewer ASGN cards preceding the first JOB card.

(ASGN card image) INVALID ASSIGNMENT

Explanation: Initialization routine has read an ASGN card that cannot be processed. (The ASGN card could be constructed internally from an assignment entered through the console printer.)

Action:

1. Press INQUIRY REQUEST.
2. Enter a five-character assignment (e.g., MDMA1).
3. Press INQUIRY RELEASE.

SHARE VIOLATION, MONITOR COMPILATION

Explanation: At System Generation, the rules concerning the assignment of System files were violated. This violation is detected during the first initialization of the System Operating File containing the invalid assignments.

Action: None. (A new System Operating File must be generated to correct the invalid assignment(s).)

JOB SEARCH OUT OF STEP

Explanation: Something has caused the Initialization routine to search past the job count entered. (This message is given only during reinitialization with rewind.)

Action: Begin reinitialization again.

Bootstrap Routine Messages

All error messages, and their explanations, issued by the Bootstrap Routine are listed below. All messages appear only on the console printer; the corrective action for each message is to begin the bootstrap operation again.

BOOTSTRAP ERROR 01

Explanation: Bootstrap routine, in a tape system, has found a header record, rather than an execute record, after the Resident Monitor record on the System Operating File. The Bootstrap routine has entered a waiting loop.

BOOTSTRAP ERROR 02

Explanation: Tape read error, other than wrong-length record, persists after re-reading nine times. The Bootstrap routine has entered a waiting loop.

BOOTSTRAP ERROR 03

Explanation: A tape read error occurred while the Bootstrap routine was trying to read a header record. The Bootstrap routine has entered a waiting loop.

BOOTSTRAP ERROR 04

Explanation: The first record after the Bootstrap routine is not a header record. The Bootstrap routine has entered a waiting loop.

BOOTSTRAP ERROR 05

Explanation: The instruction loaded by the Bootstrap routine into location 00000 is not valid because it does not contain a non-overlap character in location 00002. The Bootstrap routine has entered a waiting loop.

BOOTSTRAP ERROR 06

Explanation: Disk read error persists after re-reading five times. The Bootstrap routine has entered a waiting loop.

BOOTSTRAP ERROR 07

Explanation: The Bootstrap routine has found an unidentifiable record while loading the Resident Monitor. The Boostrap routine has entered a waiting loop.

00222

Explanation: I-address of a halt. A tape read error, other than wrong-length record, occurred while the Bootstrap 2 routine was being read. The computer has halted.

00438

Explanation: I-address of a halt. The instruction loaded by the Bootstrap routine into location 00000 is not valid because it does not contain a valid non-overlap character in location 00002. The computer has halted.

Messages During Restart

Messages, and their explanations, issued during Restart operations are listed below. All messages appear only on the console printer; no operator action is required for any message.

10128 SPU-SPR SAME UNIT

Explanation: Because the Standard Punch Unit and Standard Print Unit share the same tape unit, the Restart program will make no attempt to reposition this unit during restart.

10133 SIU U/R

Explanation: Because the Standard Input Unit is a card reader, no attempt will be made by the Restart program to reposition the Standard Input Unit during restart.

10134 AIU U/R

Explanation: Because the Alternate Input Unit is a card reader, no attempt will be made by the Restart program to reposition the Alternate Input Unit during restart.

Linkage Loader Diagnostic Messages

The following is a list of diagnostic messages, and their explanations, issued by the Linkage Loader.

All messages appear only on the Standard Print Unit. Wherever a control card is printed, this fact is indicated following the messages. There is no operator action associated with any message; corrective action, if any, taken by programmers must be based on an analysis of the message explanation.

SOF LIBRARY NAME UNEQUAL TO EXEQ CARD
Explanation: The name of the relocatable library specified in the fourth operand of the EXEQ LINKLOAD card cannot be found on the System Operating File. The Linkage Loader will end execution with a special end of program.

/LIB/ HEADER NAME UNEQUAL TO EXEQ CARD
Explanation: The physical unit assigned to LIB does not contain the library named in the fourth operand of the EXEQ LINKLOAD card. The Linkage Loader will end execution with a special end of program.

INPUT CARD INVALID
Explanation: The operand of the INPUT card cannot be identified by the Linkage Loader. The Linkage Loader will end execution with a special end of program.

***** BAD INPUT CARD (contents of card)
Explanation: The card type in column 72 or the card name in column 16 is in error. (The *input card* can be any card, not just the INPUT control card.) The Linkage Loader will end execution with a special end of program.

***** BAD INDICATOR (contents of card)
Explanation: One of the relocation indicators on the card is in error. The Linkage Loader will end execution with a special end of program.

***** CORE EXCEEDED (contents of card)
Explanation: The relocated load address plus the size of the common data area is greater than the machine's highest addressable location. If the SG mode is ON, this is a warning message; Linkage Loader processing is continued. If the SG mode is OFF, the Linkage Loader will end execution with a special end of program.

***** SYMBOL UNDEFINED (contents of card)
Explanation: The symbol in the operand (for a control card) or used as the load address (for a load card) has not been defined. This is a warning message. Linkage Loader processing is continued.

***** TABLE OVERFLOW
Explanation: The Linkage Loader symbol table has been exceeded. The Linkage Loader will end execution with a special end of program. The user should reorder the program deck so that the number of forward references of linkage symbols is reduced.

***** SNAP IGNORED (contents of SNAP card)
Explanation: This message indicates that the subprogram named in the SNAP card has not been processed. The message can also indicate that columns 56 and 57 of the SNAP card were blank. The Linkage Loader ignores the Snapshot request and continues processing.

***** BAD LOAD ADDR (contents of card)
Explanation: After relocation, the load address in the printed card is in the Resident Monitor area. The Linkage Loader will end execution with a special end of program.

***** JOB FILE NOT CREATED
Explanation: This message indicates that (1) the Linkage Loader put no output on the job file, (2) the Linkage Loader read other subprogram cards before reading a PHASE or TITLE card, or (3) the first PHASE card read did not contain an operand. The Linkage Loader will end execution with a special end of program.

***** UNRESOLVED ENTRIES NONE
Explanation: All linkage symbols and calls were resolved. The Linkage Loader continues processing.

***** UNRESOLVED ENTRIES (list of unresolved linkage symbols and calls)
Explanation: All linkage symbols and calls were resolved except the ones listed. The format of the list of entries is:

xxxxxLINKSYMBOL
or
xxxxxLINKSYMBOL

where:
xxxxx is location where definition of symbol or call was needed
LINKSYMBOL is linkage symbol that was undefined
The plus zone over the low-order position of xxxxx indicates that the entry was an imbedded call. If xxxxx contains blanks, the subprogram call was made in a Linkage Loader control card.
The Linkage Loader continues processing.

***** DUPLDEFIN (contents of second DEFIN card)
Explanation: A second DEFIN card has been read for a linkage symbol in a program. The Linkage Loader continues processing and ignores the second DEFIN card.

***** FIRST LIB REC NOT DIR
Explanation: The first record of a relocatable library on disk was not a directory. A common cause of this error is the loading of the library onto the wrong area on the disk. The Linkage Loader will end execution with a special end of program.

***** ERROR (contents of card)
Explanation: This message indicates an error in specification of a chained program. The Linkage Loader will end execution with a special end of program.

*****END OF FILE INDICATOR WHILE ATTEMPTING TO PROCESS LIBRARY
Explanation: The area assigned to /LIB/ does not contain enough tracks. A module which is being processed resides outside this area. The area assigned to /LIB/ must be increased to contain the entire library. The Linkage Loader will end executions with a special end of program.

Console Inquiries

Console inquiries are used to communicate control information to various portions of the System Monitor. The messages that can be entered through the console are divided into three classes, as follows:

A. Messages that can be accepted by the Resident Monitor during the execution of a dependent program.

B. Messages that can be accepted by the Transitional Monitor between jobs.

C. Messages that are communicated to the Transitional Monitor in reply to a request from the Transitional Monitor for specific control information.

The messages are communicated to the System Monitor by:

1. Depressing the INQUIRY REQUEST key.
2. Typing in the message.
3. Depressing the INQUIRY RELEASE key.

Class A Messages

The following messages can be accepted during execution of a dependent program. Unless otherwise stated, the entire message consists of the three characters listed.

$10

Entry Time: Any time during execution of a dependent program.
Explanation: This message causes immediate entry to the /UEP/ point of the Resident Monitor's End-of-Program routine. This message cannot be used for programs under control of the Tele-processing Supervisor. NOTE: The operator can also branch to the /UEP/ entry by pressing the COMPUTER RESET key and then START key. However, this procedure resets certain machine indicators. Therefore, records on the Core Image file may not reflect the status of storage at the time the dependent program failed. (The RESET and START key procedure should not be used in a Tele-processing system.)

$20

Entry Time: Any time during execution of a dependent program
Explanation: This message requests the Transitional Monitor to notify the operator when it can accept class B messages. Notification is given after the next JOB card or EXEQ card.

$3x (any input wanted)

Entry Time: In answer to user-specified message or any time during execution of a dependent program.
Explanation: This message causes the Resident Monitor to place x in /MCI/. The entire message is put in the input area whose high-order position is designated by /RIQ/. This message is used to provide console input to the dependent program.

$50

Entry Time: Normally after a user-specified message. For FORTRAN, the user message is PAUSE nnnnn, where nnnnn is a five-digit number.
Explanation: This message causes the Resident Monitor to exit from its Wait-Loop routine and return control to the dependent program.
NOTE: If the HALT option was specified for the Wait-Loop routine, before entering the $50 message, the operator must:
1. Press INQUIRY REQUEST.
2. Press START.

The operator must press INQUIRY RELEASE after entering the $50 message.

$70

Entry Time: Any time during execution of a program that is requesting checkpoints and can be restarted.
Explanation: This message signals the Resident Monitor to initiate immediate restart procedures. (See the *Operator's Guide* publication.)

$8x

Entry Time: In reply to a 20101, 30101, or 30102 message from the Resident IOCS.
Explanation: This message, which has three forms (x = A, R, or C), is described in the *Operator's Guide* publication.

$90b (any input of up to 16 characters)

Entry Time: Any time.
Explanation: This message is used to communicate information to the Tele-processing system. (See the *Tele-processing Supervisor* publication.)

Class B Messages

The following messages can be accepted by the Transitional Monitor. When class B messages can be accepted, the Transitional Monitor notifies the operator by the console message: ENTER B MESSAGES. Class A messages can also be accepted at this time. This message is written if the operator previously entered the $20 inquiry, and each time an END card is read from the Standard Input Unit or Alternate Input Unit. The Transitional Monitor enters a waiting loop or, if the HALT option was specified, halts after writing this message.

Unless otherwise stated, each of the following messages consists entirely of the characters listed.

$B1

Explanation: This message causes the Resident Monitor's Read routine to be altered to read from the Alternate Input Unit. This message cannot be entered if the console message ENTER B MESSAGE follows an END SIU message.

$B2

Explanation: This message causes the Transitional Monitor to close, rewind, and unload the Standard Print Unit tape. After the tape is unloaded, the operator mounts another tape on the unit.

$B3

Explanation: This message causes the Transitional Monitor to close, rewind, and unload the Standard Punch Unit tape. After the tape is unloaded, the operator mounts another tape on the unit.
NOTE: This message is used only if the Standard Punch Unit is not assigned to the same tape as the Standard Print Unit. If both units use the same tape, then the $B2 message is used for the closing function.

$B4

Explanation: This message causes the Transitional Monitor to close, rewind, and unload the Core Image file tape. After the tape is unloaded, the operator mounts another tape on the unit.

$B5ss

Explanation: This message indicates to the Transitional Monitor that the physical unit represented by assignment symbol ss is currently unavailable for use.

$B6ss

Explanation: This message indicates to the Transitional Monitor that the physical unit represented by assignment symbol ss is now available for use. The Transitional Monitor removes the unavailable indication it had set in response to a $B5ss message for this physical unit.

$B7bxxyyb

Explanation: This message assigns physical units xx and yy as SPOOL units and indicates that no editing routine is to be loaded. Only TYPE I SPOOL will be performed as a result of this message, regardless of the SPOOL type placed in the Resident Monitor. This is a valid message for any of the SPOOL control types. Odd tape parity is assumed.

$B7bxxyynnnnnnnnnnb

Explanation: This message assigns physical units xx and yy as SPOOL units and causes loading of the editing routine. The editing routine name, NNNNNNNNNN, need not be ten characters, but a blank must be entered after the last character of the name. This message is valid only for TYPE II and TYPE III SPOOL. Odd tape parity is assumed.

$B7Bxxyynnnnnnnnnnb

Explanation: Same as above, specifying odd tape parity.

$B7Uxxyynnnnnnnnnnb

Explanation: Same as above, specifying even tape parity.

$B7Rxxyy

Explanation: This message releases physical units xx and yy from their SPOOL assignments. This message is valid for all SPOOL control types.

$BX

Explanation: This message indicates that the operator has no more class B messages at this time. It causes the Transitional Monitor to exit from the waiting loop it had entered to allow console inquiries. This message must be given to enable the Transitional Monitor to resume processing.

After the $BX message is entered, the Transitional Monitor completes functions related to previous class B messages. For example, the complete procedure for use of a $B2 message is:

1. The Transitional Monitor types out ENTER B MESSAGES to notify the operator it is ready to accept class B messages and then enters a waiting loop or halts.

2. If the machine is in a waiting loop, the operator presses INQUIRY REQUEST and enters the $B2 message. If a machine HALT has occurred, the operator presses INQUIRY REQUEST, presses START, and then enters the $B2 message. After the message is entered, the operator presses INQUIRY RELEASE.

3. The Transitional Monitor performs the closing functions for the Standard Print Unit and returns to the waiting loop or halts.

4. The operator mounts another tape for the Standard Print Unit. If the machine is in a waiting loop, the operator presses INQUIRY REQUEST and then enters the $BX message. If a machine HALT has occurred, the operator presses INQUIRY REQUEST, presses START, and then enters the $BX message. After the message is entered, the operator presses INQUIRY RELEASE.

5. The Transitional Monitor opens the file for the new tape and resumes other processing related to preparation for the next job.

When the $BX message is entered in answer to an ENTER B MESSAGES following an END SIU message, the Transitional Monitor resumes reading and processing from the Standard Input Unit and without initialization. Therefore, it will ignore any daily information, such as a DATE card, and will skip to the next JOB card.

Class C Messages

The following message is given in reply to a message from the Transitional Monitor.

Monitor Message Received

20502 SUB FOR SS

Explanation: This message indicates that an ASGN card specified an assignment symbol, SS, for a physical unit that is not available.

$C1ss

Explanation: This message specifies that the assignment symbol ss is to be substituted for the one contained in the ASGN card. The substituted physical unit must be currently available.

NOTE: The $B6ss message cannot be used at this time to indicate that the unit specified on the ASGN card is now available; it must have been entered before the ASGN card was read.

Relocatable subprograms on the System Library file or Go file can be corrected or changed by placing patch cards in the Standard or Alternate Input Unit after a CALLP card. The patching is performed by the Linkage Loader immediately after it processes the called subprogram.

Considerations for patching are:

1. Calls imbedded in a subprogram by DCWS, DCWF, and CALL statements cannot be negated through patching.
2. The functions of DEFIN, BASE1, BASE2, and PRTCT cannot be overridden through patching.
3. An instruction that contains a linkage symbol for an address can be patched only if the symbol in the original instruction has been resolved before the patch is processed.

Patch cards must be punched in the formats for relocatable object cards as described below.

All cards in a subprogram must have a card type punched in column 72. The card types are summarized in Figure 12.

Number in Column 72	Card Type
0	Load card with absolute location or linkage symbol as load address
1	Load card with upward relocation factor to be applied to load address
2	Load card with downward relocation factor to be applied to load address
3	Termination card for primary subprogram
4	DEFIN,BASE1,BASE2,PRTCT,ENTRY, or CALL card
5	TITLE card
8	Set word mark or record mark card
9	Termination card for secondary subprogram
W	Clear word mark card
Y	Clear storage between limits card
Z	Random load card
blank	LINK, COMN, or END card for chained program

Figure 12. Card-Type Summary

Column 71 and down of certain cards contain relocation indicators, which give to the Linkage Loader relocation information and characteristics of instructions and data in the card. The relocation indicators are summarized in Figure 13.

NOTE 1: Constants and instructions may not be on the same card.

NOTE 2: A linkage symbol, when used as an address, must indicate NO relocation for that section of the instruction.

Relocatable Object Cards

TITLE Card

The format for the TITLE card is:

Column	*Contents*
16-20	TITLE
21-30	(1- to 10-character subprogram name, left-justified, with trailing blanks)
31-35	(Origin point to which subprogram was compiled. Usually 00000 or blanks for Autocoder programs; 00001 for FORTRAN programs; the size of IBCOBOL for COBOL programs.)
41-45	(Total number of core-storage locations required by an Autocoder subprogram for a common data area.)
72	5

NOTE: TITLE cards as generated by a language processor *may* contain additional data, not used by the Linkage Loader, in columns 6-10 and 73-80. For a complete description of the TITLE card generated by a given compiler, refer to the applicable processor language manual.

Termination Card (for Subprograms)

The format for the subprogram Termination card is:

Column	*Contents*
1	E
2	Word separator character (0-5-8 punch)
3-7	(Entry point for primary subprogram, blanks for secondary subprogram)
72	3 (for primary subprogram) or 9 (for secondary subprogram)

The entry point address in columns 3 through 7 is the compiled address. It will be relocated by the Linkage Loader using the upward relocation factor for the subprogram.

Load Card

The format for the load card is:

Column	*Contents*
1	Word separator character (0-5-8 punch)
2-6	(Load address: Low-core address of the area into which the characters in columns 13 and up are to be loaded)

Column	*Contents*
7	Word separator character (0-5-8 punch)
8-10	Zeros
11-12	(Length in core storage. Number of characters to be loaded from this card.)
13-up	(Instructions or constants to be loaded into storage, left-justified) NOTE 1: To load a word mark into storage, punch a word separator character in the column preceding the character with which the word mark is to be associated. These word separator characters are not included in the count columns 11 and 12. To enter a word separator character into storage, punch word separator characters in two adajacent columns. Add only "1" to the count. NOTE 2: Instructions, DCWS calls, and DCWF calls may appear on the same card, but cannot appear on a card containing constants.
71-down	(Relocation indicators. The indicator punched in column 71 applies to the first instruction or constant in the card, the indicator in column 70 applies to the second instruction or constant, etc. Relocation indicators are given in Figure 13.
72	0 (for load card with absolute location or ABCD/ linkage symbol as load address. If the load address is a linkage symbol, the symbol must have been defined before the Linkage Loader reads this card.) 1 (for load card with upward relocation factor to be applied to numeric load address.) 2 (for load card with downward relocation factor to be applied to numeric load address.)

DCWS and DCWF calls may be included in the instructions in columns 13 and up. To include these calls, punch a word separator character followed by the 1- to 10-character subprogram name, left-justified, with trailing blanks in a 10-position field. The factor to be added into the count in columns 11 and 12 is seven for a DCWS and five for a DCWF.

Linkage symbols appearing as an address are given a NO relocation indicator.

NOTE: There must be at least one blank between the instructions and the relocation indicators on the load card.

DEFIN Card

The format for the DEFIN card is:

Column	*Contents*
6-15	(5-character or 1- to 10-character linkage symbol, left-justified, with trailing blanks)
16-20	DEFIN
21-30	(5-character numeric address or 1- to 10-character linkage symbol, left-justified, with trailing blanks. For a numeric address, index register tags in the tens and hundreds positions are permissible. If a linkage symbol is used, its value must have been defined before the Linkage Loader reads this card.)
71	Blank (Relocate upward) P (Relocate downward) Q (NO relocation) NOTE: These relocation indicators are for the address in columns 21 through 30. The Q relocation indicator is required if a linkage symbol is used in columns 21 through 30.
72	4

Description of Relocation	Without d-Modifier			With d-Modifier		
	Relocation Indicator	Card Code for Indicator	Length in Core Storage	Relocation Indicator	Card Code for Indicator	Length in Core Storage
No address	9	9	1	I	12-9	2
X-control operation, B-address upward	J	11-1	10			
X-control operation, B-address downward	K	11-2	10			
X-control operation, no relocation for B-address	L	11-3	10			
X-control operation, no B-address	M	11-4	5			
Single address, upward	@	4-8	6	⌑	12-4-8	7
Single address, downward	:	5-8	6	⌊	12-5-8	7
Single address, no relocation	>	6-8	6	<	12-6-8	7
Two addresses, no relocation	0	0	11	?	12-0	12
Two addresses, only B-address upward, no relocation for A-address	1	1	11	A	12-1	12
Two addresses, only B-address downward, no relocation for A-address	2	2	11	B	12-2	12
Two addresses, only A-address upward, no relocation for B-address	3	3	11	C	12-3	12
Two addresses, both upward	4	4	11	D	12-4	12
Two addresses, A-address upward, B-address downward	5	5	11	E	12-5	12
Two addresses, only A-address downward, no relocation for B-address	6	6	11	F	12-6	12
Two addresses, A-address downward, B-address upward	7	7	11	G	12-7	12
Two addresses, both downward	8	8	11	H	12-8	12
Address constant, upward	N	11-5	5			
Address constant, downward	P	11-7	5			
DCWS	S	0-2	7			
Address constant, no relocation	Q	11-8	5			
Constant	blank	none				
DCWF	T	0-3	5			

Figure 13. Relocation Indicators

BASE1 Card

The format for the BASE1 card is:

Column	*Contents*
16-20	BASE1
21-30	(Left-justified linkage symbol, absolute address, *+X00, or blank)
72	4

If a linkage symbol is used, it must have been defined before the Linkage Loader reads this card.

BASE2 Card

The format for the BASE2 card is:

Column	*Contents*
16-20	BASE2
21-30	(Left-justified linkage symbol, absolute address, or blank)
72	4

If a linkage symbol is used, it must have been defined before the Linkage Loader reads this card.

PRTCT Card

The format for the PRTCT card is:

Column	*Contents*
16-20	PRTCT
21-30	(Left-justified linkage symbol, absolute address, or blank)
72	4

If a linkage symbol is used, it must have been defined before the Linkage Loader reads this card.

ENTRY Card

The format for the ENTRY card is:

Column	*Contents*
16-20	ENTRY
21-30	(1- to 10-character linkage symbol, including subprogram name, left-justified, with trailing blanks, or an absolute address)
72	4

LINK Card

The format for the LINK card is:

Column	*Contents*
16-19	LINK
21-	(1- to 10-character linkage symbol followed by a comma, which may or may not be followed by a 1- to 10-character linkage symbol or an absolute address)

COMN Card

The format for the COMN card is:

Column	*Contents*
16-19	COMN
21-	(1- to 10-character linkage symbols separated by one or two commas)

END Card for Chained Programs

The format for the END card for chained programs is:

Column	*Contents*
16-18	END

CALL Card

The format for the CALL card is:

Column	*Contents*
16-19	CALL
21-30	(1- to 10-character subprogram name, left-justified, with trailing blanks)
72	4

Random Load Card

The format of the Random Load card is:

Column	*Contents*
1	Word separator character (0-5-8 punch)
2-6	(Load address: low-order address of area into which the characters in columns 7 through 11 are to be loaded)
7-11	(Five characters to be loaded in Move mode, beginning at the location specified in columns 2 through 6. These five characters are considered to be an address and will thus be relocated. Loading the five characters will not affect any word marks set by previous load cards.)
12-22	(Same as columns 1 through 11)
23-33	(Same as columns 1 through 11)
34-44	(Same as columns 1 through 11)
45-55	(Same as columns 1 through 11)
67	(Relocation indicator for columns 45 through 55)
68	(Relocation indicator for columns 34 through 44)
69	(Relocation indicator for columns 23 through 33)
70	(Relocation indicator for columns 12 through 22)
71	(Relocation indicator for columns 1 through 11)
72	Z

The relocation indicators to be used in columns 67 through 71 are:

RELOCATION INDICATOR	LOAD ADDRESS	FIVE CHARACTERS TO BE LOADED
0	No relocation	No relocation
1	No relocation	Relocate upward
2	No relocation	Relocate downward
3	Relocate upward	No relocation
4	Relocate upward	Relocate upward
5	Relocate upward	Relocate downward
6	Relocate downward	No relocation
7	Relocate downward	Relocate upward
8	Relocate downward	Relocate downward

These relocation indicators are also given in Figure 13.

If word marks are wanted over the five characters to be loaded, they must be set before the Random Load card is loaded into storage for program execution.

If less than five fields are needed, the relocation indicator corresponding to the first unused field must be left blank.

The load address and/or the five characters to be loaded may be an ABCD/ format linkage symbol. The corresponding relocation indicator must specify NO relocation.

Clear Storage Card

The format for the Clear Storage card is:

Column	*Contents*
1	Word separator character (0-5-8 punch)
2-6	(Lowest address of area to be cleared)
7	Word separator character (0-5-8 punch)

Column	*Contents*
8-12	(Highest address of area to be cleared)
71	N (Relocation indicator when the addresses in columns 2 through 6 and 8 through 12 are to be relocated upward) P (Relocation indicator when the addresses are to be relocated downward)
72	Y

Set Word Mark or Record Mark Card

The format for the Set Word Mark or Record Mark card is:

Column	*Contents*
1	Word separator character (0-5-8 punch)
2-6	(Address of first word mark or record mark to be set)
7	Word separator character (0-5-8 punch)
8-12	(Increment factor, in storage locations, used to determine the location of the second, and succeeding, word marks or record marks to be set)
13	Word separator character (0-5-8 punch)
14-18	(Number of word marks or record marks to be set)
20	Record Mark (0-2-8 punch). (If this card is to set record marks.) Blank (If this card is to set word marks.)
71	N (Relocation indicator, when the address in columns 2 through 6 is to be relocated upward) P (Relocation indicator, when the address in columns 2 through 6 is to be relocated downward)
72	8

Clear Word Mark Card

The format for the Clear Word Mark card is:

Column	*Contents*
1	Word separator character (0-5-8 punch)
2-6	(Address of first word mark to be cleared)
7	Word separator character (0-5-8 punch)
8-12	(Increment factor, in storage locations, used to determine the location of the second, and succeeding, word marks to be cleared)
13	Word separator character (0-5-8 punch)
14-18	(Number of word marks to be cleared)
71	N (Relocation indicator when the address in columns 2 through 6 is to be relocated upward) P (Relocation indicator when the address in columns 2 through 6 is to be relocated downward)
72	W

Operation of the System Monitor

Operation of the System Monitor is divided into two categories:

1. *Initialization* to prepare the Resident Monitor for work.
2. *Functioning* of the System Monitor to control the Operating System.

The System Monitor also enables restarting dependent programs from checkpoints.

Initialization of the System Monitor

Initialization of the System Monitor begins with bringing the Resident and Transitional Monitors into core storage from the System Operating File. They are brought into storage by the Bootstrap routine, which is the first element on the System Operating File. The Bootstrap routine is also used if reinitialization of the System Monitor becomes necessary because of an emergency. (For example, a dependent program might fail in such a way that portions of the Resident Monitor are destroyed.)

The Initialization routine is brought into storage with the Resident Monitor. It is given control to perform the housekeeping needed to prepare the Resident Monitor. For this housekeeping, the installation supplies the Resident Monitor with *daily information* control cards. These cards are the Monitor DATE card and ASGN cards for certain files used by the System Monitor.

Reinitialization procedures include facilities for repositioning the Standard or Alternate Input Unit, if tape, to the next job specified by the operator.

For full details on initialization and reinitialization, see the *Operator's Guide* publication.

Functioning of the System Monitor

Example 1

To illustrate functioning of the System Monitor, assume that a program is to be compiled, relocated, and executed. Control cards for such a program are shown in Figure 14. These cards are in a card reader Standard Input Unit.

Both the Resident Monitor and Transitional Monitor are left in storage by initialization. During initialization, the Transitional Monitor had just read and processed the daily information cards on the Standard Input Unit. When the Transitional Monitor reads the first JOB card, Monitor functioning begins.

Following the JOB EXAMPLE card are COMT, ASGN, and MODE cards. Note that for the language processors the MODE GO card is placed before the EXEQ cards. The Transitional Monitor processes these cards, then reads the first EXEQ AUTOCODER card. It locates the Autocoder language processor on the System Operating File and branches to the Resident Monitor's Load routine with a request to load the Autocoder processor. A Clear Storage routine from the System Operating File is loaded immediately above the end of the Resident Monitor. The Clear Storage routine clears all of storage above the Resident Monitor, thus destroying the Transitional Monitor, except for the first 99 positions above the Resident Monitor and the last two positions at the top of core storage. After storage is cleared, the Load routine brings in the Autocoder processor.

The Autocoder processor reads its source statements from the Standard Input Unit. It uses three Work files, assigned by the daily information, as temporary storage during the compilation. The processor places its output (relocatable card-image records) on the Go file, because MODE GO was specified, and also punches out a relocatable object deck on the Standard Punch Unit. Use of the tape Go file eliminates the operator intervention that would have been required with only punched output.

When the Autocoder processor is finished, it gives control to the Resident Monitor's End-of-Program routine at the /EOP/ entry point. The Resident Monitor brings in the Transitional Monitor. The Transitional Monitor resumes reading the Standard Input Unit. The next card is the second EXEQ AUTOCODER card. For this card, the System Monitor repeats the procedure performed for the first EXEQ AUTOCODER card.

The Autocoder processor must be executed twice, because the source statements for the two subprograms are on different files.

In its second execution, the Autocoder processor places its output on the Go file after the output for subprogram A.

When the Autocoder processor is finished, it returns control to the Resident Monitor, which brings in the Transitional Monitor. The Transitional Monitor reads the Standard Input Unit, assigning the Job file (MJB) to physical unit B2.

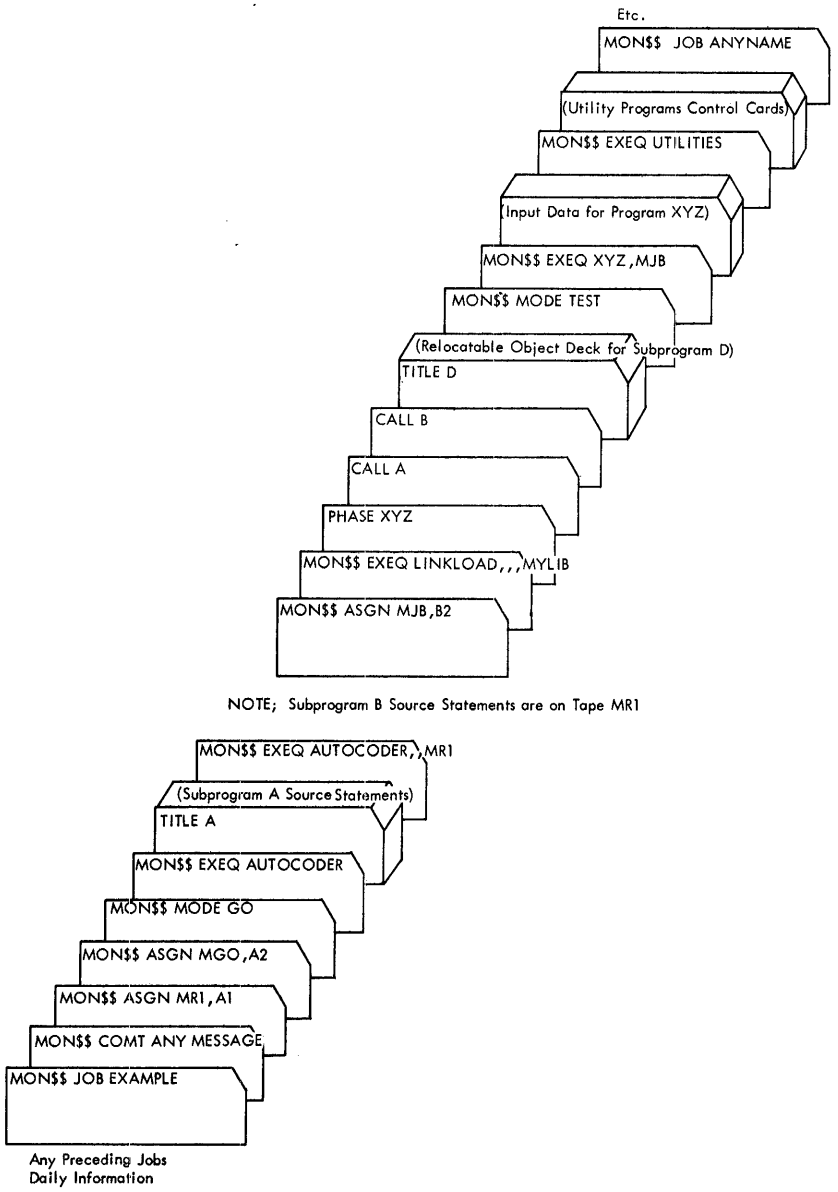
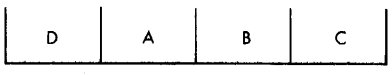


Figure 14. Control Card Deck for First Example

Then the Transitional Monitor, for the EXEQ LINKLOAD card, repeats the procedure performed for the language processor's EXEQ cards to bring the Linkage Loader into storage.

In this example, the Linkage Loader reads the CALL cards for subprograms A and B and puts the calls on the request list. Then it reads and processes subprogram D. The Linkage Loader continues reading and finds a Monitor control card. The Monitor control card forces the Linkage Loader to read in and process all subprograms on the request list. Thus, the Linkage Loader calls in and processes the newly compiled subprograms A and B from the Go file. Subprogram A contains an imbedded call for subprogram D, which has already been processed. Subprogram B contains an imbedded call for subprogram C, which is on a Library named MYLIB on the System Operating File. The fourth operand of the EXEQ LINKLOAD card designates MYLIB as the System Library file for this Linkage Loader run. The Linkage Loader, when it processes subprogram B, puts subprogram C on the request list. Because no more Linkage Loader control cards exist, the Linkage Loader then calls in subprogram C from the Library.

The memory map for these subprograms, which form program XYZ, will be:



Note that A was before B on the Go file, because A was compiled before B.

The Linkage Loader puts program XYZ, in absolute format, on the Job file. If the installation wants to keep using this program in absolute format, the operator should save the Job file tape reel after the program is executed. This reel can later be mounted on a physical unit that has been assigned to the Job file.

When the Linkage Loader is finished, it returns control to the Resident Monitor, which brings in the Transitional Monitor. The Transitional Monitor reads the MODE card. Because two of the subprograms are newly compiled, MODE TEST is specified before the program is executed. The last subprogram to be executed ends with B /UEP/ in order to return control to the Resident Monitor's End-of-Program routine at the /UEP/ entry point.

When the Transitional Monitor reads the EXEQ XYZ card, it repeats the procedure used to bring in the language processors and Linkage Loaders, except that it loads phase 001, named XYZ, from the Job file. Note that dependent programs, whether IBM-provided or user-written, are given the same processing by the System Monitor.

In this example, subprogram B was the primary subprogram. Therefore, processing control is initially given to the entry point for subprogram B.

The program, during execution, reads input data from the Standard Input Unit through the Resident Monitor's Read routine.

If the program fails, the operator gives control to the Resident Monitor's End-of-Program routine through the /UEP/ entry point. If the program executes successfully, it also gives control to the /UEP/ entry point. In either case, the End-of-Program routine will write the contents of storage on the Core Image file, and the Resident Monitor will bring in the Transitional Monitor.

The Transitional Monitor, because the unusual end of program occurred in TEST mode, will read the Standard Input Unit for *any* Monitor control cards, rather than skipping to the next JOB card. If the program fails before all input data cards are read, the Transitional Monitor will skip past them and resume processing when it finds the first Monitor control card. Thus, the Transitional Monitor will read the EXEQ UTILITIES card. It will bring the Utility programs into storage from the System Operating File in the same way it brought in the other dependent programs.

When the Utility programs are finished, they will return control to the Resident Monitor, which will bring in the Transitional Monitor. The Transitional Monitor, upon resuming reading from the Standard Input Unit, will read the JOB ANYNAME card. It will close out the previous job, named EXAMPLE, and reset the Resident Monitor for the new job.

The above example shows just one of the many job combinations that are possible. For example, the subprograms compiled could form a user-written language processor. This new processor could be relocated by the Linkage Loader and executed to compile a program. The newly compiled program could then be relocated and executed.

The example showed execution of only one program. The job could have included execution of many programs.

Example 2

The following example shows the complete control card deck for a batch. The machine configuration for this example is shown in Figure 15. Figure 16 illustrates the control card deck for this example. Figure 17 presents a chart of input/output assignments as they are made from the ASGN cards in the control card deck.

The following comments give information concerning the makeup of the control card deck and explain the changing input/output assignments summarized in Figure 17.

Cards 1-8

These cards constitute the daily information for initialization of the System Monitor. In addition to supplying the current date, these cards assign input/output units

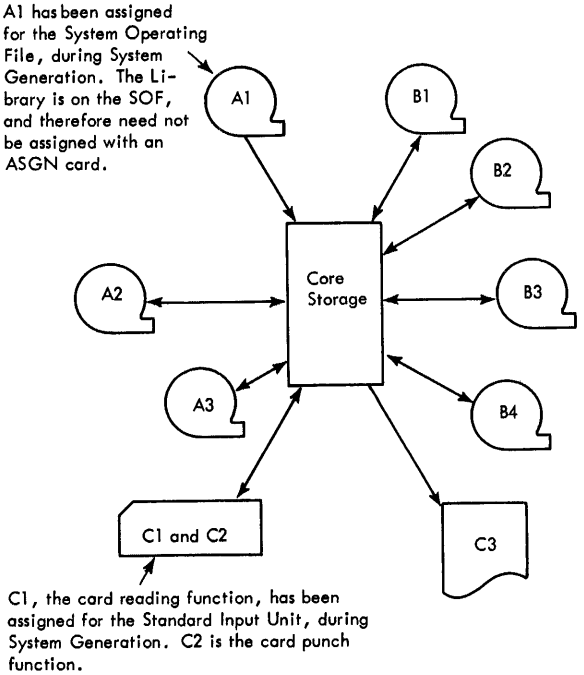


Figure 15. Machine Configuration for Second Example

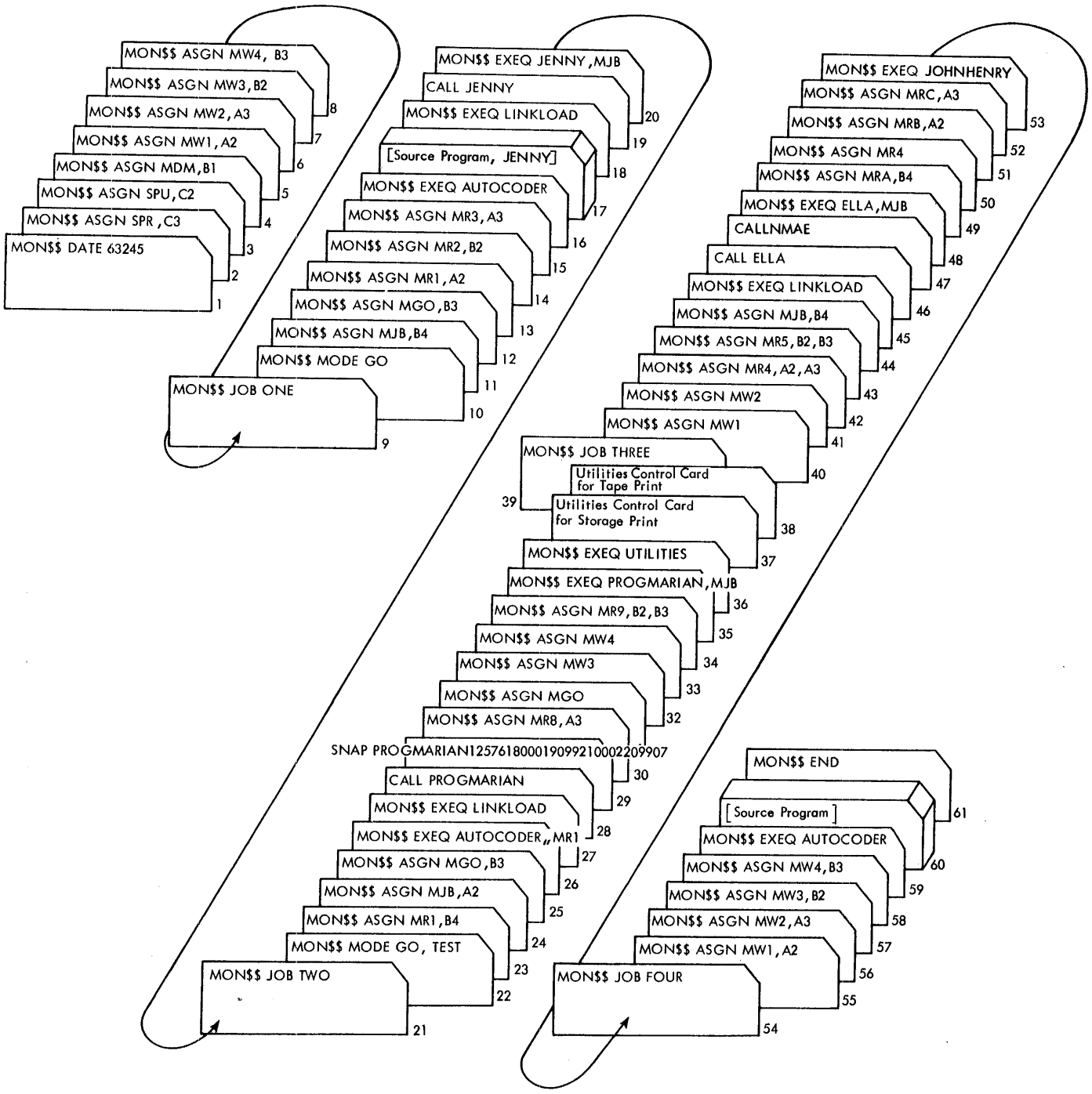


Figure 16. Sample Control Card Deck for Second Example

Units	Initial-ization	First Job	Second Job	Third Job	Fourth Job
A1	SOF	SOF	SOF	SOF	SOF
A2	MW1	MW1 and MR1	MW1 and MJB	1. MW1 cancelled 2. MR4 (base unit) 3. MR4 cancelled 4. MRB	MW1
A3	MW2	MW2 and MR3	MW2 and MR8	1. MW2 cancelled 2. MR4 (alternate unit) 3. MR4 cancelled 4. MRC	MW2
B1	MDM	MDM	MDM	MDM	MDM
B2	MW3	MW3 and MR2	1. MW3 2. MW3 cancelled 3. MR9 (base unit)	MR5 (base unit)	MW3
B3	MW4	MW4 and MGO	1. MW4 and MGO 2. MW4 and MGO cancelled 3. MR9 (alternate unit)	MR5 (alternate unit)	MW4
B4	un-assigned	MJB	MR1 source program tape for Autocoder	MJB and MRA	unas-signed
C1	SIU	SIU	SIU	SIU	SIU
C2	SPU	SPU	SPU	SPU	SPU
C3	SPR	SPR	SPR	SPR	SPR

Figure 17. Assignments of Input/Output Units for Second Example

for the Standard Print Unit, the Standard Punch Unit, the Core Image file, and the language processors' Work files.

Cards 9-20

These cards define the first job of the batch. The job consists of a compile-and-go operation for a single subprogram.

Cards 21-38

These cards define the second job. This job consists of a compile-and-go operation for a single subprogram to be tested. The Snapshot program is to be incorporated into the subprogram. The Storage Print and Tape Print programs are to be executed after execution of the subprogram. Note that for the compilation, the source statements are to be mounted on tape physical unit B4. Also note that the test program will use tape physical units B2 and B3 as base and alternate Reserve files. Since B2 and B3 have been previously assigned as base units only, the previous assignments must be canceled.

Cards 39-53

These cards define the third job. This job consists of the execution of the Linkage Loader to combine two subprograms from the Library, the execution of the resulting program, and then the execution of one of the installation's programs contained on the System Operating File.

Cards 40 and 41 cancel the assignment of two Work files to permit A2 and A3 to be used as base and alternate Reserve files. Note that the assignment made by card 43 is permissible because the last effective assignment for B2 and B3 was for a Reserve file, and that assignment was automatically canceled between jobs. Card 50 cancels the assignment made by card 42, permitting A2 and A3 to be used as base units for the next program within the job.

Cards 54-60

These cards define the fourth and last job in the batch. This job consists of a single compilation. It is included in the sample batch to illustrate the need for recording the changing input/output assignments, so that canceled assignments for symbolic units can be restored when those symbolic units are again required. In this job, the four Work files assigned during initialization must be assigned again, since during the course of the preceding jobs, all the Work file assignments have been canceled (cards 32, 33, 40, and 41).

Restarting from a Checkpoint

The System Monitor provides facilities to restart certain programs that were discontinued because of certain error conditions recognized by the operator or because a program of higher priority, to be loaded from the Alternate Input Unit, required the machine. Also, the processing of some programs requires so much time that the processing is performed in installments.

NOTE: Only programs that request checkpoints on the Core Image file can use the restart facilities of the System Monitor. (See "System File Sharing and Device Switching Feature.")

The operator discontinues programs (that have taken checkpoints and will be restarted) by using the $10 or $3x console inquiry. The $3x inquiry can be used only if the program recognizes it as a request to discontinue at some convenient point (checkpoints were taken previously) or as a request to take a checkpoint and then discontinue.

Restarting is described in detail in the *Operator's Guide* publication.

Immediate and Delayed Restarts

The Resident IOCS provides a routine for writing checkpoint records on the Core Image file. (See the *Basic Input/Output Control System* publication for details concerning the creation of checkpoint records.) These records consist of the information in core storage at the time the checkpoint is taken. From these records the System Monitor can restart a program in one of two ways:

Immediate Restart: An immediate restart can be effected by entering the $70 message through the console printer. This message causes the Resident Monitor to bring the Transitional Monitor into storage. The Transitional Monitor then executes a Restart routine, which locates the most recent checkpoint on the Core Image file and initiates a restart of the dependent program from that checkpoint. The major functions performed for the immediate restart are as follows:

1. Reloading the dependent program into core storage.
2. Restoring areas of the Resident Monitor related to the status of the dependent program at the time the checkpoint was taken.
3. Repositioning the necessary system tape files and all the user's tape files that were open at the time the checkpoint was taken.

NOTES: (1) The Standard Print and Punch Units, if assigned to the same tape unit, are not repositioned during a restart. (2) If the variable print and punch modules have been selected during System Generation, and if the Standard Print Unit, Standard Punch Unit, and Core Image file are on the same tape unit, that unit will be repositioned to a point just beyond the checkpoint taken.

4. Resuming execution of the dependent program at the point at which the checkpoint was taken.

Delayed Restart: A delayed restart is effected by initialization of the System Monitor. At the time of an initialization for the purpose of a delayed restart, the operator supplies control information specifying the particular checkpoint from which the dependent program is to be restarted. (Note the difference from the immediate restart, which uses only the most recent checkpoint.)

The functions performed for the dependent program during a delayed restart are essentially the same as those performed during an immediate restart. Care must be exercised, however, that the configuration at the time the checkpoint was taken is duplicated for the restart procedures.

Restart Conditions

The following considerations apply to programs using the checkpoint and restart facilities of the System Monitor:

1. No repositioning can be performed for unit-record equipment.
2. After the checkpoint is requested, the restart procedure cannot restore disk records that have been changed. Therefore, programs that update records in disk storage cannot be effectively restarted if the checkpoint is requested after processing of the records has begun.
3. Tape files that are closed at the time of a checkpoint cannot be repositioned during a restart from that checkpoint. (This is no problem if the tape file was rewound when it was closed.)
4. Programs operating under control of the Teleprocessing Supervisor cannot use the checkpoint and restart facilities.
5. If the Core Image file and the Temporary Storage file share the same tape unit, immediate restart cannot be used if the TP complex is open. A delayed restart must be used instead. (Before the delayed restart is processed, the TP complex is closed.)

Index

Where more than one page reference is given, the major reference appears first.

Printed in U.S.A. C28-0319-4 1065/12P

IBM

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