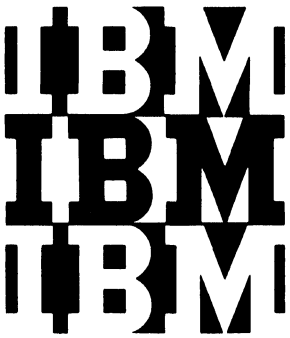


File No. S360-36
GY28-6605-4

Program Logic

IBM System/360 Operating System

Introduction to Control Program Logic

Program Number 360S-CI-505
360S-DM-508

This publication describes the components of the primary control program (PCP) configuration of IBM System/360 Operating System and indicates the program logic manuals that describe these components. It contains general descriptions of the organization of the control program in both main and auxiliary storage, the flow of program control during a job step, the flow of control for supervisor operation, and the processing for input/output operations. The appendix contains the names, numbers, and macro instructions associated with the IBM-supplied SVC routines.

Program Logic Manuals are intended for use by IBM customer engineers involved in program maintenance, and by system programmers involved in altering the program design. Program logic information is not necessary for program operation and use; therefore, distribution of this manual is limited to persons with program maintenance or modification responsibilities.

Preface

This publication is an introduction to the logic of the primary control program and to the program logic manuals that describe its components. The first section of this publication discusses the contents of the different areas of main storage. The second section describes the stages of execution of a program in the operating system. The third section discusses how CPU control is passed between the control program and a processing program. The last section describes the programs provided to move data between main and auxiliary storage.

The glossary defines many of the terms used in discussing the control program, and the appendix lists all the IBM-supplied SVC routines. The figures that trace the flow of CPU control for various operations are examples of typical uses of control program facilities.

Before using this publication, the reader should be familiar with the contents of:

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

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

IBM System/360: Principles of Operation, Form A22-6821

Fifth Edition (November, 1968)

This is a major revision of, and obsoletes, Y28-6605-2 and -3 and Technical Newsletters Y28-2180, Y28-2203, Y28-2222, Y28-2261, and Y28-2295. Discussions of checkpoint/restart processing and IBM 2361 Core Storage support have been added. Changes to the text, and small changes to illustrations, are indicated by a vertical line to the left of the change; changed or added illustrations are denoted by the symbol • to the left of the caption.

This edition applies to release 17 of IBM System/360 Operating System and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/360 SRL Newsletter, Form N20-0360, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

Contents

Illustrations

Figures

Introduction

The control program is a collection of supervisory and service routines that govern System/360 Operating System. These routines have exclusive control over all input/output and machine-oriented functions. The control program schedules work, allocates system resources, and performs input/output (I/O) operations.

The operation of the control program is described in a series of program logic manuals, each of which describes a functional area within the control program.

This publication briefly discusses the processing performed by the components of the primary control program and indicates the program logic manuals in which more detailed information can be found. The publication is divided into four sections:

- Organization of the control program.
- Program flow.
- System control.
- Input/output control.

The first section describes the organization of the control program in main and auxiliary storage; the second, the control program processing required to execute a job step; the third, some of the supervisor operations during the execution of a job step; and the last, the control program operations required to perform I/O operations.

Functions of the Control Program

Control program routines are grouped into three functional areas:

- Job management routines.
- Task management routines.
- Data management routines.

JOB MANAGEMENT ROUTINES

Job management routines provide communication between the user and the operating system by:

- Analyzing the input job stream and collecting the information needed to prepare the job for execution.
- Analyzing operator commands, and transmitting messages from a program to the operator.

There are three major job management routines:

- Master scheduler, which analyzes commands from the console and transmits messages to the operator.
- Reader/interpreter, which reads the input job stream and constructs control blocks and tables from information in the control statements.
- Initiator/terminator, which collects the information and resources needed to execute a job step and performs the operations required to terminate a job step.

The operation of these routines is described in the publication IBM System/360 Operating System: Job Management, Program Logic Manual, Form Y28-6613.

TASK MANAGEMENT ROUTINES

Task management routines monitor and control the entire operating system, and are used throughout the operation of both the control program and processing programs.

There are six functions performed by these routines:

- Interruption handling.
- Task supervision.
- Main storage supervision.
- Contents supervision (and program fetch).
- Overlay supervision.
- Time supervision.

The task management routines are collectively referred to as the "supervisor." A description of these routines is given in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual, Form Y28-6612.

DATA MANAGEMENT ROUTINES

Data management routines control all the operations associated with input/output devices: allocating of space on volumes, channel scheduling, storing, naming, and cataloging of data sets, moving of data between main and auxiliary storage and handling errors that occur during I/O operations. Data management routines are used by both processing programs and control program routines that require data

movement. Processing programs use data management routines primarily to read and write data. The control program uses data management routines not only to read and write required data, but also to locate input data sets and to reserve auxiliary storage space for output data sets of the processing programs.

There are five categories of data management routines:

- Input/output (I/O) supervisor, which performs I/O operations and processes I/O interruptions.
- Access methods, which communicate with the I/O supervisor.
- Catalog management, which maintains the catalog and locates data sets on auxiliary storage.
- Direct access device space management (DADSM), which allocates auxiliary storage space.
- Open/Close/End-of-Volume, which performs required initialization for I/O operations and handles end-of-volume conditions.

The operation of these routines is described in the following publications:

IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual, Form Y28-6616

IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual, Form Y28-6604

IBM System/360 Operating System: Indexed Sequential Access Methods, Program Logic Manual, Form Y28-6618

IBM System/360 Operating System: Basic Direct Access Method, Program Logic Manual, Form Y28-6617

IBM System/360 Operating System: Graphics Access Method, Program Logic Manual, Form Y27-7113

IBM System/360 Operating System: Catalog Management, Program Logic Manual, Form Y28-6606

IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual, Form Y28-6607

IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV), Program Logic Manual, Form Y28-6609

Different portions of the control program operate from different areas of main storage. Main storage is divided into two areas: the fixed area and the dynamic (or program) area.

The fixed area of main storage is the lower portion of main storage; its size is determined by the control program configuration. The fixed area contains those control program routines that perform a system function during the execution of a processing program.

The dynamic area is the upper portion of main storage. It is occupied by a processing program, or control program routines that either prepare job steps for execution (i.e., job management routines), or handle data for a processing program (i.e., the access methods).

On auxiliary storage, the control program resides in three partitioned data sets that are created when the operating system is generated. These data sets are:

- The NUCLEUS partitioned data set (SYS1.NUCLEUS) which contains the resident portion of the control program and the nucleus initialization program.
- The SVCLIB partitioned data set (SYS1.SVCLIB) which contains the nonresident SVC routines, nonresident error handling routines and the access method routines.
- The LINKLIB partitioned data set (SYS1.LINKLIB) which contains the other nonresident control program routines and the IBM-supplied processing programs.

Figure 1 shows the main storage areas into which the routines from each partitioned data set are loaded.

Resident Portion of the Control Program

The resident portion of the control program (the nucleus) resides in the NUCLEUS partitioned data set. This portion of the control program is made up of those routines, control blocks, and tables that are brought into main storage at initial-program-loading (IPL) time and that are never overlaid by another part of the operating system. The nucleus is loaded into the fixed area of main storage.

The resident task management routines are: all of the routines that perform interruption handling, main storage supervision, and time supervision; and some of the routines that perform task supervision, contents supervision, and overlay supervision. These routines are described in IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

The only resident job management routine is that portion of the master scheduler that receives commands from the operator. This routine is described in IBM System/360 Operating System: Job Management, Program Logic Manual.

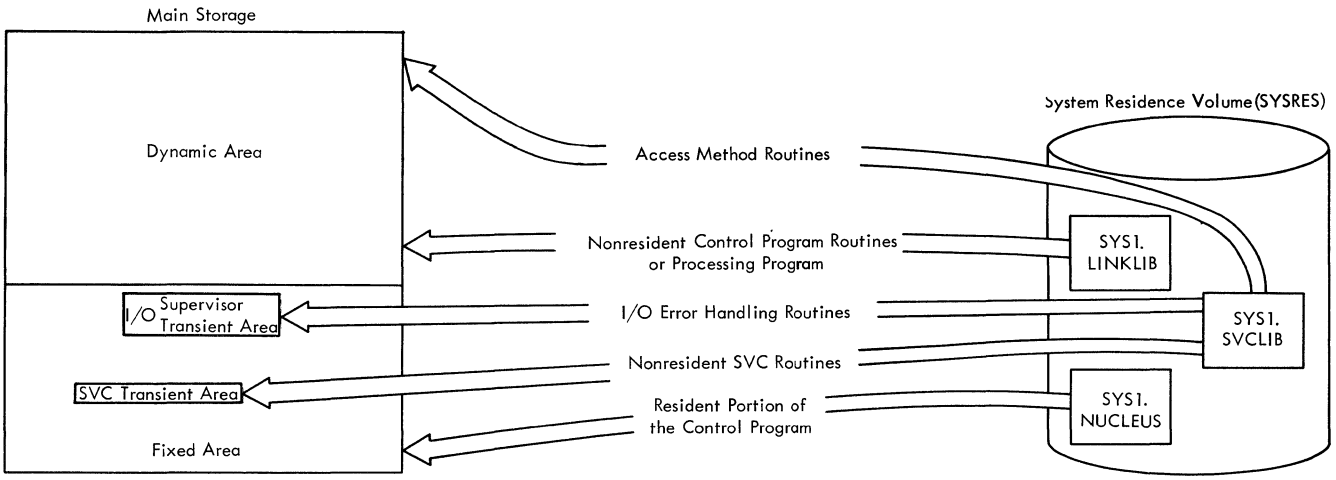


Figure 1. Loading of the Control Program

The resident data management routines are the I/O supervisor and the BLDL routine. These routines are described in IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual and IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual.

The user may also select access method modules to be made resident. These modules are referred to as the resident access method (RAM). They are loaded during IPL rather than during open processing. RAM modules reside in the fixed area adjacent to the higher end of the nucleus.

SVC routines and modules of SVC routines that are normally nonresident (i.e., type 3 and 4 SVC routines) can be made part of the resident portion of the control program. When the system is generated, the user specifies that he wants nonresident SVC routines to be made resident. When IPL is performed, he specifies which SVC routines he wants resident and the nucleus initialization program loads them into the fixed area adjacent to the RAM modules, or to the nucleus if there are no RAM modules.

The test translator (TESTRAN) has an SVC routine in the nucleus. This routine provides a linkage to nonresident TESTRAN routines. The operation of TESTRAN is described in IBM System/360 Operating System: TESTRAN, Program Logic Manual, Form Y28-6611.

In systems that support graphic devices, the graphics access method has an SVC routine and a table in the nucleus. Both are described in IBM System/360 Operating System: Graphics Access Method, Program Logic Manual.

Nonresident Portion of the Control Program

The nonresident portion of the control program is made up of those routines that are loaded into main storage as they are needed, and can be overlaid after their completion. The nonresident routines operate from the dynamic area and from two sections of the fixed area called transient areas.

TRANSIENT AREAS

The transient areas are two blocks of main storage defined in the nucleus and embedded in the fixed area. The first, the SVC transient area, is reserved for nonresident SVC routines. The second, the I/O supervisor transient area, is used by nonresident I/O error handling routines that are brought in by the I/O supervisor. Each transient area contains only one routine at a time. When a nonresident SVC or error handling routine is required, it is read into the appropriate transient area. All routines read into the transient areas reside in SYS1.SVCLIB.

DYNAMIC AREA

The dynamic or program area is all main storage that is not part of the fixed area. It is used for all processing programs as well as for the access method routines and the nonresident job management routines of the control program. Processing programs are brought from either SYS1.LINKLIB, or a user-specified partitioned data set. Access method routines are brought in from SYS1.SVCLIB. Job management routines are brought in from SYS1.LINKLIB. When the control program needs main storage to build control blocks or for a work area, it obtains this space from the dynamic area.

The dynamic area is subdivided as shown in Figure 2. Job management routines, processing programs, and routines brought into storage via a LINK, ATTACH, or XCTL macro instruction are loaded into the lowest available portion of the dynamic area. The highest portion of the dynamic area is occupied by a table (the task input/output table) built by a job management routine. This table is used by data management routines and contains information about DD statements. It remains in storage for the whole job step. Access method routines and routines brought into storage via a LOAD macro instruction are placed in the highest available locations in the dynamic area.

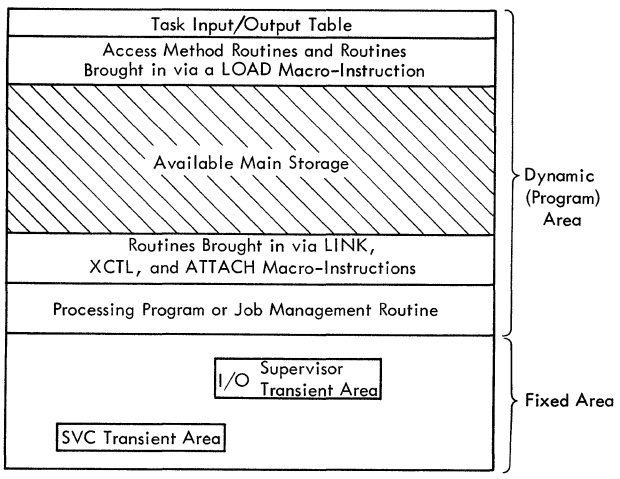


Figure 2. Divisions of Main Storage for the Operating System

The dynamic area may be expanded by including IBM 2361 Core Storage in the system. Main storage hierarchy support for IBM 2361 Models 1 and 2 permits selective access to the processor storage (hierarchy 0) or IBM 2361 (hierarchy 1) portions of main storage. Extensions are made to the various assembler language macro instructions to permit specification of either hierarchy as desired. If the system does not include IBM 2361 Core Storage, requests for storage within hierarchy 1 are obtained from hierarchy 0.

Program Flow

The stages to execution of a program under System/360 Operating System are:

0. Loading the nucleus into main storage (IPL).
1. Reading control statements.
2. Initiating a job step.
3. Executing a job step.
4. Terminating a job step.

The operating system is given control of the computer when the control program nucleus is loaded. Thereafter, jobs may be processed continuously without reloading the nucleus.

When the user introduces a job into the input stream, the initial processing required to prepare his job for execution is performed by job management routines. Control statements for a complete job are read during stage 1. (When a user data set is included in the input stream, only control cards for job steps preceding this data set are read during stage 1.)

Stage 2 is the processing required to initiate the execution of a user's job step. Stage 3 occurs when CPU control is passed to that job step. Stage 4 occurs when the job step terminates; job management routines perform termination procedures for the step (and, when applicable, for the job).

Upon completion of a job or a job step that had a data set in the input job stream, control passes back to stage 1. If further job step control statements had been read during stage 1, control passes to the initiation of the next job step (stage 2).

When a job terminates, and there are no succeeding jobs in the input job stream, the control program places the CPU in the wait state. As long as the nucleus remains intact in main storage, the user can introduce new jobs into the input job stream without reloading the nucleus.

Reading control statements and initiating a job step are performed by the reader/interpreter and initiator/terminator routines, respectively. Descriptions of these routines are given in the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

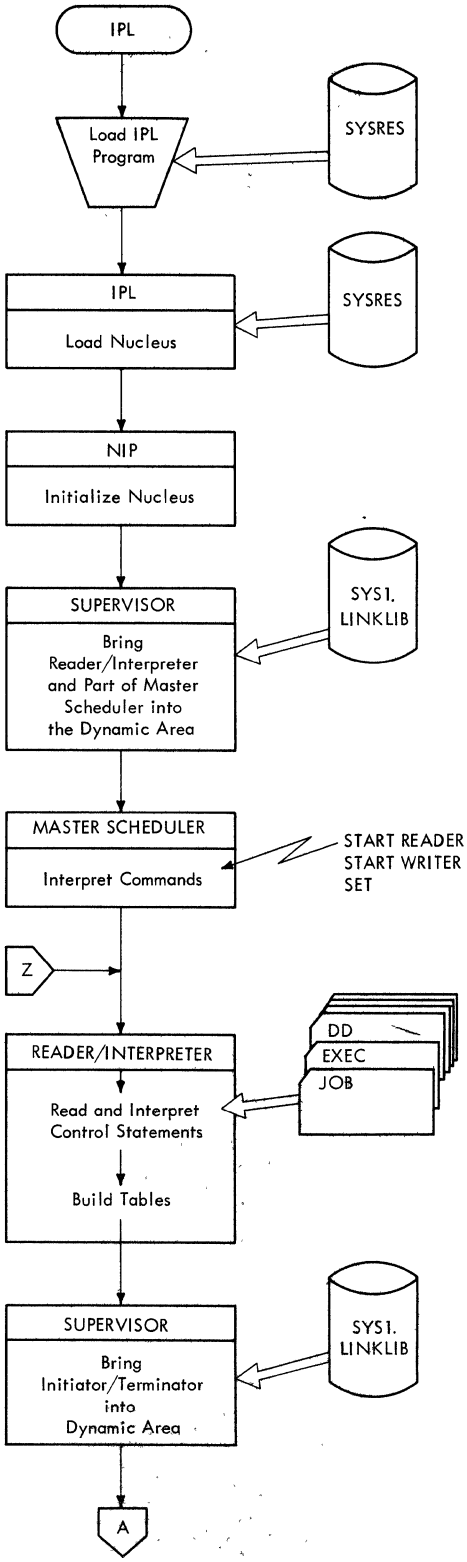
A job step is performed by a user-written (e.g., a payroll program), or IBM-supplied (e.g., linkage editor, COBOL) processing program.

Terminating a job step is performed by the initiator/terminator and the supervisor. Descriptions of these routines are given in the publications IBM System/360 Operating System: Job Management, Program Logic Manual and IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

Job steps that have been terminated abnormally can be restarted. Restart functions are performed by job management and supervisor checkpoint/restart routines, also described in the Job Management and Fixed-Task Supervisor Program Logic Manuals.

Figure 3 shows the overall flow of CPU control through the job processing cycle. This figure describes the processing performed by various components of the control program as it loads the nucleus, reads control statements, initiates the job step, and terminates the job step. Control program processing performed during the execution of a job step is discussed in the subsequent chapters of this publication.

Figures 4 and 5 show the flow of CPU control for checkpoint/restart processing. Figure 4 describes the flow for the checkpoint function, and Figure 5 describes the flow when a job is restarted. Only processing related to checkpoint/restart functions is shown in these figures. Once a checkpoint is taken, or after a step is restarted, the job processing cycle shown in Figure 3 is resumed.



To load the nucleus, the operator specifies the device on which the system residence volume is mounted, and presses the LOAD button on the console. This action causes an IPL record to be read and to be given CPU control. This record reads a second IPL record which, in turn, reads the rest of the IPL program into main storage.

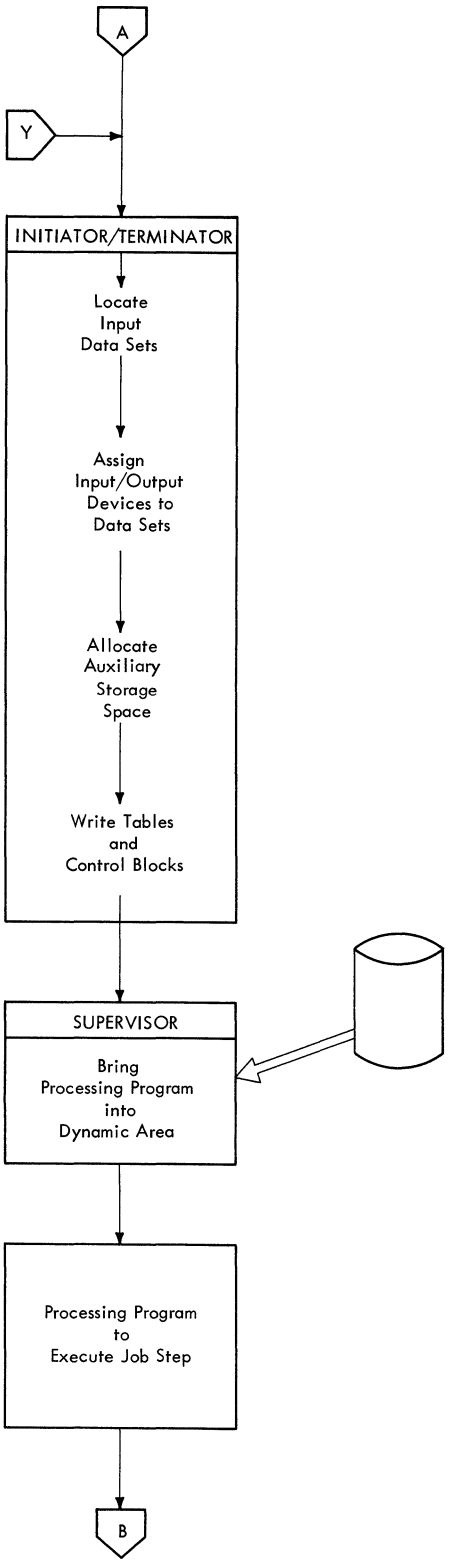
The IPL program searches the volume label to locate the volume table of contents (VTOC) of the system residence volume. The volume table of contents is then searched for the SYS1.NUCLEUS. The nucleus is brought into the fixed area, and the nucleus initialization program (NIP), which is also part of SYS1.NUCLEUS, is brought into the dynamic area. NIP receives CPU control from the IPL program, and initializes the nucleus. After completing its processing, NIP causes itself to be replaced in the dynamic area with the reader/interpreter and the master command routine of the master scheduler.

When the start reader (START RDR), start writer (START WTR), and set date (SET) commands are issued, the resulting interruption causes CPU control to be given to the master command routine. After processing the commands, this master scheduler routine passes CPU control to the reader/interpreter. The master scheduler and reader/interpreter are described in the publication IBM System/360 Operating System: Job Management, Program Logic Manual.

The reader/interpreter reads the control statements from the input job stream. Information from the JOB, EXEC, and DD statements is used to control the execution of job steps. This information is used to construct a job control table (JCT) for the job being read, a step control table (SCT) for the job step being read, and a job file control block (JFCB) and step input/output table (SIOT) for each data set being used or created by the job step. Information from these tables and control blocks is combined with information in the data control block (DCB) and data set control block (DSCB) or label when a data set is opened during step execution.

The reader/interpreter has itself replaced by the initiator/terminator routine.

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Part 1 of 3)



After receiving CPU control, the initiator/terminator prepares to initiate the job step that has been read and interpreted. Using the data which the reader/interpreter extracted from DD statements, the initiator/terminator:

Locates Input Data Sets: The initiator/terminator determines the volume containing a given input data set from the data definition (DD) statement, or from a search of the catalog. This search is performed by a catalog management routine that is entered from the initiator/terminator. A description of the routines that maintain and search the catalog is given in the publication IBM System/360 Operating System: Catalog Management, Program Logic Manual.

Assigns I/O Devices: A job step cannot be initiated unless there are enough I/O devices to fill its needs. The initiator/terminator determines whether the required devices are available, and makes specific assignments. If necessary, messages to the operator direct the mounting of volumes (tapes, etc.).

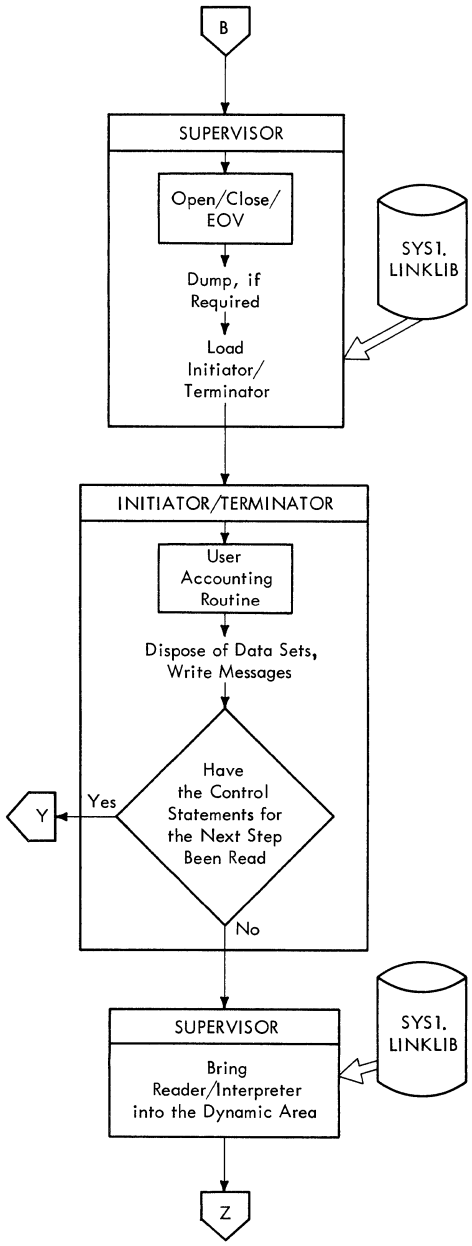
Allocates Auxiliary Storage Space: Direct access volume space required for output data sets of a job step is acquired by the initiator/terminator, which uses DADSM. A description of the operation of DADSM is given in the publication IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual.

The JFCB, which contains information concerning the data sets to be used during step execution, is written on auxiliary storage. This data is used when a data set is opened, and when the job step is terminated (e.g., disposition).

The initiator/terminator causes itself to be replaced by the processing program to be executed.

The processing program can be one of the IBM-supplied processors (e.g., COBOL, linkage editor), or a user-written program. The processing program uses control program services for operations such as loading other programs, and performing I/O operations.

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Part 2 of 3)



When the processing program terminates, the supervisor receives CPU control. The supervisor uses the OPEN/CLOSE/EOV routines to close any open data control blocks. These routines are described in the publication IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV), Program Logic Manual.

Under abnormal termination conditions, the supervisor may also provide special termination procedures such as a storage dump.

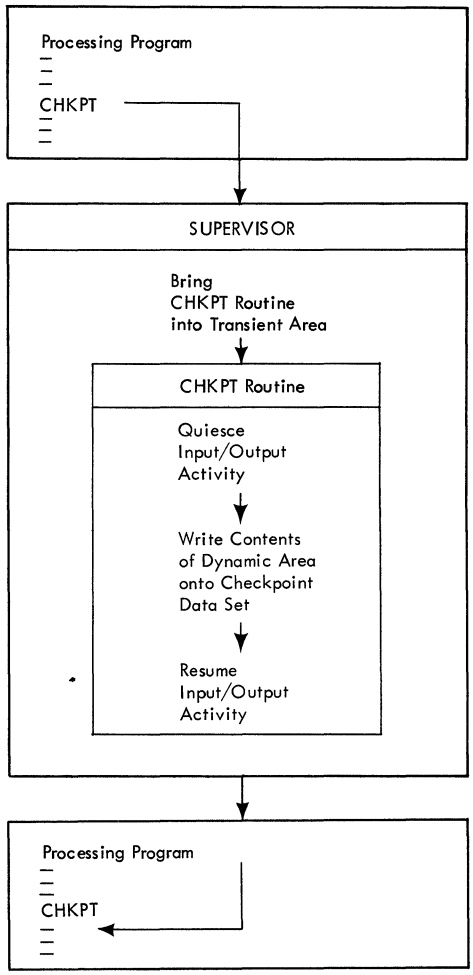
The supervisor passes control to the initiator/terminator, which is brought into the dynamic area replacing the processing program.

The initiator/terminator performs functions required to terminate individual job steps and complete jobs. It executes an installation accounting routine if one is provided.

The initiator/terminator releases the I/O devices, and disposes of data sets used and/or created during the job step. (This requires reading tables prepared during initiation, which include information such as disposition of data sets.)

If the control statements for the next job step were read and interpreted, the initiator/terminator initiates that step (entry point Y on page 12). If the statements were not read, the initiator/terminator is replaced with the reader/interpreter which starts the read-initiate-execute-terminate cycle for the next job step or job (entry point Z on page 11).

Figure 3. Example of CPU Control Flow for a Job Processing Cycle (Part 3 of 3)



To cause the checkpoint function to be performed, the programmer places a CHKPT macro instruction in his program. The expansion of the CHKPT macro instruction includes an SVC 63 instruction, which causes CPU control to be passed to the supervisor.

The CHKPT routine is a nonresident SVC routine.

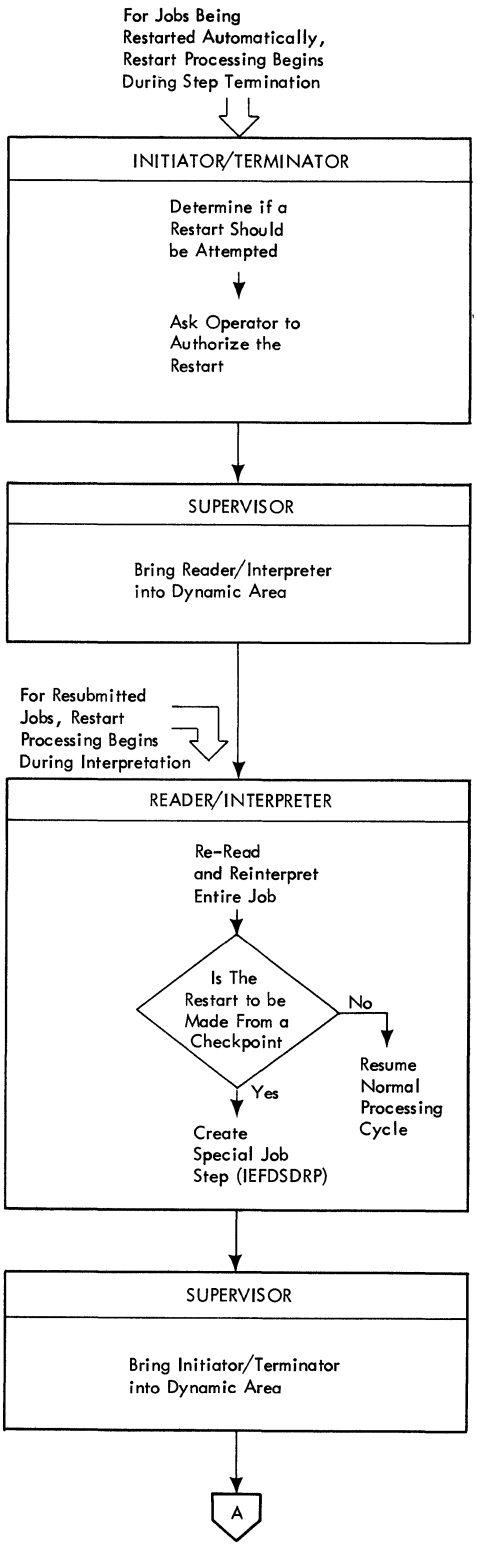
Any input/output operations in progress are allowed to complete, and no new input/output operations are begun.

Enough information about the status of the job step is recorded to allow a restart to be made from the place in the processing program where the checkpoint is taken.

Input/output operations, delayed so the checkpoint could be taken, are started.

Step execution is resumed.

•Figure 4. Example of CPU Control Flow for the Checkpoint Function



When an error condition causes step termination to begin prematurely, the initiator/terminator determines if the programmer has requested an automatic restart. If so, it then determines if a restart is likely to be successful (some types of errors can make a successful restart unlikely).

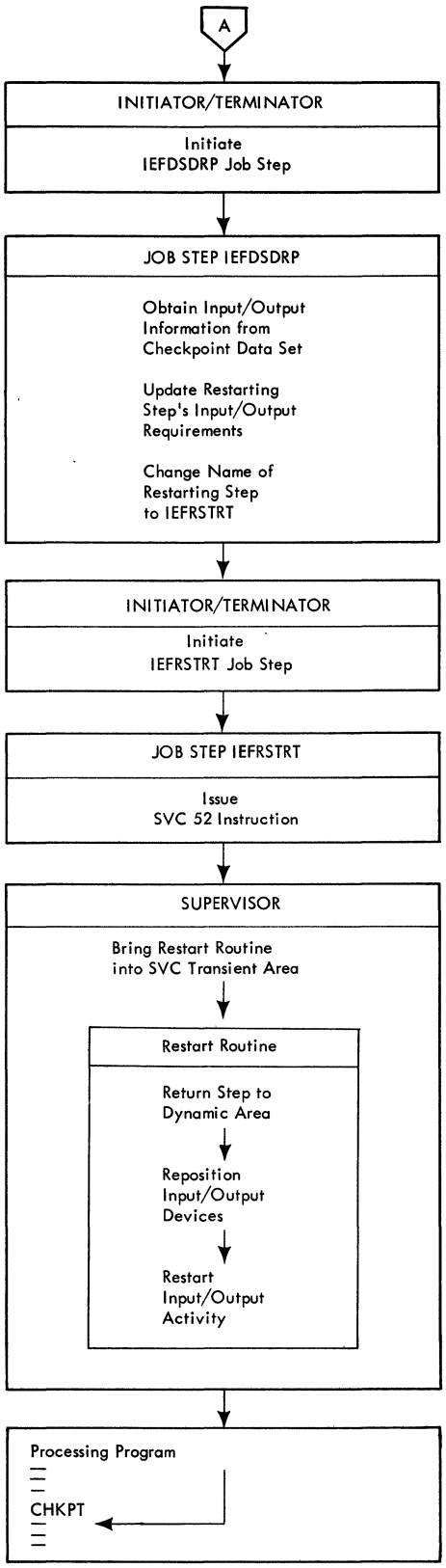
A request for operator authorization consists of a message to which the operator may reply by indicating whether or not the step is to be restarted.

If the restart is to be made from the beginning of a step, the normal read-initiate-execute-terminate job processing cycle will be resumed with initiation of the step to be restarted.

If the restart is to occur at a place where a checkpoint was taken, however, the normal cycle cannot be resumed until the partially executed step is retrieved from the checkpoint data set.

A special job step, IEFDSDRP, will be executed immediately before the step to be restarted. It is used for input/output allocation purposes.

•Figure 5. Example of CPU Control Flow When a Job is Restarted (Part 1 of 2)



During initiation for job step IEFDSDRP the initiator/terminator assigns a device to the checkpoint data set. If necessary, the operator is instructed to mount the volume containing the checkpoint data set.

The restarting step's input/output device and space requirements, recorded in control blocks by the reader/interpreter, reflect what is needed to start execution at the beginning of the step. The checkpoint data set contains a record of input/output assignments that existed when the checkpoint was taken. Where differences exist, the control blocks are changed so that requirements for the restart will reflect what is needed to start execution from the checkpoint.

During initiation for job step IEFRSTRT, the initiator/terminator assigns input/output devices to the restarting step, and instructs the operator to mount any needed volumes.

The first instruction of the restarting step was made to be in SVC 52 instruction, intended to cause CPU control to pass to the restart SVC routine.

The restart routine is a nonresident SVC routine.

The step is restored to the dynamic area and an environment similar to that which existed when the checkpoint was taken is re-created.

CPU control is returned to the processing program at the instruction following the CHKPT macro instruction.

• Figure 5. Example of CPU Control Flow When a Job is Restarted (Part 2 of 2)

During execution of a processing program (a job step), CPU control is passed back and forth between the control program and the processing program. This section of the publication discusses how CPU control is passed between programs and routines in the fixed and dynamic areas of main storage.

Control Flow to the Control Program

CPU control is passed from a processing program to the control program via an interruption; an interruption causes CPU control to be given to an interruption handler routine of the control program. The interruption handler gives CPU control to a routine that processes the interruption. This routine returns CPU control to the supervisor which, in turn, causes CPU control to be passed to the appropriate processing program. A description of the interruption handlers is given in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

The processing performed by the control program is determined by the type of interruption and falls into five categories:

- SVC interruption. The processing program requires a control program service and requests that service via an SVC instruction.
- Input/output interruption. An input/output operation terminates, or the operator issues a command.
- Timer/external interruption. An event (e.g., a timer or external signal) indicates the need for control program processing.
- Program interruption. The processing program execution generates an unexpected need for control program processing either because an invalid operation is attempted (e.g., execution of a privileged instruction by a program in the problem state), or a data error (e.g., overflow) is detected.
- Machine-check interruption. A computer error signal indicates that a recognizable machine error has occurred.

An SVC interruption is the only type of interruption that must be initiated in the interrupted program; the other types of interruptions occur because of an event that is generally not anticipated by the program that is interrupted.

SVC INTERRUPTIONS

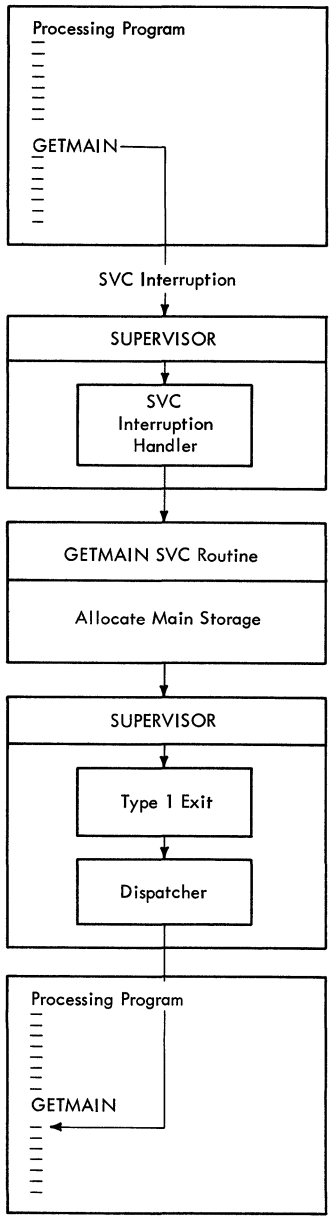
SVC interruptions result from program requests for contol program services. The SVC interruption handler saves critical information about the interrupted program before passing CPU control to the SVC routine that performs or initiates the requested service.

There are four types of SVC routines:

- Type 1 SVC routines are part of the resident portion of the control program and are disabled (masked) for all types of interruptions except machine-check interruptions.
- Type 2 SVC routines are part of the resident portion of the control program, but may be enabled (interruptable) for part of their operation.
- Type 3 SVC routines are part of the nonresident portion of the control program, may be enabled, and occupy not more than 1024 bytes of main storage when loaded.
- Type 4 SVC routines are nonresident, may be enabled and are larger than 1024 bytes. They are brought into main storage in segments of 1024 bytes or less.

When the requested service is performed by a type 3 or type 4 routine, the supervisor fetches it into the SVC transient area, unless it was loaded by the nucleus initialization program. A list of the SVC routines, their types, and the program logic manuals in which they are described is given in Appendix A.

A processing program uses the SVC facilities to request services that can be performed only by the control program. One of these services, dynamic allocation of main storage, is requested via the GETMAIN macro instruction. Figure 6 shows the overall flow of CPU control when the expansion of a GETMAIN macro instruction is executed in a processing program. The flow of CPU control for all SVC interruptions is similar to the flow for that of GETMAIN; the SVC interruption handler receives CPU control,



Expansion of a GETMAIN macro instruction includes an SVC instruction and instructions that set up the parameters of the requested main storage area. Execution of the SVC instruction causes an SVC interruption to occur. CPU control is passed to the SVC interruption handler which saves the program status word and register contents.

The SVC interruption handler analyzes the cause of the interruption. Since, in this case, the SVC routine that is to receive CPU control is a type 1 routine, the SVC interruption handler passes CPU control directly to the GETMAIN SVC routine.

The GETMAIN routine determines whether the request is valid and whether there is enough available main storage to fill the request. When storage is allocated, indicators are set, showing that this storage is in use. The GETMAIN SVC routine is described in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

When the GETMAIN SVC routine has completed its processing, it passes CPU control to the type 1 exit routine which determines whether the routine that issued the GETMAIN macro instruction is enabled or disabled. Since a processing program (i.e., an enabled program) issued the request, CPU control is passed to the dispatcher which, in turn, passes CPU control to the processing program.

Figure 6. Example of CPU Control Flow for an SVC Interruption

determines which SVC routine is to perform the requested service, brings that routine into storage if it is not already there, and passes CPU control to it. After the SVC routine performs its service, it returns control to the supervisor which in turn, returns control to the processing program.

OTHER INTERRUPTIONS

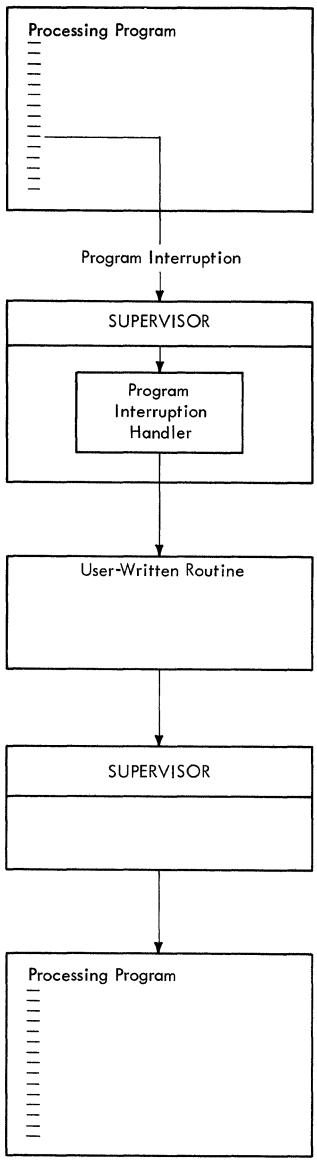
Program, timer/external, input/output, and machine-check interruptions give CPU control to the control program when certain computer or program events occur. The main characteristic of these interruptions is that they are asynchronous (i.e., they

occur at an undetermined point in the program). Each of these interruptions is processed in a unique way but the overall flow of CPU control is similar for all of them. Figure 7 uses a program interruption as an example of the overall flow of CPU control for an asynchronous interruption.

When such an interruption occurs, the appropriate interruption handler receives CPU control and, in turn, passes control to the routine that processes the interruption. After this routine completes its processing, it passes control to the supervisor. The routine that now receives CPU control is determined by the result of the interruption processing.

The routines that process asynchronous interruptions are described in the publications:

- IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.
- IBM System/360 Operating System: Job Management, Program Logic Manual.
- IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual.



During execution of a processing program, a program interruption occurs causing CPU control to be passed to the program interruption handler routine. This routine stores register contents and the PSW of the interrupted program, and determines the cause of this interruption.

If the user has not indicated (via a SPIE macro instruction) a routine to process this particular program interruption type, the job is abnormally terminated. If such a routine is provided, the program interruption handler passes CPU control to this user-written routine.

When the user-written routine has completed its processing, CPU control is returned to the supervisor which returns control to the interrupted program.

Figure 7. Example of CPU Control Flow for an Asynchronous Interruption

Control Flow to a Processing Program

Two conditions that can exist when the control program is to pass CPU control to a processing program are:

- The supervisor must return CPU control to a processing program that previously relinquished control for some control program service or supervisory function.
- The supervisor must pass CPU control to a new processing program whose execution was requested via a supervisor-assisted linkage.

The supervisor passes CPU control to any program in the dynamic area by loading the program status word (PSW) for that program. The supervisor routine that loads the PSW is the dispatcher; its operation is referred to as dispatching.

RETURNING CONTROL TO A PROCESSING PROGRAM

Normally after the control program has performed some requested service or required supervisory function for a processing program, the supervisor returns CPU control to that processing program. When this processing program had been interrupted, certain registers and the PSW under which it was operating were stored by the interruption handler. The dispatcher restores the registers to their previous values and loads this PSW, returning CPU control to that program.

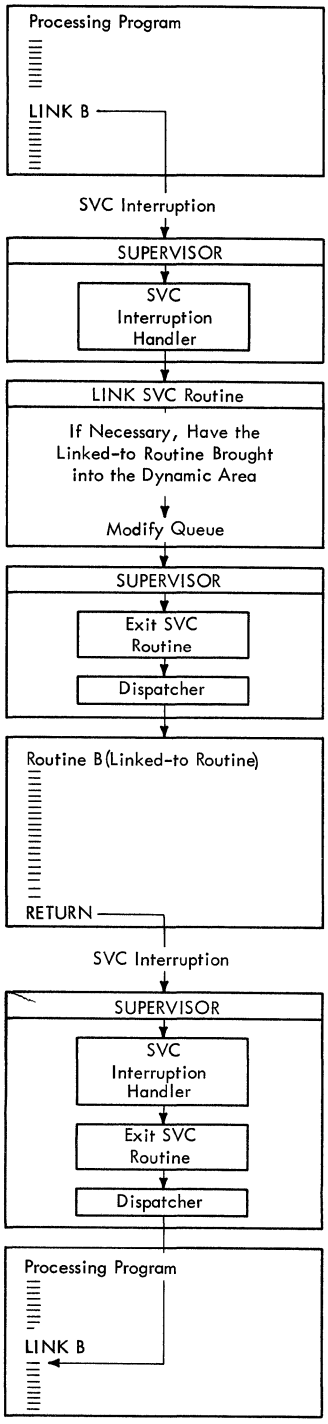
SUPERVISOR-ASSISTED LINKAGES

By using certain SVC macro instructions, the user can have the control program perform all the linkages required to pass CPU control from one program or routine to another. In general, when a supervisor-assisted linkage such as a LINK, ATTACH, or XCTL macro instruction is used, the control program brings the desired program into storage, builds a PSW for that program, passes CPU control to this program, and saves the information required to pass CPU control to the proper program when the 'linked-to' program is complete.

Figure 8 shows the flow of CPU control when a processing program uses the LINK macro instruction to pass CPU control to another program. The SVC interruption handler receives CPU control as a result of the interruption, and passes control to the link SVC routine. This routine uses program fetch to bring the linked-to routine into storage and constructs a control block so that the supervisor can control this routine. The link SVC routine then returns CPU control to the supervisor which dispatches the linked-to routine.

When the linked-to routine is complete, CPU control is returned to the supervisor which dispatches the processing program that issued the LINK macro instruction.

Supervisor-assisted linkages are used not only by processing programs, but also by certain control program routines. Job management routines use supervisor-assisted linkages to pass CPU control. When the initiating of a job step is completed, the initiator/terminator uses an XCTL macro instruction to pass CPU control to the processing program that is to execute the job step. This macro instruction causes the program that issued it (in this case the initiator/terminator) to be overlaid by the program that is to receive CPU control.



Expansion of the LINK macro instruction includes an SVC instruction and instructions that set up parameters. Execution of the SVC instruction causes an SVC interruption. CPU control is passed to the SVC interruption handler which saves the program status word and register contents, and analyzes the cause of the interruption. Since the link SVC routine is a type 2 routine, the SVC interruption handler can pass CPU control directly to the LINK routine.

If necessary, the link SVC routine has a copy of the linked-to routine brought into main storage. The link SVC routine then modifies a request block queue so that the 'linked-to' routine will receive CPU control, and passes CPU control to the exit SVC routine. This routine passes control to the dispatcher which dispatches the 'linked-to' routine. The link SVC routine, and request block queue are discussed in the publication IBM System/360 Operating System: Fixed-Task Supervisor, Program Logic Manual.

When the 'linked-to' routine is completed, it issues a RETURN macro instruction. In this case, an SVC interruption occurs causing CPU control to pass first to the SVC interruption handler and then to the exit SVC routine. The exit SVC routine restores the conditions required by the program that issued the LINK macro instruction. CPU control is then passed to the dispatcher which returns CPU control to the program that issued the LINK macro instruction.

Figure 8. Example of CPU Control Flow for a Supervisor-Assisted Linkage

Input/Output Control

The I/O control facilities provided by the operating system perform services associated with the moving of data between main and auxiliary storage. These services are performed by the control program's data management routines and consist of:

- Catalog management.
- Direct access device space management.
- Input/output support.
- Input/output operation control.

The catalog management routines maintain a catalog of certain data sets and the volumes on which they reside. These routines locate cataloged data sets, and add and delete items from the catalog. A description of the catalog management routines is given in the publication IBM System/360 Operating System: Catalog Management, Program Logic Manual.

The catalog management routines are SVC routines which operate from the SVC transient area. The initiator/terminator of job management uses catalog management routines both to locate cataloged input data sets for a job step being initiated, and to catalog specified data sets created during the job step. The IEHPROGM utility program uses catalog management routines to add, change, or delete the components of the catalog.

The direct access device space management (DADSM) routines allocate and release space on direct access volumes. A description of the DADSM routines is given in the publication IBM System/360 Operating System: Direct Access Device Space Management, Program Logic Manual.

The DADSM routines are SVC routines that operate from the SVC transient area. They are used by the initiator/terminator when a job step is being initiated to get direct access storage space for output data sets, and by I/O support routines to acquire additional space and to release unneeded space.

The I/O support routines are SVC routines that perform operations directly associated with an I/O operation. These operations are:

- Opening a data control block.
- Closing a data control block.
- Performing end-of-volume procedures.

The I/O support routines operate from the SVC transient area. They are used by processing programs, the supervisor, and job management routines, and are described in the publication IBM System/360 Operating System: Input/Output Support (OPEN/CLOSE/EOV), Program Logic Manual.

Control of I/O operations occurs both when an I/O operation is to be initiated, and when an I/O operation terminates.

Initiation of an I/O operation normally is performed by access method routines and the I/O supervisor. The access method routines operate from the dynamic area, and the I/O supervisor operates from the fixed area. The access methods are described in the publications:

- IBM System/360 Operating System: Sequential Access Methods, Program Logic Manual.
- IBM System/360 Operating System: Indexed Sequential Access Methods, Program Logic Manual.
- IBM System/360 Operating System: Basic Direct Access Method, Program Logic Manual.
- IBM System/360 Operating System: Graphics Access Methods, Program Logic Manual.

The I/O supervisor is described in the publication IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual.

Termination of an I/O operation causes another part of the I/O supervisor to receive CPU control. During its processing, the I/O supervisor uses error handling routines and access method routines called appendages. The I/O supervisor and the error handling routines are described in the Input/Output Supervisor, Program Logic Manual. The access method appendage routines are described in the program logic manuals for the various access methods.

Processing Input/Output Operations

The processing directly associated with I/O operations is performed when a data control block is opened and when an I/O operation is executed.

OPENING A DATA CONTROL BLOCK

Before any information can be read from or written into a data set, initialization must be performed. This initialization is referred to as 'opening' the data control block of the data set, and consists of:

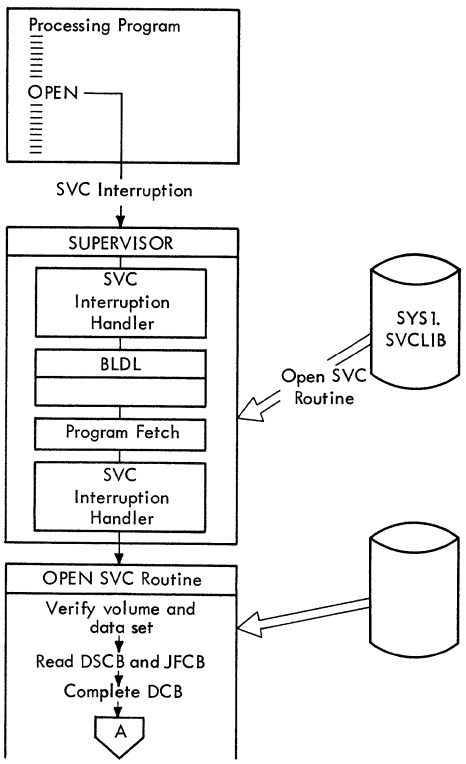
- Completing the data control block (DCB), job file control block (JFCB), and data set control block (DSCB) of the data set.
- Building the data extent block (DEB).
- Acquiring the access method routines that are to operate on the data set.
- Priming buffers when the queued sequential access method (QSAM) is used for input.

Figure 9 shows the flow of CPU control when an OPEN macro instruction is issued for the data control block of a data set residing on a direct access volume.

The program that uses or creates a data set specifies the opening of its DCB via an OPEN macro instruction. The execution of this macro instruction causes an SVC interruption. The SVC interruption handler passes CPU control to the data management open SVC routine.

When the open routine has completed its processing, it returns CPU control to the supervisor which, in turn, dispatches the processing program.

The open routine completes the DCB so that sufficient information is available to perform I/O operations on the associated data set. Empty fields in the DCB are filled from information in the DSCB or label (if the data set is input) and the JFCB. The DSCB is the data set label for data sets residing on a direct access volume; the JFCB was built when the job step was read and interpreted, and contains information from the DD statement. The formats of the DCB, JFCB, and DSCB are given in the publication IBM System/360 Operating System: System Control Blocks, Form C28-6628.



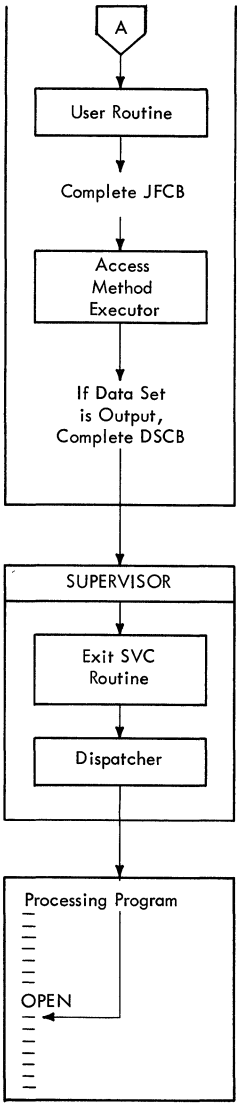
Expansion of an OPEN macro instruction includes an SVC instruction which causes an SVC interruption when executed. CPU control is passed to the SVC interruption handler.

If a table of record addresses for type 3 and 4 SVC routines was not set up at initial-program-loading time, the SVC interruption handler uses the BLDL routine of data management to find the open SVC routine on SYS1.SVCLIB. Program fetch then loads the first segment of the open SVC routine into the SVC transient area.

The open SVC routine first verifies that the proper volumes are mounted, and then finds the data set on the volume. If the data set has password-protection, the open routine issues a message to the operator requesting the proper password. If the operator provides a wrong password twice, the job step is terminated.

The open routine reads the JFCB and DSCB into an open work area obtained from the dynamic area. Information from the DSCB is used to fill empty fields in the JFCB. The JFCB is now used to fill empty fields in the DCB.

Figure 9. Example of CPU Control Flow When an OPEN Macro Instruction Is Issued (Part 1 of 2)



If the user has provided a DCB exit routine, CPU control is passed to this routine. Upon completion of this user routine, control returns to the open SVC routine which merges fields from the now complete DCB into the JFCB. For input data sets, this merge fills only empty fields; for output data sets, it also overlays any affected field except the JFCDSORG field.

The segment of the open SVC routine that is now loaded and given CPU control is called an access method executor. These executor routines perform processing that is unique to the access method specified; they are described in the program logic manuals for the various access methods.

The access method executor builds a DEB and loads the required access method routines into the dynamic area of main storage.

If the data set is output, fields in the JFCB are now merged into the DSCB, overlaying as well as filling any affected fields except the DSORG field. The now-complete DSCB is written on auxiliary storage overlaying the DSCB that was read in initially.

The open SVC routine passes CPU control to the exit SVC routine. This routine, in turn, passes control to the dispatcher which dispatches the processing program.

Figure 9. Example of CPU Control Program Flow When an OPEN Macro Instruction is Issued (Part 2 of 2)

EXECUTING AN INPUT/OUTPUT OPERATION

Executing an I/O operation is discussed in two phases: processing required to start the operation, and processing performed when the operation is terminated. The flow of CPU control during the execution of an I/O operation is shown in Figure 10.

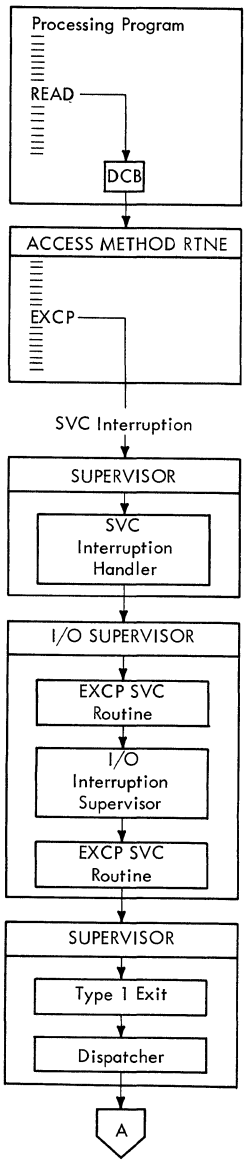
Starting an Input/Output Operation

The expansion of an I/O macro instruction specified in the processing program results in a branch to the access method routines. These routines gather information used to initiate the I/O operation and place this information in control blocks. An access method routine then issues an EXCP macro instruction causing an SVC interruption. The SVC interruption handler gives CPU control to the I/O supervisor.

The I/O supervisor initiates the I/O operation and returns CPU control to the supervisor. The supervisor then returns CPU control to the access method routine which finishes its processing, and returns control to the processing program.

Terminating an Input/Output Operation

When an I/O operation terminates, an I/O interruption occurs causing CPU control to pass to the I/O interruption handler. This routine passes control to the I/O supervisor which performs the termination processing. The I/O supervisor may pass control to error handling routines and access method appendage routines to perform special processing. When its processing is complete, the I/O supervisor passes control to the supervisor which dispatches the processing program.

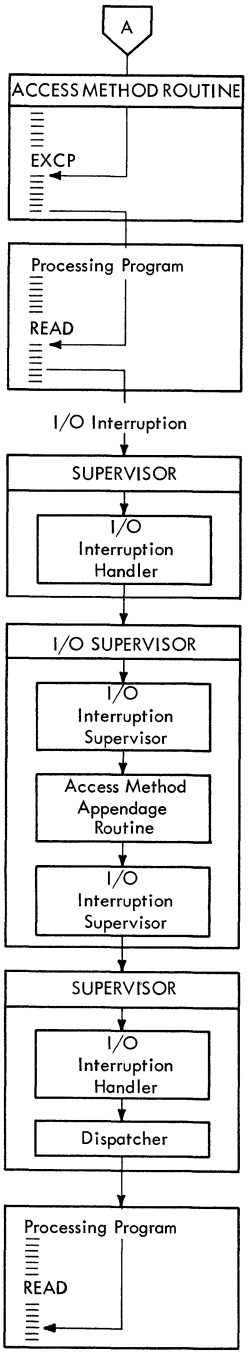


To execute an I/O operation, an I/O macro instruction such as READ is specified in the processing program. The expansion of this macro instruction includes a branch to an access method routine. The address of this routine is obtained from the DCB where it was stored when the DCB was opened.

The access method routines set up a channel program and an input/output block (IOB) so that the I/O supervisor can initiate the I/O operation. The access method routine then issues an SVC instruction.

An SVC interruption occurs causing CPU control to be passed through the SVC interruption handler to the EXCP SVC routine of the I/O supervisor. The EXCP SVC routine passes CPU control to the I/O interruption supervisor portion of the I/O supervisor. This routine checks queues of pending I/O operations and, if possible, starts some of these operations. The EXCP SVC routine and the I/O interruption supervisor are described in the publication IBM System/360 Operating System: Input/Output Supervisor, Program Logic Manual. CPU control is now passed from the I/O interruption supervisor through the EXCP SVC routine to the type 1 exit routine. Since the program that issued the SVC is enabled, the type 1 exit routine passes control to the dispatcher which in turn, returns control to the access method routine.

Figure 10. Example of CPU Control Flow for an I/O Operation (Part 1 of 2)



The access method routine completes its processing and passes CPU control back to the processing program. The requested I/O operation can be in one of three conditions: complete, in process, or queued as a pending request. In this last case, the operation will be started as part of the processing of some subsequent interruption.

When an I/O operation terminates, an I/O interruption occurs causing CPU control to be passed through the I/O interruption handler to the I/O interruption supervisor portion of the I/O supervisor which processes the I/O interruption.

The I/O interruption supervisor branches to access method appendage routines to perform processing unique to the access method used. These routines are discussed in the program logic manuals for the various access methods. The I/O interruption supervisor then services pending I/O interruptions and tries to restart free channels before passing CPU control back to the I/O interruption handler. This routine gives control to the dispatcher which returns control to the processing program at the point where the I/O interruption occurred.

Figure 10. Example of CPU Control Flow for an I/O Operation (Part 2 of 2)

Glossary

access method executor: A routine that is entered during the performance of the open, close, or end-of-volume function, and performs processing unique to the access method to which it applies.

appendage: A routine that performs a function of the input/output supervisor but is not a part of the input/output supervisor. Appendages are provided by the user of the input/output supervisor (e.g., Access Method).

asynchronous: Without regular time relationship to the affected program; as applied to program execution, unpredictable with respect to instruction sequence.

catalog: One or more data sets that specify the volumes upon which cataloged data sets reside.

checkpoint:

1. A point at which information about the status of a job step can be recorded so that the job step can be restarted.
2. To record such information.

checkpoint/restart: A facility of the operating system that can minimize time lost in reprocessing a job step that terminated abnormally due to a program or system failure. It allows a restart to be made from a checkpoint or from the beginning of a job step.

disabled: (masked) A state of the CPU that prevents the occurrence of certain types of interruptions. The types of interruptions for which the CPU is masked is determined by the current program status word. Unless qualified, the terms 'masked' and 'disabled' apply to I/O and External interruptions.

dynamic area: That portion of main storage from which processing programs, job management routines, access method routines and appendages operate.

enabled: (interruptable) A state of the CPU that allows the occurrence of certain types of interruptions. The types of interruptions for which the CPU is interruptable is determined by the current program status word. Unless qualified, the terms 'interruptable' and 'enabled' apply to all types of interruptions.

fixed area: That portion of main storage occupied by the resident portion of the control program (nucleus).

interruptable: See Enabled.

masked: See Disabled.

nonresident portion of control program: Those control program routines that are loaded into main storage as they are needed, and can be overlaid after their completion.

nucleus: That portion of the control program that is loaded into main storage at initial-program-loading time and is never overlaid by another part of the operating system.

SVC routine: A control program routine that performs or initiates a control program service specified by a supervisor call (SVC).

system residence volume: The direct access volume that contains SYS1.NUCLEUS, SYS1.SVCLIB, a catalog data set, and the data set reserved for recording data about machine errors (i.e., SYS1.LOGREC).

SYS1.LINKLIB: The partitioned data set that contains the IBM-supplied processing programs and part of the nonresident portion of the control program. It may also contain user-written programs.

SYS1.NUCLEUS: The partitioned data set that contains the resident portion of the control program (i.e., the nucleus).

SYS1.SVCLIB: The partitioned data set that contains the nonresident SVC routines, nonresident error-handling routines, and the access method routines.

transient areas: Two areas of main storage defined in the nucleus; one is reserved for nonresident SVC routines, the other is used by nonresident error-handling routines.

Appendix A: SVC Routines

This appendix contains two lists: the first is a list of those macro instructions whose expansion includes an SVC instruction and the SVC number (decimal) associated with that instruction; the second is a list of the routines that perform the services requested via the SVCs and the program logic manuals (PLMs) in which these routines are described.

MACRO INSTRUCTION	SVC NUMBER
ABEND	13
ATTACH	42
BLDL	18
BSP	69
CATALOG	26
CHAP	44
CHKPT	63
CIRB	43
CLOSE	20
CLOSE (TYPE=T)	23
DELETE	09
DEQ	48
DETACH	62
DEVTYPE	24
ENQ	56
EOV	55
EXCP	00
EXTRACT	40
FEOV	31
FIND	18
FREEDBUF	57
FREEMAIN	05
GETMAIN	04
IDENTIFY	41
INDEX	26
IOHALT	33
LINK	06
LOAD	08
LOCATE	26
OBTAIN	27
OPEN	19
OPEN (TYPE=J)	22
POST	02
PURGE	16
RELEX	53
RENAME	30
RESTART	52
RESTORE	17
SCRATCH	29
SEGLD	37
SEGWT	37
SETPRT	81
SPIE	14
STAE	60
STIMER	47
STOW	21
SYNCH	12
TIME	11
TTIMER	46
WAIT	01
WAITR	01
WTO	35
WTOR	35
WTL	36
XCTL	07

In the following list, 'ROUTINE NAME' indicates the name by which each SVC routine is referred to in the associated PLM. Two entries in the 'TYPE' field indicate that at system generation time, the user can choose either type for this SVC routine. The first number indicated is the dominant one and is the type assigned unless the second number is explicitly specified.

Use of an SVC number that has '**' in the 'ROUTINE NAME' field causes the SVC interruption handler to abnormally terminate the job step. All unassigned and some nonsupported SVCs fall into this category.

Use of the remaining nonsupported SVC numbers is effectively a no-operation instruction. An interruption will occur, but after the SVC interruption handler analyzes the SVC, it immediately passes CPU control to the SVC exit routine. Nonsupported or unassigned SVC numbers cannot be assigned to user-written SVC routines.

SVC NUMBER	ROUTINE NAME	TYPE	PLM
00	EXCP	1	Input/Output Supervisor
01	Wait	1	Fixed-Task Supervisor
02	Post	1	Fixed-Task Supervisor
03	Exit	1	Fixed-Task Supervisor
04	Getmain	1	Fixed-Task Supervisor
05	Freemain	1	Fixed-Task Supervisor
06	Link	2	Fixed-Task Supervisor
07	XCTL	2	Fixed-Task Supervisor
08	Load	2	Fixed-Task Supervisor
09	Delete	1	Fixed-Task Supervisor
10	Getmain/Freemain	1	Fixed-Task Supervisor
11	Time	1	Fixed-Task Supervisor
12	SYNCH	2	Fixed-Task Supervisor
13	ABEND	4	Fixed-Task Supervisor
14	SPIE	3,2	Fixed-Task Supervisor
15	ERREXCP	1	Input/Output Supervisor
16	Purge	3	Input/Output Supervisor
17	Restore	3	Input/Output Supervisor
18	BLDL	2	Sequential Access Methods
19	Open	4	Input/Output Support (OPEN/CLOSE/EOV)
20	Close	4	Input/Output Support (OPEN/CLOSE/EOV)
21	Stow	3	Sequential Access Methods
22	OpenJ	4	Input/Output Support (OPEN/CLOSE/EOV)
23	Tclose	4	Input/Output Support (OPEN/CLOSE/EOV)
24	DEVTYPE	3	Input/Output Supervisor
25	Track Balance	3	Sequential Access Methods
26	Catalog	4	Catalog Management
27	Obtain	3	Direct Access Device Space Management
28	CVOL	4	Catalog Management
29	Scratch	4	Direct Access Device Space Management
30	Rename	4	Direct Access Device Space Management
31	FEOV	4	Input/Output Support (OPEN/CLOSE/EOV)
32	Allocate	4	Direct Access Device Space Management
33	IOHALT	3	Input/Output Supervisor
34	Master Command EXCP	4	Job Management
35	Write to Operator	3	Job Management
36			Not supported in this configuration
37	Overlay Supervisor	2	Fixed-Task Supervisor
38	Resident SVC	2	TESTRAN

SVC NUMBER	ROUTINE NAME	TYPE	PLM
39	Label	4	Utilities
40	Extract	3,2	Fixed-Task Supervisor
41	Identify	3,2	Fixed-Task Supervisor
42	Attach	3,2	Fixed-Task Supervisor
43	CIRB	3	Fixed-Task Supervisor
44	CHAP	1	Supported in MVT configuration
45	Overlay Supervisor	2	Fixed-Task Supervisor
46	Ttimer	1	Fixed-Task Supervisor
47	Stimer	2	Fixed-Task Supervisor
48	DEQ	1	Control Program with MFT
49	Ttopen1	3	TESTRAN
50	**		Unassigned
51	ABDUMP	4	Fixed-Task Supervisor
52	Restart	4	Fixed-Task Supervisor
53	RELEX	3	Supported in MFT configuration
54	Disable	2	
55	EOV	4	Input/Output Support (OPEN/CLOSE/EOV) Sequential Access Methods
56	ENQ	1	Control Program with MFT
57	Freedbuf	3	
58	REQBUF	1	Basic Telecommunications Access Method
59	OLTEP	3	On-Line Test Executor
60			Not supported in this configuration
61	Save	3	TESTRAN
62	DETACH	2	Supported in MVT configuration
63	CHKPT	4	Fixed-Task Supervisor
64	RDJFCB	3	Input/Output Support (OPEN/CLOSE/EOV)
65	QWAIT	2	Queued Telecommunications Access Method
66	QTAM test	4	Queued Telecommunications Access Method
67	QPOST	2	Queued Telecommunications Access Method
68	SYNADAF	4	
69	Backspace	3	Sequential Access Methods
70	GSERV	2	Graphics Access Method
71	ASGNBFR	3	Graphics Access Method
72	IEECVCTR	4	Control Program with MFT
73	SPAR	3	Graphics Access Method
74	DAR	3	Graphics Access Method
75	Dequeue	3	Graphics Access Method
76-80	**		Unassigned
81	SETPRT	4	Sequential Access Methods
82	GETALT	4	Utilities
83-199	**		Unassigned
200-255	Available for assignment to user-written SVC routines. Until a number is assigned, its use in a processing program causes termination.		

Index

READER'S COMMENT FORM

IBM System/360 Operating System
Introduction to Control Program Logic

GY28-6605-4

- Is the material:

	Yes	No
Easy to read?	☐	☐
Well organized?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Accurate?	☐	☐
Suitable for its intended audience?	☐	☐

- How did you use this publication?
  - ☐ As an introduction to the subject
  - ☐ For additional knowledge
  - Other

- Please check the items that describe your position:
  - ☐ Customer personnel
  - ☐ IBM personnel
  - ☐ Manager
  - ☐ Systems Analyst
  - ☐ Operator
  - ☐ Programmer
  - ☐ Customer Engineer
  - ☐ Instructor
  - ☐ Sales Representative
  - ☐ Systems Engineer
  - ☐ Trainee
  - Other

- Please check specific criticism(s), give page number(s), and explain below:
  - ☐ Clarification on page(s)
  - ☐ Addition on page(s)
  - ☐ Deletion on page(s)
  - ☐ Error on page(s)

Explanation:

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS PLEASE . . .

This publication is one of a series which serves as reference for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

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

POSTAGE WILL BE PAID BY

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

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

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

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

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