SPERRY UNIVAC
Operating System/3 (OS/3)

Introduction to the Supervisor

SPERRY UNIVAC

UP-8058
Rev. 1

H

This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

Sperry Univac is a division of Sperry Rand Corporation.

FASTRAND, PAGEWRITER, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are trademarks of the Sperry Rand Corporation.

 PRINTED IN U.S.A.

preface

This manual is one of a series designed to introduce the software available with the SPERRY UNIVAC Operating System/3 (OS/3). The actual programming procedures required to use the software described are not included in this introductory series. Such detailed information is beyond the scope and intent of these manuals and is included in the appropriate UNIVAC Operating System/3 user guide. For a complete introduction to the Executive, which includes the Supervisor and Job Control, you should also read the introduction to job control, UP-8053 (current version).

introduction

Today's multijob data processing demands have made management increasingly aware of the need for more sophisticated software elements. The needed elements must efficiently, as well as concurrently, handle the ever-expanding volumes of input. Operating System/3 (OS/3) offers users the Supervisor component of the Executive to act as a master controller/coordinator of multijob processing. It accomplishes this control through a unique organization of system software routines that you are about to examine.

Because the early data processing systems were capable of performing only one program at a time, the user of these systems was always limited to fewer jobs; multijob processing was not possible. In effect, lack of advanced technology dictated the system capacities and capabilities available to the user. At that time, the user simply accepted these limitations and worked within their restrictions. Gradually, demands for greater efficiency in handling the constantly increasing volume of data at much greater speeds led to the introduction of operating systems, and eventually led to the introduction of multijobbing systems like OS/3, which is capable of executing up to seven jobs concurrently.

supervisor concepts

The Executive is the master system element that loads, initiates, executes, and terminates user jobs (composed of one or more job steps or programs), and controls and coordinates these actions. The component of the Executive that performs the job initiation and step-to-step transition is Job Control, a set of nonresident system software routines. The component of the Executive that controls and coordinates these operations and provides the interface and functions for achieving optimal system utilization is the Supervisor. The interface between the Supervisor and Job Control provides a close relationship in which Job Control functions as a dynamic extension of the Supervisor.

The Supervisor exercises control via the *task*, the lowest viable entity that can compete for central processing unit (CPU) time.

When a job initially enters the system, the system establishes one task per job. This original task is called a *job step (primary) task*. When additional tasks are needed for a job, a Supervisor macro requests their creation. Task creation and management play an important role in Supervisor control functions because CPU control is determined by internal priorities based on the task.

The Supervisor achieves coordination between executing jobs and operating system software through *interrupts*, which pass control to the Supervisor modules. An interrupt is a break in the normal flow of job-step execution so that processing can be diverted to some other function and then returned to the task the system was executing when interrupted or to some new task as prescribed by the interrupt.

There are six types of interrupts that can be caused by external or internal actions:

Interrupt Type	Interrupt Cause
Supervisor call (SVC)	Requests for Supervisor routines
Interval timer	Timer
Input/output	Input/output completion or error
Program error	Erroneous or unusual conditions in the executing programs
Hardware error	Hardware malfunctions
Operator request interrupt	External operator requests

Interrupts help to coordinate processing by allowing the system to overlap the execution of job steps from different jobs and by permitting communication between job steps and system components.

The SVC interrupt passes control to the Supervisor in order to request the performance of some function on behalf of the requesting task. The interval timer interrupt signals the passage of some predetermined element of time for clocking purposes and task synchronization. Input/output interrupts signal the Supervisor of the occurrence of discrete events on the input/output (I/O) channels. The program error interrupt indicates to the Supervisor that an error condition has occurred in a user job step. Hardware error interrupts indicate system malfunctions. The operator request interrupt enables the operator to communicate with the operating system.

When an interrupt occurs in a multijobbing environment, many jobs are in different stages of execution throughout the system. Consequently, the Supervisor software modules or I/O devices requested are not always available. Nevertheless, the interrupt request is valid and still must be serviced. While interrupt requests await servicing or attention, they are placed on lists of entries, called *queues*, maintained by the Supervisor. At interrupt time, the Supervisor examines the queues to determine which task gets a resource next.

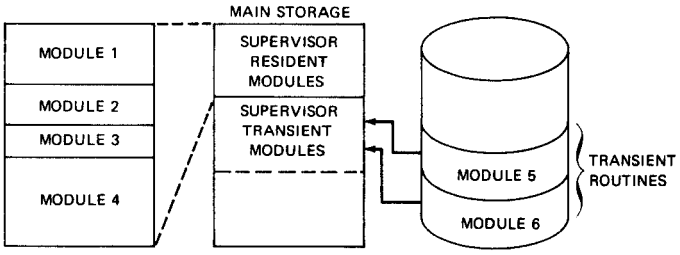
OS/3 queues operate on a *priority* arrangement. There are two types of priority in the OS/3 system: the scheduling priority and the switching priority. *Scheduling priority* is a Job Control function dealing with job initialization. *Switching priority* is a Supervisor function that determines the order in which CPU time should be distributed. Both scheduling and switching priorities are indicated to the system via Job Control Language (JCL).

In a multijobbing environment, interrupt requests prompt the Supervisor to allocate system resources and queues operating on a priority basis store the requests awaiting available resources.

supervisor structure

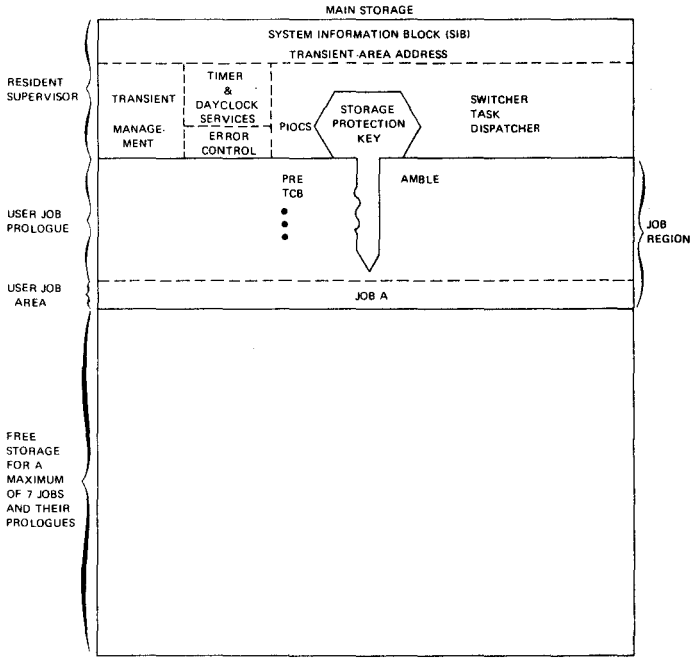
GENERAL DESIGN CONCEPTS

The OS/3 Supervisor design is modular, with each module performing a single function or service. Some Supervisor modules, are permanently resident in main storage, while other Supervisor modules are non-resident, or *transient*, routines called into main storage from disc when they are needed. To avoid impeding Supervisor performance, transient retrieval and loading is designed to execute very efficiently.



Supervisor design employs minimal main storage space for the resident functions continually needed for normal job and job step execution while efficiently using peripheral storage devices to supply transient functions when needed. Resident modules may be modified or other modules may be included as resident modules at system generation time to provide the capabilities desired for your Supervisor. However, the more resident modules you place in main storage at system generation, the more you reduce the amount of main storage available for user job step execution.

OS/3 is capable of concurrently processing a maximum of 7 jobs, each job step having up to 256 tasks. A maximum of 256 tasks can be specified by the user, and each task may be assigned up to 62 priority levels. Main storage, with the permanently resident Supervisor modules and temporary (reusable) areas, could look like this:



Generally, the resident Supervisor modules, which can be modified or expanded according to user needs, consist of:

- Physical Input/Output Control System (PIOCS)
- The task dispatcher
- Transient management
- Timer and dayclock services

- Error control

- Supervisor request handler

The Physical Input/Output Control System (PIOCS) consists of a set of functionally related modules that control the dispatching, queueing, and interrupt processing for all I/O devices directly connected to the system.

The task dispatcher, as the hub of the Supervisor, controls allocation of the CPU for task execution according to a *switch list* of internal switching priorities. There are from 3 to 64 internal switching priorities available to determine the order in which CPU control should be distributed (1 to 62 for user tasks). The switch list is a directory of task queue addresses, with each address pointing to a Task Control Block (TCB). The switch list is the task dispatcher's guide for priority processing.

Transient management schedules, locates, and loads the noncritical (interruptable) transient modules that perform the transient Supervisor functions. One segment of transient management, the transient scheduler routine, operates in the critical (noninterruptable) Supervisor environment; and the other segment, the transient loader, operates in the Supervisor task environment. Transient modules are self-relocating programs residing on the System Resident

Device (SYSRES) and are called into main storage to be executed as Supervisor overlays when required by a user job or system service.

The timer and dayclock services module is an optional part of the resident Supervisor that provides an interface between task execution and the hardware high-resolution timer by allowing a task to request an interrupt after any time interval greater than 1 millisecond. The calling task may specify, in milliseconds or seconds, the wait interval at which this interrupt is to occur. In addition, the timer provides the time of day that is used, when requested by user programs or the Supervisor, for time-stamping log messages and job accounting entries. The timer and dayclock services module supports two macro instructions:

- GETIME

  Obtains the current time and date from the dayclock.

- SETIME

  Sets an elapsed time counter to execute the job's user recovery subroutine for the requesting task when a specified time interval has elapsed.

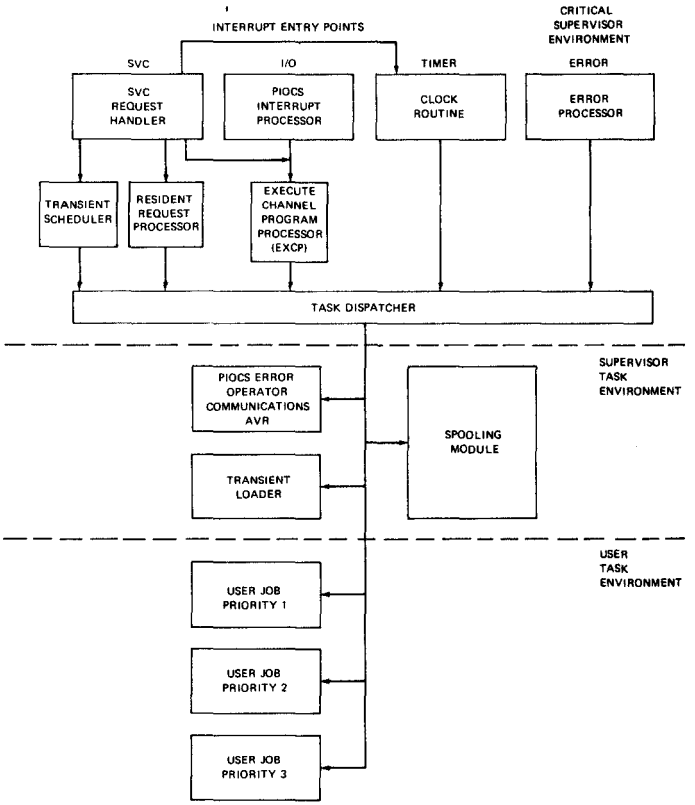
The error control module handles unresolved I/O, machine check, and program check interrupts, and schedules user recovery subroutines or overlay functions to handle error conditions appropriately. Recovery subroutines are attached to a job step via the STXIT Supervisor macro and exited via the EXIT Supervisor macro. If recovery subroutines are not supplied by the user, the Supervisor takes appropriate action to recover (if possible) or cancels the job.

The Supervisor request handler determines which function is requested by the Supervisor and routes control to the routine which satisfies the Supervisor request. If the requested function is nonresident, the Supervisor request handler routes control to transient management; otherwise, it passes control to the specific resident routine.

SUPERVISOR ELEMENTS AND INTERFACES

Because the modular design of the OS/3 Supervisor is function oriented, the internal correlation of Supervisor activities is accomplished by control modules that manage the Supervisor elements. Macros pertinent to each facet of Supervisor management provide interfaces needed to relate Supervisor elements to other elements, to job steps, or to operator communications. Input/output control, task management, resource allocation, transient management,

console management, spooling, Automatic Volume Recognition (AVR), and error control are all Supervisor elements handled by the various control modules via specific functions that we will examine more closely here. You will find the following diagram helpful in visualizing the intermodular control flow that correlates Supervisor activities and results in the ultimate execution of your jobs.



PHYSICAL INPUT/OUTPUT CONTROL SYSTEM (PIOCS)

PIOCS controls dispatching, queueing, and interrupt processing for all I/O devices. It is the interface between the CPU and I/O devices. Basically, the PIOCS is divided into three distinct sections:

1. The PIOCS interrupt processor

2. The Execute Channel Program (EXCP) processor module

3. The error processor

Each section contains related modules. The PIOCS interrupt processor handles I/O interrupts caused by the asynchronous ending of I/O operations. The EXCP processor contains routines used in validating, queueing, and conditionally executing I/O requests based on channel and device availability.

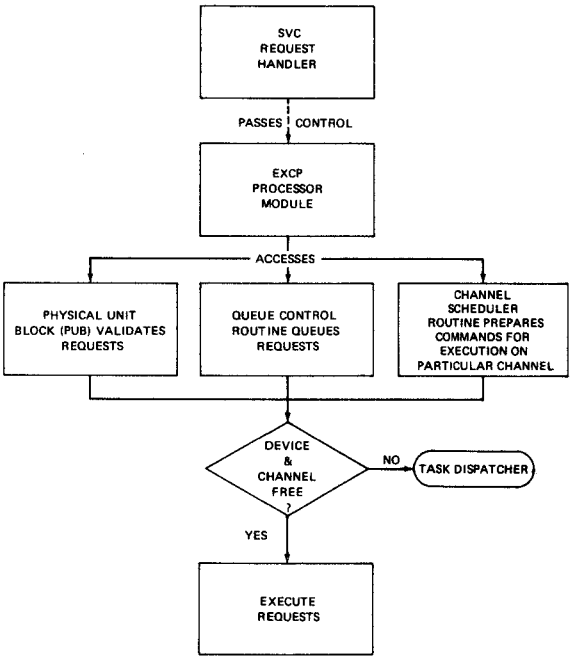
The EXCP processor module receives control from the SVC request handler and accesses the following routines:

- The PUB routine validates the availability of the device being called once a request is submitted and locates devices associated with hardware interrupts, answered operator communications, and communications interrupt suspension.

- The queue control routine places the request on the I/O queue, retrieves it from the queue and, in the case of program errors, deletes a request from the queue.

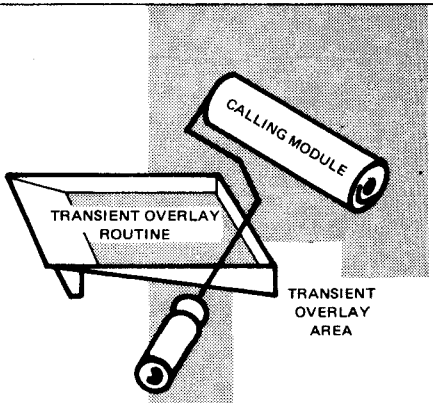
- The channel scheduler routines (one for each type of channel) prepare requests or commands for execution on a particular channel.

The standard exit from the EXCP processor module is via an unconditional branch to the task dispatcher. Operation of the EXCP processor is shown in this diagram:

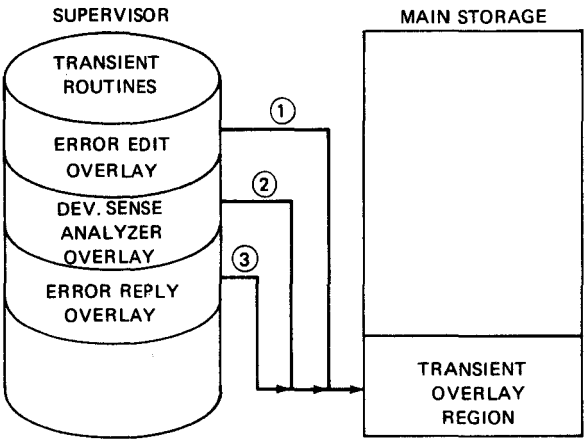


In order to prepare requests or commands for execution on the different channels, the programmer can use a set of PIOCS declarative and imperative macros to communicate with the PIOCS channel scheduler. Essentially, the declarative macros build control blocks that contain information about files, channels, or devices to be used. The imperative PIOCS macros communicate directly with the I/O scheduler to submit I/O requests, locate and read file control blocks, and exchange various control block addresses.

The PIOCS module also includes the error processor, which processes all PIOCS error conditions via the error editing root overlay, device sense analyzer overlay, and error reply overlay. An *overlay* is a transient routine that is called into main storage and overlays, or replaces, all or part of another transient routine that is no longer needed. Overlaying in main storage is a technique that uses the same blocks of internal storage during different stages of job execution. When an overlay routine is no longer needed in main storage, it is replaced partially or entirely by another transient overlay routine.



The PIOCS error control module alerts the error editing overlay routine when operator communication is needed. When the error editing overlay comes into the main storage transient overlay area, it processes I/O error messages to the operator, validates and prepares reply options, and calls the appropriate device sense analyzer overlay routines. Their function is to convert the sense information of a particular error into an English-language message to the operator and to call the error reply overlay. Finally, the error reply overlay processes an answered I/O error message to the operator.



NOTES:

(1) calls (2) overlay into transient overlay region.

(2) calls (3) overlay into same transient overlay region.

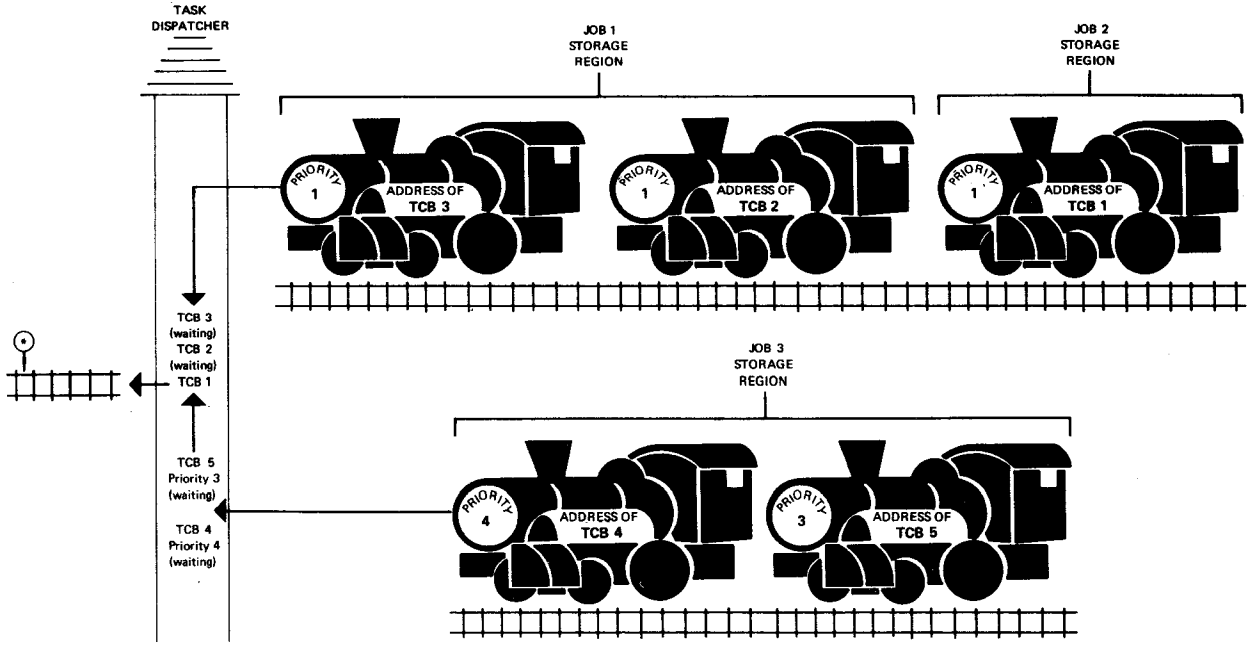
TASK DISPATCHER AND TASK MANAGEMENT

Earlier, we briefly described the task dispatcher, located in main storage, as the hub of the Supervisor. The task dispatcher allocates CPU time to tasks based on priorities. The switch list contains a position for each priority level. These positions have address pointers for each priority level at which tasks exist. A pointer indicates the first task to be evaluated for dispatching. If the first task is not dispatchable, the chain of tasks is rotated until each task at that priority level is evaluated prior to scanning the next lower priority level.

We learned that a maximum of seven jobs can be activated by the job scheduler for concurrent execution and that job steps can consist of one or more tasks executable by internal priority levels. Every task is identified to the Supervisor by a Task Control Block (TCB), which contains or points to all the control information associated with a task. The switch list is composed of priority levels that contain entries of TCB addresses taken from your job prologue areas in main storage.

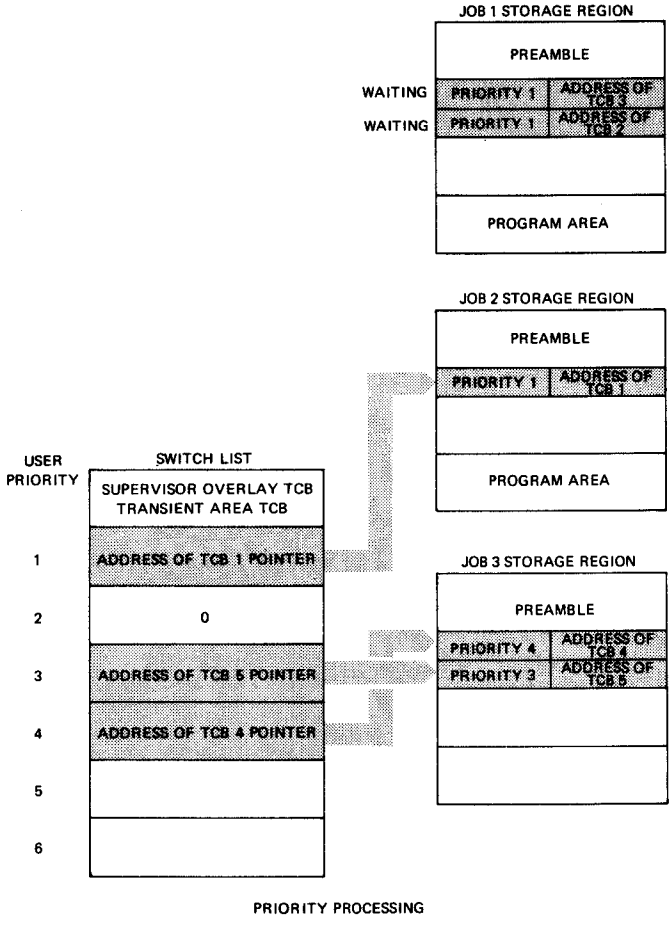
When the task dispatcher gains control, it selects the highest priority active task and dispatches control. A task is active if it can use the CPU and inactive if some event must take place before the task can use the CPU.

The task dispatcher's priority function is very similar to tower control in a busy railroad yard. Many different trains (TCBs) arrive from various locations, each carrying a load (the TCB address of the place where the task control information is stored). The trains arrive on tracks coming from all directions (the job storage regions). The tower (task dispatcher) must then analyze or scan the loads (TCB addresses) of waiting trains in relation to all traffic in the railroad yard (system) for a ready TCB. The tower then dispatches the load (ready TCB) to the mainline track (control of CPU).



*PRIORITY NUMBER INCREASES AS TCB'S ARE PROCESSED

Now compare the tower functions to the task switcher dispatcher's functions by following the control flow arrows in this diagram:



In addition to the primary task inherited from Job Control at job step initialization time, you can specify the creation of a task and a priority for the task via the ATTACH macro. Other task management macros provide the interface that jobs need to activate, deactivate, delete, control, and synchronize additional tasks in a multitasking environment. Your programs are interfaced by another set of macros that load parts of programs, relocate address constants, and transfer control.

To begin task termination functions, a task executes a DETACH macro that determines whether the DETACH macro was executed on normal or abnormal termination of the job step.

RESOURCE ALLOCATION

Although the Supervisor is capable of allocating certain *resources*, it does *not* allocate devices to jobs. The special capability of the OS/3 Supervisor to allocate or extend permanent and temporary files during job step execution is a definite asset. Disc space management routines provide you with a completely automatic space accounting and maintenance feature. Thus, you don't need to be concerned with the precise contents of direct access volumes. Like other elements of the OS/3 Supervisor,

disc space management provides you with a set of macros that obtain information, allocate and extend disc space, or scratch and rename files.

TRANSIENT MANAGEMENT

In our discussion of the OS/3 Supervisor basic design, we talked about a resident and transient Supervisor. Transient management is the Supervisor element that handles transient routines. It is designed to locate and load a transient routine very efficiently by requiring only one access to the disc. There are two routines involved in transient management: the transient scheduler and the transient loader. The *transient scheduler* receives control from the SVC request handler and executes as a Supervisor critical (no-ninterruptable) function. The transient scheduler allocates a main storage transient area and, via the task dispatcher, schedules the transient loader to receive control as the task associated with the main storage transient area. The *transient loader* then computes the disc address of the needed transient routine and initiates a read operation of that transient routine. Once the read operation is completed, control passes to the transient routine, and transient management services are continued to completion.

CONSOLE MANAGEMENT

The operator can activate console management (a Supervisor transient module) via system console messages and commands. He does this by hitting the MESSAGE WAITING (MSG) key, causing an attention interrupt. The programmer can activate the same module by using these macro instructions in the program:

OPR

Displays a message on the system console.

WTL

Writes a message in the log files.

WTLD

Writes a message in the log file after displaying it on the system console.

GETMSG

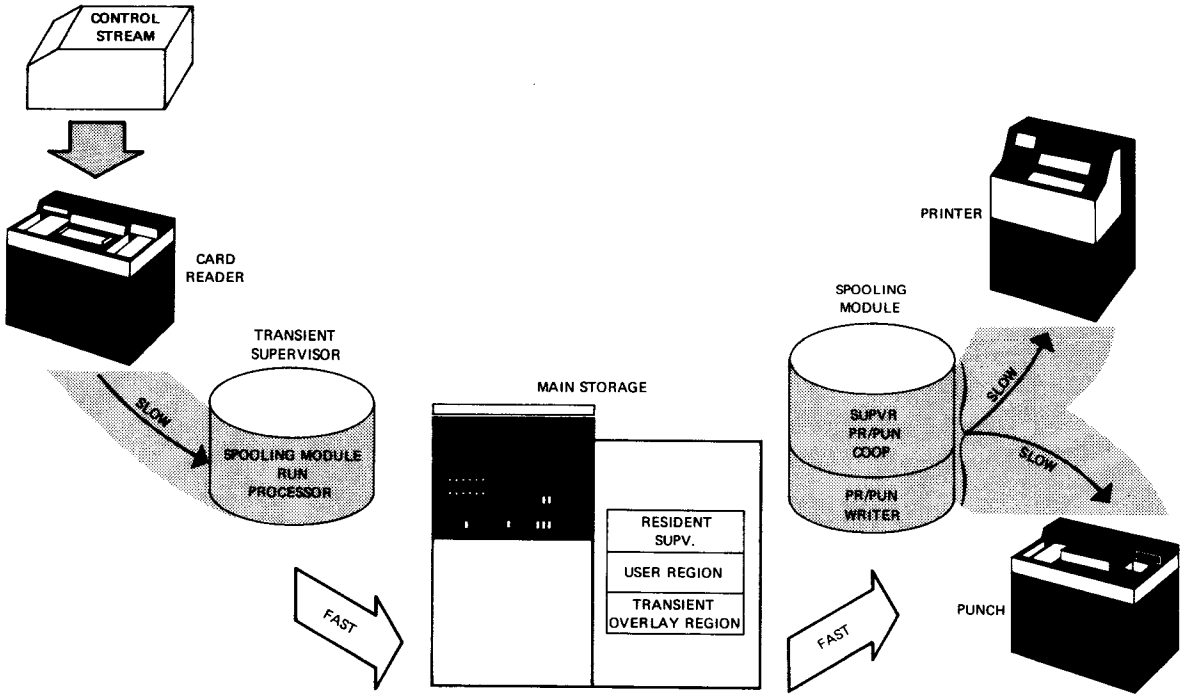
Receives messages from the canned message file.

SPOOLING

Another valuable OS/3 Supervisor module is the spooling *(Simultaneous Peripheral Operation On Line)* control module. It has a RUN processor routine, a Supervisor printer/punch spooling cooperative routine, and output printer/punch writer routines.

These routines buffer, on disc, data files received from low-speed input devices or intended for low-speed output devices. Since they operate concurrently with user and system program execution, they are called *symbiont* routines. The spooling module permits a high-speed transfer of card input file from disc to main storage and, of the output file from main storage to disc for printer output.

The RUN processor reads embedded data cards from a Job Control stream into the run library and other card data onto a disc file specified by the user. The Supervisor printer/punch cooperative acts in concert with the output printer/punch writers. User printer/-punch requests are diverted to disc files that are transferred at job termination time to printer/punch devices at compatible speeds.


CONTROL STREAM
CARD READER
TRANSIENT SUPERVISOR
SLOW
SPOOLING MODULE RUN PROCESSOR
FAST
MAIN STORAGE
RESIDENT SUPV.
USER REGION
TRANSIENT OVERLAY REGION
SPOOLING MODULE
SUPVR PR/PUN COOP
PR/PUN WRITER
SLOW
SLOW
FAST
PRINTER
PUNCH

AUTOMATIC VOLUME RECOGNITION

In addition to spooling, you have a convenient feature, Automatic Volume Recognition (AVR), activated by the attention interrupt. Attention interrupts are caused by turning on most devices. AVR checks to see if required tapes and discs are already mounted. This reduces job setup time by the operator and the number of responses required of him at the system console.

ERROR CONTROL

There is an error control module contained in the resident Supervisor that examines any error causing a program check or machine check interrupt and takes appropriate action for the interrupts via recovery subroutines, which are given control when certain error conditions occur. If user recovery subroutines are not supplied, or if error recovery fails, abnormal termination occurs. This error control module is not the same as the PIOCS error processor overlay routines, which are transients.

DIAGNOSTIC AND DEBUGGING AIDS

After exploring all the Supervisor resident and transient features, you might ask what diagnostic aids are available for debugging programs. Earlier, we talked about symbionts and their functions as parts of the spooling operation. The monitor/trace symbiont is another routine that concurrently transfers information about the environment of each instruction to print as a trace of the program's execution. This is a valuable debugging tool. You can also request a partial main storage printout at given points in your program by using the SNAP macro, and there is always the option of overriding the SNAP macro via Job Control.

conclusion

OS/3 SUPERVISOR FEATURES

An effective way of realizing the value of OS/3 Supervisor features is to examine them in the categories of CPU and I/O functions. There are OS/3 features relating to the following CPU functions:

- Resident Supervisor
- Main storage management
- Job step management

One very attractive feature of the OS/3 Supervisor is the small amount of main storage (6K–7K) it requires when supporting a minimum configuration, including the Integrated Peripheral Channel (IPC) and the Integrated Disc Adapter (IDA).

The Supervisor features an optional number of transient execution areas, allowing for increased capacity in software expansion possibilities.

In addition, the Supervisor maintains a variable number of priority levels. The number of internal priority levels is relative to the number of entries in the switch list. This feature permits large task priority

hierarchies. The Supervisor transient loader module loads transient routines and variable-length overlays efficiently with one disc access.

Included in the Supervisor main storage management features is the support of serially reusable and private copy transients. If a request for a transient is received and that transient has already been loaded for a previous operation and is serially reusable, it can be reused without reloading.

OS/3 Supervisor features for job step management include:

- CPU dispatching and I/O execution on a programmer-selectable task priority basis
- Multijobbing of up to 7 user jobs
- Multitasking of up to 256 tasks per job step
- Automatic relocation of job steps requiring relocation

The programmer can request a CPU dispatching priority via the EXEC control statement in his Job Control stream. Any subtask created by the job step task or other subtask can have a dispatching priority specified on the ATTACH call macro. In effect, the

OS/3 Supervisor is flexible enough to allow the programmer some control via the Job Control Language (JCL) and Supervisor Macro Interfaces.

The Supervisor capability of multijobbing up to 7 jobs with a maximum of 256 tasks offers greater processing efficiency and speed, as well as increased job step management flexibility.

The loader automatically relocates job steps requiring relocation via either the hardware relocation register or, in some cases, via the LOADR macro. Here you can see the advantages of hardware and software interfacing.

The OS/3 Supervisor offers the following I/O functional capabilities and provisions:

- It maintains peripheral device independence for other system software through System Access Technique (SAT).

- It provides an optional backup transient file.

- Its symbiont capabilities allow unlimited dynamic extension to the Supervisor.

- Its space management provides split-cylinder allocation and automatic file extensions.

- It supports Automatic Volume Recognition (AVR).

Device independence means more efficient I/O implementation, a significant advantage when you're faced with an increased volume of data processing.

Since transient areas are recurrently overlaid, they tend to be more vulnerable areas when unrecoverable I/O errors occur. OS/3 provides an optional transient backup file, which gives your programs added security and protection.

Symbiont processing is another timesaver, which increases system throughput without requiring additional resident Supervisor storage. Extensions to the Supervisor are made possible by the symbionts, which operate concurrently with the user and other system programs.

Split-cylinder allocation is a special type of allocation that may be made when two or more related disc files share the same cylinders. This type of allocation minimizes access arm movement. Automatic file extensions are also available for split files via the EXTEND macro.

AVR checks to see if required tapes and discs are already mounted. In addition, AVR reduces the operator job setup time and the number of system console responses required.

SUPERVISOR EFFECTIVENESS IN MULTIJOB APPLICATION

Let's assume that you have compiled and stored on disc several jobs composed of two or more job steps. Job A is a series of payroll calculations, reports, and paychecks; Job B, a comprehensive analysis of company statistics; and Job C, an engineering design calculation. Each job contains job steps relative to its objectives but not necessarily relative to the objectives of the other jobs in the multijob environment. Your jobs are loaded, initiated, executed, and terminated by the appropriate portion of OS/3; however, to provide all the jobs with execution time and resources, the Supervisor is called into action to control and coordinate the activities of the Job Control with the user job steps and other established system software on a multijob basis.

When Jobs A through C are loaded into main storage and initiated for execution, the Supervisor builds a prologue for each job. Much of the information in the prologue is supplied by Job Control, and the prologue provides continuous storage from job step to job step for system and user information or information address locations. The following diagram illustrates the multijob layout and main storage in multijob execution from job step to job step. Jobs execute concurrently in a multijobbing environment. This means that only one job step per job can be executed at any one point in time.

Each job step has a job step or primary task inherited at job step initialization from Job Control, and any further tasks are created via the Supervisor ATTACH macro. In addition, the Supervisor structures one switch list to indicate task priorities among all jobs in the system.

MULTIJOB LAYOUT

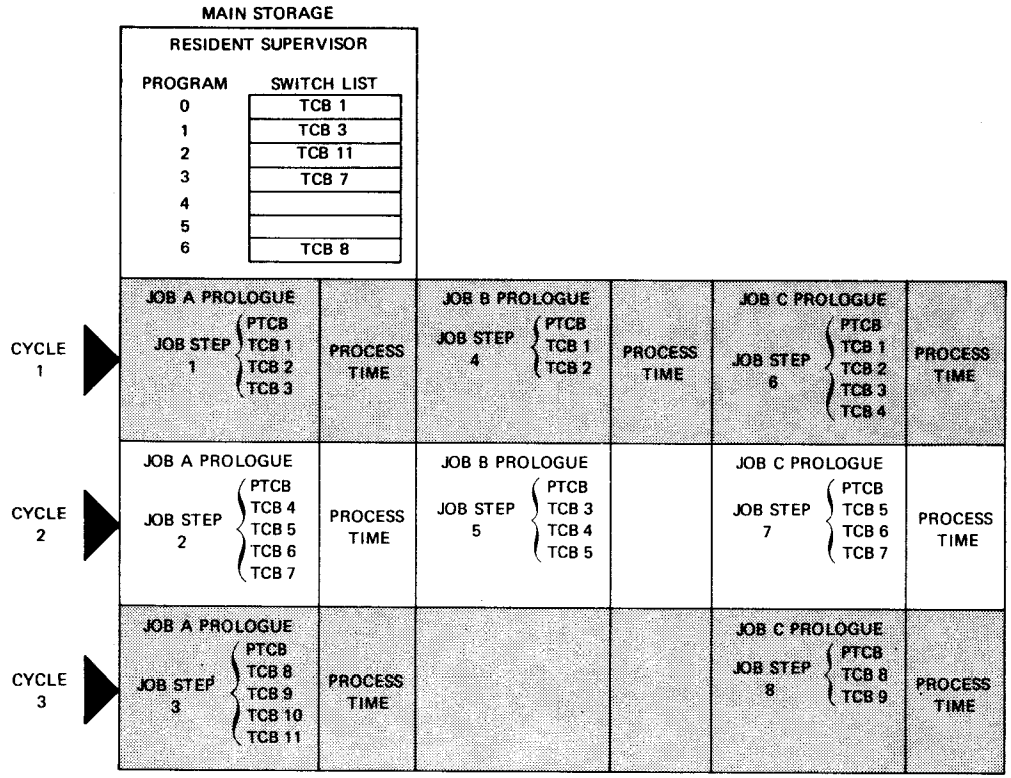
JOB A PAYROL
JOB STEP 1
JOB STEP 2
JOB STEP 3

JOB B COSTAT
JOB STEP 4
JOB STEP 5

JOB C EGRDSN
JOB STEP 6
JOB STEP 7
JOB STEP 8

MULTIJOB EXECUTION IN MAIN STORAGE

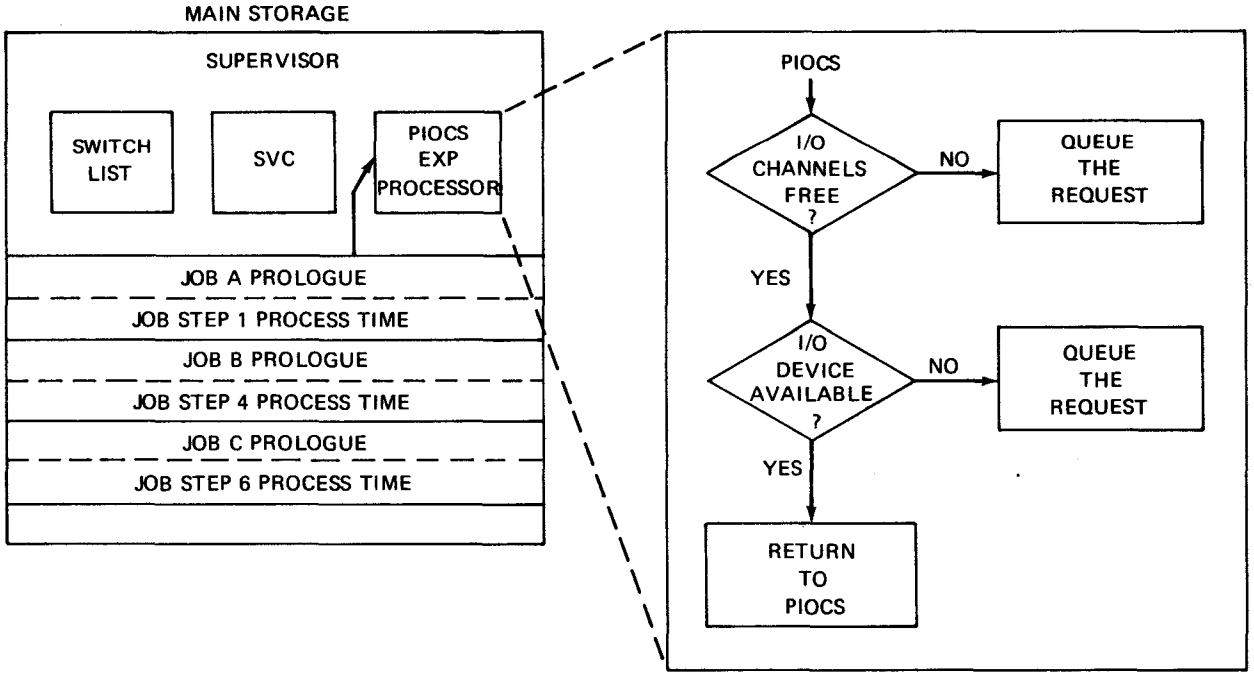
		JOB A PAYROL		JOB B COSTAT		JOB C EGRDSN
CYCLE 1	PROLOGUE JOB A	JOB STEP 1	PROLOGUE JOB B	JOB STEP 4	PROLOGUE JOB C	JOB STEP 6
CYCLE 2	PROLOGUE JOB A	JOB STEP 2		JOB STEP 5	PROLOGUE JOB C	JOB STEP 7
CYCLE 3	PROLOGUE JOB A	JOB STEP 3			PROLOGUE JOB C	JOB STEP 8



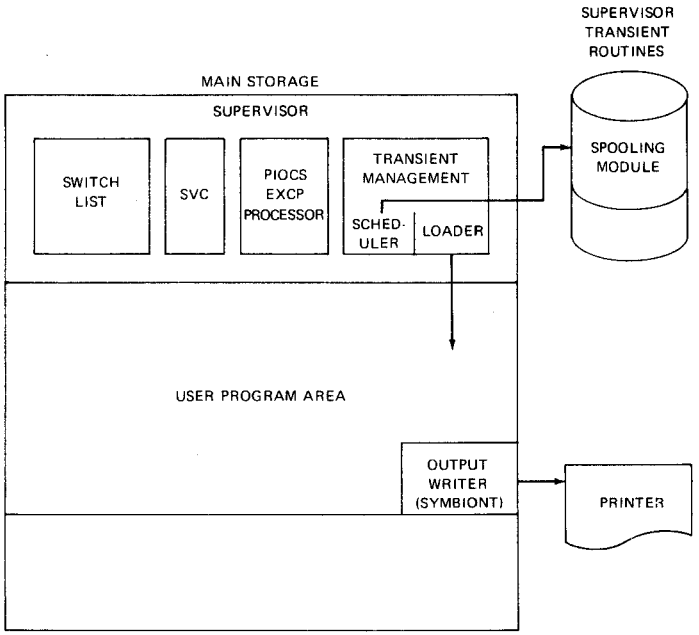
PTCB = PRIMARY TASK CONTROL BLOCK
TCB = TASK CONTROL BLOCK

Once job execution begins, the interrupt processing technique implements multijobbing. Let's suppose that the first job step in your Job A issues a READ instruction. Your job step is requesting an I/O read device. Let's say that some payroll information must be entered as data embedded in the control stream and that your OS/3 system is equipped with the spooling module (a transient Supervisor routine). Consequently, you spool in your input data.

First the READ instruction in your job step issues an I/O interrupt to the Supervisor. The Supervisor Call (SVC) instruction, which interfaces your program to the Supervisor, accesses the Execute Channel Program (EXCP) processor of the Supervisor PIOCS module. The PIOCS module begins its test for I/O channel and device availability. Since we have seen that queues are formed when system services are not available, we will assume that the I/O channels and device are free and available.


MAIN STORAGE
SUPERVISOR
SWITCH LIST
SVC
PIOCS EXP PROCESSOR
JOB A PROLOGUE
JOB STEP 1 PROCESS TIME
JOB B PROLOGUE
JOB STEP 4 PROCESS TIME
JOB C PROLOGUE
JOB STEP 6 PROCESS TIME
PIOCS
I/O CHANNELS FREE ?
NO
QUEUE THE REQUEST
YES
I/O DEVICE AVAILABLE ?
NO
QUEUE THE REQUEST
YES
RETURN TO PIOCS

The next module that the Supervisor needs to access is the Supervisor spooling control module (a Supervisor transient routine). The Supervisor accesses transient modules via the transient scheduler module, which transfers control to the transient loader. The transient loader initiates a read of the transient routine (in this case, the spooling module), and the spooling procedure files your job step input data onto the spooling disc. Interjob and intrajob step execution occurs via Job Control under Supervisor control.



If job step or machine errors occur during Supervisor operations, the error control module handles the error via error interrupts.

With optional timer and dayclock services available in the Supervisor, CPU time can be more precisely controlled; however, the interrupt processing technique produces a very efficient multijobbing result, with error recovery routines provided in the resident as well as transient Supervisor modules.

You should realize that interrupts for Job B or C during the execution of Job A cause requests from Job B and/or C (up to a maximum of seven jobs) to be recognized by the Supervisor and either queued or their execution begun, according to the task priorities listed on the task switch list.

summary

The OS/3 Supervisor incorporates a powerful set of software modules with minimal main storage usage. In combination with a valuable transient overlay structure, it gives you a high-efficiency, low-cost operating system that is flexible and easily expanded.

A more detailed description of the features and capabilities of the UNIVAC Operating System/3 Supervisor, complete with illustrations and examples drawn from actual practice, is available in the supervisor user guide, UP-8075 (current version).