TDOS

CONTROL SYSTEM

REFERENCE MANUAL

RCA
Information
Systems

SPECTRA 70

TAPE-DISC OPERATING SYSTEM (TDOS)

# Control System
# Reference Manual

SPECTRA 70

TAPE-DISC OPERATING SYSTEM (TDOS)

# Control System Reference Manual

~~70-00-611~~
July 1969

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.


| | |
|---|---|
| First Printing: | January 1967 |
| Reissued: | May 1967 |
| Reissued: | July 1969 |

# CONTENTS

# CONTENTS
(Cont'd)

# CONTENTS
(Cont'd)



**LIST OF TABLES**

# CONTENTS
(Cont'd)

**LIST OF TABLES (Cont'd)**

# 1. INTRODUCTION

◆ The Spectra Tape-Disc Operating System (TDOS) is a disc resident multiprogramming system having multichannel communications capabilities. The following components may comprise the TDOS System:

1. Executive Control System (ECS)
   a. Executive (Resident Element)
   b. Monitor (Nonresident Element)
2. Multichannel Communications System (MCS)
   a. Communications Interrupt Analysis (CIA)
   b. Multichannel Communications Program (MCP)
   c. Communications Users Program (CUP)

All programs within the TDOS environment are executed under control of the Executive. Specific TDOS programming components are required to operate under control of the Monitor, which is an Executive extension that controls the job stream environment. This manual describes the Executive Control System in detail and explains how the ECS relates to and communicates with programs operating under its control. In addition, a brief description of the use of the TDOS system is included to describe those functions relating to the execution of programs within the system.

The Multichannel Communications System consists of Communications Interrupt Analysis (CIA), the Multichannel Communications Program (MCP), and the Communications Users Program (CUP). CIA is a resident extension to the Executive that handles communication interrupts and acts as an interface between the Executive and the communication programs. This manual describes CIA only as it interfaces with the Executive. Communications programming descriptions (MCP, CUP) are covered in detail in the TDOS Multichannel Communications System Manual (70-00-612).

## TDOS PROGRAMMING COMPONENTS

◆ The Spectra 70 Tape-Disc Operating System (TDOS) is an operator-controlled multiprogramming system that provides a standard operating and programming environment for the 70/35, 45, and 55 Systems. The Tape-Disc Operating System facilitates the preparation, execution, and maintenance of application programs. The components of TDOS are the Control System (Executive, Monitor, and File Control Processor), Multichannel Communications System (MCS), Language Translators, Library Maintenance routines, Peripheral routines, Sort/Merge Generator, Automatic Integrated Debugging System, Diagnostic routines and Transcriber routines. All programs are controlled by the Executive to ensure that maximum efficiency is maintained through the use of common programming facilities, input/output devices, and other equipment features in the system. All information and programs either entering or leaving the system are under the direct control of the Executive Control System.

**Executive**

◆ The Executive Control System maintains complete control over the system and its physical resources. These resources are core memory, peripheral devices, processor time, and supervisory functions. In controlling the system and its physical resources, the Executive Control System controls the initiation, execution, and termination of programs. Program requests for the allocation of memory and the assignment of devices are submitted to the Executive. The Executive determines if these resources are available and, if available, assigns them to the requesting program and protects the assigned resources from other programs operating on the system. ECS also maintains control over which program should be serviced first according to priority in the multiprogramming mode. In addition to its program control function, the Executive decodes and services all interrupts, maintains complete control over the Console Typewriter, and controls all physical input/output operations.

**Monitor**

◆ The Monitor, the nonresident component of the Executive, controls a run-to-run sequenced operation as determined by a job stream with little or no operator's intervention. Program preparation (language translation and binding) is performed under control of the Monitor. Maintenance of the system's libraries is performed under control of either the Executive or Monitor.

**File Control Processor**

◆ The File Control Processor enables the program to perform input/output operations without regard to the physical constraints and controls required by the Executive. In effect, the File Control Processor is the interface between logical and physical input/output requirements. The File Control Processor provides for the processing of files recorded in a serial mode on conventional devices, such as magnetic tape. It also provides for the processing of files recorded in a serial, sequential, or random mode on direct-access devices, such as disc.

**Communications Interrupt Analysis**

◆ The Communication Interrupt Analysis (CIA) is a resident extension to the TDOS Executive that interfaces with the Multichannel Communication Program (MCP) to act upon communications interrupts. When a TDOS Executive with communications capabilities is used, the CIA is always resident.

**Language Translators**

◆ The TDOS system provides four language translators: Assembly, COBOL, FORTRAN, and the Report Program Generator (RPG). Any one or a combination of these language translators may be used to state the solution to a problem.

**Library Maintenance**

◆ The three libraries that comprise TDOS are: System Library, Call Library, and Program Load Library. These libraries are used in the preparation, maintenance, and execution of both installation and RCA-supplied programs.

All of the libraries are modular and may be expanded or contracted as requirements dictate, thereby permitting centralization or decentralization of the libraries and their contents. To maintain the system libraries and provide for their modularity, TDOS includes a set of Library Maintenance routines.

**Peripheral Routines**

◆ Peripheral routines are provided in TDOS to simplify data conversion. Data can be converted from a system-supported medium with a minimum of parameters. Many of the peripheral programs permit editing of the data and printing of data during the conversion process.

**Sort/Merge**

◆ The TDOS Sort/Merge Generator provides for the generation of a tailored Sort/Merge program based upon supplied parameters. Own-code may be bound into the generated programs and, if desired, overlay modules of own-code may be constructed by the bind routine. The output of the Sort/Merge Generator must be bound before the Sort/Merge program can be executed.

**Automatic Integrated Debugging System**

◆ TDOS provides an Automatic Integrated Debugging System (AIDS) to facilitate the preparation and testing phases of program production. AIDS provides the ability to automatically perform the testing of one or more programs without requiring the programmer to be present. AIDS also provides the facility to perform the console-controlled testing of any program.

**Diagnostic Routines**

◆ In addition to the Automatic Integrated Debugging System, TDOS provides a complete package of Diagnostic routines. These routines are designed to meet all program testing requirements. In addition to program-initiated diagnostic functions, facilities are provided for dumps of memory and devices caused by a machine condition failure.

**Transcriber Routines**

◆ Transcriber routines are provided in TDOS to transcribe the Executive, Program Load Library, and Call Library to disc or drum.

## PREPARATION OF THE APPLICATION

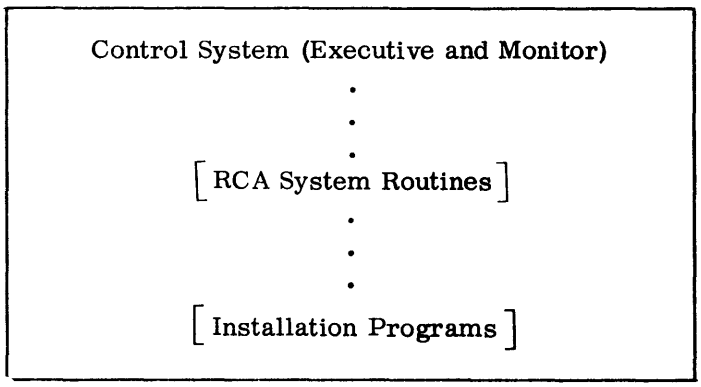
◆ To prepare an application for use in the system, the user must be thoroughly familiar with the capabilities and limitations of the total system. Of particular importance, however, is a thorough knowledge of the TDOS libraries required by the system and their use; of the processing requirements for program preparation; and the various methods by which programs may be executed in the system.

## TDOS LIBRARIES

◆ There are three libraries in the Tape-Disc Operating System: the System Library, the Call Library, and the Program Load Library. Each of these libraries is modular in construction; therefore, the contents of each library may be deleted or added as installation requirements dictate. This modular construction provides the installation with the capability to optimize its library requirements and thereby to eliminate needless processing. It also provides the installation with the option of operating with all the libraries on disc and drum or with the program libraries on disc, drum, and tape.

**System Library**

◆ The System Library, which must be resident on disc or drum, is required to execute programs on the system. This library is constructed as follows:




*Notes:* 1. There may be more than one System Library in an installation, but only one may be active at a time.

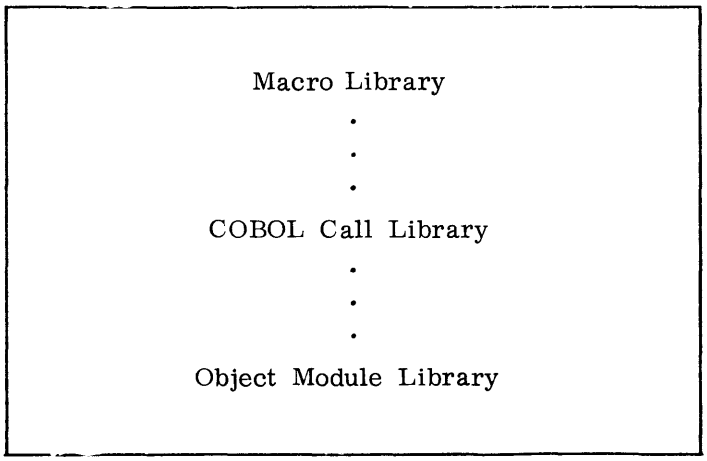
2. The selection as to which RCA system routines appear on this library is determined by the installation. Only those RCA system programs that are required need appear on this library.

3. Installation programs may optionally reside on this library.

4. Each program, except the Executive, is floated at loading time.

**Call Library**

◆ The Call Library which must be resident on disc or drum is required for program preparation processing. It is used by the various Language translators and the Linkage Editor routine to facilitate the preparation of programs. This library is constructed as follows:

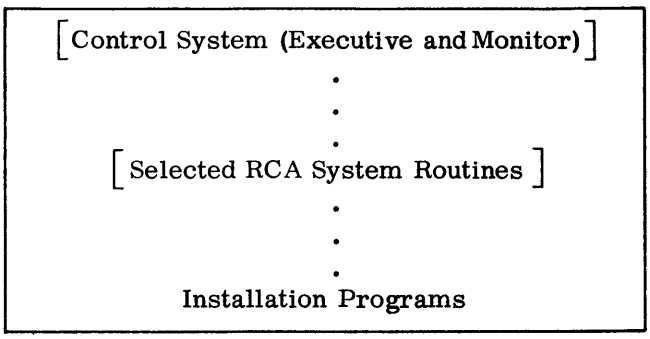



*Notes:* 1. Each library component may appear on disc or drum.

2. More than one Call Library may not be active at a time. However, more than one Call Library may be used during a program preparation session, each being used at different times during the session.

**Program Load Library**

◆ The Program Load Library which may be resident on tape, disc, or drum is required for installation production processing. This library is constructed as follows:




*Notes:*
1. There may be more than one Program Load Library in an installation, but only two alternate disc/drum and one alternate tape program load libraries may be active at a time.
2. Selected RCA System routines may optionally appear on this library.
3. This library may optionally reside on tape, disc, or drum.

**Summary of Library Use**

◆ 1. A System Library is required to run any program in the system.

2. Two libraries (System Library and Call Library) are required during program preparation processing.

3. One or two libraries (System Library or System Library and Program Load Library) are required for production processing.

**PROGRAM PREPARATION**

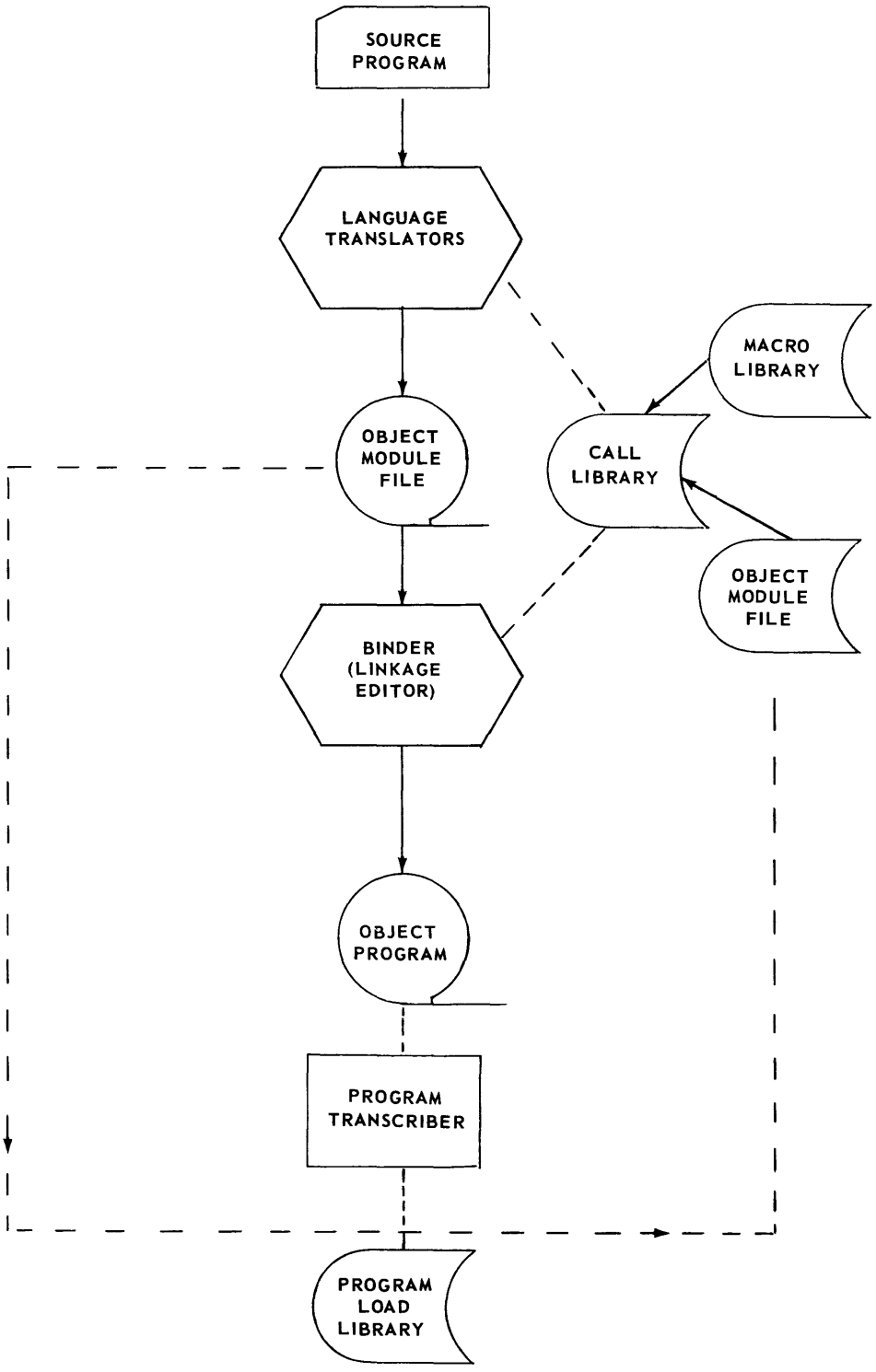
◆ All program preparation on the Tape Disc Operating System is performed under the control of the TDOS Monitor, a nonresident Executive extension, which controls the sequential execution of programs contained in a job stream. In preparing programs for execution, two processing operations are required: language translation and program binding. Each language translator produces a common output format which is termed an object module file. An object module file is card-image formatted and represents a single program memory load (program segment). Each object module file is acceptable input to the binding routine (Linkage Editor). During the binding operation, object module files are bound together (if required) and reformatted into core-image library format to facilitate loading and search operations.

**Modular Program Construction**

◆ Common language translator output (object module file) is the basis for providing modular program construction capabilities. Because the language translators produce a common output, any combination of these outputs may be bound together to form a program. To eliminate the need for retranslating object modules that are common to many programs or to provide a program back-up level beyond source program translation, object module files may be retained by means of an update routine on the object module library. When object modules are retained on the library, they are selectively bound during Linkage Editor processing with other object modules to form new programs.

**Sample Process Flow**

◆ A simplified process flow illustrating program preparation is presented below:



## PROGRAM EXECUTION

◆ The multiprogramming capability of the system enables the concurrent execution of up to six programs in a non-communication system. In a communication system one communication user program and five additional non-communication may be multiprogrammed. However, this does not preclude *job scheduling* methods to be employed which indicate that many programs are to be executed sequentially (batched processing) without operator intervention between programs. In explaining job scheduling, it is first necessary to define the three methods which may be used for program execution:

1. A program may be executed independent of any other program on the system.

2. A program may be executed as part of a chain of programs which are to be executed sequentially and are related to each other in a logical process flow (successor chaining).

3. A program may be executed under the control of TDOS Monitor, which permits the sequential execution of jobs as defined by a job input stream.

Any combination of the above methods up to six programs (successor chains and a Monitor job stream constitute one program each with regard to multiprogramming) may be executed concurrently. The one exception is that only one Monitor job stream may be executed at any given time. Thus, the determination of the program mix (combination of programs executing independently, in a successor chain or under Monitor control) is left to the discretion of the application in order that the application may optimize its system usage.

An example of a program mix showing three programs (one independent program, a successor chain, and a Monitor controlled session) is given below:

| Independent Program | Successor Chain | Monitor Session |
|---|---|---|
| Program A | Program B | Program D |
| | Program C | Program E |

When Program B completes, it internally calls Program C, and if desired, may pass devices to it (i.e., output from Program B becomes input to Program C). The same is also true regarding the relationship between Programs D and E. There is no specified limit to the number of programs that may be chained together or run under Monitor control.

**PROGRAM EXECUTION (Cont'd)**

In the preceding example, let us assume that each of the three execution methods have been initiated and their respective programs are running on the system; that is, Programs A, B, and D are concurrently sharing the processor. Now Program B completes while Programs A and D are still running. Program B, at completion, calls Program C. Program C is then loaded into memory, without operator intervention, and the program mix is continued. Only now the mix consists of Programs A, C, and D. The next program to complete is Program A. Since Program A is operating independently no other program is to be called at its completion. Therefore, the number of programs sharing the processor is reduced from three programs to two, leaving Programs C and D to share the processor. Program D now terminates and calls Program E to be loaded into memory (without operator intervention). The program mix now is changed to Programs C and E. Program C is next to complete and since it is the last program in the chain, the chain is complete leaving Program E to singularly use the processor. At the completion of Program E the operation as defined is complete.

*Note:* This example emphasizes the job scheduling capabilities available in the system as opposed to its multiprogramming capabilities. Therefore, the number of programs sharing the processor was three (a maximum of six permitted) and new programs were not initiated when the number of programs sharing the processor was reduced from three to two to one to zero.

Aside from this example, any time the number of programs currently sharing the processor is less than six and the memory required by a program is available, that program may be initiated for execution.

# 2. USE OF THE TAPE-DISC OPERATING SYSTEM

◆ Implementation and use of the Tape-Disc Operating System require the user to be thoroughly familiar with both his application and the Tape-Disc Operating System. This section describes the concepts of the Tape-Disc Operating System.

## SYSTEM INITIATION

◆ To initiate the Tape-Disc Operating System the operator performs the following:

1. Places the channel and device number of the disc or drum where the Executive is located into the LOAD UNIT switches on the console.
2. Presses the GEN RES pushbutton.
3. Presses the LOAD pushbutton.

Pressing the LOAD pushbutton causes the Bootstrap (Record 1 of Cylinder 0, Track 0) to be read into memory at location zero. Control is then transferred to location zero and the processor enters into the $P_1$ state (Processing State).

### Bootstrap

◆ The Bootstrap enables the Executive to be loaded by performing the following:

1. Reads Record 2 (Initial Program Loader) and Record 3 (Standard Volume Label) to contiguous areas in upper core.
2. Moves the device address of the disc or drum to the Initial Program Loader.
3. The Bootstrap then transfers control to the Initial Program Loader.

### Initial Program Loader

◆ The IPL then picks up the Standard Volume Label record, which contains the address of the Volume Table of Contents (VTOC) for the disc. The VTOC is then searched for the TDOS entry EXCLIB and the first record for TDOS, the Program Directory, is read into the IPL work area. The Program Directory supplies the disc address of the TDOS Executive Load Directory and the first entry in the Load Directory is read into memory. The first load of the TDOS Executive is now loaded and the IPL passes the device address of the disc along with necessary Load Directory information. The initialization phase of the Executive is now entered.

### Initializer

◆ The initialization phase of the TDOS Executive performs the following functions:

1. Sets the base registers used by the Executive.
2. Sets the four Interrupt Mask registers and Interrupt Status registers.

**Initializer** *(Cont'd)*

3. Sets the Program Counters for states 3 and 4.
4. Sets all of memory to a storage key of $(F)_{16}$ if memory protect is present.
5. Makes an entry in the Load Library Table for the system device.
6. Requests the date, which is stored at the beginning of the Executive Communication Region.
7. Requests the time, which is converted to a binary value and stored in the Time of Day slot (if the timer is present).
8. Sets the System Time Clock to its maximum value if timer is present.
9. Performs the initiation required for direct control if DC is present.
10. Gives control to the Exec Exit.

## PROGRAM INITIATION

◆ Program initiation in the Tape Disc Operating System is controlled by the operator. When the operator presses the COIN (Console Interrupt) button, the Console Control portion of the Executive initiates a read to the Console Typewriter. The operator responds with a load program request (see Executive Console Request), which causes the following:

1. The format of the operator request is validated.
2. The operator request is analyzed.
3. Tables within the Executive are updated with information concerning the program to be loaded.
4. If run-time parameters (see Device Assignment) are required, they are edited and stored.
5. If the Program Load Library is on tape, it is positioned to this program.
6. Memory is allocated to the program to be loaded.
7. The program is loaded and floated, and its operation list entry is initialized.

Once the above functions have been performed, control is returned to the Executive to continue processing.

## MEMORY ALLOCATION

◆ TDOS permits the user to write programs without concern for the actual memory locations that these programs occupy during execution. Memory is assigned by the Executive based upon the requirements of the program and the available memory at program initiation time.

When the operator or a program requests that a program be loaded, the Executive searches the tape for the program header or the disc/drum for the program directory entry associated with the program to be loaded. When the program header or program directory has been found, the Executive extracts the information concerning the amount of memory required by the program. (The memory requirement contained in the program header or program directory can be overridden by the operator when the program load is initiated.) The Executive determines, in conjunction with the Free Memory Table, if the amount of memory required by the program is available. If memory is not available, the operator is notified and the program is not loaded. If memory is available, the Program Load Header or the Load

## MEMORY ALLOCATION (Cont'd)

Directory is located and the text blocks following the load header are loaded into memory. Any modifier blocks (blocks containing the relative addresses of any address constants contained in the preceding text block) encountered among the text blocks are used to float the address constants contained in the loaded text blocks.

### Free Memory Table

◆ Memory allocation in the TDOS Executive is controlled by a table contained within the Executive called the Free Memory Table. The Free Memory Table defines the areas in memory that are not assigned at any given time. If the memory protect feature is present, memory is always assigned and recorded in multiples of 2,048 bytes. For example, a program requiring 4,000 bytes of memory would be assigned a 4,096-byte area (two units of 2,048 bytes each). If the memory protect feature is not present, memory is assigned in multiples of eight bytes.

The Free Memory Table contains room for up to nine entries and has the following format:

| Entry | Starting Address (4 bytes) | Ending Address (4 bytes) |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | | |

The first eight entries in the table can contain the starting and ending address of available contiguous memory. Entry nine is a sentinel and contains all zeros. Only one entry (entry 1) is required if all of memory is available. More than one entry is required only when available memory is not contiguous. The presence of zeros in entries 1 through 8 indicates that the remaining entries of the table are also zeros.

When the system is initiated at the beginning of the day (before any programs have been loaded) only entry 1 is utilized. It indicates the starting and ending address of the entire memory available for programs.

*Example 1*

◆ Processor size = 65,536 bytes.

Available memory for programs = from 16,385 to 65,536 bytes.

The Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(16,385)_{10}$ | $(65,536)_{10}$ |
| 2 | (zeros) | (zeros) |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

As programs are loaded, the available memory indicated in the table decreases.

*Example 2*

◆ Processor size = 65,536 bytes.

Available memory for programs = from 16,385 to 65,536 bytes.

Assume three programs are loaded as follows:

Program 1 - requiring 4,096 bytes (occupies from 16,385 to 20,480)

Program 2 - requiring 6,144 bytes (occupies from 20,481 to 26,624)

Program 3 - requiring 12,288 bytes (occupies from 26,625 to 38,912)

The Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(38,913)_{10}$ | $(65,536)_{10}$ |
| 2 | (zeros) | (zeros) |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

When a program is completed, the memory allocated to it is made available.

*Example 3*

◆ Assume three programs have been loaded as in Example 2; i.e.,

Program 1 occupies from 16,385 to 20,480
Program 2 occupies from 20,481 to 26,624
Program 3 occupies from 26,625 to 38,912

Assume that program 2 terminates, thereby causing the memory allocated to it to become available.

The Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(26,624)_{10}$ |
| 2 | $(38,912)_{10}$ | $(65,536)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Note:* Two entries are required in the Free Memory Table because there are two noncontiguous areas of memory available.

When a program is loaded, the Executive normally places it in the lowest available memory where it best fits.

*Example 4*

◆ Assume that a program requiring 4,096 bytes is to be loaded and the Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(26,624)_{10}$ |
| 2 | $(49,153)_{10}$ | $(53,248)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Notes:*

1. Entry 1 indicates available memory to 6,144 bytes.
2. Entry 2 indicates available memory to 4,096 bytes.
3. The program (requiring 4,096 bytes) is placed in the memory area indicated by entry 2 because this area, although not the lowest available memory area, is a *better fit* than the area indicated by Entry 1.

When a program is to be loaded, contiguous memory *must* be available.

*Example 5*

◆ Assume that a program requiring 8,192 bytes is to be loaded and the Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | (26,624) |
| 2 | $(49,153)_{10}$ | (53,248) |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Notes:* 1. Entry 1 indicates available memory of 6,144 bytes.
2. Entry 2 indicates available memory of 4,096 bytes.
3. Although the total available memory is 10,240 bytes, the program (requiring 8,192 bytes) is not loaded because contiguous memory of this size is not available.

When the operator initiates a program load, he has the ability to specify that the program be loaded into the highest available memory. In this case, the Executive does *not* try to find the best fit. The program is loaded into the highest available memory in which it fits.

*Example 6*

◆ Assume that a program requiring 4,096 bytes is to be loaded, the operator specifies a high memory load and the Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(24,576)_{10}$ |
| 2 | $(49,153)_{10}$ | $(55,296)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Notes:* 1. Entry 1 indicates available memory of 4,096 bytes.
2. Entry 2 indicates available memory of 6,144 bytes.
3. The program (requiring 4,096 bytes) is placed in the memory area indicated by Entry 2 because:
   a. A high load was specified.
   b. Entry 2 indicates the highest available memory in which the program fits even though it is not the best fit.

*General Notes*

◆ 1. The starting and ending addresses contained in the Free Memory Table are four-byte binary addresses.

2. The lowest available memory is always indicated in the first entry of the table.

## MEMORY DEALLOCATION

◆ When a program is terminated, the memory that is used is deallocated and returned to the system. The Free Memory Table is updated to include the additional available memory as follows:

1. If the deallocated memory is contiguous to a segment of free memory, the two segments are combined and are represented by a single Free Memory Table entry.

2. If the two segments are combined and this combined segment is contiguous to the available memory represented in the next entry, the two entries are combined and the remaining entries are moved up in the table.

3. If the deallocated memory is between that represented by two Free Memory Table entries but is <u>not</u> contiguous to either, a new entry is made for the deallocated memory and inserted between the other two entries. The entries following the new entry are moved down.

*Example 1*

◆ Assume that the Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(26,624)_{10}$ |
| 2 | $(55,297)_{10}$ | $(65,536)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Example 1 (Cont'd)*

A program terminates and memory from 26,625 to 47,104 becomes available. The Free Memory Table now appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | (47,104) |
| 2 | $(55,297)_{10}$ | $(65,536)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Note:* Because the deallocated memory is contiguous to the available memory represented in Entry 1, it is combined with Entry 1.

*Example 2*

◆ Assume that the Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(26,624)_{10}$ |
| 2 | $(55,297)_{10}$ | $(65,536)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Example 2 (Cont'd)*

A program terminates and memory from 26,625 to 55,296 becomes available. The Free Memory Table now appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(65,536)_{10}$ |
| 2 | (zeros) | (zeros) |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Note:* Because the deallocated memory is contiguous to the available memory represented in Entries 1 and 2, it is combined with both entries and one entry results.

*Example 3*

◆ Assume that the Free Memory Table appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(26,624)_{10}$ |
| 2 | $(55,297)_{10}$ | $(65,536)_{10}$ |
| 3 | (zeros) | (zeros) |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Example 3 (Cont'd)*

A program terminated and memory from 49,153 to 53,248 becomes available. The Free Memory Table now appears as follows:

| Entry | Starting Address | Ending Address |
|---|---|---|
| 1 | $(20,481)_{10}$ | $(26,624)_{10}$ |
| 2 | $(49,153)_{10}$ | $(53,248)_{10}$ |
| 3 | $(55,297)_{10}$ | $(65,536)_{10}$ |
| 4 | (zeros) | (zeros) |
| 5 | (zeros) | (zeros) |
| 6 | (zeros) | (zeros) |
| 7 | (zeros) | (zeros) |
| 8 | (zeros) | (zeros) |
| 9 | (zeros) | (zeros) |

*Note:* Because the deallocated memory is not contiguous to any of the available memory represented in the table, a new entry must be created and inserted between Entry 1 and Entry 2.

## DEVICE ASSIGNMENT

◆ The TDOS device assignment routine is designed to take advantage of the multiprogramming capabilities of the system. It allows the user the flexibility of entering run-time parameters at program initiation, waiting until the first I/O command is issued to a particular device, or executing an ASSGN Executive Communication macro, at which time the device may be assigned. This capability allows the user to program with little concern for the actual peripheral device to be used at object time.

Device deallocation is also designed to take advantage of the multiprogramming capabilities of the system. The user may wait for program termination for a device to be deallocated, issue an Executive communication macro in his program to deallocate the device, or deallocate the device via a console request by the operator.

### Use of Run Time Parameters

◆ Parameters may be entered at program initiation to supply the Executive with the information needed to assign actual devices to the user's program. The Executive is notified that these parameters are present through the LOD program console request. At program initiation the Executive reads and validates these parameters and places them at the upper end of the user's program.

### Operator Responses

◆ When the first I/O command is issued to a device and run-time parameters are not present, the operator is requested to type in the information needed to assign a device.

**Processing of Device Assignment**

◆ When the first I/O command is issued, the Executive checks to see if an actual device has been assigned to the symbolic device specified by the command. This is done by looking at the Command Control Block (CCB) for this file. If this file has not been assigned, the device assignment routine within the Executive is entered. The Device List (DL) is searched to see if the same symbolic name is assigned to another CCB in this program. If the same symbolic name is assigned, this indicates there are multiple CCB's for the same file or that this device is being passed to a successor program and device assignment is not required.

Upon determining that a random access device must be assigned, the Volume Displacement parameters are scanned because random access devices do not contain symbolic device names. When a match is made the associated volume is assigned. Program sharing of random access devices is allowed under TDOS. This means that more than one program may be assigned to a random access device.

For nonrandom access devices, a check is made to see if there are run-time parameters for this program. If there are, the run-time parameters are examined to see if there is an assignment for this symbolic device and; if there is, an attempt is made to assign the device. If there are no run-time parameters for this device, or if the run-time parameters are invalid, a message is typed to the operator to assign a device for the symbolic device name; or, in the case of an optional file, to reply if the file is or is not present. After the operator replies with a type in, an attempt is made to assign the device.

If the device is not available, the operator is notified and he may choose to assign another device or suspend the program until that device is available. In either case when a valid assignment is made or the program is suspended, control is returned to the Executive to continue processing.

If the program is running under control of the Monitor, and device assignment has not been made, the assignment via Monitor equate table routine is entered. The Monitor Equate Table is scanned for the symbolic device name. If not found, the assignment via the operator routine is entered. If found, the Device List address of the equated device is picked up out of the Table. The assignment is then made by placing this address into the assignment halfword of the equated device's CCB. Control is then returned to the Executive to continue processing.

**Device Deallocation**

◆ Devices may be deallocated in one of three ways. If the user wishes to take advantage of the multiprogramming capabilities of the Tape-Disc Operating System and deallocate devices as soon as he is finished with them in his program, a Deallocate Device Executive Communication macro (DDEV) is issued, making the device available to another program before the first program terminates. The second method of deallocation is to issue a Deallocate Device (DDV) operator request which deallocates the device at the time the request is issued. The final and probably the most commonly used method of deallocation is at program termination. All devices assigned to a program are deallocated at program termination, unless termination is caused by a TERMS macro.

**Processing of Device Deallocation**

◆ When a device is deallocated, the entry is removed from the CCB and the Device List (DL); thus informing the Executive that the device is now available for use by another program. At this time the Device List (DL) is checked to see if another program is waiting for this device to become available. If there is one waiting, the device is assigned. After assignment, or if there is not another device assignment to be made, control is returned to the Executive to continue processing.

**Run-Time Device Assignment Parameters**

◆ Five parameters define device assignment information to the Executive. These parameters are: ASSGN, FILES, VOL, TPLAB, and VDC.

*ASSGN*

◆ The ASSGN parameter defines device assignment information for a particular file. This parameter indicates the actual device which is to be assigned to the symbolic device name. It also indicates write control information for 7-level tapes.

*FILES*

◆ The FILES parameter defines tape positioning information for a particular file. This parameter indicates the symbolic device name and how many tape marks are to be skipped.

*VOL*

◆ The VOL parameter defines the volume label information. This parameter indicates the symbolic device name and the file name associated with the DTF macro.

*TPLAB*

◆ The TPLAB parameter defines header label information which indicates the label information used to check and write labels.

*Notes:*

A contiguous set of FILES, VOL, and TPLAB parameters must be provided every time a file is OPENed.

*VDC*

◆ The VDC parameter defines random access volumes that are to be used with a particular file.

**Device List**

◆ An integral part of the Executive is the Device List. This list contains a 32-byte entry for each peripheral device that is present at a user installation. Only one entry exists for all communication devices. These devices are defined to a System Generator routine which generates a Device List for that installation only, and includes this list as a permanent part of the Executive. The user, therefore, can maintain a Device List in memory that is only as large as is required for the installation. As additional devices are added to an installation, the System Generator is again used to include these devices in the Device List.

**Command Control Block**

◆ A Command Control Block is an area in the user's program in which data is passed between the user and physical I/O device that is accessed in a program.

**UPDATE ON-LINE CATALOG**

◆ The Update On-Line Catalog Executive console routine posts the random access volumes that are on-line to the system to an executive memory resident catalog. This routine is, in essence, a form of device assignment and gives the File Control Processor (FCP) the ability to address individual volumes on a multivolume device, such as the 70/568 Mass Storage Unit. FCP uses the On-Line Catalog routine at OPEN time.

The Update On-Line Catalog routine reads the volume serial number from each volume that is on-line to the system and enters that serial number into the proper location of the catalog. It then appends that entry in the catalog with the address of the device list entry that corresponds with that serial number, and in the case of the 70/568, develops a bb (BIN) address of 0-7. A description of the processing functions performed by this routine is as follows:

1. Scans the device list searching for random access devices.
2. Determines the status of each device; local or remote.
3. Reads the volume serial number located in the standard volume label.
4. Posts that serial number in the Executive Resident Catalog, the address of which is maintained in the Communication Region.
5. Appends that serial number with a two-byte address of that portion of the Executive Device List that corresponds with the particular entry.
6. Develops a bin address, (0-7) for the 70/568.

The on-line catalog update must be executed for any random access volume brought into the operating system that will have files or extents opened by logical level FCP.

## PROGRAM TERMINATION

◆ Program termination in the Tape-Disc Operating System can be initiated in one of three ways as follows:

1. The program can issue an Executive Communication macro that informs the Executive that the program is to be terminated.
2. The operator can issue a console request that informs the Executive that a program is to be terminated.
3. The Executive can automatically terminate a program when an Executive function determines that this program can no longer continue.

When the Executive determines that a program is to be terminated, it performs the following functions:

1. Deallocates all devices assigned to the program and returns to the system.
2. Deallocates memory assigned to the program and returns to the system.
3. Deletes entries in the Executive tables for the programs to be terminated.
4. Types a message to the operator informing him of the termination.

Once the above functions have been performed, termination is complete and control is returned to the Executive to continue processing.

## CHECKPOINT

◆ The purpose of Checkpoint is to save the program and its pertinent environment on a specified secondary storage medium so that at a future time, the program can be reinitiated and entered at a specified address. Checkpoints can be taken to a tape, disc, or drum.

Program checkpoints are taken whenever the program requiring a checkpoint issues the Checkpoint Executive Communication Command (CKPT Macro). Program checkpoints are not taken automatically by the system based on a specified file block count or end-of-volume condition.

### Processing

◆ Before issuing a program checkpoint, the device to which the checkpoint records are to be written must be assigned to the program requesting the checkpoint. During checkpoint processing, the checkpoint records are written to the specified device. Each checkpoint contains a header record which contains information to distinguish checkpoint records from data records, to identify the program, and to identify the particular checkpoint in the case where more than one checkpoint is taken during program processing. As each checkpoint is taken, the program name and identification for the current set of checkpoint records as well as the installation mnemonic of the device on which the checkpoint has been written are logged on the Console Typewriter.

**Processing** *(Cont'd)*

The console message is:

02K0Δmn,ID

mn = checkpoint device mnemonic.

ID = checkpoint identification.

At the completion of normal checkpoint processing, control is returned to the program at the instruction following the CKPT macro.

**Checkpoint Record Formats**

◆ Program checkpoint consists of three types of records, a Checkpoint Header Record, Checkpoint Text Records, and a Checkpoint Trailer Record.

*Checkpoint Header Record*

◆ ///Δ CHKPTΔ//PPPPPPIIIIIXXXXYYYYZZZZSSSSSS

PPPPPP = Program Name.
IIIII = Checkpoint Identification.
XXXX = Number of bytes in the program to be checkpointed.
YYYY = Program End Address.
ZZZZ = Program Executive Storage Area (ESA) Address.
SSSSSS = Symbolic Device Name of Checkpoint Device.

*Checkpoint Text Records*

◆ B.....

B = Binary representation for each byte contained within the program limits in memory. When checkpoint is taken, the Executive Control System divides and writes memory in blocks of 32,768 bytes. If the program size is less than 32,768 (one block) or if the program exceeds 32,768 but is not evenly divisible by 32,768 (last block only), the block is truncated and contains only the required number of bytes.

*Checkpoint Trailer Records*

◆ ///ΔCHKPTΔ//

**Additional Considerations**

◆ Special considerations regarding the use of checkpoint are as follows:

1. External requests, such as console requests occurring during Checkpoint, are not remembered at restart.
2. Checkpoint records are not included in the block count for a data set.
3. If the checkpoint device is a 7-channel tape, the Pack/Unpack mode must be used.
4. If an error condition occurs during the issuance of the CKPT macro or during checkpoint processing, the condition indicator is set in the checkpoint error byte and control is returned to the program's error routine address.

**Additional Considerations** *(Cont'd)*

5. If the checkpoint device is disc or drum, the record overflow feature must be present.
6. Checkpoint should not be used while in the unrecoverable routine of STXIT.
7. If any input data tapes have Checkpoint records interspersed with data records, CKPTREC = YES must be specified in the DTFSR.

*Note:*

1. Also see "Additional Considerations" for Restart (page 2-18).
2. For a description of the use of the Checkpoint macro, refer to the TOS/TDOS File Control Processor and Executive Communication Macros Reference Manual (No. 70-00-608).

## RESTART

◆ The Restart routine reestablishes the existence of a program and its data in the operating system environment and provides for the program to continue operation at the restart address supplied at checkpoint time.

**Processing**

◆ The Restart routine is initiated by an executive console request which indicates the program to be restarted, the program priority number, and the mnemonic name of the device containing the program. The console request is as follows:

EΔRSTΔProgname , mn, p

Progname = name of program to be restarted
mn = program's load library mnemonic
p = priority of program

After entering the request for restart, control is transferred to the Executive Restart routine. Upon receipt of control, the Restart routine requests parameter information from the operator to be entered from the Console Typewriter. The console message requesting parameters is:

02R0AΔPARAMS

This parameter information indicates the name of the checkpoint (checkpoint identification) and the mnemonic name of the device containing the checkpoint records. The format of the parameter information is:

mn,ID [,wc]
or
mn,ID,,CCCH
or
RHLT

mn = checkpoint device mnemonic
ID = checkpoint identification (five bytes)
wc = optional write control code for seven level tapes
CCCH = Disc (or Drum) cylinder and head address (in decimal) of where checkpoint information is located
RHLT = halts program being restarted

If erroneous parameter information is entered, the operator is notified on the Console Typewriter and he is required to reenter the information.

**Processing** *(Cont'd)*

If the restart device is a disc or drum the checkpoint records must exist at the specified address or termination will result. If the restart device is magnetic tape, it is first rewound and then searched until the specified checkpoint records or the logical end-of-tape (double tape mark) is encountered. An unsuccessful search results in the termination of the restart procedure and the operator is notified. Upon locating the specified Checkpoint Header Record, restart compares the historical memory location for the program to be restarted to determine if required area is currently available. When programs are restarted they must occupy the same area of memory that they occupied at the time the checkpoint was taken (restart programs are not relocatable). If this area is not available, the restart procedure is terminated and the operator is notified.

Restart extracts the symbolic device name from the Checkpoint Header Record and places it in the Device List entry for the checkpoint device. This insures that the user program, after being restarted, is able to refer to the checkpoint device by the symbolic device name used prior to checkpoint. Thus the device containing the checkpoint records is assigned to the program by the Executive Control System during restart processing.

The protection key (if applicable) for the restarted program is then set in the memory modules to be occupied by the program, the checkpoint device list entry, and the Program Table entry. Restart then loads the program's memory image from the checkpoint device into the designated memory area.

If an unrecoverable error occurs during a restart I/O operation on the checkpoint device, the operator is notified. The only option available to the operator, if a retry is unsuccessful, is to request program termination. On termination, the checkpoint device is deallocated by the Executive Restart routine.

The Executive Restart routine performs only those functions indicated above; it does not perform device reassignment, label checking (if applicable), and file positioning. These functions are performed in one of two ways:

1. *Logical FCP Users* - For magnetic tape files processed at the logical FCP level (DTFSR only) an optional entry is provided in the Checkpoint macro to indicate that restart processing is required. When this option is employed, the following is performed at restart:

    a. After reestablishing the program, the Executive System returns control to the program at the address specified in the Checkpoint entry. At this address an Executive call (FLOAD) is generated during language translation which calls, loads, and transfers control to the Logical FCP Restart routine which resides on the system device as an Executive overlay. During FCP Restart routine processing, device assignment, file location, label checking (if applicable), and file positioning are performed. At termination of this routine, control is transferred to either the return address specified in the Checkpoint macro or to the

**Processing** *(Cont'd)*

instruction following the Checkpoint macro if the return address is not specified in the Checkpoint macro.

2. *Non FCP Users* - for files processed by the program using physical I/O macros to define and process the file. Under these conditions, the following is performed at restart:

   a. After reestablishing the program, the Executive System returns control to either the return address specified in the Checkpoint macro or to the instruction following the Checkpoint macro if the return address is not specified in the Checkpoint macro.

   b. In either case, it is the program's responsibility to perform device assignment, label checking, and file positioning.

**Additional Considerations**

1. Run-time parameters may not be entered at restart time. Specifically, ASSGN cards are not accepted at restart time. Therefore, device assignment following restart must be made by a console response.

2. Only programs independent of Monitor can be Restarted.

3. If FCP is used, Registers 14 and 15 (State 1) must be stored by the user prior to CKPT and restored after CKPT.

4. It is assumed that all random access files are available at restart time in exactly the same state of information storage as they were at Checkpoint time.

## DEVICE ERROR RECOVERY

### Introduction

◆ The Tape Disc Operating System features a continuous, nonstop operation. To implement this feature, standard error recovery routines have been provided to handle most error conditions that can occur on all devices supported by the system. The action taken depends on the error condition, the device type, and the options specified by the program.

Separate error recovery segments are provided for the following devices:

70/432, 441, 442, 445, 581 Magnetic Tape Devices (Read)

70/432, 441, 442, 445, 581 Magnetic Tape Devices (Write)

70/221 Paper Tape Reader/Punch (Gapped Tape Only)

70/234 Card Punch

70/236 Card Punch (Models 10, 20, 21)

70/237 Card Reader

70/242, 243 Printers

70/248 Bill Feed Printer (Print Function only)

70/564 Disc Storage Unit

70/565 Drum Memory Unit

70/568 Mass Storage Unit

IBM 2540 and 1402 Card Punches (70/293 Controller)

All error recovery segments reside as overlay segments on the systems resident device along with the Executive. All of the segments can be loaded with one access of the disc or drum.

When a device error occurs, the appropriate error recovery segment is entered. (If the segment is not in memory in the overlay area, it must be loaded.) An attempt is then made to recover from the error if reasonable to do so by programming. The number of times recovery is attempted is a variable figure depending on the device type. For example, error recovery for magnetic tape is attempted 25 times; for the card reader it is attempted once per error. If the error persists after error recovery has been attempted the maximum number of times per device, a message is typed to the operator indicating the device in error and the error condition. The operator must then reply with one of the following codes:

0 - causes error recovery to again attempt to recover the maximum number of times. If the error persists, this procedure is repeated.

1 - causes control to be returned to the Executive with the "unrecoverable error bit" and the "termination bit" set in the executive flag byte in the CCB. The Executive examines the "accept unrecoverable error bit" in the user flag byte in the CCB to determine if the program accepts control on unrecoverable errors. If the

**Introduction** *(Cont'd)*

program indicates the acceptance of unrecoverable errors, control is transferred to the program. If the program does not accept unrecoverable errors, the job is terminated with an option to dump if the program is written at the logical FCP level.

*Program Written Using Physical Level FCP*

◆ When a program is written using physical level FCP, it normally has the option of obtaining control or terminating the job when an unrecoverable error condition occurs because the program determines the bit settings in the user flag byte.

*Program Written Using Logical Level FCP*

◆ When a program is written using logical level FCP, it is normally terminated when an unrecoverable error condition occurs unless the unrecoverable error condition is one of the following:

*Tape or Direct-Access Read Error* - If this error occurs, the program can ignore the error, skip the block containing the error, or obtain control at a specified location.

*Direct-Access Write Error* - If a record is not capable of being written, the program can obtain control at a specified location.

**Error Conditions Causing Transfer to Standard Error Recovery**

◆ The conditions that cause standard error recovery to take place, the messages to the operator, the responses, and the action taken are all described in the TDOS Operator's Guide (70-35-404).

*General Notes*

◆ 1. Data chaining is the only chain operation supported by TDOS error recovery for devices other than Random access. If command chaining or mixed chaining is used, the user program must provide its own error recovery.

2. Command chaining is the only chain operation supported by TDOS error recovery for Random Access devices. If data chaining or mixed chaining is used, the user program must provide its own error recovery.

3. Time dependent operations (stacker selection, continuous feed devices, etc.) are not supported by TDOS.

4. All magnetic tape illegal operation alarms with the exception of the case of an illegal operation caused by the absence of a write ring on a tape reel, cause the program to be terminated.

5. For card punch error recovery, a four-byte address immediately following the CCB must point to the punch back up area.

## MULTI-PROGRAMMING

### Introduction

◆ The Spectra 70 TDOS Executive Control System permits the initiation and execution of a maximum of six programs concurrently. Each program that has been initiated is allocated a portion of memory and shares central processor time with other programs that have been initiated. Each program initiated is assigned a priority number ranging from one to six, with six being the highest. The highest priority program is serviced first and control is not given to the next higher priority program until the higher priority program temporarily relinquishes (interrupt occurs) central processor time. This process is repeated for all programs being executed in a multiprogramming environment.

In TDOS a Multichannel Communication Program (MCP) and Communication User Program (CUP) may be run, as one program, in the program mix along with five additional programs. The MCP, CUP program must receive the highest priority.

The Executive Control System also provides for program chaining or run-to-run sequenced operations. This may be accomplished in one of three ways: 1) under Monitor control, 2) by the use of run-time parameters, or 3) by making use of successor call (TERMS) in installation programs. A series of programs being executed in one of these three ways is considered as one program with one priority in the multiprogramming environment. Since the Monitor is an extension of the Executive, only one Monitor session is permitted to be in operation at any given time.

In order to facilitate a multiprogramming capability, the Executive Control System requires the use of three tables:

1. Executive Storage Area
2. Operation List
3. Current Operation Slot

**Introduction** *(Cont'd)*

The Executive Storage Area is used to store the machine conditions for a given program at the time of interrupt. This area is generated and assigned memory immediately preceding the user's program.

The Operation List is used to record the current status of each user program and of certain Executive functions. Within this list is an indication specifying whether the program is "Free" or "Locked" and the program's return address.

The Current Operation Slot is used to record information about the last program to have control of the central processor.

**Processing**

◆ During program initiation, memory is allocated for the program (including its Executive Storage Area) and the program number and start address are put into their priority slot in the Operation List. The first segment of the program is then loaded into memory and the Operation List entry is "freed".

**Interrupt Decoder**

◆ When an interrupt condition occurs, the program currently executing in Processing State 1 is interrupted and control is given to the Interrupt Control routine of the Executive which stores the Interrupt Mask Register, Interrupt Status Register, P-Counter, General Purpose Registers, and Floating Point Registers. The Interrupt Decoder then decodes the interrupt and transfers control to the appropriate routine for servicing. Depending on the type of interrupt that occurred, the program may be either *freed* or *waited* by the Interrupt Decoder setting the proper indication in the Operation List. A *freed program* is one that can utilize the processor immediately upon gaining control. A *waited program* is one that has relinquished control and is waiting for input/output termination or programming routine availability. For example, a program is locked when it issues an EXCPW or an EXCP followed by a WAIT.

**Executive Exit**

◆ When interrupt servicing is completed, control goes to the Executive Exit routine. It is the function of this routine to search the Operation List for a free program and to return control to this selected program. The search begins with the highest priority slot in the list. When a free program is found, its Interrupt Mask Register, Interrupt Status Register, and P-Counter are restored. If the selected program was not the last to have control, as determined by the Current Operation Slot, the general purpose and floating-point registers are also restored, and the Current Operating Slot is updated. Control then is given to the selected program.

If the Operation List search indicates that all programs are *waited* and therefore cannot use the processor, the Executive system idles until an interrupt occurs and the cycle is repeated.

**Operational Considerations**

◆ When executing programs in the multiprogramming environment, a decision is required regarding whether:

1. a given program is to be completed in the fastest possible processing time or,
2. a balance between programs is to be achieved.

If a program is to be completed in the fastest possible program processing time, regardless of whether the processor is being used as efficiently as it might be, it should be assigned the highest priority. However, if the objective is to achieve a balance between the programs operating in the multiprogramming environment, the points below should be considered:

1. I/O bound programs should be assigned a high-priority classification.
2. Processing bound programs should be assigned a low-priority classification.

## SUCCESSOR PROGRAM CHAINING

### Introduction

◆ The Executive Control System provides for program chaining or run-to-run sequenced operations. This may be accomplished under control of the Monitor as a Monitor session, (described under Monitor, Section 5) by using the Executive macros within the user program or by run-time parameters. Programs running under control of either method are considered as one program. Therefore, when the chain is initiated, a single priority is assigned and memory is reserved for the largest program within the sequenced operation. This is accomplished during program initiation by the operator specifying the requirements in the E LOD console request.

### Executive Communication Macros

*TERMS*

◆ This macro is issued at the completion of a program when a successor program is to be initiated. This macro does not deallocate devices; therefore, all devices that are not used by the called program are deallocated by using the DDEV macro before issuing TERMS. This allows other programs operating in a multiprogramming environment to have access to the unused devices. Also, the memory allocations of the calling program are not affected and the called program is loaded into the same area as an overlay. This means that the initial program in the chain must reserve memory for the largest program called. Any devices that were assigned and were not deallocated by the calling program remain assigned to the called program.

It should be noted that programs written using the TERMS macro must be run as part of a successor chain since this macro always calls a successor program.

An additional consideration should be noted if a run-time Job parameter is present. This is determined by the run-time parameter, Successor Call Presence Flag, being set in the Program Table. In this case the Job parameter program name takes precedence over the successor program name in the TERMS macro. If JOB parameters are not present for every program in the chain, the chain is terminated when the Executive attempts to load the next program and finds that the Job parameters are exhausted.

*TERM*

◆ This macro is issued at the end of a program to inform the system and the operator that the job is finished. When this macro is executed all devices assigned to the program are deallocated and the run-time parameter Successor Call Presence Flag in the Program Table is examined. If this flag is not set, memory is deallocated and control returns to the Executive. If this flag is set, it indicates the presence of a successor job; memory is not deallocated and the successor program is located and loaded. Since all devices were initially deallocated when the TERM macro was issued, it is necessary to reassign the devices.

*TERMD*

◆ With regard to successor program chaining, this macro causes the same action to be taken as described under the TERM macro.

**Operator Console Requests**

*E HLT*

◆ This request is handled in the same manner as the TERM macro.

*E DUM*

◆ This request is handled in the same manner as the TERMD macro.

*E KIL*

◆ This command allows the operator to cancel all unstated job orders in the program's run-time parameter area. The job that is currently being executed is completed, but no additional jobs are initiated.

**Special Consideration**

◆ When an Executive function determines that a program cannot be allowed to continue, it is halted and any successor chaining is treated as if a TERM or E HLT had occurred.

## DATA EXCHANGE CONTROL SUPPORT

◆ The Data Exchange Control (DXC) enables two Spectra 70 processors to communicate with each other via the RCA standard input/output interface. Data transmission may be in either direction, but in one direction at a time. Either processor can originate a transmission at any time provided the proper programs have been loaded in each processor.

An external DXC interrupt (i.e., a Write command issued by the sending processor) must be handled by a user-supplied routine for each DXC connected to a receiving processor (i.e., the processor that issues a Read command to receive the transmitted information).

The user supplies the address of his DXC routine to the Executive by issuing a STDXC macro. This macro must be issued before the receiving processor can receive information. General Registers 10 and 11, and the P1 program counter are stored in the CCB extension at the time the Executive transfers to the DXC routine, thus leaving the user free to utilize the two registers. When the user gets control in this routine he must issue an EXCP or EXCPW to read or receive information from the DXC. After all DXC processing takes place, the user exits from this DXC routine by issuing a DXCXT macro. This allows the Executive to give control back to the user's main program path. The Executive restores the P1 program counter, and General Registers 10 and 11 before returning control to the user's main program path. This macro must be issued before the Executive will allow this routine to process another DXC interrupt.

## DATA EXCHANGE CONTROL SUPPORT (Cont'd)

An entry for each DXC to be handled must be contained in the TDOS Executive Device List. The DXC is assigned in much the same manner as any other nonrandom access device, i.e., each DXC can be assigned to only one user program at a time. The assignment is accomplished by the operator or via run-time parameters, at the first call for this device, or by the ASSGN macro. The main difference between the DXC and other devices is that external interrupts being caused by a Write from a connected processor may occur. Although the Executive processes all DXC interrupts, it is the responsibility of the user program to control the exchange of information between the two processors via DXC. To properly utilize the DXC, compatible programs have to run, and compatible actions have to be performed.

When the Executive recognizes a DXC external interrupt the following steps are executed:

1. The Device List entry for this DXC is checked to find the address of the user CCB (previously set by a STDXC macro).
2. The address specified in the CCB is placed into the Operation List entry for this program. The return address and General Registers 10 and 11 are stored into the CCB.
3. When the user-supplied routine gets control, he must initiate an EXCP (w) to read from the DXC. The same CCB must be used for all calls to a given DXC. At the end of the routine a DXCXT macro must be issued.

To transmit data between two processors via the DXC, a program in processor A issues a Write to the DXC. This sets the External Interrupt bit ($2^7$ of the standard device byte) in processor B. The compatible program in processor B must previously have had the DXC assigned to itself, must have informed the Executive that he will process the external interrupt, and must have supplied to the Executive the address of a routine which will handle the interrupt. When the program in processor B gets control it must issue a Read (EXCP or EXCPW) to the DXC. This completes the transfer of information between the two processors. The user will be supplied with the standard device and sense bytes associated with the I/O. All abnormal terminations must be handled by the user.

If an external interrupt occurs in processor B and no program is assigned to the DXC, an error message is issued to the operator. This type of error lies strickly in the operation of the two processors and must be settled by the operator. He may reply to the message indicating that he will load a compatible program in processor B, thus allowing the Manual Request Expected bit in the Device List to remain set. If no compatible program can be loaded, the operator replies to the message indicating this fact. This will cause the Executive to issue a Write to processor A (the sending computer) and $2^1$ of the sense byte of each processor to be set, indicating noncompatible programs exist in the two processors.

If an external interrupt occurs in processor B, and a program is assigned to the DXC, but is currently processing a previous interrupt (has not yet issued a DXCXT macro) the interrupt is processed by the Executive and the device list entry reflects the fact that this interrupt is to be processed.

## DIRECT CONTROL SUPPORT

◆ The Direct Control feature enables one 70/35-45-55 processor to signal directly up to five other processors over an interface independent of the input/output channels. The inclusion of the Direct Control coding and associated tables (Direct Control Line Table, Direct Control Program Table) into the Executive is specified at System Generation time. Only one byte of information can be transferred between processor memories when using this Direct Control feature. A given Direct Control trunk can be assigned to only one program, however, a program may control more than one Direct Control trunk.

The TDOS Executive responds to all external signal interrupts (a remote processor transmitting to the user's processor through the DC-trunk) by passing the transmitted byte to the user's program disregarding its content (except certain control bytes which have special significance to the Executive itself). Control bytes are used by the Executive to indicate that a Direct Control trunk is "out of service" (X'00') or "null" (X'01') meaning no transmission is taking place over an "in service" trunk. The TDOS Executive also services all user program requests associated with the Direct Control feature. The user macros associated with the Direct Control feature are:

DCCB - Direct Command Control Block

SETDC - Set Address of User's Direct Control Routine

WRTDC - Execute Write Direct

DCWT - Direct Control Wait

DCXT - Exit from User's Direct Control Routine

RELDC - Release User of Direct Control

**Note:** See TOS/TDOS FCP - Executive Communication Macros Reference Manual 70-00-608 for a detailed description of these macros.

Upon loading of the TDOS Executive into a processor which is linked to other processors through the Direct Control feature, the Executive samples all static inlines. Those containing a zero byte will be marked out of service. A null signal (X'01') is then written to all processors indicating that this processor is in service.

When an external signal causes an interrupt, the processors causing the interrupt is marked in service, if it has not been previously so marked.

**Executive Communication Macros**

**DCCB**

◆ The user program which is to use the Direct Control feature must define each DC-trunk to be referenced by including separate DCCB macros for each DC-trunk. Each DCCB also specifies the address of the user's Direct Control routine for the handling of external interrupts on this DC-trunk.

**SETDC**

◆ The user program must then issue a SETDC macro to notify the Executive that it is going to use a specific Direct Control line. Separate SETDC macros should be issued for each Direct Control line to be assigned to the user program. The Executive then does the following:

1. Checks to make sure that the Direct Control trunk specified by the DCCB for the SETDC macro has not been already assigned. If the trunk is already in use, the SETDC macro is ignored and the operator is notified.

2. Stores the DCCB address for this trunk in the Direct Control Line Table.

3. Stores the program number in the Direct Control Line Table for reference.

4. Marks the line in use for this program.

**WRTDC**

◆ Once a Direct Control line has been assigned by the SETDC macro, the line is available to transmit data. To issue a Write to a Direct Control line the user must issue a WRTDC macro. A single issuance of a WRTDC macro may transmit data over more than one Direct Control line at a time. The DC-trunk byte is used to designate which trunks are to receive the transmitted DC-byte. The following takes place when a WRTDC macro is issued:

1. The DC-trunk byte, indicated by the WRTDC macro, is tested to ensure that the lines specified are both in-service and assigned to this program. The bit for any line failing this test is set to 0. If this results in the entire byte being cleared, (i.e., all lines failed the test) the Write request is ignored.

2. If an internal Waiting For Acknowledge (WFA) byte indicates that the Executive is waiting for acknowledgement of a previous write, the DC-trunk and DC-byte are saved in the DC Program Table and control passes to Exec Exit.

3. The static in-lines corresponding to the out-lines specified by the DC-trunk byte are sampled to ensure that all of them are either null (X'01') or out of service (X'00'). If any one of them is otherwise, (i.e., a transmission is taking place), the two bytes are saved, and control passes to Exec Exit. If there is already a queued Write Direct, the program counter and the Operation List scan are reset as for a DCWT macro.

**WRTDC** *(Cont'd)*

4. The DC-byte is placed on the static out-lines to indicate to the other processor(s) that this processor is readying a Write Direct. When the byte is placed on the line, it can be sensed by the Executive in another processor, but is not transmitted.

5. The appropriate static in-lines are sampled once again to make sure that none of the receiving external processors have also been readying a Write Direct in the interim. If a positive byte indicates that another processor is readying a Write, the null byte is placed on the lines and the Write operation is queued as indicated in 3 above.

6. If the second check, as shown in 5 above, shows all lines open and clear, the actual Write Direct instruction is executed as requested and the DC-trunk byte in the DC Program Table is zeroed.

7. The DC-trunk mask is placed in the WFA byte and control passes to Exec Exit.

**DCWT**

◆ After a Direct Control Write macro (WRTDC) has been issued and before another one can be issued or a RELDC macro can be issued, a Direct Control Wait macro (DCWT) should be issued. Once the Wait macro is issued, the following takes place:

1. The byte holding the queued DC-trunk mask in the DC Program Table is tested. If it is all zeros, meaning that the Write Direct instruction has been executed, control passes to Exec Exit.

2. If it is not zero, an attempt is made to execute the Write Direct as outlined above.

3. If the attempt is not successful, the P-counter of the user is modified to return to this DCWT macro and the Operation List scan is reset to the next lower priority program.

4. If the attempt is successful, control passes normally to Exec Exit.

**DCXT**

◆ When the user has finished processing in his Direct Control routine he must issue a Direct Control Exit macro (DCXT) to return to his in-line program. When a DCXT macro is executed the Executive checks for a backup area byte (see Handling of DC External Signal Interrupt) and either restores the user's P1 program counter to the main program flow or resets the P1 program counter to the Direct Control routine entry, depending upon the result. General Registers 10 and 11 are always saved on entrance to the user's routine and are restored on exit from the routine.

**RELDC**

◆ When the user relinquishes control of any (or all) DC-trunks currently assigned for its use, the RELDC macro is used. When an RELDC macro is recognized, the TDOS Executive performs the following:

1. Checks the DC-trunk byte against the lines-in-use for this program to ensure only those lines are released.

2. Resets the corresponding bits of the lines-in-use byte and queued DC trunk mask in the Direct Control Program Table.

**RELDC** *(Cont'd)*

3. Resets the corresponding bits of the WFA and lines allocated bytes, writing the null byte (X'01') on the out-lines when the resulting WFA byte is zero.

4. Clears the corresponding entries in the Direct Control Table and goes to the Exec Exit.

Since no "quieting" function is performed on Direct Control trunks, it is necessary that all WRTDC macros are followed by a DCWT macro to insure that all queued requests are completed prior to the release of a DC-trunk.

**Handling of DC External Signal Interrupt**

◆ When an interrupt takes place due to an external signal, the following steps are executed.

1. If the bit corresponding to this line in the Waiting for Acknowledge (WFA) byte is set, an acknowledgement is assumed. The bit in the WFA byte corresponding to this line is cleared, and if this results in clearing the byte, an attempt is made to execute any queued Write Direct operations. If there are none that can be executed, the null byte (X'01') is written on the static out-lines.

2. If the WFA byte is already 0 when the interrupt is serviced, the DCCB for this line is located. If none has been specified, (i.e., the line is not assigned), the operator is notified, and the byte is ignored. If a user Direct Control routine has been furnished, a check is made to ensure that it is free (i.e., the last 12 bytes of the DCCB are all zero). If it is not, the input byte is placed in the backup area (byte 3 of DCCB expansion) pending execution of the DCXT macro. Otherwise, the byte is placed into byte 1 of the DCCB expansion for this trunk, and the P1 counter saved in the user's operations list entry is set to the address of the user Direct Control routine, if one has been furnished. In addition, if the user program has been Waited with Nothing to Do, it is unwaited.

3. Whether or not a DCCB has been specified for this line, an acknowledgement (null byte with an interrupt) is sent to the originating processor.

When a power failure interrupt occurs in a processor, the power failure byte (X'00') is written by the Executive to any external processors immediately.

When a power failure byte is received:

1. The line is marked out of service.
2. The WFA bit for this line is cleared.
3. The byte is transmitted to the appropriate user program.
4. When an interrupt occurs for an external line marked Out of Service the appropriate user, if there is one, is notified of the return to service.

## ELAPSED TIME CLOCK

◆ When an elapsed time clock has been installed and is specified at system generation time, the TDOS Executive uses this feature to provide job accounting functions, which provide time of day, actual program time, setting of program timer interrupt, servicing of this interrupt, and informative executive messages.

### Job Accounting

◆ The TDOS Executive provides job accounting functions for all user program's including CUP and MCP. The desired information to perform job accounting is provided to the TDOS Executive and/or the user program via Executive Communication macros (GETOD, SETIC, GEPRT, STXIT-IT) and to the operator via console typeouts.

### Communications

◆ The MCP provides the TDOS Executive with a timer setting via the CMINT Initialization macro. At the end of this predetermined time interval, the resulting interrupt is processed by the CIA portion of the TDOS Executive.

### User Programs

◆ User programs may use the Elapsed Time Clock in the following three ways, each utilizing Executive Communication macros to communicate with the TDOS Executive.

1. Time of day may be obtained by utilizing the GETOD macro.
2. Actual program running time may be obtained by utilizing the GEPRT macro.
3. The elapsed program time interrupt may be specified by utilizing the SETIC macro and serviced by utilizing the STXIT-IT macro.

**Time Table**

◆ The time table generated at system generation is used to record information regarding the running time of each program.

1. There are eight entries in the time table:
   a. Idle.
   b. User Program 1.
   c. User Program 2.
   d. User Program 3.
   e. User Program 4.
   f. User Program 5.
   g. User Program 6 or CUP.
   h. MCP.
2. The 16-byte table entry for each program follows:

   0:0 Interval Setting.
   4:0 Interval Remainder.
   8:0 Program Time Accumulator.
   12:0 Unused.

**Executive Messages**

◆ There are two TDOS Executive messages typed, in the form indicated below, if the existance of an Elapsed Time Clock is specified at systems generation.

1. V PPPPPP 02Ln hhmmss
2. V PPPPPP 02NH hhmmss hhmmss

   P = program name.
   n = program number.
   hh = hours.
   mm = minutes.
   ss = seconds.

The first typeout denotes the current time of day upon program initiation. The second typeout denotes the current time of day and the actual elapsed program running time, immediately after program termination.

## DISC ORGANIZATION

◆ In order to utilize TDOS, certain basic steps must be taken to initialize the systems resident device. To do this, the following three records must be present on the disc in the addresses specified:

| | |
|---|---|
| 1. Bootstrap | Cylinder 0, Track 0, Record 1 |
| 2. IPL | Cylinder 0, Track 0, Record 2 |
| 3. Standard Volume Label | Cylinder 0, Track 0, Record 3 |

Contained in the Standard Volume Label is a pointer to the Volume Table of Contents (VTOC). The VTOC record is a standard entry in all random access devices, and as such is independent of all files on the device. The VTOC consists of records with a 44-byte key which identifies a file on this disc, and a 96-byte data portion. Each such record defines the attributes of a file which is resident on the disc. There would be such an entry for the TDOS Executive, and another for user programs which run under the TDOS Executive. In addition, there could be VTOC entries on other discs which contain user programs that run under the TDOS Executive. Other VTOC records on the systems disc would refer to files used by other programs for data storage. Each entry contains a pointer to the disc areas allocated to the respective files on the disc. The VTOC for a systems disc or drum must not begin on cylinder 0, track 0.

Prior to the placing of the TDOS Executive and user program on a disc the Random Access Initializer and Allocator programs must be run (in that order) to prepare the disc for subsequent use. These programs define the size of the Bootstrap and IPL records, record the Standard Volume Label (hence location of VTOC), record the home address and R0 of each track, perform surface analysis of the disc, and generate entries in the VTOC.

The VTOC entries for the TDOS Executive and the user programs (assigned by the user via the Allocator) normally specify that only one contiguous area be reserved for the TDOS Executive, and one contiguous area for the user programs.

Within each area there are three sections:

1. Program Directory (PD).
2. Load Directories (LD's).
3. Loads.

These elements are generated by the Program Transcriber; the transcriber being the program which places programs and their associated directories into the prescribed disc areas.

## DISC ORGANIZATION (Cont'd)

The PD is a list of all the programs contained on the disc which run under Executive control of TDOS. Each PD entry has the following format:

| | |
|---|---|
| 0:0 | Program Name |
| 6:0 | Initial Load Entry Point |
| 9:0 | Core requirements (max., min.) |
| 15.0 | Disc Address of Load Directory |
| 20.0 | Date |
| 26.0 | Version |
| 30:0 | 30 bytes for each entry |

Each program consists of an initial load and any number of overlays. An LD is a list of all the loads associated with a particular PD entry. Each LD entry has the following format:

| | |
|---|---|
| 0:0 | Load Name |
| 6:0 | Disc Origin of Load (CCHHR) |
| 11:0 | Program Relative Load Address of 1st Text Byte |
| 14:0 | 14 bytes for each entry |

The Loads have the following format:

Text blocks - (Normally 1 per load)

Modifier blocks - (Fixed length, except last; variable number)

## 70/350 MASTER SWITCH CONTROLLER

◆ The 70/350 Master Switch Controller, when used to house 70/310 Manual Remote Standard Interface Switch(s), allows one or more processors to access one or more devices.

The Master Switch Controller controls by program up to eight manual remote standard interface switches. Up to four Spectra 70 Processors can use the switch controller. Only one processor can execute the controller at any time. Others are held off by means of the control busy bit of the controller's standard device byte.

Upon acceptance of a new instruction, which specifies a switch and a switch position, the controller tests the desired switch's Active Indicator found in the second sense byte. If the indicator is on, the command is terminated and a secondary status indicator is set.

If the switch Active Indicator is not on, the controller turns the indicator on and then transmits the proper signal to the addressed switch. At the end of a 10 ms period, the controller terminates and indicates a device end condition to the initiating processor by means of the controller's standard device byte. The controller is then free to accept another command. The controller remains busy during command chaining.

## 70/350 MASTER SWITCH CONTROLLER (Cont'd)

The TDOS Executive does not support error recovery or multi-processor use of the switch controller; the user must exercise extreme care when accessing the controller.

The following rules apply to the user of the controller:

1. The device must be defined at SYSGEN time.
2. The device can be assigned to only one program.
3. The device is addressed by issuing a Write Control CCW at the physical level.
4. Write Control Operations which specify non-existent switches or switch positions will terminate normally with no indication of the error.

In order for a user to activate a switch successfully, the user must specify the switch position (one of four positions, determined by the switch type) and the switch number (up to eight switches may be connected).

Therefore, the user must claim the controller by issuing a Write Control command specifying a switch position, switch number, and when necessary, reset the switch active indicator.

Once the I/O has been completed, the user must issue a CCW to reset the controller. Failure to do this prevents any other program from making a Claim on the Controller.

To claim the controller, two types of Write Control commands may be issued:

1. Claim - claims the controller if it is not busy; or if the controller is busy, allows the Executive to hold off the user until the controller becomes free.
2. Unconditional Claim - claims the controller, disregarding any I/O's that may be going on.

### Example of Claim

◆ A user from processor 2 wants to use the printer which is connected to the controller via switch 2:

| CC | Data Address | User Flag | Byte Count | Comments |
|---|---|---|---|---|
| 07, | TAG1, | X'00', | 1 | Claim controller for processor 2, switch 2. |
| PROCESS EXCP(s) or EXCPW(s) | | | | |
| 07, | TAG2, | X'00', | 1 | Reset controller. |
| *TAG1 DC | X'11' | | | |
| *TAG2 DC | X'51' | | | |

*See page 2-36 for an explanation of the bits designating positions and switch numbers.

**Example of Unconditional Claim**

◆ A User from processor 4 wants to use the punch which is connected to the controller via switch 7. At the time this unconditional claim is made, processor 3 is using switch 8.

| CC | Data Address | User Flag | Byte Count | Comments |
|---|---|---|---|---|
| 07, | TAG3, | X'40', | 1 | Reset controller from processor 3, switch 8. |
| 07, | TAG4, | X'00', | 1 | Claim controller for processor 4, switch 9. |
| PROCESS EXCP(s) or EXCPW(s) | | | | |
| 07, | TAG5, | X'00', | 1 | Reset controller. |
| *TAG3 DC | X'67' | | | |
| *TAG4 DC | X'36' | | | |
| *TAG5 DC | X'76' | | | |

*See below for an explanation of the bits designating positions and switch numbers.

**Definition of Write Control Command**

| Bit Setting | | | | | | | | Definition |
|---|---|---|---|---|---|---|---|---|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | |
| X | 0 | X | X | X | X | X | X | Connect switch to designated position only if specified switch active indicator is off, and turns the switch active indicator on. |
| X | 1 | X | X | X | X | X | X | Reset switch active indicator associated with specified switch, previous connection is broken and switch goes to no-connection. |
| X | X | 0 | 0 | X | X | X | X | Designates position #1. |
| X | X | 0 | 1 | X | X | X | X | Designates position #2. |
| X | X | 1 | 0 | X | X | X | X | Designates position #3. |
| X | X | 1 | 1 | X | X | X | X | Designates position #4. |
| X | X | X | X | X | 0 | 0 | 0 | Specifies Switch #1. |
| X | X | X | X | X | 0 | 0 | 1 | Specifies Switch #2. |
| X | X | X | X | X | 0 | 1 | 0 | Specifies Switch #3. |
| X | X | X | X | X | 0 | 1 | 1 | Specifies Switch #4. |
| X | X | X | X | X | 1 | 0 | 0 | Specifies Switch #5. |
| X | X | X | X | X | 1 | 0 | 1 | Specifies Switch #6. |
| X | X | X | X | X | 1 | 1 | 0 | Specifies Switch #7. |
| X | X | X | X | X | 1 | 1 | 1 | Specifies Switch #8. |

The results of a switch operation are conveyed to the program via a secondary indicator.

**Example of Unconditional Claim** *(Cont'd)*

It is the responsibility of the user to issue the proper Write Control commands to the switch controller to ensure that the interface switch is set to the proper configuration before issuing an I/O command to the affected trunk. The unconditional claim of a switch must be exercised with extreme care; if processor A executes an unconditional claim of a switch belonging to processor B while an I/O operation is in progress in processor B through the trunk connected to the claimed switch, the entire channel associated with the affected trunk in processor B will be locked out indefinitely since a terminate will never be received on that trunk in processor B.

Before issuing an unconditional claim to the controller, the user program must determine which processor is using the controller. This information enables the user program to issue a reset CCW for that processor and that particular switch setting. This can be accomplished as follows:

1. The user knows by definition which processor(s) are connected to a switch.

2. The user issues a claim to a switch and if it is busy, receives an indication of the secondary indicator set. The secondary indicator shows which switch is active.

3. The user issues a CCW to reset the particular processor connected to a given active switch, and continues as outlined in the above example for an Unconditional Claim.

## 70/310 MANUAL REMOTE STANDARD INTERFACE SWITCH

◆ The following types of 70/310 switches are acceptable under TDOS:

| | |
|---|---|
| 1 x 2 | two-bi |
| 1 x 3 | three-bi |
| 1 x 4 | four-bi |

Each switch is bi-directional; that is, they may be considered as:

| | |
|---|---|
| 2 x 1 | two-bi |
| 3 x 1 | three-bi |
| 4 x 1 | four-bi |

A two-bi switch may specify one processor connected to two peripheral devices. (See figure 2-1.)

**70/310 MANUAL REMOTE STANDARD INTERFACE SWITCH (Cont'd)**

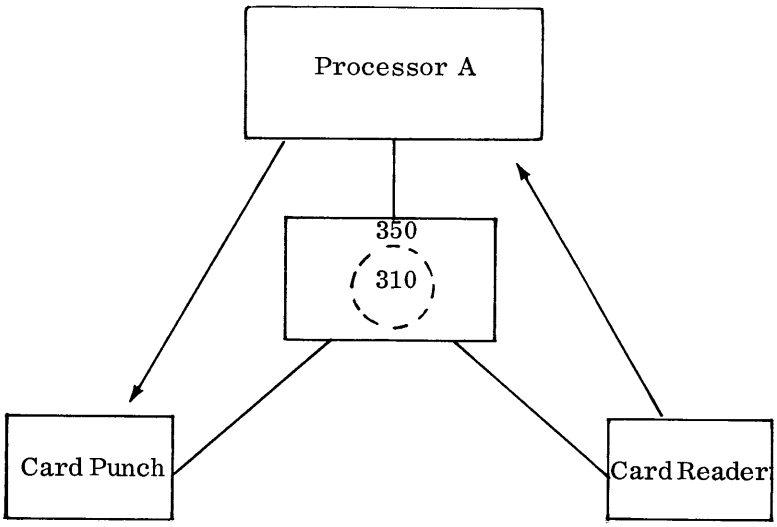



**Figure 2-1.**

Processor A may receive information from the card reader, process it and send it to the card punch.

The reciprocal may be true under a two-bi switch; two processors may be connected to one peripheral device. (See figure 2-2.)

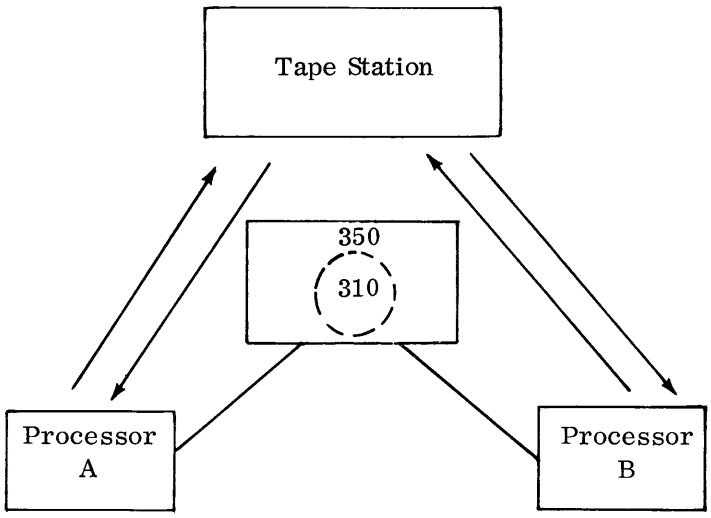



**Figure 2-2.**

Processor A and Processor B may send and receive information to and from the tape station.

The same rule applies to both the three and the four-bi switches.

Figure 2-3 is an example only and illustrates a 2 x 2 switching complex.

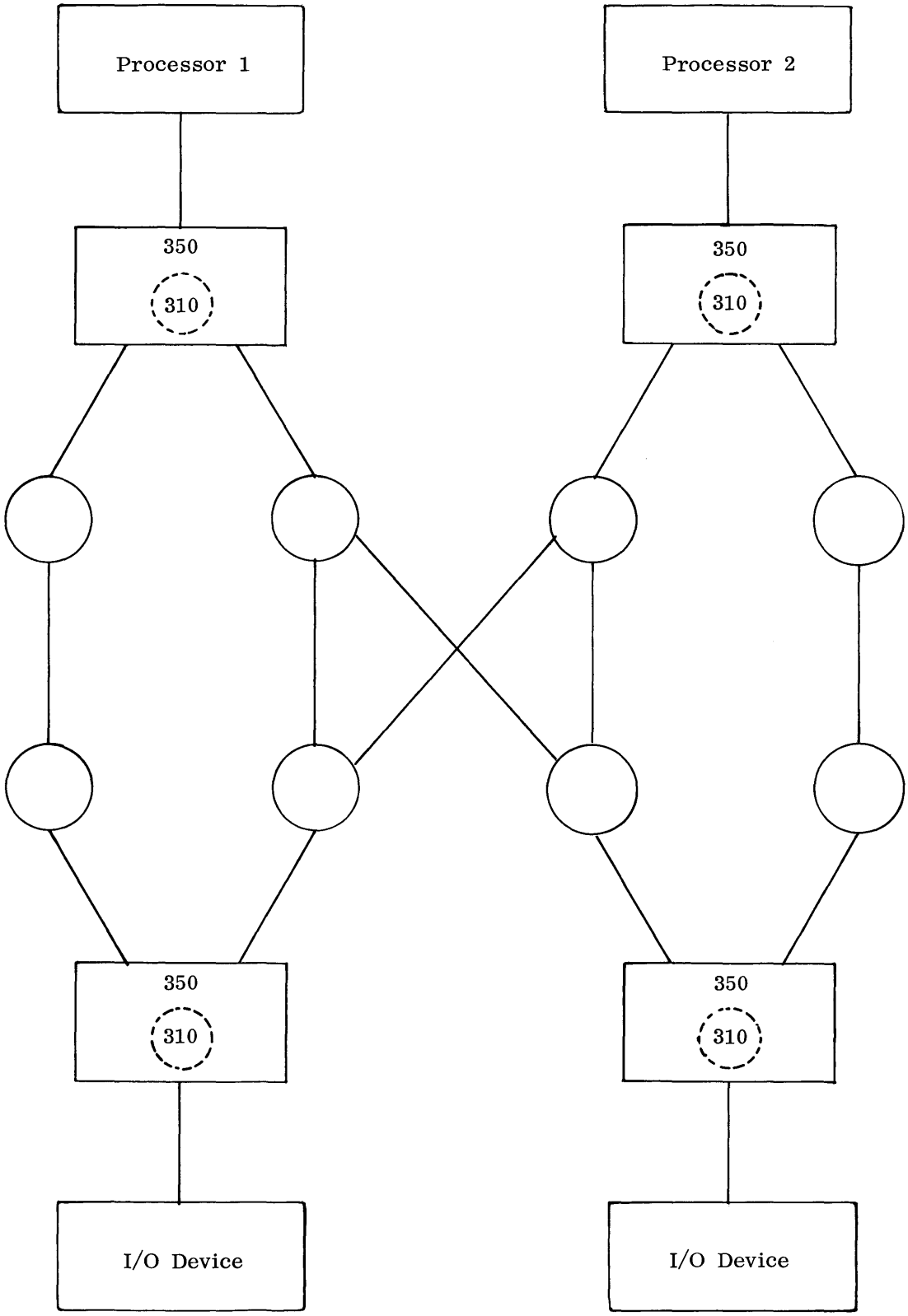



**Figure 2-3. Example of 2 x 2 Switching Complex**

**Standard Device Byte**

| Bit | Meaning |
|---|---|
| $2^7$ | Manual Request - always zero. |
| $2^6$ | Termination Interrupt Pending in Device. |
| $2^5$ | Device Busy. |
| $2^4$ | Control Busy. |
| $2^3$ | Device End. |
| $2^2$ | Secondary Indicator. |
| $2^1$ | Inoperable. |
| $2^0$ | Status Modifier. |

**First Sense Byte**

| Bit | Meaning |
|---|---|
| $2^0$ | Illegal Operation. |
| $2^1$ | Designated Switch is Active. |

**Second Sense Byte**

| Bit | Meaning |
|---|---|
| $2^0$ | Switch #1 Active. |
| $2^1$ | Switch #2 Active. |
| $2^2$ | Switch #3 Active. |
| $2^3$ | Switch #4 Active. |
| $2^4$ | Switch #5 Active. |
| $2^5$ | Switch #6 Active. |
| $2^6$ | Switch #7 Active. |
| $2^7$ | Switch #8 Active. |

## 5513 MULTICHANNEL SWITCH

◆ The 5513 Multichannel Switch permits a Model 70/551 Random Access Controller to be utilized by two selector channels with switching accomplished under program control. The multichannel switch is physically located within the 70/551 Controller cabinet.

The Multichannel Switch has three positions, channel A, channel B, and neutral. When the multichannel switch is in the neutral position, the 70/551 Controller is selected by the first channel to request it. If both channels request the Controller simultaneously, the switch decides which channel will be connected.

Each device attached to a 70/551 Controller can be reserved for use by one of the two channels. A device attached to the controller that has not been reserved by either channel will operate with either channel.

When off-line seeks are performed, the seek complete interrupt is presented only to the channel that the device is reserved for. Therefore off-line seeks can only be given on devices that are reserved; otherwise the seek complete interrupt will be given to the processor on channel A.

Once the controller has been selected by a channel, it remains selected to that channel until all chained operations are completed. Upon completion of the final operation, the multichannel switch returns to the neutral position unless a Secondary Indicator is set. In this case, the multichannel switch will return to the neutral position after a sense command has been executed to the specific device.

If processor A requests control of the 70/551 Controller via the multichannel switch (See Figure 2-4), and the controller is busy with processor B, processor A is informed of this fact by the device busy ($2^4$) and status modifier ($2^0$) bits set in the standard device byte. When the multichannel switch switches from processor B to processor A, an interrupt is sent to processor A with status modifier ($2^0$), manual request ($2^7$) and device end ($2^3$) bits set in the standard device byte. The switch will remain connected to processor A indefinitely until the processor responds with any command or chain of commands. A sense instruction could be used to release the switch should service not be desired.

In the device address sent when an interrupt is generated from the multichannel switch changing processors, the upper two bits are the base address of the controller. The least six bits of the device address should be ignored.

Two additional commands are recognized by the random access controller when the multichannel switch option is installed. These are:

Device Reserve

Device Release

A detailed description of these commands follows.

General reset of the selected channel or controller will place the switch in the neutral position and will release all device reservations.

**5513 MULTICHANNEL SWITCH (Cont'd)**

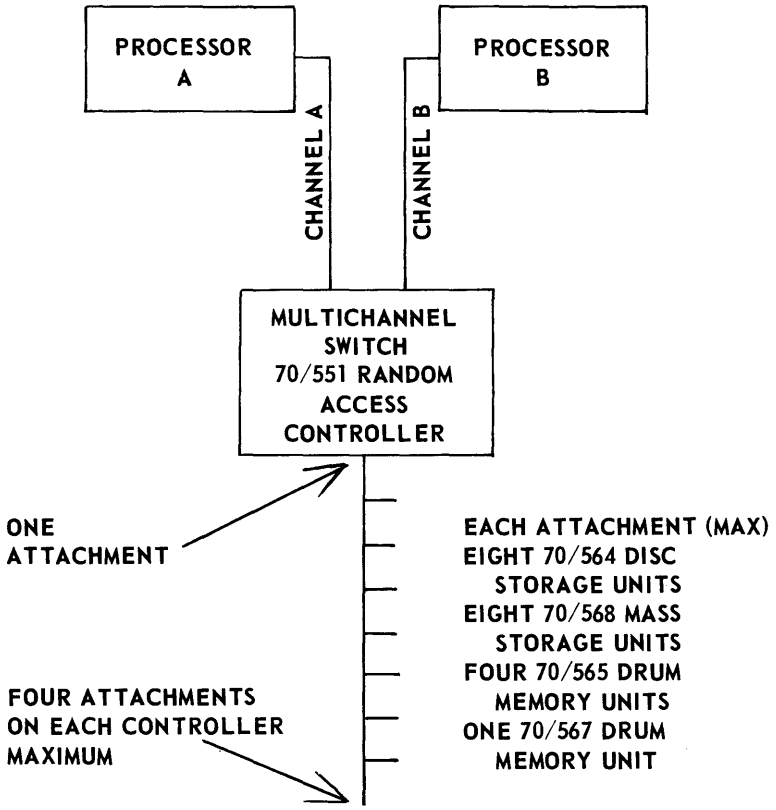



**Figure 2-4. Multichannel Switch**

**Device Reserve**

◆ This command causes the addressed device to become reserved to the processor issuing the command via the multichannel switch. Once a device becomes reserved to a processor, it remains reserved until a Device Release command is issued to that device.

The Device Reserve command is rejected with secondary indicator ($2^2$) set in the standard device byte and command code reject ($2^0$) set in sense byte 1 by a 70/551 Controller that does not have a multichannel switch option installed. If an attempt is made to reserve a device that is already reserved or address a device reserved by the other processor, the command will be terminated with the secondary indicator ($2^2$) of the standard device byte and the device reserve ($2^3$) of sense byte 3 set. A Device Reserve command is executed regardless of any abnormal device status conditions (i.e., inoperable, etc.).

When offline seeks are performed, the seek complete interrupt is sent to the processor that has reserved the device. If a device has not been reserved by either processor, the processor on channel A of the multichannel switch will receive the seek complete interrupt. Offline seeks should be performed only on devices that have been reserved.

**Device Reserve** *(Cont'd)*

A device that has not been reserved can operate with either processor via the multichannel switch. There is a chance that if the device is not reserved the wrong processor may be interrupted.

*Note:*

When a seek complete interrupt is generated either by turning on a 70/568 or 70/565 from local to remote, the interrupt will be sent to the processor that has reserved the device just as the offline seek operation does.

**Device Release**

◆ This command causes the reservation of the addressed device to be terminated.

If the 70/551 Controller does not have the multichannel switch feature installed and a Device Release command is sent to the controller, the command is rejected.

If an attempt is made to release a device that has been reserved by the other processor, the Device Release command will be terminated with secondary indicator ($2^2$) of the standard device byte and device reserve ($2^3$) set in sense byte 3.

A Device Release command is executed regardless of any abnormal device status conditions (i.e., inoperable, etc.).

# 3. EXECUTIVE CONTROL SYSTEM (ECS)

## INTRODUCTION

◆ The Executive Control System (ECS) is an integrated disc or drum resident programming system that controls the processing environment of the Spectra 70/35, 45, or 55 Processors. In maintaining control over the processing environment, the Executive Control System is responsible for memory allocation, interrupt analysis, program loading, and program segment loading. In addition, ECS is responsible for the performance of the logical functions required to initiate and to terminate programs that are executed as either an independent program or as part of a successor program chain. ECS controls all input/output operations performed on the system and their associated error recovery processing. Multiprogramming, which permits the execution of up to six programs concurrently, is also monitored by the Executive Control System.

The Executive Control System takes full advantage of the equipment capabilities of the Spectra processors it services. As such, the Executive Control System is totally a combined programming/hardware system which is primarily driven by recognizing and servicing the various types of equipment interrupts that can occur within the processor. In so doing, the Executive remains resident in memory throughout an entire day's operation and occupies the equipment processing states $P_2$, $P_3$, and $P_4$. The installation's programs and other RCA-supplied routines require core memory residency only for the duration of the program and they occupy processing state $P_1$.

Whenever an equipment interrupt occurs, control is automatically passed by the equipment from processing state $P_1$ to processing state $P_3$ or $P_4$ depending on the type of interrupt. The Executive Control System receives control and, in general, determines the type of interrupt and selects the proper component of the Executive Control System to service the interrupt. When all interrupts have been serviced control is returned to processing state $P_1$.

## ECS COMPONENTS

◆ The Executive Control System consists of six interrelated control components. These major components and their functions are as follows:

1. *Interrupt Control* - decodes the interrupt and based upon the type of interrupt, either services the interrupt or passes control to another Executive component to service the interrupt. All entrances to the Executive System are directed to this component. After interrupt servicing is completed all returns or exits from the Executive Control System are made from this component.

2. *Console Control* - maintains exclusive control of the Console Typewriter. It services all console requests and termination interrupts as well as program message requests.

## ECS COMPONENTS (Cont'd)

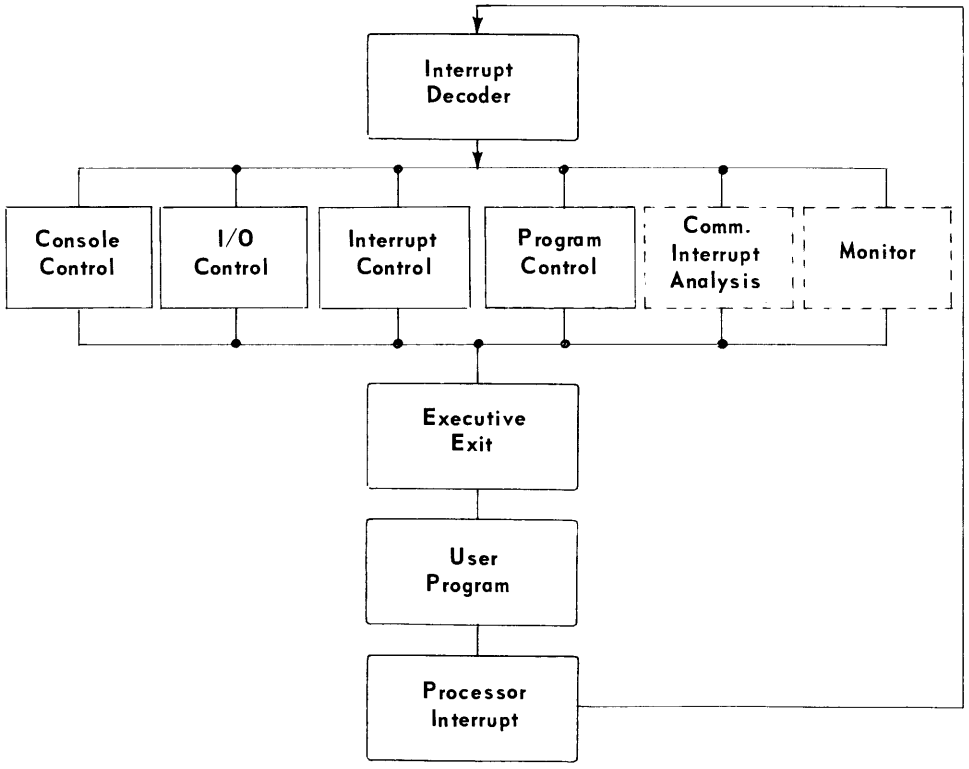
3. *Program Control* - services all requests requiring the use of the System and Program Load Libraries. As such, the initiation of programs, the loading of program overlays, program checkpoint, program termination, and program restart are controlled by this component.

4. *Input/Output Control* - services all requests for input/output operations on all devices except the Console Typewriter.

5. *Monitor* - this nonresident component controls and supervises programs run in a job stream.

6. *Communications Interrupt Analysis* - is a special routine which handles interrupts pertaining to the Multichannel Communications Program.

A sample process flow of the interrelationships of the six major components of the Executive Control System is presented below:




**ECS Sample Process Flow**

## INTERRUPT CONTROL

### Functional Description

◆ The Interrupt Control decodes all interrupts, services Executive Calls. services the timer, and handles machine failure conditions.

Whenever a program operating in programming state $P_1$ is interrupted, control goes immediately to the Interrupt Decoder entrance of Interrupt Control except for machine failure interrupts. The interrupt is then analyzed and linkage is established to the proper servicing routine. When servicing is complete, control returns to Interrupt Control, and then to the programming state $P_1$ program.

### Interrupt Control Components

◆ The Interrupt Control consists of the following:

1. Interrupt Decoder
2. Executive Exit

### *Interrupt Decoder*

◆ Whenever a program operating in programming state $P_1$ is interrupted, control goes immediately to the Interrupt Decoder, which is the only entrance to the Executive (other than that for machine failure interrupts). The interrupt is analyzed in the following manner:

1. The P-counter in programming state $P_1$ is stored in the Operation List bytes 5 through 7 for this program.
2. Bits $2^{16}$ through $2^{31}$ of the Interrupt Flag Register (which indicate potential interrupts) are stored in the Operation List bytes 2 and 3.
3. The weight register (GR15P3) is used to access an address from the Interrupt Address Table, and control is transferred to that address.
4. At the jump address for an Executive Call, a code is used to access an address from the Address Table, and control goes to the proper Executive routine.

Interrupts are classed as follows:

*Program Independent*

1. Machine Failure
   a. Power Failure
   b. Machine Check
2. External Signal
3. Selector or Multiplexor channel (I/O).
4. Console Request
5. Elapsed Time Clock

*Interrupt Decoder (Cont'd)*

*Program Dependent*

1. Executive Call
2. Unrecoverable Error
   a. Privileged Operation
   b. Address Error
3. Recoverable Error
   a. Op-Code Trap
   b. Exponent Overflow
   c. Divide Error
   d. Significance Error
   e. Exponent Underflow
   f. Decimal Overflow
   g. Fixed-Point Overflow
   h. Data Error
4. Test Mode

*Machine Failure Interrupt* – When either a Power Failure or a Machine Check occurs, the processor is set to idle. In the case of a Machine Check, a message is typed out before idling.

*External Signal Interrupt* – All external signal interrupts are inhibited by the Executive unless the Direct Control feature is supported.

*Selector or Multiplexor Channel Interrupt* - All selector and multiplexor interrupts are routed to the I/O Control Device Return routine, which services all I/O interrupts except those from the Console or communication devices. Console interrupts go immediately to Console Control. Device return stores interrupt information in response to communication device termination and then routes them directly to CIA for examination.

*Console Request Interrupt* – An interrupt caused by pressing the Console Interrupt (COIN) button is routed directly to the Console Control routine.

*Executive Call Interrupt* – When an Executive Call interrupt occurs, the Interrupt Decoder examines the identifying code and gives control to the routine which responds to the call.

*Unrecoverable Error Interrupt* – Whenever a Privileged Operation, or an Address Error causes an interrupt, the interrupted program's ESA is tested for an unrecoverable error procedure. (See STXIT Executive Call.) If so, the address of the error procedure is taken from the ESA and put into the program's Operation List entry. Then control goes to Exec Exit. This procedure is permitted so that the error may be further analyzed before the program is terminated; the error procedure must end by issuing a TERM or TERMD Executive communication macro.

*Interrupt Decoder (Cont'd)*

If there is no unrecoverable error procedure, an indication of the error is typed, and the operator is given the option of dumping before the program is terminated.

*Recoverable Error Interrupt* – Whenever an Op-Code Trap, an Exponent Overflow, a Divide Error, a Significance Error, an Exponent Underflow, a Decimal Overflow, Fixed-Point Overflow, or a Data Error causes an interrupt, the interrupted program's ESA* is tested for a program check error routine. (See STXIT and EXIT Executive Calls.) If so, the normal return information is stored in the program's ESA and the address of the error procedure is taken from the ESA and put into the program's Operation List entry. Then control goes to Exec Exit. The program can determine the specific interrupt from the weight stored in his ESA.

If there is no user program check error routine or if the user is already in his program check routine, an indication of the error is typed and the operator is given the option of dumping before the program is terminated.

*Test Mode Interrupt* – When a Test Mode interrupt occurs, the Interrupt Decoder gives control to the procedure whose address is in the test mode slot in the Interrupt Address Table. When AIDS or Monitor is operating, the slot contains the address of the appropriate routine. Otherwise, this slot contains zeroes.

*Elapsed Time Clock* – When an Elapsed Time Clock interrupt occurs, and communications is supported, an indicator is set and control is transferred to the CIA routine. When communications is not supported, this interrupt can only occur when the maximum timer value has expired.

**Exec Exit**

◆ Return to the user program from the Executive is always through the Exec Exit. The code to be executed is found by scanning the Operation List (OL), which contains, in priority order, entries for Executive functions, for user programs, and for idling when all other entries are waiting. When the Exec Exit procedure has selected a user program to receive control, the Time Clock is serviced prior to executing the Program Control instruction to return to programming state $P_1$.

*ESA - Executive Storage Area - See page 3-31.

## CONSOLE CONTROL

**Functional Description**

◆ Console Control maintains exclusive control of the Console Typewriter. It provides for operator commands to the Executive, messages to the operator from the Executive and user programs, and replies to these messages. Entrance to Console Control may be caused by an Executive Call, the Console Request button (COIN) on the Console Typewriter, or a Console I/O termination.

**Console Control Components**

◆ The Console Control consists of the following:

User Executive Message Request (TYPE)

Console Request Interrupt (COIN)

Console Termination Interrupt

*User Message Request*

◆ When a user issues the TYPE macro, the Console Control List (CCL) is examined to insure that no previously issued TYPE (by this program) remains incomplete before the caller's request is logged in the CCL. If there is an incomplete write, the Operation List entry for the caller is set to the address of the call, and Exec Exit is set to scan the next entry in the Operation List before control is given to Exec Exit. If an incomplete write is not logged in the CCL, the user's request is now logged. The output message is then formatted and typed when the console is not busy. Formatting of the message consists of obtaining an initial character to prefix the message based on the user's Program Table Entry, obtaining the address of the program name as a function of the user's program number, and various addresses in the CCW's to be issued to the console.

*Console Request Interrupt*

◆ When the Console Interrupt (COIN) button is pressed by the operator, an interrupt occurs in the processor and control goes to the Console Request Interrupt routine. If the Console is busy typing in, or the Key-in Area is full (i.e., the last operator command to the Executive or the Monitor has not been serviced yet) the interrupt is ignored, and control goes to the Exec Exit. If the Console is not already busy, the interrupt is logged and a read is issued to the Console. Control then goes to the Exec Exit.

Certain Executive functions are available to the operator through the Console Typewriter. The first character of a type-in identifies it as a command to the Executive (E), a command to the Monitor (M), a reply to an Executive typeout (7, 6, R, T, U, V, W, X, Y, or Z), or a reply to a user program (1, 2, 3, 4, 5, or 6). The second character is always a space. The remainder of the Key-in Area is either delivered to the user program or Executive function requesting the reply, *or* moved into the Executive Key-in Area for analysis by Program Control or Monitor.

*Console Termination Interrupt*

◆ Whenever an operation on the Console terminates without error, the CCL is scanned on a priority basis, and another request is started if one exists in the list. If one does not exist in the list, the Console is no longer busy.

If there was an error on the operation just completed and the operation was a write, it is repeated and control goes to Exec Exit. If a read is found to be in error, the ERROR key is pressed, the message read is ignored and the COIN that initiated the read is considered satisfied. Then the CCL is scanned for another request as described above.

## PROGRAM CONTROL

### Functional Description

◆ Program Control exercises control over the initiation and termination of all programs. It contains routines which handle the locating and loading of programs and overlays, memory allocation, and deallocation, program termination, checkpoint, and restart.

### Program Control Components

◆ Program Control consists of the following:

1. Program Control Interface
2. Validate Key-in
3. Initiate Program
4. Edit Run Time Parameters
5. Memory Allocator
6. Load User Overlay Function
7. Dump Program
8. Terminate Program
9. Memory Deallocator
10. Change Priority
11. Checkpoint
12. Restart

### *Program Control Interface*

◆ All entrances to Program Control are via the Program Control Interface. Program Control consists of five areas: (1) Loading of User Overlays, (2) Control and Interface of Executive Overlays, (3) Control and Interface of Non-Resident Error Recovery Overlays, (4) Loading of System Overlays, (5) All other Non-I/O-Oriented Program Control Functions.

When one of the Program Control functions is required, the request is either performed or queued, depending on whether the function is free or busy. When the Program Control function is requested and is found busy, the caller is queued in the Program Table for this function and Program Control goes to Exec Exit.

If the function is not busy, the caller is placed in control of the function, the function is set busy, and the function unwaited in the Operation List. When the function is completed, Program Control determines if other users are waiting for it. If so, the highest priority user is selected, and the function unwaited in the Operations List; that is, made available for use.

In any event, the user which has just finished with the function is unwaited in the Operations List, and Program Control goes to Executive Exit.

*Program Control Interface (Cont'd)*

Some Program Control functions run through to completion and hence are never busy. These functions do not have entries in the Operation List, and are always executed immediately upon request. The following Program Control functions do not run through to completion when initiated and have waiting queues associated with them:

1. Request for a user overlay.

2. Request for an Executive overlay.

3. Request for a error recovery overlay.

Each of these functions has an entry in the Operations List (error recovery having three; one for nonpriority users, one for CUP, and one for MCP). These Program Control Operation List entries are unwaited and waited as they are needed by the users. These entries are selected from the Operations List on a priority basis.

Requests for a given Program Control function are handled on a priority basis; user program 1 lowest, MCP highest. Any user can have up to three Program Control functions being performed for him at one time. These are:

1. Loading a user overlay.

2. Loading or performing an Executive overlay.

3. Loading or performing error recovery.

In the case of loading a user or Executive overlay, only one such request at a time per user is allowed.

The spec packet address associated with the requested program control function is stored in the proper place in the user program table entry. For error recovery, a list consisting of one entry for each device which can be assigned to the user is maintained.

*Validate Key-In*

◆ The Validate Key-in system overlay is used only for the load (LOD) and restart (RST) Executive operator requests. This overlay also is used for the Communication User Program (CUP) and Multichannel Communication Program (MCP) when supporting communications. The prime function of this routine is to assign a slot in the Program Table for the program being loaded or restarted. If the operator request indicates that a program priority is to be assigned, the corresponding Operation List entry is assigned if it is available; otherwise, the command is rejected. If no priority was indicated, the highest available Operation List entry is assigned. If a load device is specified, and it is valid, it is then placed in the Load Library Table. If it is invalid the Load (LOD) is rejected. If the command is Restart (RST), the Restart overlay is fetched, and control goes to Program Control Interface.

*Initiate Program*

◆ The Initiate Program System overlay is entered after the Validate Key-In overlay has processed a load (LOD) command. It executes all the functions necessary to prepare the program in the LOD command for execution. The first time the device is accessed for the loading of user programs, the Standard Volume label and Volume Table of Contents (VTOC) are brought in to obtain the disc address of the Program Directory entry. This data is saved in the appropriate Load Library Table entry. On subsequent requests to load user programs from the same device, the PD is searched using the CCHHR in the Load Library table. If run-time parameters are specified in the Load, the Edit run-time parameter is brought in. Otherwise, the Memory Allocate overlay is accessed.

*Edit Run-Time Parameters*

◆ When a load (LOD) Executive Operator Request indicates that there are run-time parameters to be processed, Program Control loads the Edit run-time parameter after the Initiate Program Overlay. This routine assigns the parameter device (RTP) to itself, then finds the largest segment of free memory, and reserves it for processing the parameters. Parameters of a given type must be together, and only one group of each type is allowed. The valid types and the order in which the groups must appear are as follows:

1. ASSGN CARDS (Device Assignment)
2. FILES, VOL, and TPLAB CARDS (taken as one type)
3. VDC CARDS (Volume Displacement)
4. JOB CARDS (Successor Program Call)
5. END CARD (No more Run Time Parameters)

When a JOB card is found, the RTP Successor Call Presence Flag is set in the Program Table.

If shuffled or invalid parameters are encountered, the run-time parameter routine deallocates the run-time parameter device, the LOD command is rejected, and the operator is notified.

After all run-time parameters have been processed the run-time parameter device is deallocated and the memory used to process the parameters is returned to the free memory table.

*Memory Allocator*

◆ The Memory Allocator bases its control of memory on the Free Memory Table, which contains nine slots, each of which may define an unassigned memory segment. If the Memory Protect feature is present, the processor protects memory in 2,048 byte units. If the Memory Protect feature is not present, memory is assigned in multiple of eight bytes.

*Memory Allocator (Cont'd)*

The Memory Allocator obtains the memory requirements from the Program Directory if loading from disc or drum, or from the Program Descriptor Block if loading from tape, unless this is overridden by a memory specification in the E LOD command. In either case, this routine scans the free memory table for segments not less than the required sizes. If the operator requires that high memory be assigned or allocated, the first ample segment is assigned. Otherwise, the Free Memory Table is scanned from the top and a record kept of the closest fit. If an exact fit is found, it is assigned. However, if the scan ends, the closest fit is assigned. Then a key is set in each 2,048 byte unit of memory assigned if the Memory Protect feature is present.

When the Memory Allocator has completed its function, the packed run-time parameters are relocated if necessary; the Executive Storage Area is initialized; memory protect and storage protect keys are set if necessary; and the Operation List entry is created.

*Load User Overlay Function*

◆ The Load User Overlay function is a separate Program Control function and is activated by an E LOD call or by any of the following macros:

1. LPOV - Load program overlay
2. LPOVR - Load program overlay and return
3. FLOAT - Float program overlay
4. FLODR - Float program overlay and return
5. TERMS - Call Successor Program

The Load User Overlay function is performed as follows:

1. Search of the proper load directory for the desired load. This information supplies the disc origin of the load, and its program relative load address in memory.
2. Actual loading of the text and modifiers. Text occurs first in a load, and is placed directly in the desired core locations. The modifiers follow the text sequentially on the disc, and are read into a work area within Program Control. Then, they are applied to the text already in core.

*Dump Program*

◆ The Executive Dump is executed as a result of a Dump (DUM) Executive operator command, a Terminate and Dump (TERMD) Executive Call, or a reply to an Executive terminate message in which the operator elects to dump before terminating.

The following information is edited and written to magnetic tape so that when printed, it has both actual memory locations and program relative location of the program coding.

1. The P1 General Purpose and Floating Point registers.
2. The Executive Tables.
3. The program's Program Table entry and Device List entries including random access devices.
4. The program, including its Executive Storage Area and run-time Parameter Area (if any).

When the dump is complete, the Halt bit is set in the dumped program's Operation List entry and control goes to the Terminate Program routine.

*Terminate Program*

◆ Executive Communications Macro - when a termination Executive Call (TERM) interrupt occurs, and run-time parameters are not present, the program does the following:

1. All devices are quieted and deallocated.
2. Memory is deallocated.
3. The Operation List and Program Table entries are marked unassigned.
4. The operator is notified.
5. Exec Exit is entered.

If the call was a TERMD, the DUMP function is performed prior to the above. If the call was TERMS, only step 4 (above) will be accomplished before entering Program Control with a request to load the successor program.

When a termination Executive Call (TERM) interrupt occurs, and run-time parameters are present, the following is accomplished:

1. All devices are quieted and deallocated.
2. The operator is notified.
3. The run-time parameter area is searched for JOB entries.

When an entry is found, Program Control is entered with a request to load the successor program using the same program table and Operation List entries. If no more JOB entries exist, the Memory is deallocated and the Program Table and Operation List entries are marked unassigned before notifying the operator.

If the call was TERMS, the devices are not deallocated and the JOB name found overrides the name in TERMS call.

*Operator Command Entrance*

◆ When a termination operator command (HLT or DUM) is typed, the devices being used by the program are quieted and all device entries in the I/O queue are deleted before control goes to the Executive communication macro entrance.

*Executive Call Entrance*

◆ When an Executive function determines that a program cannot be allowed to continue the Termination bit and the Termination Reason bit are set in the program's Operation List. The setting of the Halt bits causes the Exec Exit to bypass this program until all its other pending operations are completed. Then, Exec Exit gives control to the Executive communication macro entrance.

*Memory Deallocator*

◆ The Memory Deallocator tests successive Free Memory Table slots to determine if:

1. the deallocated memory is contiguous to a segment of free memory, in which case the two segments are combined and represented by a single Free Memory Table entry;

2. the combined segment is contiguous to the free memory represented by the next entry, in which case the two entries are combined and the remaining entries are pushed upward;

3. the deallocated memory is between that represented by two Free Memory Table entries; but is contiguous to neither in which case the second of the two and all subsequent entries are pushed down and a new entry is made for the deallocated memory.

*Change Priority*

◆ The Change Priority routine is called when the operator command PRY is typed which causes the program specified to be given a priority of 6. The Operation List entries for any other programs are pushed down, and the new Operation List entry is inserted. As each entry is relocated, its indices in the Program Table, Device List, Current Operation Slot, and Console Control List are adjusted. When all entries have been relocated, control goes to the Program Control Interface.

*Checkpoint*

◆ The Checkpoint routine saves the user program and its pertinent environment on some secondary storage medium so that at a future time, the program may be reinitiated and entered at a specified address. The Checkpoint routine of Program Control is divided into two subroutines that accomplish the major functions required to service the user Executive Communication macro (CKPT). These two subroutines are, (1) Device Quieting, and (2) Program Environment Dumping.

*Device Quieting* - When a Checkpoint macro is executed the Device Quieting routine, which is resident, is entered to validate the macro parameters. If the parameters are valid, Device Quieting delays further operation of the user program and Checkpoint until all the user's started and queued input/output requests have been fulfilled. By completing all I/O's; the user is ensured of valid information in the affected Command Control Blocks when restarting. If Device Quieting finds erroneous parameters, byte 25 in the macro expansion is set to $(01)_{16}$ and control is given to the user's error routine. When this condition occurs the Checkpoint is not taken.

*Program and Environment Dumping* – The Program and Environment Dumping routine writes the user program and other pertinent information to a specified storage device. This is necessary in order to restart the program. This routine is entered from the Device Quieting routine.

*Restart*

◆ The restart procedure is initiated by an operator console request which describes the program to be restarted. Upon receiving the operator request, Program Control is utilized to load the Restart routine into the Executive Overlay area. The Restart routine is then given control and exercises its function of reestablishing the user program and its data in the operating system environment. Restart provides for the program to continue operation at the restart address supplied at checkpoint time.

## INPUT/OUTPUT CONTROL

### Functional Description

◆ Input/Output Control services requests for operations on all devices supported by TDOS except the Console Typewriter. This includes device assignment, servicing of I/O termination interrupts, selector channel scheduling, execution of I/O channel programs, error recovery, and updating of the random access on-line catalogue.

### Input/Output Control Components

◆ Input/Output Control consists of the following:

1. Input/Output Dispatcher
2. Device Assignment
3. Error Recovery
4. Device Return Handling
5. Wait
6. Channels and Devices
7. Selector Channels
8. Multiplexor Channel
9. Dual Channel Tape Stations
10. Seven Channel Tape Stations
11. Random Access Devices
12. Program Sharing of Random Access Devices
13. Random Access Off-Line Seek
14. Random Access Multichannel Switch Controller
15. Communication Devices
16. Data Exchange Control

### *I/O Dispatcher*

◆ The Input/Output Dispatcher initiates all input/output operations on all devices except the Console Typewriter. It also queues I/O requests when the channel needed is busy.

When the Interrupt Decoder receives an EXCP or EXCPW Executive Call, control is given to the Input/Output Dispatcher, and the Command Control Block (CCB) is examined to determine if the file has been assigned. If the file has not been assigned, the Device Assignment routine is entered. However, if the file has previously been assigned, control is passed to either the Start I/O or Test Device Procedure (routines within the I/O Dispatcher) according to the Request Type.

The Test Device Procedure initiates and executes the Test Device instruction, and gives control to the normal terminate entrance of Device Return or to the Channel Inoperable Procedure.

*I/O Dispatcher (Cont'd)*

In the Channel Inoperable Procedure a comment is typed, and Exec Exit is entered, preventing the user program from proceeding until the operator informs the Executive that the channel is operable. This does not occur if the device has an alternate channel which is operable.

In the Start I/O Procedure the Channel Address Word (CAW) is set up and the Start I/O Instruction is initiated and executed. Depending upon the condition code, control is transferred to Executive Exit, the Channel Inoperable Procedure, or the Abnormal Terminate path of Device Return, except when the device is busy.

If the device is busy the request is put at the bottom of the queue and control goes to the Start Listed Request Procedure which performs a channel scheduling function and returns control to the Start I/O Procedure.

*Device Assignment*

◆ Device Assignment takes place on the first physical I/O command to a symbolic device. The Device Assignment routine is entered from I/O Dispatcher when the CCB indicates that is has not been used to execute a channel program. Since several CCB's may be used for a given device, Device Assignment first scans the Device List to see if the Symbolic Device Name in the CCB is in a Device List entry with this program's priority. If the scan finds the Symbolic Device Name with the correct priority in the Device List, the file has already been assigned and this assignment is applied to the current CCB. Control then returns to the I/O Dispatcher. If the scan does not find the proper Symbolic Device Name in the Device List, a test is made to see if this program has Assignment run-time parameters. If not, or if they are invalid, control goes to the Operator Assignment Procedure. However, if the parameters are present and valid, the assignments are made and control returns to the I/O Dispatcher.

Because communication devices can be handled by the MCP one and only one Device List entry exists for all devices no assignment is necessary.

In the Operator Assignment Procedure a typewriter message requests the operator to assign a device for the Symbolic Device Name. The type-in gives the installation's mnemonic for the device being assigned. If the file is an optional input and is not present, the operator types in the reply number followed by "NO". This causes the No File bit to be set in the CCB. If the device is nonexistent, the operator is requested for a new assignment. For devices already assigned but unavailable, the operator is notified and may reply with another assignment or the same assignment followed by a W. W indicates that the program is to wait for the device previously assigned. Only one program may wait for a given device, and that program is suspended until the device is released by its present user. After the assignments have been made, control returns to the I/O Dispatcher.

*Error Recovery*

◆ The channel status, standard device, and sense bytes are tested to determine the type of error. If an unrecoverable error has occurred the CCB is checked to see if the user can accept the error. If so, the unrecoverable error bit is set and control is given to the I/O Dispatcher. If unrecoverable errors are not accepted the operator is notified and is given the option of terminating or dumping and terminating.

*Error Recovery (Cont'd)*

For recoverable errors, the CCB is checked to determine if standard error recovery is to be bypassed. If so, the CCB is marked as having terminated and control is given to Device Return.

When an Error Recovery routine is to be used, the request for error recovery is queued for one of the three error recovery overlay areas according to the priority type of the call. The Error Recovery Bit in the Device List entry is set and all termination information is stored in the CCB. If the overlay area is not busy a call is issued for the overlay and control is returned to Exec Exit.

If error recovery is not successful the operator is given the option of: (1) retrying the Error Recovery routine, (2) ignoring the error and giving control back to the user. Acceptance by the user program gives control to Device Return after the indication of the abnormal termination has been recorded in the CCB. If the user program has not provided for acceptance, the program is terminated.

*Device Return Handling*

◆ This routine is entered from the Interrupt Decoder when a termination occurs on the multiplexor or a selector channel. Initially the pertinent scratch-pad memory information is stored in the calling program's Command Control Block (CCB). A check is made to determine if the termination was caused by the Console Typewriter, in which case control is immediately given to Console Control. If the termination is from a sense command, or the standard device byte or channel status byte indicates an abnormal termination, control is transferred to the Error Recovery routine. If the termination is on the multiplexor and the I/O was issued by MCP, control is transferred to CIA; otherwise, it is transferred to the Exec Exit. If the termination occurred on a selector channel, the channel queue is checked for a waiting request. If none are queued, Exec Exit is entered, or for MCP calls control is transferred to CIA. If there is a queued request, the proper Device List and Channel List entries are updated and the request is removed from the queue. When the return is found to be a normal termination, Device Return is allowed to follow its normal path into the I/O Dispatcher.

*Wait*

◆ This routine is entered when a WAIT Executive Call is issued. Before the WAIT Call is executed, the Termination bit in this WAIT's CCB is tested. If it is set the WAIT is bypassed; otherwise, an interrupt occurs. In either case, control goes to the Exec Exit.

*Channels and Devices*

◆ The TDOS system maintains a channel queue for each selector channel which contains more than one device. When an input/output request is encountered for a device whose channel is busy, the request is placed into the channel queue. This operation is to be performed at some later time when the channel is not busy. In TDOS queuing and subsequent initiating are accomplished on a first-in, first-out basis provided that communications devices are not being supported. If communications is being supported, requests from communications programs receive priority.

*Channels and Devices (Cont'd)*

Any device that is supported by TDOS, defined at systems generation time, and is to be handled by the generated Executive, has an entry in the Device List. This DL entry describes the device and the information needed to assign the device to a program. Although the format of the DL entry is identical for all devices, it does include certain additional information for shared random access devices.

*Selector Channels*

◆ Up to six channels can be handled. There is no restriction as to the device configuration. Because a selector channel can handle only one I/O operation at a time, additional requests have to be queued. At the termination of one I/O operation the next one in the queue is attempted. A Channel List includes an eight-byte entry for each channel, pointing to a channel queue for this channel. This queue contains a two-byte entry for each possible device request to this channel. The type of priority code and the DL entry are stored in the queue. The channel queues are filled dynamically from the bottom and emptied from the top. Request for MCP and CUP as well as the appropriate error recoveries have priority (MCP, CUP, users). The requests are issued on a first-in, first-out basis.

*Multiplexor Channel*

◆ One multiplexor channel with up to nine subchannels can be connected to a 70/35-45-55 Processor and are handled by I/O Control. Only the Console Typewriter is handled by Console Control. With the exception of the Multichannel Communications Controller (CCM) only one device per subchannel can be handled at a time and no operation in burst mode is allowed if other I/O operations are currently using the multiplexor. The operations of the CCM and the connected communication devices must be handled by the MCP only. Because of these restrictions the I/O Control can issue each I/O request immediately.

*Dual-Channel Tape Stations*

◆ This device needs as additional information the address of the co-channel. If an I/O operation on this device is requested, it is queued in both channel queues, unless one of the channels is not busy handling another I/O. When the request is fired from one channel queue it has to be cancelled on the other. Because the position of the request in the queue is not known, a complicated routine would have to be used to seek and cancel the request. To avoid this effort of time, the address of the channel used for firing is set in the DL entry indicating that there is a queued request which is not to be fired. When the turn for this request comes in the queue the indicator is reset, the request is ignored, and the next I/O in the queue is fired. If another request for the device is given and must be queued, this indicator is reset and the request is only queued in this channel queue. Only at the point of deallocation is a request in the queue cancelled to avoid confusion if the tape is to be assigned to a program of a different priority type.

*Seven-Channel Tape Stations*

◆ At the assignment time of a seven-channel tape station a device control byte is supplied to define the mode of the tape. This byte is stored in the DL entry for the particular tape. Prior to any I/O operation issued to this device a Write Control is fired using this byte to set the proper seven-level mode at the controller.

*Random Access Devices*

◆ Random access devices may be shared by all user programs. The DL entry for this device type must be modified. The DL entry points to an extension list containing seven entries, one for each program. Each entry contains the parameters particular to the requesting program and is eight bytes long. The SDN is not included, and when a request is issued only the program number is checked to insure that the device is assigned to the calling program. For these devices the address of the extension minus 16 is stored in the channel queue to find the right request. Via the CCB address in the extension, the DL entry can be found. Requests from the Executive are queued the same way but the parameters are delivered with the appropriate call.

There are five possible calls for the systems device (Disc/Drum) in addition to the seven user requests:

3 for error recovery overlays

1 for Exec overlays

1 for user overlays

Because all programs can use any random access device, it is the responsibility of the user to avoid erroneous writing to the devices. The user cannot be sure that the device is in the position he left it and must take care of proper head positioning with any I/O operation. Off-line seeks should only be used if no other program has access to the device.

No deallocation (DDEV) SVC should be issued for a random access device, because the SDN is not included in the DL entry. The deallocation is not necessary because all programs have access.

Error recovery for random access devices is not resident because the system overlay routine performs its own error recovery.

*Program Sharing of Random Access Devices*

◆ In TDOS, a random access device can be assigned to more than one program. However, the TDOS Executive does not provide protection (for areas being used by one program) from being erroneously overwritten by another program. A program is allowed to execute one EXCP at a time to the device. If two consecutive EXCP's are executed to the same device without an intervening WAIT, the program is prohibited from continuing until the original EXCP is terminated. This means that the TDOS Executive queues requests to a given random access device by program.

*Random Access Off-Line Seek*

◆ Program sharing of random access devices is permitted in TDOS; therefore, a user should be very careful when issuing off-line seeks. He should insure that no other program is using the same device. If other users are sharing the device, erroneous results may occur when using an off-line seek, because there would be no guarantee that some other user did not move the head after a previous off-line seek was completed. When an off-line seek is issued, two interrupts are processed by the Executive: 1) when the controller responds with a termination interrupt indicating that the channel is now free; and 2) when the seek operation is completed. If the user wishes to check which condition exists he can examine the standard device byte in the CCB. Bits $2^3$ (Device End) and $2^6$ (Termination Interrupt Pending) are set when the seek is received by the Controller. Bits $2^3$ (Device End) and $2^7$ (External Device Request Interrupt Pending) are set when the seek is completed.

*Random Access Multichannel Switch Controller*

◆ This option permits a 70/551 Random Access Controller to be utilized by two selector channels with switching accomplished under program control. The TDOS Executive does not directly support this option; however, if it is used, certain actions are performed by the Executive. When the Executive attempts to issue an I/O to a device whose controller is selected to another channel, bit $2^4$ (Control Busy) is returned. The Executive does not attempt to issue any further I/O's to devices on that channel until an interrupt occurs with bits $2^0$ (Status Modifier) and $2^7$ (Manual Request) set in the standard device byte indicating that the switch has returned to the neutral position.

*Communications Devices*

◆ The Communications Devices are completely under the control of the MCP. The Device List contains only one DL entry for all devices. The device address is supplied with each call. All requests are fired immediately because the devices are connected to the multiplexor channel only. After any request or interrupt belonging to MCP (including tape and random access operations) control is given to CIA. The proper information is stored in the CCB. The MCP handles the right sequence of operations and any error recovery procedures.

*Data Exchange Control*

◆ The DXC can be assigned to a program and is handled as a normal device. Provisions are also made to handle manual requests. No error recovery is provided by the Executive to handle abnormal conditions.

## MONITOR

**Functional Description**

◆ The Monitor supervises and controls programs that are run in a job stream environment. When a job stream is to be executed, the Monitor must be called by an Operator Console Request. The Executive then allocates memory and loads the Monitor which remains resident until it terminates itself at the completion of the total job stream. Processing of the total job stream is called a Monitor session, and any number of jobs may be performed within a session.

**Monitor Components**

◆ The Monitor consists of the following:

Job Stream Language Processing

Monitor Executive Call Command Servicing

Monitor Operator Calls Servicing

Device Assignment

Monitor Initiation

Monitor Termination

Monitor Overlay Call

Monitor I/O Processing

DUMP Function

Program Termination

SNAP Function

PATCH Function

Section 5 describes the Monitor in detail.

## COMMUNICATIONS INTERRUPT ANALYSIS

**Functional Description**

◆ The Communications Interrupt Analysis (CIA) component is a resident extension of the TDOS Executive Control System. The primary function of the CIA is to analyze all interrupts associated with the communications processing environment. In carrying out this function, the CIA analyzes the CCM interrupts, records them for subsequent processing, and initiates the appropriate function(s) needed to properly service the interrupt. The CIA also analyzes and records for subsequent processing, interrupts associated with devices other than the CCM. Section 7 describes the Communication Interrupt Analysis routine.

## EXECUTIVE DATA

◆ This section defines the tables, input areas, and work areas used by the Executive, and the areas established in the user program by the Executive.

### Executive Data Contents

◆ The Executive Data defined in this section are listed below in alphabetical order.

**Executive Communication Region**

◆ The 86-byte Executive Communication Region (see table 3-1) is a source of data and addresses for non-Executive programs. Its address is provided to user programs when the ADEXT macro instruction is executed.

**Table 3-1. Executive Communication Region**

| Absolute Memory Location | Byte | Bit | Meaning |
|---|---|---|---|
| 000F0 | 0-1 | | Month. |
| 000F2 | 2-3 | | Day. |
| 000F4 | 4-5 | | Year. |
| 000F6 | 6-8 | | Day of Year. |
| 000F9 | 9 | 0-2 | Memory Size: 011 = 65K; 100 = 131K; 101 = 262K; 110 = 524K |
| | | 3 | Memory Protection Option: 1 if present, 0 if not present. |
| | | 4 | System Timer Option: 1 if present, 0 if not present. |
| | | 5 | Direct Control Option: 1 if present, 0 if not present. |
| | | 6-7 | Processor Type: 00 = 70/35; 01 = 70/45; 10 = 70/55 |
| 000FA | 10-11 | | Device List Address. |
| 000FC | 12-13 | | Program Table Address. |
| 000FE | 14-15 | | Load Library Table Address. |
| 00100 | 16-17 | | Interrupt Address Table Address. |
| 00102 | 18-19 | | SVC Jump Table Address. |
| 00104 | 20-21 | | Current Operation Slot Address. |
| 00106 | 22-23 | | Last Device List Address. |
| 00108 | 24-25 | | Error Recovery Interface Queue Address. |

**Executive Communication Region** *(Cont'd)*

**Table 3-1. Executive Communication Region** *(Cont'd)*

| Absolute Memory Location | Byte | Bit | Meaning |
|---|---|---|---|
| 0010A | 26-27 | | Operation List Address. |
| 0010C | 28-29 | | Free Memory Table Address. |
| 0010E | 30-31 | | On-line Catalog Address. |
| 00110 | 32-33 | | Console Control List Address. |
| 00112 | 34-35 | | End Address of Common Data Area. |
| 00114 | 36-37 | | Exec Overlay Area Address. |
| 00116 | 38-39 | | Program Control Entrance for Type-ins. |
| 00118 | 40-43 | | Address To Branch To On Test Mode Interrupt. |
| 0011C | 44-45 | | Exec Overlay Area List Address. |
| 0011E | 46-47 | | User Error Recovery Overlay Area List Address. |
| 00120 | 48-49 | | MCP Error Recovery Overlay Area List Address. |
| 00122 | 50-51 | | Start Address of Common Data Area. |
| 00124 | 52-53 | | Communications Device List Entry Address. |
| 00126 | 54-55 | | Executive Load Directory Address. |
| 00128 | 56-59 | | Load Address of Last User Overlay Loaded. |
| 0012C | 60-63 | | Aids Switch for Handling Executive Communication SVC Interrupts. |
| 00130 | 64-65 | | Interrupt Decoder Routine Address. |
| 00132 | 66-67 | | DXC Routine Address. |
| 00134 | 68-69 | | Checkpoint Parameters Address. |
| 00136 | 70-71 | | Timer Routine Address. |

**Executive Communication Region** *(Cont'd)*

**Table 3-1. Executive Communication Region** *(Cont'd)*

| Absolute Memory Location | Byte | Bit | Meaning |
|---|---|---|---|
| 00138 | 72-73 | | CUP Error Recovery Overlay Area List Address. |
| 0013A | 74-75 | | Direct Control Routine Address. |
| 0013C | 76-77 | | Monitor Snap Shot Parameter Area Address. |
| 0013E | 78-79 | | CIA Address - Executive Console Typein Area. |
| 00140 | 80-81 | | CIA Address - Device Scratchpad where Channel Registers are stored when an I/O interrupt occurs. |
| 00142 | 82-83 | | CIA Address - Four-Byte Work Area containing an address that points to the address of CCB at Selector Channel I/O interrupt time. |
| 00144 | 84-85 | | CIA Address of Executive. Routine to Process Invalid SVC. |

**Operation List**

◆ The eight-byte Operation List (see table 3-2) is used to record the current status of each user program and of certain Executive functions. It is arranged in the reverse order of the priority assigned to each entry, and it is scanned in reverse to find the most eligible code to be executed.

**Table 3-2. Operation List**

| Byte | Bit | Meaning |
|---|---|---|
| *0 | 0 | Terminate Flag. |
| | 1-2 | Reserved. |
| | 3 | Wait for Completion of Executive Operation. |
| | 4 | Reserved. |
| | 5 | Wait Nothing to do. |
| | 6 | Wait for I/O Termination. |
| | 7 | Wait for LPOV. |
| 1 | 0 | Run-Time Parameters Presence Bit. |
| | 1-7 | Coded Reason for Terminating Program. |
| 2-3 | | Interrupt Flag Register bits $2^{16}$ through $2^{31}$. |
| 4 | | Program Number, (Shifted Left five Bits, for users only). |
| 5-7 | | Return Address, (P1 Program Counter). |

* If byte 0 ='FF', this operation list entry has not been used.
= '00', this operation list entry is unwaited and is available for control.

The 15 entries for the Operation List are as follows:

1. Idle.
2. User Program Priority 1.
3. User Program Priority 2.
4. User Program Priority 3.
5. User Program Priority 4.
6. User Program Priority 5.
7. User Program Priority 6.
8. Monitor.
9. Executive Overlay.
10. Program Control.
11. User Error Recovery Overlay.
12. CUP.
13. CUP Error Recovery Overlay.
14. MCP.
15. MCP Error Recovery Overlay.

**Current Operation Slot**

◆ The eight-byte Current Operation Slot table (table 3-3) keeps track of the program that had control when an interrupt occurs.

**Table 3-3. Current Operation Slot**

| Byte | Meaning |
|---|---|
| 0-3 | Address of Operation List Entry. |
| 4-5 | Program Number (Shifted Left 5 Bits). |
| 6-7 | Program Priority (Shifted Left 3 Bits). |

**Interrupt Address Table**

◆ The Interrupt Address Table consists of 32 halfwords corresponding to the 30 potential interrupts which cause control to be given automatically to program state P3. The entries are indexed by the weight register in the processor (GR15-P3) when an interrupt occurs. Each halfword contains the address of the routine that resolves the interrupt corresponding to that weight.

**Supervisor Call Address Table**

◆ The Supervisor Call Address table consists of halfwords, each having a one-bit indication of the state in which the call routine is to be executed, and the two byte address of the routine which resolves the call. The entries in the table are indexed by using the Call Byte of the Interrupt Status Register, into which the processor stores the second byte of the Executive Communication macro when the interrupt occurs.

**Console Control List**

◆ This list (see table 3-4) contains an eight-byte entry for each user program and for each Executive function that uses the Console Typewriter.

**Table 3-4. Console Control List**

| Byte | Bit | Meaning |
|---|---|---|
| 0 | 0 | Order Not Issued, or Termination Not Received. |
| | 1 | Reply Pending; that is, Operator Response Not Received. |
| | 2-7 | Reserved. |
| 1 | 0 | User has OP List Entry. |
| | 1 | Return to User when Request is Satisfied. |
| | 2 | 1 = User Message, 0 = Exec Message. |
| | 3 | Return to CIA on Termination. |
| | 4 | Do Not Accept Program Number of Zero. |
| | 5-7 | Reserved. |
| 2-3 | | Program Number (Shifted Left Five Bits). |
| 4-7 | | Address of Specification Packet. |

**Console Control List** *(Cont'd)*

The 16 entries for the Console Control List are as follows:

1. User Program 1 (1).
2. User Program 2 (2).
3. User Program 3 (3).
4. User Program 4 (4).
5. User Program 5 (5).
6. User Program 6 (6).
7. Monitor (U).
8. Executive Overlay (Y).
9. Load Program Overlay Function (Program Control) (V).
10. User Error Recovery Overlay (X).
11. CUP (6).
12. CUP Error Recovery Overlay (W).
13. MCP (7).
14. MCP Error Recovery Overlay (R).
15. Resident Executive (T). - No OP List Entry.
16. Console Interrupt Request (Z).

**Program Table**

◆ The Program Table (see table 3-5) contains one 32-byte entry for each user program and each Executive function that can call on Program Control.

**Table 3-5. Program Table**

| Byte | Bit | Meaning |
|---|---|---|
| 0 | | Program Number. |
| 1 | | Exec Overlay Number. |
| 2-7 | | Program Name. |
| 8 | | Program Priority (Shifted Left 3 Bits). |
| 9 | 0 | Monitor Subprocessor or Program Entry Flag. |
| | 1 | Run-Time Parameters Successor Call Presence Flag. |
| | 2-3 | Reserved. |
| | 4-7 | Protection Key. |
| 10-14 | | Disc Address of Load Directory. |
| 15 | | SVC Code for Program Control. |
| 16-19 | | Address of LPOV Specification Packet. |

**Program Table** *(Cont'd)*

**Table 3-5. Program Table** *(Cont'd)*

| Byte | Bit | Meaning |
|---|---|---|
| 20-23 | | Program End Address. |
| 24-27 | | Address of Executive Storage Area. (Zeroed out at program termination.) |
| 28-29 | | Address of next Device List Entry to be Quieted During Checkpoint-User Program Entries Only. |
| 30-31 | | Load Library Pointer. |

The 9 entries for the Program Table are as follows:

1. User Program 1.
2. User Program 2.
3. User Program 3.
4. User Program 4.
5. User Program 5.
6. User Program 6 or CUP. CUP is always assigned Priority 11.
7. MCP. MCP is always assigned Priority 13.
8. Monitor. Monitor is always assigned Priority 7.
9. Operator Command (Key-In) Contains Only First 8 Bytes of an entry.

**Load Library Table**

◆ This table (see table 3-6) contains five 10-byte entries in which the current status of each of the (maximum of) five Load devices is recorded.

**Table 3-6. Load Library Table**

| Byte | Meaning |
|---|---|
| 0-1 | Installation Mnemonic Device Name. |
| 2-7 | Disc Address of Program Directory or Last Encountered User Program Name. |
| 8-9 | Assignment Halfword. |

The following five Load Library Table entries are:

1. Random Access System Device.
2. Alternate Random Access Device.
3. Alternate Random Access Device
4. Alternate Tape Loading Device.
5. Compile-and-go Device.

**Free Memory Table**

◆ Program Control uses the Free Memory Table to record the boundaries of unassigned segments of memory. The table has nine entries, the last a sentinel, the other eight to accommodate the most fragmented arrangement of memory (i.e., five user programs MCP, and CUP, contiguous with neither each other nor the limits of user memory). Each entry contains either two fullword addresses or zero, except the last, which is always zero. The presence of zero in any entry indicates that the rest of the table is also zero. In a nonzero entry, the first fullword contains the address of the first byte of a 2,048-byte unassigned memory unit (i.e., the rightmost 11 bits are zeros). The second fullword containes the address of the last byte of that 2,048-byte unit, or the last of a series of unassigned units contiguous with that addressed in the first fullword of the entry (i.e., the rightmost 11 bits in the second fullword are always ones).

**Executive Key-In Area**

◆ The Key-In Area is an 80-byte storage area into which Console Control puts Operator Requests for the Executive and the Monitor. When the first byte of the area is zero it is considered "empty"; otherwise, it is considered "full".

When an operator command E INT is for the MCP, control is transferred to CIA which moves the message from the Executive Key-In Area to a Communications Key-In Area.

**Executive Storage Area**

◆ The 184-byte Executive Storage Area (see table 3-7) is prefixed immediately to the user program by Program Control when the program is initiated. It is the area where the Executive *records* and *accesses* information peculiar to the program when it is interrupted.

*RESTRICTION:*

The user cannot use Checkpoint while in the unrecoverable STXIT routine. The STXIT operator communication operand (E INT) will be ignored, if Checkpoint is in progress. After the Checkpoint has completed, the E INT command can be retried.

**Table 3-7. Executive Storage Area**

| Byte | Meaning |
|---|---|
| 0-3 | $P_1$ Interrupt Mask Register. |
| 4-7 | $P_1$ Interrupt Status Register. |
| 8-11 | $P_1$ Program Counter. |
| 12-15 | General Register 0 |
| 16-19 | General Register 1 |
| 20-23 | General Register 2 |
| 24-27 | General Register 3 |
| 28-31 | General Register 4 |

**Executive Storage Area** *(Cont'd)*

**Table 3-7. Executive Storage Area** *(Cont'd)*

| Byte | Meaning |
|---|---|
| 32-35 | General Register 5. |
| 36-39 | General Register 6. |
| 40-43 | General Register 7. |
| 44-47 | General Register 8. |
| 48-51 | General Register 9. |
| 52-55 | General Register 10. |
| 56-59 | General Register 11. |
| 60-63 | General Register 12. |
| 64-67 | General Register 13. |
| 68-71 | General Register 14. |
| 72-75 | General Register 15. |
| 76-83 | Floating-Point Register 0. |
| 84-91 | Floating-Point Register 2. |
| 92-99 | Floating-Point Register 4. |
| 100-107 | Floating-Point Register 6. |
| 108-111 | Program Base Address (beginning address) + Program Size (calculated by Linkage Editor) -1. |
| 112-115 | Address of Run-Time Parameter Area. |
| 116-119 | Interrupt Weight Code (from GR15P3). |
| 120-123 | Address of Program Check Routine (STXIT). |
| 124-127 | $P_1$ Program Counter stored when program check interrupt occurs. |
| 128-131 | $P_1$ General Register 10 stored when a program check interrupt occurs. |
| 132-135 | $P_1$ General Register 11 stored when a program check interrupt occurs. |

**Executive Storage Area** *(Cont'd)*

**Table 3-7. Executive Storage Area** *(Cont'd)*

| Byte | Meaning |
|---|---|
| 136-139 | Address of Interval Timer Routine (STXIT). |
| 140-143 | $P_1$ Program Counter Stored When an Interval Timer Interrupt Occurs. |
| 144-147 | $P_1$ General Register 10 Stored When an Interval Timer Interrupt Occurs. |
| 148-151 | $P_1$ General Register 11 Stored When an Interval Timer Interrupt Occurs. |
| 152-155 | Address of Operator Communication Routine (STXIT). |
| 156-159 | $P_1$ Program Counter Stored When an Operator Communication Interrupt Occurs. |
| 160-163 | $P_1$ General Register 10 Stored When an Operator Communication Interrupt Occurs. |
| 164-167 | $P_1$ General Register 11 Stored When an Operator Communication Interrupt Occurs. |
| 168-171 | Address of Unrecoverable Error Routine (STXIT). |
| 172-175 | $P_1$ Program Counter Stored When an Unrecoverable Error Interrupt Occurs. |
| 176-179 | $P_1$ General Register 10 Stored When an Unrecoverable Error Interrupt Occurs. |
| 180-183 | $P_1$ General Register 11 Stored When an Unrecoverable Error Interrupt Occurs. |

Bytes 172-183 are used by the Checkpoint routine when a Checkpoint is taken. They are redefined as follows:

| Byte | Meaning |
|---|---|
| 172 | Operation List entry byte 1 at Checkpoint. |
| 173-175 | Restart Address at Checkpoint. |
| 176-177 | Interrupt Flag Register (IFR) at Checkpoint. |
| 178-179 | Assigned Halfword of Checkpoint device. |
| 180 | Program Table byte 9 at Checkpoint. |
| 181-183 | CKPT Macro Expansion Address. |

**Run-Time Parameter Area**

◆ The Run-Time Parameter Area is set up by Program Control and contains the packed data from control cards supplied at program initiation. The beginning address of the Run-Time Parameter Area is stored in the Executive Storage Area. The end address of the area is the Program End Address, which is stored in the Program Table.

The Run-Time Parameter Area has two sections:

1. *A series of fullword absolute addresses of the following stacks. (See table 3-8).*

**Table 3-8. Run-Time Parameter Area**

| Byte | Meaning |
|---|---|
| 0-3 | Device Assignments Parameter Address. |
| 4-7 | VOL-FILES-TPLAB Parameters Address. |
| 8-11 | Volume Displacement Parameters Address. |
| 12-15 | Successor Program Name Parameter Address. |
| 16-19 | Last Packed Parameter Byte + 1. |
| 20-23 | Reserved. |

2. *A succession of contiguous stacks of packed parameters.* There may be any number of each of the four types of parameters: Device Assignments, VOL-FILES-TPLAB, Volume Displacement, and Successor Program Name. If there are none of a given type, its stack address (see table 3-8) equals the next stack address.

**Table 3-9. Run-Time Parameter-Device Assignments**

| Byte | Bit | Meaning |
|---|---|---|
| 0-5 | | Symbolic Device Name. |
| *6 | 00 | Flag = 0 (if input format is POS X 'Cuu'). |
| | 1-3 | Ignored. |
| | 4-7 | Channel (from Cuu). |
| 7 | 0-3 | Unit (from Cuu). |
| | 4-7 | Unit (from Cuu). |
| **6-7 | | TDOS Installation Mnemonic Device Name. |
| 8-9 | | Control Code (two bytes from X'wc'). |

*POS/TOS/TDOS. **TOS/TDOS only.

**Run-Time Parameter Area**

**Table 3-10. Run-Time Parameter-VOL-FILES-TPLAB**

| Byte | Meaning |
|---|---|
| 0-1 | Number of Tape Marks to be skipped. |
| 2-8 | DTF Name. |
| 9-77 | TPLAB Information. |
| 78-83 | Reserved for FCP use. |

**Table 3-11. Run-Time Parameter — Volume Displacement (minimum of 61 bytes)**

| Byte | Meaning |
|---|---|
| 0-1 | Parameter Size (PS) - Binary Byte Count. |
| 2-8 | DTF Name. |
| 9-10 | Matrix Size (MS) - Binary Byte Count. |
| 11-(MS+10) | Extents Matrix. |
| (MS+11-(MS+54) | Label Information. |
| (MS+55-(MS+60) | First Volume Serial Number. |

*Note:*

The number of six-byte Serial Numbers is indicated by the Parameter Size (PS).

**Table 3-12. Run-Time Parameter — Successor Program Name**

| Byte | Meaning |
|---|---|
| 0-5 | Successor Program Name. |

*Note:*

The number of six-byte Successor Program Names can be calculated from the entry Last Packed Parameter Byte + 1 (see table 3-8).

**Command Control Block**

◆ The Command Control block is a 40-byte information area generated by the CCB macro instruction to serve as an interface between I/O Control and users of Physical I/O.

*Note:*

If the CCB is for a card punch, the first four bytes following the CCB must be the absolute address of the backup area (supplied by user). This area is used by error recovery.

For DXC devices the CCB is extended by an additional 16 bytes.

**Command Control Block** *(Cont'd)*

**Table 3-13. Command Control Block**

| Byte | Bit Position | Title |
|---|---|---|
| 0-5 | | Symbolic Device Name or Channel and Device Address for Communication Devices Used by MCP. |
| 6 | | Device Class. |
| | 00 | Magnetic Tape. |
| | 01 | Printer. |
| | 02 | Bill Feed Printer. |
| | 03 | Card Punch 234. |
| | 04 | Card Punch 236. |
| | 05 | Card Reader. |
| | 06 | Paper Tape Reader/Punch. |
| | 07 | Disc Storage Unit. |
| | 08 | Drum Storage Unit. |
| | 09 | Mass Storage Unit. |
| | 0B | Data Exchange Control. |
| | 0C | Communication Device. |
| | 0D | Switch Controller. |
| | 0E | 70/567 Drum Storage Unit. |
| | FF | Accept Assignment of any Device Class. |
| 7 | | User Flag Byte. |
| | 0-1 | Type of I/O Operation:<br>00 = Start I/O.<br>01 = Test Device.<br>10 = Halt I/O for Communication Devices Only. |
| | 2 | Bypass Device Error Recovery. |
| | 3 | Accept Unrecoverable Error. |
| | 4 | Optional File. |
| | 5 | Inhibit Error Recovery Messages. |
| | 6 | Card Punch Recovery Flag. |
| | 7 | Reserved-Must be Zeros. |
| 8-11 | | Address of first Channel Command Word (CCW). |
| 12-13 | | MCP Code Passed to CIA. |
| 14-15 | | Assignment Halfword (Points to Device List Entry for this Device) 0000 = unassigned. |

**Command Control Block** *(Cont'd)*

**Table 3-13. Command Control Block** *(Cont'd)*

| Byte | Bit Position | Title |
|---|---|---|
| 16 | | Device Number (from CAR) (Cleared to Binary O at I/O Initiation). |
| 17-19 | | Address of Next CCW (from CAR).<br>If byte 17 is ($FF_{16}$), the E BSY console routine found this CCB to have an I/O outstanding. |
| 20 | | Flags (from CCR-II). |
| | 0 | Data Chaining. |
| | 1 | Command Chaining. |
| | 2 | Suppress Length Indicator. |
| | 3 | Skip. |
| | 4 | Program Controlled Interrupt. |
| | 5-7 | Zeroes. |
| 21 | | Channel Status Byte (from CCR-II). |
| | 0 | PCI Interrupt. |
| | 1 | Incorrect Length. |
| | 2 | Program Check. |
| | 3 | Protection Check. |
| | 4 | Channel Data Check. |
| | 5 | Channel Control Check. |
| | 6 | Reserved for Processor Use. |
| | 7 | Terminating Interrupt. |
| 22-23 | | Remaining Byte Count (from CCR-II). |
| 24 | | Command Code (from CCR-I). |
| | 0-3 | Zeros. |
| | 4-7 | Command Code:<br>01 = Sense.<br>02 = Read Reverse.<br>03 = Write.<br>04 = Write Erase.<br>05 = Read.<br>07 = Write Control.<br>09 = Transfer In Channel. |
| 25-27 | | Data Address of first byte or location of new CCW if command is TIC (from CCR-I). |
| 28-30 | | Data byte - 70/55 only (from Assembly/Status Register). |

**Command Control Block** *(Cont'd)*

**Table 3-13. Command Control Block** *(Cont'd)*

| Byte | Bit Position | Title |
|---|---|---|
| 31 | | Standard Device Byte (from Assembly/Status Register). |
| | 0 | External Device Request Pending. |
| | 1 | Termination Interrupt Pending. |
| | 2 | Device Busy. |
| | 3 | Control Busy. |
| | 4 | Device End. |
| | 5 | Secondary Indicator. |
| | 6 | Inoperable. |
| | 7 | Status Modifier. |
| 32-34 | | Device Sense Bytes. |
| 35 | | Executive Flag Byte: |
| | 0 | Termination (initially = 1):<br>Wait Issued and I/O Not Terminated. |
| | 1 | Abnormal Condition at Termination. |
| | 2 | Unrecoverable Error. |
| | 3 | Program Controlled Interrupt (PCI). |
| | 4-5 | Condition Codes:<br>0 = 00<br>1 = 01<br>2 = 10<br>3 = 11 |
| | 6 | Sense Information Lost. |
| | 7 | No File.<br>1 - If Optional file is not present.<br>0 - If optional file is present. |
| 36-39 | | Channel Address Register if PCI Occurs. |

**Command Control Block Extension (DXC)**

◆ The CCB extension is an additional 16 byte entry that is generated onto the CCB for the DXC. The format of the extension is shown in table 3-14.

**Table 3-13. Command Control Block Extension**

| Byte | Bit | Meaning |
|---|---|---|
| 40-43 | | Address of user-supplied routine to handle a DXC interrupt. |
| 44-47 | | Saved return address at DXC interrupt. |
| 48-51 | | Saved GR10 at DXC interrupt. |
| 52-55 | | Saved GR11 at DXC interrupt. |

**Command Control Block Extension (Card Punch)**

◆ The CCB extension is an additional four-byte entry that is furnished by the user. It immediately follows the CCB generated by the CCB macro instruction.

**Table 3-15. CommandControl Block Extension (Card Punch)**

| Byte | Bit | Meaning |
|---|---|---|
| 40-43 | | Absolute Address of Backup Area for Card Punch Using Physical Level I/O. |

**Direct Command Control Block**

◆ The Direct Command Control Block is a 20-byte area generated by the DCCB macro instruction to serve as an interface between the Executive routine handling the Direct Control feature and the user program handling a DC external interrupt. The area is shown in table 3-16.

**Table 3-16. Direct Command Control Block**

| Byte | Bit | Meaning |
|---|---|---|
| 0 | | DC trunk handled. |
| 1 | | Used to store input byte. |
| 2-3 | | Reserved. |
| 4-7 | | Address of user DC routine. |
| 8-11 | | Saved returned address at DC interrupt. |
| 12-15 | | Saved GR10 at DC interrupt. |
| 16-19 | | Saved GR11 at DC interrupt. |

**Executive Device List**

◆ The Device List is built by the System Generator and contains one 32-byte entry for each device, except communication devices connected to the Processor. Only one entry exists for all communications devices with channel and device addresses undefined. Bytes 0-6, 19, 28-31 are supplied by the System Generator, bytes 8-18 by Device Assignment, and bytes 20-23 by I/O Control. For each random access device, an extension is generated containing an eight-byte entry per program for all seven programs (including MCP) plus another eight-byte entry for Monitor if the Random Access is specified.

**Table 3-17. Executive Device List**

| Byte | Bit | Meaning |
|---|---|---|
| 0-1 | | Installation Mnemonic Device Name. |
| 2 | | Device type (hexadecimal codes).<br>01 = 70/242 Printer.<br>02 = 70/243 Printer.<br>03 = 70/248 Printer.<br>04 = 70/234 Card Punch.<br>05 = 70/236-10,-11 Card Punch. IBM 1402, IBM 2540.<br>06 = 70/237 Card Reader.<br>07 = 70/221 Paper Tape Reader/Punch.<br>0A = Magnetic Tape which uses Write Control Code.<br>0B = 9-Channel Magnetic Tape.<br>0C = 70/564 Disc Storage Unit.<br>0D = 70/565 Drum Storage Unit: 32 cylinders.<br>1D = 70/565 Drum Storage Unit: 64 cylinders.<br>2D = 70/565 Drum Storage Unit: 96 cylinders.<br>3D = 70/565 Drum Storage Unit: 128 cylinders.<br>4D = 70/565 Drum Storage Unit: 160 cylinders.<br>5D = 70/565 Drum Storage Unit: 192 cylinders.<br>6D = 70/565 Drum Storage Unit: 224 cylinders.<br>7D = 70/565 Drum Storage Unit: 256 cylinders.<br>8D = 70/567 Drum Storage Unit: 100 cylinders.<br>9D = 70/567 Drum Storage Unit: 200 cylinders.<br>0E = Reserved.<br>0F = 70/568 Mass Storage Unit.<br>10 = 70/627 Data Exchange Control.<br>11 = 70/668 Communication Control Multi-channel. |

**Executive Device List** *(Cont'd)*

**Table 3-17. Executive Device List** *(Cont'd)*

| | Byte | Bit | Meaning |
|---|---|---|---|
| | | | 12 = Switch Controller.<br>15 = 70/236 Card Punch (Models 20 and 21). |
| | 3 | | Device Class (see CCB, byte 6). |
| | 4 | | Channel Address. |
| | 5 | | Co-Channel Address (FF if no co-channel). |
| | 6 | | Device Address. |
| | 7 | | Channel Just Used (Shifted Left 3 Bits). |
| * | 8-13 | | Symbolic Device Name. |
| * | 14 | | Program Number. |
| * | 15 | 0-3 | Reserved. |
| | | 4-7 | Protection Key. |
| * | 16-17 | | Priority (Shifted Left 3 Bits). |
| | 18 | | Write Control Code (hexadecimal) for 7-channel tapes only: |

| Code | Density | Parity | Pack | Translate |
|---|---|---|---|---|
| 60 | 200 | Odd | Off | Off |
| A0 | 556 | Odd | Off | Off |
| E0 | 800 | Odd | Off | Off |
| 40 | 200 | Even | Off | Off |
| 80 | 556 | Even | Off | Off |
| C0 | 800 | Even | Off | Off |
| 68 | 200 | Odd | Off | On |
| A8 | 556 | Odd | Off | On |
| E8 | 800 | Odd | Off | On |
| 48 | 200 | Even | Off | On |
| 88 | 556 | Even | Off | On |
| C8 | 800 | Even | Off | On |
| 70 | 200 | Odd | On | Off |
| B0 | 556 | Odd | On | Off |
| F0 | 800 | Odd | On | Off |

*Unused for shared devices.

**Executive Device List** *(Cont'd)*

**Table 3-17. Executive Device List** *(Cont'd)*

| Byte | Bit | Meaning |
|---|---|---|
| 19 | 0 | Reserved. |
| | 1 | Request Queued, or Executed. |
| | 2 | Termination Expected. Manual Request Expected (DXC only). |
| | 3 | Sense Command Issued. |
| | 4 | Error Recovery in Progress. |
| | 5 | Random Access Device. Shared Device = 1, Non-Shared Device = 0. |
| | 6 | Manual Request. |
| | 7 | Purge Bit or Manual Request Handled (for DXC only). |
| 20-23 | | CCB Address (Not Used for Shared Random Access Devices). |
| 24-25 | | Error Count (Binary Count). |
| 26 | | Number of the Program Awaiting this Device. |
| 27 | | Write Control Code for Awaited Device. |
| 28 | 0-1 | 00 = 7-channel Tape (other than below).<br>01 = 581 Tape.<br>10 = 441 Tape.<br>11 = 1600 BPI Tape. |
| | 2-7 | Address of TDOS Device List Extension -24 (Shared Random Access Device only). |
| 29 | | |
| 30-31 | | Address of CCB Address -20 Pending. |

**Executive Device List Extension (Random Access Only)**

◆ The Device List Extension consists of a 56-byte area for each Random Access device. Each 56-byte area is made up of seven 8-byte entries of information for each program. The extension is set to zero by the System Generator. One additional 8-byte entry (64-byte area) is generated when the Monitor Random Access option is specified.

**Table 3-18. Device List Extension**

| Byte | Bit | Meaning |
|---|---|---|
| 0-1 | | Priority (Pointing to proper OP List Entry). (Shifted Left 3 Bits). |
| 2 | 0 | Zero. |
| | 1 | Set to 1 for Extension for a Device List Entry. |
| | 2 | Zero. |

**Executive Device List Extension (Random Access Only)** *(Cont'd)*

**Table 3-18. Device List Extension** *(Cont'd)*

| Byte | Bit | Meaning |
|---|---|---|
| | 3 | Reserved. |
| | 4-7 | Protection Key. |
| 3 | 0 | Waiting for I/O Termination. |
| | 1 | Request Queued or Executed. |
| | 2 | Termination Expected. |
| | 3 | Sense Command Issued. |
| | 4 | Error Recovery in Process. |
| | 5 | Set 1 Indicating an Extension. |
| | 6-7 | Reserved. |
| 4-7 | | CCB Address. |

There are seven (or eight) Device List Extensions in the following order:

1. User Program 1
2. User Program 2
3. User Program 3
4. User Program 4
5. User Program 5
6. User Program 6
7. MCP
8. Monitor (optional if Random Access option specified)

**Direct Control Line Table**

◆ The Direct Control Line Table consists of six 4-byte entries which contain current information about each of the External Signal lines. The Direct Control Line Table is defined in table 3-19.

**Table 3-19. Direct Control Line Table**

| Byte | Bit | Meaning |
|---|---|---|
| 0 | | One byte for program number. |
| 1-3 | | Three bytes for address of Direct Control Block. |

**Direct Control Program Table** *(Cont'd)*

◆ The Direct Control Program Table contains a four-byte entry for each program. An entry is defined as shown in table 3-20.

**Table 3-20. Direct Control Program Table**

| Byte | Bit | Meaning |
|---|---|---|
| 0 | 0-5 | Lines in use (1 bit per line). |
| | 6-7 | Reserved. |
| 1 | | Queued out-line mask. |
| 2 | | Queued out-byte. |
| 3 | | Reserved. |

**Channel List**

◆ The Channel List contains an eight-byte entry for each selector channel that points to the last entry in the appropriate Channel Queue. The Channel List currently can be found at hexadecimal address 0002DC.

**Table 3-21. Channel List**

| Byte | Bit | Meaning |
|---|---|---|
| 0-1 | | Scratch Pad Address. |
| 2 | | DL Entry of the pending I/O. |
| 3 | 0-6 | DL Entry of the pending I/O. |
| | 7 | Normally set to zero. A one indicates that a sense for the device in the control channel (2) could not be fired because of a pending interrupt. Sense will be repeated after the pending interrupt terminates. The bit will then be reset to zero. |
| 4-5 | | Number of queued requests (shifted left 1 bit). |
| 6-7 | | Address of the entry in the queue which will be executed next. |

There are eight Channel List entries in the following order:

1. Multiplexor Channel (has no channel queue).
2. Selector Channel No. 1.
3. Selector Channel No. 2.
4. Selector Channel No. 3.
5. Selector Channel No. 4.
6. Selector Channel No. 5.
7. Reserved for Error Recovery.*
8. Selector Channel No. 6.

*Entry 7 is used to store certain information about the Error Recovery Interface Queue. The System Generator places the address of the ER Interface Queue into the third and fourth bytes of this entry.

**Channel Queue**

◆ The Channel Queue is built for each selector channel. The queue consists of a two-byte entry for each possible request to this channel. The size of the queue is calculated as follows:

2 (r + 7s + 5t) = number of bytes per queue list.

where: r = number of non Random Access devices.

s = number of Random access devices.

t = 0 if no Disc/Drum is connected.
1 if at least one Disc/Drum is connected.

Each two-byte entry is defined as follows:

| Byte | Bit | Meaning |
|---|---|---|
| 0 and 1 | 0-7 and 0-5 | Address of DL entry for non-shared random access devices.<br>Address of DL extension entry minus 16 for shared random access devices.<br>Address of the spec pack minus 16 for Executive requests.<br>All addresses are 14 bits in length. |
| 1 | 6-7 | 11 = MCP and MCP error recovery.<br>10 = CUP and CUP error recovery.<br>00 = the rest of the requests<br>User programs 1-5.<br>Error recovery 1-5.<br>Load User overlay function.<br>Executive functions. |

The queue is serviced by priority with MCP being the highest.

**Error Recovery Interface List**

◆ The Error Recovery Interface List contains the parameters for the three Error Recovery Overlay Areas and the Error Recovery Interface Queue.

**Error Recovery Interface Queue**

◆ The Error Recovery Interface Queue consists of two-byte entries for each possible request for error recovery for each of the three possible error Recovery overlays.

The size of the queue is calculated as follows:

2 (r + 7s + 1) = number of bytes per queue.

where: r – number of non RA devices.

s = number of RA devices.

Each entry has the same format as the entries in the channel queue.

**Area List**

◆ The Area List is used to store parameters pertaining to each of the possible four overlay areas. There is one entry for each of the following:

1. Executive Overlay Area.
2. User Error Recovery Overlay Area.
3. CUP Error Recovery Overlay Area.
4. MCP Error Recovery Overlay Area.

Each Area List entry consists of 20 bytes and is defined as follows:

**Table 3-22. Area List**

| | Byte | Bit | Meaning |
|---|---|---|---|
| | 0-1 | | Priority of the Area (shifted left three bits). |
| | 2-3 | | Reserved - '8004' hexadecimal - indicates call for Systems Overlays and Device List Extension. |
| | 4-7 | | Address of CCB used to load the overlay. |
| | 8-9 | | Address of CCB address minus 20, for which Error Recovery is being accomplished; or the Program Table Address of the user in control (Executive Area List entry only). |
| | 10-11 | | Entrance to Error Recovery overlay area. |
| | 12-13 | | Return Address (P1 Program Counter). |
| | 14 | | Error Recovery Error Counter. |
| | 15 | | Overlay Number to be loaded. |
| * | 16-17 | | Number of queued requests awaiting this Area List entry (shifted left one bit). |
| * | 18-19 | | Address of entry in Error Recovery Interface Queue to be executed next; that is, next function to use the Area List to load an overlay. |

*Bytes 16-19 are not used for Executive Overlay Area.

*Notes:*

When Error Recovery is entered (see bytes 10-11), General Purpose Register 1 of P2 State will contain the address of the appropriate Area List entry.

**On-Line Catalog**

◆ The On-Line Catalog is built by the System Generator and contains one 9-byte entry for each random access volume on the system.

**Table 3-23. On-Line Catalog**

| Byte | Meaning |
|---|---|
| 0 - 5 | Volume Serial Number |
| 6 - 7 | Device List Address |
| 8 | Bin Number of 70/568 |

**Program Directory**

◆ The Program Directory is a directory of all programs of a disc/drum which contains 10 entries per block with each having a six-byte key field indicating the program name of the last entry in the block.

**Program Directory**

| Byte | Bit | Meaning |
|---|---|---|
| 0-5 | | Program name - must be in ascending sequence. |
| 6-8 | | Initial load entry point. |
| 9-11 | | Maximum core requirements of the program. |
| 12-14 | | Minimum core requirements of the program. |
| 15-19 | | Disc address of Load Directory (CCHHR). |
| 20-21 | | Month. |
| 22-23 | | Day. |
| 24-25 | | Year. |
| 26-29 | | Version number of program. |

**Load Directory**

◆ The Load Directory is a directory of each load of a program on disc/drum which contains 14 byte entries for each load.

**Load Directory**

| Byte | Bit | Meaning |
|---|---|---|
| 0-5 | | Load Name. |
| 6-10 | | Disc address of load (CCHHR). |
| 11-13 | | Program relative load address of first text byte. |

# 4. COMMUNICATING WITH THE EXECUTIVE

◆ The Executive controls the overall operation of both users' programs and systems programs. To perform this control, the Executive responds to all Interrupts, controls and I/O devices, and provides a means of operator communication with the Executive.

## EXECUTIVE MACROS

◆ Macros are provided for communication between the user's program and the Executive, and also for addressing information in the Executive Tables. These macros are classified as physical I/O macros, used with physical level FCP and Executive Communication Macros. A detailed description of these macros can be found in the Spectra 70 TOS/TDOS FCP and Executive Communication Macros Manual. A summary of these macros is as follows:

**EXCP** ◆ Requests that an I/O operation be performed.

**EXCPW** ◆ Requests that an I/O operation be performed and that the calling program be "waited" until the I/O operation has completed.

**WAIT** ◆ Is issued when the program requires that a previously initiated I/O operation be completed before continuing.

**CHECK** ◆ Checks the status of an I/O operation.

**CPCI** ◆ Determines if a program-controlled interrupt has occurred during an I/O operation.

**CCW** ◆ Defines the I/O operation to be performed.

**TYPE** ◆ Enables the program to communicate with the operator via the 70/97 Console Typewriter.

**COMTY** ◆ Ensures that a previously issued typewriter request has been completed.

**LPOV** ◆ Is issued when the program requires that another program segment be loaded for execution.

**TERM** ◆ Is used at the normal completion of a program to initiate the termination of a program.

**TERMS** ◆ Is used at the completion of a program when a successor program is to be loaded and initiated.

**TERMD** ◆ Dumps the program to magnetic tape when abnormal termination is necessary.

**CKPT** ◆ Enables the program to write checkpoint records to a magnetic tape. (See note below.)

**STXIT** ◆ Enables the program to specify to the Executive the addresses of one or more of the program's interrupt routines. (See note below.)

*Note:*

The user cannot use Checkpoint while in the unrecoverable STXIT routine. The STXIT operator communication operand (E INT) will be ignored if Checkpoint is in progress. At the time Checkpoint has completed, the E INT command may then be retried.

| | |
|---|---|
| **EXIT** | ◆ Is issued at the end of a program's interrupt routine to return to the point in the program where the interrupt occurred. |
| **ADEXT** | ◆ Enables the program to access information stored in the Executive Communication Region and the Program Table entry for this program. |
| **ASSGN** | ◆ Enables the program to assign a device without an access to that device. |
| **DMODE** | ◆ Enables the program to store the 7-channel tape control byte for a given symbolic unit. |
| **SMODE** | ◆ Enables the program to change the mode of a 7-channel tape. |
| **DTYPE** | ◆ Enables the program to store the device type for a given symbolic unit. |
| **DDEV** | ◆ Returns a device to the system. |
| **ASCII** | ◆ Enables the program to set the machine code to USASCII. |
| **EBCD** | ◆ Enables the program to set the machine code to EBCDIC. |
| **SETIC** | ◆ Enables the program to set the time clock. |
| **GETOD** | ◆ Enables the program to get the current time of day. |
| **GEPRT** | ◆ Enables the program to get the elapsed program run time |
| **QUIET** | ◆ Causes all outstanding I/O operations to be completed before continuing. |
| **LPOVR** | ◆ Is issued when the program requires that another program segment be loaded and control be returned to the program before the segment is actually loaded. |
| **FLOAT** | ◆ Is issued when the program requires that another program segment be loaded into an absolute memory location for execution. |
| **FLODR** | ◆ Is issued when the program requires that another program segment be loaded into an absolute memory location and control be returned to the program before the segment is actually loaded. |
| **WTOV** | ◆ Is issued to wait a program until an overlay has been loaded. |
| **STDXC** | ◆ Notifies the Executive of the address of the users DXC routine. |
| **DXCXT** | ◆ Is issued at the end of the users DXC routine to return control to the point in the program where the interrupt occurred. |
| **SETDC** | ◆ Is issued to assign a Direct Control trunk to a program. |
| **RELDC** | ◆ Is issued to release a Direct Control trunk from a program. |
| **WRTDC** | ◆ Is issued to transmit a byte of information over the Direct Control. |
| **DCWT** | ◆ Is issued to notify the Executive that a Direct Control write is complete. |
| **DCXT** | ◆ Is issued at the end of the users Direct Control routine to return control to the point in the program where the interrupt occurred. |
| **UPR** | ◆ Enables a program to relinquish control to the program below it in priority without waiting itself. |
| **LOADP** | ◆ Permits a program to initiate another program internally without being terminated itself. |

**CMCCM** ◆ Loads a CCM memory.

**CMINT** ◆ Initializes communication by MCP.

**CMGET** ◆ GETS a communication message.

**CMPUT** ◆ PUTS a communication message.

**PR** ◆ Waits a communication user program with nothing to do and gives up control to the highest program in the program mix.

**TOCOM** ◆ Moves data to a common data area.

**EXCOM** ◆ Retrieves data from a common data area.

## EXECUTIVE CONSOLE ROUTINES

◆ The operator communicates with the Executive by means of console routines through the console typewriter. The applicable function is initiated after a console request is issued. The routines are an external means of communicating with the Executive to request an action to be performed.

A functional description of each console routine is described in this section. Further information relating to the operating procedures and message formats for each of these routines are presented in the TDOS Operators' Guide. Table 4-1 summarizes the Executive Console Routines.

**Table 4-1. Executive Console Routines**

| Console Routines | Function |
|---|---|
| E LOD | Load Program. |
| E RST | Restart Program. |
| E HLT | Terminate Program. |
| E DDV | Deallocate Device. |
| E PRY | Change Priority. |
| E DUM | Dump and Terminate Program. |
| E INT | Interrupt Program. |
| E DDA | Display Device Assignments. |
| E DMA | Display Memory Assignments. |
| E MIX | Display Program Mix. |
| E MEM | Display Free Memory. |
| E DUD | Display Unassigned Devices. |
| E OLC | Update On-Line Catalog. |
| E KIL | Terminate Successor Chain. |
| E CNG | Change Memory Requirement. |
| E MCP | Load Multichannel Communications Program. |
| E CUP | Load Communications User Program. |
| E BSY | Test Busy Status of a Device. |

**E LOD (Load Program)**

◆ The Load Program console routine initiates the loading and execution of a program. It may also inform the Executive of the priority of the program, the device on which it is located, the device from which run-time parameters are to be read, the relative position in memory it is to be loaded, and the decimal number of bytes to be assigned to the program.

**E RST (Restart Program)**

◆ The Restart Program console routine initiates the restart of a previously checkpointed program. The program name, priority, and device on which the program is located must be supplied.

**E HLT (Terminate Program)**

◆ The Terminate Program console routine terminates the specified program, by specifying its program number.

**E DDV (Deallocate Device)**

◆ The Deallocate Device console routine deallocates any device assigned to a user's program. The device to be deallocated is specified by its installation mnemonic and is quieted before deallocation.

**E PRY (Change Priority)**

◆ The Change Priority console routine changes the priority of a user's program. The designated programs priority is changed to 6, the highest priority. If necessary, the other programs in the Operation List are pushed down, in order to change the designated program to a 6 priority. The program number is indicated.

**E DUM (Dump and Terminate Program)**

◆ The Dump and Terminate console routine causes the specified program and its Executive Storage Area (ESA) to be dumped and the program to be terminated. The program number is indicated.

**E INT (Interrupt Program)**

◆ The Interrupt Program console routine causes the specified program to jump to the Operator Communication subroutine specified in the program Set Contigency Routine Address (STXIT-OC) macro. The program number of the program to be interrupted must be indicated. If this command is for MCP, information may be appended to the E INT command to be passed to MCP. The program must issue the STXIT macro prior to using the Interrupt Program console routine. The STXIT operator communication operand (E INT) will be ignored if Checkpoint is in progress. At the time Checkpoint has completed, the E INT command can be retried.

**E DDA (Display Device Assignments)**

◆ The Display Device Assignment console routine displays all devices assigned to the program. The program number must be supplied.

**E DMA (Display Memory Assignments)**

◆ The Display Memory Assignment console routine displays the low and high boundaries of the program including the Executive Storage Area and Run Time Parameter storage. The program number must be supplied.

**E MIX (Display Program Mix)**

◆ The Display Program Mix console routine displays the program names, up to six, of the programs currently running. The format given is program number, program name, priority, and an * if the program is running under Monitor control.

**E MEM (Display Free Memory)**

◆ The Display Free Memory console routine displays the low and high order boundaries of contiguous segments of unassigned memory. The low-order memory address is the first byte of a 2,048-byte unassigned memory unit and the high order address is the last byte of the 2,048 unit or a multiple of it.

**E CNG (Change Memory Requirements)**

◆ The Change Memory Requirements console routine increases or decreases the amount of memory assigned to a program. Changes are in decimal and are preceded by a + or - sign. The program number must be supplied.

**E DUD (Display Unassigned Devices)**

◆ The Display Unassigned Devices console routine displays the installation mnemonic device name for all unassigned devices on the system. Shared random access devices are always designated as unassigned devices.

**E OLC (Update On-Line Catalog)**

◆ The Update On-Line Catalog console routine updates the On-Line Catalog by posting the serial numbers of the random access volumes that are on line at the current time. Whenever the system is initialized, or random access volumes are changed the On-Line Catalog must be updated with this console routine.

**E KIL (Terminate Successor Chain)**

◆ The Terminate Successor Chain console routine cancels all unstarted job orders in the program's run-time parameters. The program currently running runs to completion. Identification is accomplished by specifying program number.

**E MCP (Load Multichannel Program)**

◆ This operator command is identical to E LOD; however, the program to be loaded is a Multichannel Communications program that has a fixed entry in the program table and a fixed priority in the Operations List. The Executive verifies that the Communications Interrupt Analysis routine is present within the Executive. If not, the operator command is rejected.

**E CUP (Load Communication User Program)**

◆ This operator command is identical to E LOD; however, the program to be loaded is a Communications User Program which has a fixed entry in the program table and a fixed priority in the Operations List. The Executive verifies that a Multichannel Communications Program has been previously loaded, if not, the operator command is rejected.

**E BSY MN (Device Status)**

◆ The Device Status console routine requests the status of the indicated mnemonic device. When this device has no outstanding I/O, the following message is typed:

Y MN NOT BUSY

If there is an outstanding I/O for this device, an indicator ($FF_{16}$) will be set in byte 17 of the CCB pending, and the following message will be typed:

*****Y X RSEXEC MN IS BUSY

The operator may ignore or terminate the device from the console. When the operator terminates the device and the I/O terminates normally, the indicator is cleared and the 'NOT BUSY' typeout is given.

*Note:*

An outstanding I/O is one for which no termination has been received.

**E BSY MN (Device Status)** *(Cont'd)*

If the mnemonic is omitted from the E BSY typein, the count of unidentifiable I/O interrupts received by the Executive will be typed. Hardware information (CAR, CSB, and SDB) relating to the last nine of these interrupts also will be typed. After typing their contents, the count and the storage area will be cleared to zero.

The format of this typeout is:

Y RSEXEC NNNN CCCCCCCCAABB$_1$ CCCCCCCCAABB$_2$
...CCCCCCCCAABB$_9$

where:

NNNN - Indicates the count of unidentifiable interrupts received.

CCCCCCCCAABB$_n$ - Indicates the CAR, CSB, SDB for the nth most recent unidentifiable interrupts received.

## EXECUTIVE RUN-TIME PARAMETERS

◆ The Executive run-time parameters are a set of object time parameters that the Executive accepts and supplies to a user's program. These parameters are optional and only accepted if the E LOD program console routine indicates that they are present. All of the Executive run-time parameters follow these general rules.

The first two characters of a parameter are two slashes in the first two positions which identify the parameter as an Executive control parameter. At least one space must follow the second slash. The control identifier follows and indicates the type of Executive run-time parameter. This identifier is a maximum of six characters in length and is followed by at least one space. The required operands follow with commas separating all operands. Operands may not be split over two input parameters. More than one control statement of the same type may be written. The last operand is followed by a space or is indicated by the end of the input parameter. The first space in the operand field indicates the end of all parameters in this statement.

The parameters must appear in groups. The valid types of parameters and the order in which the groups must appear are as follows:

1. ASSGN
2. { FILES, VOL, TPLAB }
3. VDC
4. JOB
5. END

The FILES, VOL, and TPLAB must appear together for each file defined. They also must appear each time the file is opened.

**ASSGN**

◆ This parameter may be used to assign a device to a user's program. The actual device indicated in this parameter is assigned to the symbolic device name in the user's program. Any or all of the devices for a given program or succession of programs may be assigned in this way. An ASSGN card will be used as often as requested by any program in a job stream (that is, once used, it is not deleted). Identical symbolic device names within the same job stream cannot be assigned to different devices.

The format of the ASSGN parameter is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | ASSGN | Symbolic, $\left\{\begin{matrix}\text{mn}\\ \text{X'cuu'}\end{matrix}\right\}$ [,[ dt ][ ,X'wc']] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - ASSGN.

*Operand Field* -

Symbolic - The symbolic device name, one to six characters in length, assigned to the device.

mn - Two-character installation mnemonic device name to be associated with the symbolic.

X'cuu' - Indicates the device to be assigned by channel and unit in hexadecimal.

c = 0 - multiplexor channel.

1 - selector channel 1.

2 - selector channel 2.

3 - selector channel 3.

4 - selector channel 4.

5 - selector channel 5.

7 - selector channel 6.

uu = 00 to FF indicating the unit.

dt - This entry is present for compatibility. It is ignored by TDOS.

X'wc' - Indicates applicable write control information used for 7-level tapes.

**ASSGN** *(Cont'd)*

| wc | Density bytes/in. | Parity | Pack/unpack mode | Translate mode |
|---|---|---|---|---|
| 60 | 200 | ODD | OFF | OFF |
| A0 | 556 | ODD | OFF | OFF |
| E0 | 800 | ODD | OFF | OFF |
| 40 | 200 | EVEN | OFF | OFF |
| 80 | 556 | EVEN | OFF | OFF |
| C0 | 800 | EVEN | OFF | OFF |
| 68 | 200 | ODD | OFF | ON |
| A8 | 556 | ODD | OFF | ON |
| E8 | 800 | ODD | OFF | ON |
| 48 | 200 | EVEN | OFF | ON |
| 88 | 556 | EVEN | OFF | ON |
| C8 | 800 | EVEN | OFF | ON |
| 70 | 200 | ODD | ON | OFF |
| B0 | 556 | ODD | ON | OFF |
| F0 | 800 | ODD | ON | OFF |

**FILES**

◆ This parameter may be used to position tape files. This entry overrides the MRKCTR entry in the DTFSR.

The format of the FILES parameter is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | FILES | Symbolic,n |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - FILES.

*Operand Field* -

Symbolic - The symbolic device name, one to six characters, assigned to this tape file.

n - The decimal number of tape marks to be skipped (from present position) 1-9999.

**VOL**

◆ This parameter is for checking or writing standard labels for a tape file. It associates the symbolic device name with the file name. It is rejected if it follows a FILES card and their symbolic device names do not agree.

The format of the VOL parameter is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | VOL | Symbolic,fffffff |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - VOL.

*Operand Field* -

Symbolic - The symbolic device name, one to six characters, assigned to this tape file.

fffffff - The file name, one to seven characters, which is TAG used to identify this CCB or DTF macro.

**TPLAB**

◆ This parameter contains the label information for label checking and writing. This parameter must immediately follow the volume (VOL) entry that it is associated with. File labels may require two parameters, the second being a continuation parameter. Label fields 3-10 are always written just as they appear in the label. These are the only fields used for label checking purposes. The additional fields (11-13) may be included; if the file is output they will be written in the output label, otherwise they are ignored.

The format of the TPLAB parameter is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | TPLAB | 'L_ _ _ _ _L' |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - TPLAB.

*Operand Field* -

'L_ _ _L' = Input Files. A 49-byte character string, written within apostrophes, identical to fields 3-10 of input files HDR1 label.

'L_ _ _L' = Output Files. Fields 3-13, 69-byte character string, written within apostrophes. If these fields are too long to be written in a single parameter the character string must extend to position 71 with a continuation (nonblank) punch in position 72. The string is completed on the continuation parameter (beginning in position 16). On the continuation parameter, positions 1 to 15 are blank. When the string has less than 69 characters, it is space filled to the right. When the string contains more than 69 characters, only the first 69 are accepted.

Information required for processing labels during program execution is specified by the two parameters just specified: VOL and TPLAB, Edit Run-Time Parameters convert this information into a convenient form for the label routines and stores it in upper memory for their use. A total of 83 bytes in memory is reserved for each set of VOL and TPLAB parameters.

**Fields 3-13 of the Standard Header Label**

| Field | Description | Card Columns |
|---|---|---|
| 3 | File Identifier | 17 |
| 4 | File Serial Number | 6 |
| 5 | Volume Sequence Number | 4 |
| 6 | File Sequence Number | 4 |
| 7 | Generation Number | 4 |
| 8 | Version Number of Generation | 2 |
| 9 | Creation Date | 6 |
| 10 | Expiration Date | 6 |
| 11 | File Security | 1 |
| 12 | Block Count | 6 |
| 13 | Reserved | 13 |

**VDC**

◆ This parameter defines the volume serial numbers assigned to a direct access file. This entry is required for direct access files.

The format of the VDC parameter is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | VDC | Filename, Matrix, File ID, Serial 1, Serial n |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - VDC.

*Operand Field* -

Filename - The one-to seven-character name of the DTFSR, or DTFDA.

Matrix - The number (up to four decimal digits) of bytes to be allocated for the file extent matrix if it was not specified at assembly time or an override of the size specified at assembly time is desired. When packed, the specified number of bytes is preceeded by two bytes containing the binary equivalent of the value specified in the matrix; if matrix is omitted, these two bytes are set to zeros.

FILE ID - The 44-character name in the file label which, when less than the maximum of 44 characters, is left justified and spaced filled. If more than 44 characters appear, only the first 44 will be accepted.

Serial 1 - Serial 1 thru Serial n are the six character volume serial numbers belonging to this file.

Serial n - A volume serial number must be included for each volume that is to be open. If the volume serial numbers require more than one card, a continuation (nonblank) character is punched in column 72 and the number is continued in column 16 of the next card.

**JOB**

◆ A succession of programs will be automatically executed if a LOD operator command specifies run time parameters, and there is at least one JOB card in the parameters. All jobs must exist on the original LOD device. Once the program specified in the LOD command has been executed it will be succeeded by all of the programs called by JOB cards (in the order in which they appear in the parameters); and only those programs, regardless of how any program in the succession terminates, unless the KIL operator command is used to stop the succession. A JOB card will override a TERMS command.

The format of the JOB entry is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | JOB | Progname |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - JOB.

*Operand Field* -

Progname - the name of the user's program to be called, one to six characters in length.

**END**

◆ This parameter indicates the end of the Run-Time Parameters.

The format of the END parameter is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | END | |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - END.

*Operand Field* - Blank.

## EXAMPLE OF RUN-TIME PARAMETERS FOR ONE PROGRAM

| Statement Number | Name | Operation | Operand |
|---|---|---|---|
| 1 | // | ASSGN | SYSLST,L1 |
| 2 | // | ASSGN | SYS000,02 |
| 3 | // | ASSGN | WORK,08,,X'F0' |
| 4 | // | FILES | SYS000,5 |
| 5 | // | VOL | SYS000,MAS100 |
| 6 | // | TPLAB | 'MASTERΔFILEΔ100ΔΔ<br>0000010001000500010 1Δ67013Δ68013' |
| 7 | // | VOL | WORK,WORK |
| 8 | // | TPLAB | 'ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ000800' |
| 9 | // | VDC | STAKEY,,KOΔFILE3,000777,000001 |
| 10 | // | END | |

**Explanation**

◆ 1 - ASSGN statement for assignment of SYSLST to installation mnemonic L1.

2 - ASSGN statement for assignment of SYS000 to installation mnemonic 02.

3 - ASSGN statement for assignment of WORK to installation mnemonic 08, and a write control code of X'F0'for the seven-channel tape.

4 - 5 - 6 - Must be in sets.

4 - Positions the tape forward five tape marks to the desired file on this multifile reel.

5 - 6 - Supplies the label information needed for this input file.

7 - 8 - Must be in sets.

Supplies the label information needed for this output file.

9 - Assigns a random access file for logical level FCP. This file is located on two volumes, volumes 000777 and 000001.

10 - Terminates the run-time parameters.

## EXAMPLE OF RUN-TIME PARAMETERS FOR THREE SUCCESSOR PROGRAMS

| Statement Number | Name | Operation | Operand | Program in which Parameters Belong |
|---|---|---|---|---|
| 1 | // | ASSGN | SYSLST,L1 | (First Program) |
| 2 | // | ASSGN | SYS000,02 | (First Program) |
| 3 | // | ASSGN | WORK,08,,X'F0' | (First Program) |
| 4 | // | ASSGN | MASTER,X'107' | (Second Program) |
| 5 | // | ASSGN | TAPE1,03 | (Second Program) |
| 6 | // | ASSGN | TAPE4,03 | (Third Program) |
| 7 | // | FILES | SYS000,5 | (First Program) |
| 8 | // | VOL | SYS000,MAS100 | (First Program) |
| 9 | // | TPLAB | 'MASTERΔFILEΔ100ΔΔ 00001000100050001 01Δ67013Δ68013' | (First Program) |
| 10 | // | VOL | WORK,WORK | (First Program) |
| 11 | // | TPLAB | 'ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ 000800' | (First Program) |
| 12 | // | FILES | MASTER,2 | (Second Program) |
| 13 | // | VOL | MASTER,MAS002 | (Second Program) |
| 14 | // | TPLAB | 'MASTERΔFILEΔ002ΔΔ 00002' | (Second Program) |
| 15 | // | VOL | TAPE1,OUTPUT | (Second Program) |
| 16 | // | TPLAB | 'MASTERΔΔΔΔΔΔΔΔΔΔΔ 00100' | (Second Program) |
| 17 | // | VOL | TAPE4,INPUT | (Third Program) |
| 18 | // | TPLAB | 'MASTER ΔΔΔΔΔΔΔΔΔΔΔ 00100' | (Third Program) |
| 19 | // | VDC | STAKEY,,KOΔFILE3, 000777,000001 | (First Program) |
| 20 | // | VDC | MAST,91,FICAΔΔMASTERΔ 1967,000777 | (Third Program) |
| 21 | // | VDC | STAKEY,,KOΔFILE3, 000001 | (Third Program) |
| 22 | // | JOB | TWO | (First Program) |
| 23 | // | JOB | THREE | (Second Program) |
| 24 | // | END | | |

**Explanation**

◆ 1 - ASSGN statement for assignment of SYSLST to installation mnemonic L1 for program one.

2 - ASSGN statement for assignment of SYS000 to installation mnemonic 02 for program one.

3 - ASSGN statement for assignment of WORK to installation mnemonic 08 and a write control code of X'F0' for the seven level tape for program one.

4 - ASSGN statement for assignment of MASTER to channel 1, device 07 for program two.

5 - ASSGN statement for assignment of TAPE1 to installation mnemonic 03 for program two.

6 - ASSGN statement for assignment of TAPE4 to installation mnemonic 03 for program three.

7 - 8 - 9 - Label statements for files must be in sets.

7 - Positions the tape forward five tape marks to the desired file on device SYS000 for this multi-file reel for program one.

8 and 9 - Supplies the label information needed for this input file on device SYS000 for program one.

10,11 - Label statements for files must be in sets.

10 and 11 - Supplies the label information needed for this output file WORK for program one.

12,13,14 - Label statements for files must be in sets.

12 - Positions the tape forward two tape marks to the desired file on device WORK for this multifile reel for program two.

13,14 - Supplies the label information needed for this input file on device WORK for program two.

15,16 - Label statements for files must be in sets.

15 and 16 - Supplies the label information needed for this output file on device TAPE1 for program two.

17,18 - Label statements for files must be in sets.

17 and 18 - Supplies the label information needed for this input file on device TAPE 4 for program three.

19 - Assigns a random access file for logical level FCP. This file is located on two volumes, volumes 000777 and 000001 and is used in program one.

20 - Assigns a random access file for logical level FCP. This file is located on volume 000777 and the size of extent matrix is being changed to 91 bytes and is used in program three.

21 - Assign a random access file for logical level FCP. This file is located on volume 000001 and is used in program three. NOTE: This file was used in program one but it must have another VDC to be used in program three.

**Explanation** *(Cont'd)*

22 - This JOB statement calls program two at the termination of program one.

23 - This JOB statement calls program three at the termination of program two. Note: At the termination of program three the successor chain terminates because there is not another JOB successor call.

24 - Terminates the run-time parameters.

# 5. MONITOR

## INTRODUCTION

◆ The TDOS Monitor, a nonresident function of the TDOS Executive Control System, monitors the sequential execution of jobs as determined by a job input stream. The Monitor maintains complete control over the job input stream and the systems output information. The term nonresident is applicable because the use of the Monitor is optional when running a user program. However, during a Monitor session, memory allocated to the Monitor remains allocated to it until the session is terminated.

A job input stream consists of control statements, source language statements, and/or program sections. The control statements directed to the Monitor identify the job process to be executed, indicate the Monitor-controlled devices to be assigned, and specify the Monitor control options to be employed. A single job within the job stream may contain source program sections written in more than one language. Processing of the total job stream is called a Monitor session and any number of jobs may be performed within a given session.

## FUNCTIONAL DESCRIPTION

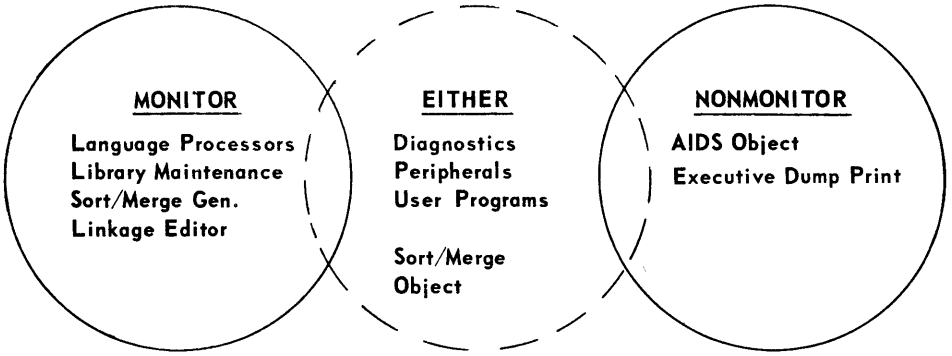
◆ Any program running under Monitor control and requiring input from the job stream or output to the system output must direct its request to the Monitor. Program input/output requests to devices other than the job stream or system output are under control of the program.

Each job to be performed is described by control language input in the job stream. A job may include any number of programs. The Monitor executes the job sequentially, requests the Executive to load each program in the job, and maintains control over the execution of each. Control information directing the programs and/or input to the programs is also included in the job stream. This input is directed by the Monitor to the program. When one program completes execution, the Monitor terminates the program. Monitor then determines from the job stream the next program to be initiated. At completion of the job stream, the Monitor terminates itself.

The Tape-Disc Operating System requires that all program preparation be performed under Monitor control. Program preparation includes language translation, linkage editing, and library maintenance of both the Object Module and the Program Load Library. Library maintenance may be performed under either Executive or Monitor. In addition to program preparation, Monitor job stream input may request execution of either installation programs or RCA system programs other than those mentioned above. The Monitor may also be used as a testing tool for installation programs. When used in this manner the program being executed specifies testing aids to be executed under Monitor control. These facilities include a dump of program memory, a snapshot dump, and memory patching. The testing routines are requested by appropriate control statements in the job stream.

## FUNCTIONAL DESCRIPTION (Cont'd)

Certain RCA-supplied programs are executed only under the control of the Monitor, which may be considered as an extension to the Executive. The figure below shows which programs may be executed under Monitor control, and which may be executed either under or outside of Monitor control.




## MONITOR SYSTEM DEVICES

◆ Devices which are designated and required by the system in the execution of its functions are called systems devices. Monitor requires and utilizes some of the system devices. The symbolic names of these system devices are indicated by the prefix SYS. All system devices except SYSIPT, SYSOPT, and SYSLST can be deallocated by the Monitor DEALOC control statement or by the deallocation facilities (DDEV macro or EDDV console routine) of the Executive. The symbolic device names assigned for Monitor are described below.

**SYSRES**

◆ Systems resident device containing the Executive, Monitor, and Systems Programs. Installation programs may optionally reside on this device. This device is allocated at system initiation and can only be accessed by the Executive.

**SYSLIB**

◆ Systems Library or Call Library containing the Macro Library (ML), the Object Module Library (OML), and a copy of the Executive.

**SYSIPT**

◆ A card reader, 9 level tape, paper tape reader, disc or drum which is the system input device from which the job stream is read. This device is assigned at Monitor initiation and may only be accessed through requests issued by the Monitor. The operator must indicate the SYSIPT device in the load Monitor request. This device is only deallocated upon Monitor termination.

**SYSLST**

◆ A printer, tape, disc or drum that is the systems output device to which listings, error notations, and messages are recorded. This device may be assigned to the Monitor by an ASSGN parameter or at the time of the execution of the first I/O requiring this device. If both SYSLST and SYSOPT are assigned to tape they must be assigned to the same device. This device is only deallocated upon Monitor termination.

**SYSOPT**

◆ An optional card punch, tape, disc or drum which is used as an output unit for system programs. This device may be assigned to the Monitor by an ASSGN parameter or at the time of the execution of the first I/O requiring this device. If both SYSLST and SYSOPT are assigned to tape, disc or drum, they must be assigned to the same device. This device is only deallocated upon Monitor termination.

**DEALLOCATION OF MONITOR DEVICES**

◆ Monitor devices (SYSIPT, SYSLST, and SYSOPT) must not be deallocated during a Monitor session.

**SYSUT1**

◆ A work tape that is used by system programs. All object module output such as Assembly, COBOL, or FORTRAN output, are produced on SYSUT1.

**SYSUT2**

◆ A work tape that is used by system programs. The Linkage Editor generates bound loadable object modules to this tape.

**SYSUT3**

◆ A work tape that is used by system programs.

**SYSUT4**

◆ An optional work tape that is used by systems programs which may be assigned to eliminate the use of SYSUT1 as a work tape.

**SYSUT5**

◆ An alternate Assembler source output device.

## MONITOR DISC I/O

◆ The Monitor Disc I/O option must be included at system generation time. (See EXECP card under Section 8). This option allows the user to specify disc or drum assignment for any or all of the Monitor I/O devices (SYSIPT, SYSOPT, SYSLST). When a Monitor session is initiated, a normal 'E LOD MON' request is given. A mnemonic of a random access device in the SYSIPT operand of the call indicates that SYSIPT is on disc or drum. The 'Y' operand, indicating the inclusion of the I/O batching area for Monitor, must be included in the Monitor call if any Monitor device is to use disc or drum.

Example:

```
E LOD MON,,A1,Y,50000
```

Output files to random access are indicated by an assignment of SYSLST and/or SYSOPT to a disc or drum. If both SYSLST and SYSOPT are assigned to random access, it must be the same file.

**Input Description**

◆ SYSIPT records on disc or drum are contained in a file allocated as 'SYSIPT'. The format of this file is the same as tape input format to Monitor, i.e., 80-byte records, blocked 4 to a block, with 9 blocks per track. No beginning label is used, therefore, data begins on track 0 of the first cylinder. The end of the input file is designated by the // ENDMON card.

**Output Description**

◆ SYSOPT and/or SYSLST may be selected as random access destined records. If both SYSOPT and SYSLST are indicated as random access files, they must be the same file. This file must be allocated as 'SYSOPT'. The format of this file is the same as tape output format, i.e., up to 400-byte variable blocked records, with 7 blocks per track. No beginning label is used, therefore, date begins on track 0 of the first cylinder. The end of file is designated by a record with a byte count of zero.

## MONITOR CONTROL STATEMENTS

◆ A Monitor control language is used to direct the processing of jobs contained within a given session. This control language consists of a series of control language parameter statements, with each statement indicating a particular function that is to be performed. All of the Monitor parameter statements follow these general rules:

The first two characters of a control statement are two slashes which identify the statement as a Monitor control statement. At least one space must follow the second slash. The control identifier follows and this indicates the type of Monitor Control statement. This identifier is a maximum of six characters in length and is followed by at least one space. The required operands follow with commas separating all operands. Operands may not be split over two input statements. More than one control statement of the same type may be written. The last operand is followed by a space or is indicated by the end of the input statement. The first space in the operand field indicates the end of all entries in this statement.

All Monitor Control Statements are written to SYSLST unless a NOLOG statement is given. Only DUMP, SNAP, and/or PATCH can appear in the job stream between LOAD and an EXEC. Any other control statement results in the LOADed program being overlaid by the Job Control overlay.

Erroneous conditions detected in a control statement are printed or typed. Appropriate action is taken by the Monitor. When a program calls its input from SYSIPT, a user return address must be supplied in the parameters for the Monitor SVC. This address is used to return control to the user when a Monitor control statement is read. If an address is not supplied and a control statement is identified, the program will terminate.

### STARTM

◆ The STARTM statement must be the first input record in the job stream. This parameter identifies the Monitor job stream.

The format for the STARTM statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | STARTM | Identification |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - STARTM.

*Operand Field* - An optional user-defined operand for identifying the Monitor session. Used only as an identification written to SYSLST.

**ASSGN**

This statement may appear in the job stream to define and to allocate devices required by the Monitor session. This statement may appear anywhere in the job stream prior to the first access to the device. All devices remain allocated to the job stream until a JOB parameter is encountered. At this time all devices but systems devices, devices prefixed with (SYS), are deallocated. Systems devices (SYS) remain allocated to the Monitor session until Monitor terminates and then they are deallocated. An ASSGN statement for SYSLST or a NOLOG statement must be the second statement in the job stream.

*Notes:*

1. Assignment of random access devices causes this assignment to be entered in the equate table thus reducing the number of acceptable equate statements.
2. The same symbolic device name can not be assigned to a different physical device unless first deallocated through the use of the DEALOC parameter or as mentioned above.
3. SYSIPT need not be assigned with a run-time parameter, since it is assigned in the E LOD MON message.

The format for the ASSGN statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | ASSGN | Symbolic, $\left\{\begin{matrix}\text{mn}\\ \text{X'cuu'}\end{matrix}\right\}$ [, [dt] [ ,X'wc'] ] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - ASSGN.

*Operand Field* -

Symbolic - The symbolic device name, one to six characters in length, assigned to the device.

mn - Two-character installation mnemonic device name to be associated with the symbolic.

X'cuu' - Indicates the device to be assigned by channel and unit in hexadecimal.

c = 0 - multiplexor channel.
1 - selector channel 1.
2 - selector channel 2.
3 - selector channel 3.
4 - selector channel 4.
5 - selector channel 5.
7 - selector channel 6.

uu - 00 to FF indicating the unit.

dt - This entry is present for compatibility, it is ignored by TDOS.

X'wc' - Indicates applicable write control information used for 7-level tapes.

**ASSGN**
*(Cont'd)*

| wc | Density bytes/in. | Parity | Pack/unpack mode | Translate mode |
|---|---|---|---|---|
| 60 | 200 | ODD | OFF | OFF |
| A0 | 556 | ODD | OFF | OFF |
| E0 | 800 | ODD | OFF | OFF |
| 40 | 200 | EVEN | OFF | OFF |
| 80 | 556 | EVEN | OFF | OFF |
| C0 | 800 | EVEN | OFF | OFF |
| 68 | 200 | ODD | OFF | ON |
| A8 | 556 | ODD | OFF | ON |
| E8 | 800 | ODD | OFF | ON |
| 48 | 200 | EVEN | OFF | ON |
| 88 | 556 | EVEN | OFF | ON |
| C8 | 800 | EVEN | OFF | ON |
| 70 | 200 | ODD | ON | OFF |
| B0 | 556 | ODD | ON | OFF |
| F0 | 800 | ODD | ON | OFF |

**JOB**

◆ This parameter indicates that a job is to be initialized in the Monitor job stream. Any number of jobs may be included in one Monitor session. A job may consist of any number of individual program loads. The monitor encountering a JOB parameter deallocates devices other than the systems devices (SYS) and initializes all job indicators and flags. The following is a list of initialization functions performed:

1. Error flags set by abnormal termination of a subprocessor are reset.
2. JOB identification name is set to the new name.
3. Indicators denoting successful completion of job functions are reset.
4. PARAM table is reset to the pre-set options.
5. EQUATE table is initialized.
6. SYSUT1 is rewound. Any object module file(s), generated previously by subprocessors, can be overwritten by subsequent programs. Care should be used, therefore, during stacked assemblies or compilations.
7. Monitor Job Accounting tables are reinitialized. 03RT time is outputted to SYSLST and/or console typewriter unless overridden.

The format for the JOB statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | JOB | Name |

**JOB** *(Cont'd)*

*Name Field* - Two slashes in the first two positions.

*Operation Field* - JOB.

*Operand Field* - An identification which labels the job in the Monitor session. Typeouts to SYSLST for this job are identified by the name.

**SUBPROCESSOR**

◆ This parameter requests the loading and execution of a system program from the system device SYSRES. If a subprocessor is requested from an alternate library the LOAD or EXEC statement must be used.

The format for the subprocessor statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | Subprocessor Name | |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - The symbolic name for the subprocessor that is to be loaded and executed:

ASSMBL = Assembly
COBOL = COBOL
FORTRN = FORTRAN
LNKEDT = Linkage Editor
LLU = Load Library Update
MLU = Macro Library Update
OMLU = Object Module Library Update
RPG = Report Program Generator
SRTGEN = Sort Generator
CLU = COBOL Library Update

*Operand Field* - Blank.

**PARAM**

◆ The PARAM statement provides the user with the facility to indicate subprocessor options for the job. This parameter may only indicate options for a subprocessor. If this statement is used for a subprocessor, it must precede the subprocessor control statement. Any one or all of the options may be requested. If any option is omitted, standard usage is assumed. Standard usage is set only when a //JOB card is encountered. Any number of options may be entered per card, up to and including column 71.

**PARAM**
*(Cont'd)*

The format for the PARAM statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | PARAM | parameter [,parameter] [, parameter] etc. |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - PARAM.

*Operand Field* - The options are indicated by the parameters listed below. They may be in any order. The underlined condition in each of the following parameters indicates the assumed option, when no parameter is entered for that option. The options are as follows:

TAPE=YES - Indicates tape output is to be generated by the translator.

=NO - Indicates tape output is not to be generated by the translator.

CARD=YES - Indicates card image output is to be written to SYSOPT.

=NO - Indicates card image output is not to be written to the SYSOPT device by the translator.

LIST=YES - Indicates a COBOL or FORTRAN source program listing is to be written to SYSLST by the translator.

=NO - Indicates a COBOL or FORTRAN source program listing is not to be written to SYSLST by the translator.

MAP=YES - Indicates to COBOL that a locator map listing is to be written to SYSLST and, to Assembly and FORTRAN that an alphabetic and numeric map listing of all variables and statement numbers is to be written to SYSLST.

=NO - Indicates to Assembly, COBOL, and FORTRAN that the listings outlined above are not to be written to SYSLST.

OBJLST=YES - Indicates a listing of a COBOL generated object program is to be written to SYSLST.

=NO - Indicates a listing of a COBOL generated object program is not to be written to SYSLST.

DIAG=YES - Indicates that a COBOL diagnostic listing is to be written to SYSLST.

=NO - Indicates that COBOL diagnostic listing is not to be written to SYSLST.

XREF=YES - Indicates that a COBOL or Assembly cross-reference listing is to be written to SYSLST.

=NO - Indicates COBOL or ASSEMBLY cross-reference listing is not to be written to SYSLST.

**PARAM**
*(Cont'd)*

DUPL=YES - Indicates that COBOL object modules generated by a "stacked compilation" can be bound into separate loadable programs without submitting additional parameters to the Linkage Editor.

=NO - Indicates the COBOL object modules generated by a "stacked compilation" will be bound into one loadable program, unless overridden by Linkage Editor parameters.

DEBUG=YES - Indicates that a trace is to be provided in each FORTRAN subprogram.

=NO - Indicates that a trace is not to be provided in each FORTRAN subprogram.

CODE=1 - Indicates that EBCDIC code is used for source program input.

=2 - Indicates that 7094 code is used for source program input.

=3 - Indicates that 3301 code is used for source program input.

INPUT=SYSIPT - Indicates that the source input is on the SYSIPT device.

=Symbolic - Indicates the device (other than SYSIPT) which contains the source input.

OUTPUT=SYSUT1 - Indicates that the translator generated object modules are to be written to SYSUT1.

=Symbolic - Indicates the device (other than SYSUT1) to which the translator generated object modules are to be written.

WORK=YES - Indicates that an additional work tape (SYSUT4) is assigned to the Monitor session.

=NO - Indicates that an additional work tape (SYSUT4) is not assigned to the Monitor session.

LIBRY=YES - Indicates there is a Macro Library.

=NO - Indicates there is not a Macro Library.

SOURCE=SYSUT5 - Indicates that the updated Assembler source output device is SYSUT5.

=Symbolic - Indicates the device (other than SYSUT5) which is assigned to the updated assembler source output.

**PARAM**
*(Cont'd)*

ERRLST=YES - Indicates that an Assembly error list is to be produced.

=NO - Indicates Assembly error listing is not to be produced.

ASMLST=YES - Indicates an Assembly source listing is to be written to SYSLST.

=NO - Indicates an Assembly source listing is not to be written.

An "X" below indicates the standard assumed usage for the individual parameters if they are omitted. The standard usage is set every time a //ΔJOB parameter is encountered.

| | YES | NO | 1 | SYSIPT | SYSUT1 | SYSUT5 |
|---|---|---|---|---|---|---|
| TAPE | X | | | | | |
| CARD | | X | | | | |
| LIST | | X | | | | |
| MAP | X | | | | | |
| OBJLST | | X | | | | |
| DIAG | X | | | | | |
| XREF | | X | | | | |
| DUPL | | X | | | | |
| DEBUG | | X | | | | |
| CODE | | | X | | | |
| INPUT | | | | X | | |
| OUTPUT | | | | | X | |
| WORK | | X | | | | |
| LIBRY | X | | | | | |
| SOURCE | | | | | | X |
| ERRLST | X | | | | | |
| ASMLST | X | | | | | |

**LOAD**

◆ The LOAD control statement requests a program to be loaded by the Monitor. The loaded program is not executed until an EXEC statement is encountered. If the LOAD control statement is followed by control statements other than DUMP, PATCH, SNAP, or EXEC the loaded program is released from memory.

The format for the LOAD statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | LOAD | [Program name, [UT2 / ALT / mn]] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - LOAD.

*Operand Field* -

Program name - The program name which is the load name of the program to be brought into memory. When this operand is omitted, the program previously bound by the Linkage Editor in this JOB and contained on SYSUT2 is loaded into memory. If no program has been bound by this JOB and object modules generated by a language translator in this JOB are present on SYSUT1, these modules are bound before they are loaded.

UT2 - Specifies that the program is to be loaded from SYSUT2 (use of this parameter allows the loading of any program previously bound to SYSUT2 by the Linkage Editor). If UT2 or ALT is not specified and program name is included, the program is loaded from SYSRES. This parameter is permitted only when the program name is specified.

ALT - Specifies that the program is to be loaded from the alternate load tape as specified in the E LOD MON command. If ALT or UT2 is not specified and program name is included, the program is loaded from SYSRES. This parameter is permitted only when the program name is specified.

mn - Installation mnemonic of the alternate disc or drum that contains the program to be loaded. If the mn is not specified and program-name is included, the program is loaded from SYSRES. This parameter is permitted only when the program-name is specified.

**EXEC**

◆ Calls for execution of a program which has been loaded or is to be loaded as a result of this control statement.

The format for the EXEC statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | EXEC | [Program-name, [UT2 / ALT / mn]] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - EXEC.

*Operand Field* -

Program name - The name of the program to be loaded and executed. The presence of the operand indicates that the program has not been loaded. When this operand is omitted and a program has not been loaded, the program previously bound by the Linkage Editor in this JOB and contained on SYSUT2 is loaded and executed. If a program has not been bound by this JOB and object modules generated by a language translator in this JOB are present on SYSUT1, then these object modules are bound before they are loaded and executed.

ALT - Specifies that the program is to be loaded from the alternate load tape.

UT2 - Specifies that the program to be loaded is on a program library tape previously assigned to SYSUT2.

mn - Installation mnemonic of the alternate disc or drum that contains the program to be loaded. If the mn is not specified and program-name is included, the program is loaded from SYSRES. This parameter is permitted only when the program-name is specified.

**DUMP**

◆ The DUMP control statement requests the program or a portion of the program last loaded by Monitor to be dumped to the SYSLST device.

The format for the DUMP statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | DUMP | [LLLLLL,RRRRRR] [,FN] [,0] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - DUMP.

*Operand Field* - Dumping options are provided within this operand for area format, grouping and time to dump. If the dumping options are not used, the operand field is left blank and, dumping of the entire program is performed at program termination. (See output formats page 5-14.)

The dumping options are: 3

LLLLLL - The leftmost relative address in hexadecimal format of memory to be dumped. If this address is other than zero, the Executive Storage Area is not dumped.

RRRRRR - The rightmost relative address in hexadecimal format of memory to be dumped. These addresses are required only when less than the entire program is to be dumped.

F = Format type:

- H (or 8) - hexadecimal.
- G (or 7) - graphic (and literal, indicated with a grouping factor of zero).
- F (or 6) - floating-point (double-precision).
- S (or 2) - floating-point (single-precision).
- E (or 5) - logical.
- M (or 4) - halfword hex with its corresponding instruction mnemonic.
- C (or 3) - hexadecimal with its graphic equivalent.
- I (or 9) - integer.

If this entry is omitted hexadecimal with graphic format is assumed.

N = 0 - grouping factor of zero.

1 - grouping factor of one.

2 - grouping factor of two.

4 - grouping factor of four.

If this entry is omitted a grouping factor of four is assumed. 0 is permitted with format type G (or 7) only.

0 = 1 - dump is required immediately.

2 - dump is required at program termination.

3 - dump is required at error termination.

If this entry is omitted the dump at program termination is assumed.

**Dump Output Formats**

◆ The various output formats for the DUMP areas follows:

Hexadecimal - H (or 8)

```
Four-byte   XXXXXXXX      XXXXXXXX
Two-byte    XXXX    XXXX      XXXX
One-byte    XX    XX    XX      XX
```

where X = 0-9 and A-F

Graphic - G (or 7)

```
Four-byte   GGGG      GGGG      GGGG
Two-byte    GG    GG    GG    GG    GG
One-byte    G   G   G   G   G   G   G
```

Literal - G (or 7) with a grouping factor of zero

GGGGGGGGGGG

where G = the printable graphic character

Floating Point - F (or 6) double-precision

(Two words) - XXXXXXXXXXXX XX E ± XX

Floating Point - S (or 2) single-precision

(One word) - XXXXXX E ± XX

where X (mantissa) = hexadecimal format

E = exponent, XX is a power to which the number 16 is raised

± = sign

An even number of entries is placed on each printed line.

Logical = E (or 5)

F _ _ F _ _ T _ _ T _ _ F _ _ T _ _ XX _ T _ _

where T = the first byte of a word or a byte that contains ($FF_{16}$)

F = the first byte of a word or a byte that contains ($00_{16}$)

Half-word Hex with Mnemonic - M (or 4)

```
XXXX  XXXX  XXXX  XXXX  XXXX
MM          MMM
```

where X = 0-9 and A-F

M = operation code mnemonic

**Dump Output Formats**
*(Cont'd)*

Hex with Graphic - C (or 3)

```
                G G    G          G    G
Four-byte       XXXXXXXX          XXXXXXXX

                G G          G          G
Two-byte        XXXX         XXXX       XXXX

                G                  G       G
One-byte        XX      XX         XX      XX
```

where X = 0-9 and A-F, and
G = printer graphic equivalent

Integer - I (or 9)

```
Four-byte    XXXXXXX   -XXX      XX   -XXXXXXX
Two-byte      -XXXXX    XXX   -XXXX         -X
```

where X = integer. A space before the integer indicates a positive power; a minus indicates a negative power. Leading zeros are suppressed.

**SNAP**

◆ The SNAP control statement-requests a memory dump at specified intervals of the resident program in memory running under Monitor control. The SNAP control statement must be between the LOAD statement for this program and the EXEC statement. Only the first load of a program may use the SNAP statement. There may be a maximum of 101 SNAP statements for each program. When the SNAP Control statement is used, the program name entry in the E LOD console routine typein must be MONSNP.

The format for the SNAP statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | SNAP | IIIIII,A,B [,LLLLLL,RRRRRR] [,FN] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - SNAP.

*Operand Field* -

IIIIII - Leftmost relative address in hexadecimal format of the instruction in the user program at which the snapshot dump is to be taken.

A - The decimal numbers from 1 to 3 digits; not greater than $255_{(10)}$ of the times the instruction is to be executed before the snapshot dump is to be taken. Zero (0) indicates that the dump is to be taken on every encounter. The snapshot dump is taken before the instruction is executed.

Example - If six (6) is specified for A, the instruction is executed five (5) times and then the snapshot dump is performed before the instruction is executed the sixth time.

B - The decimal numbers from 1 to 3 digits, not greater than $255_{(10)}$ of the number of the last time the instruction is to be executed for a snapshot dump. Zero (0) indicates that the dump is to be taken on every encounter. The snapshot dump is taken before the instruction is executed.

Example - If nine (9) is specified for B, the snapshot is performed for the last time just before the ninth execution of the instruction.

The remaining elements of the SNAP request are explained under // DUMP format.

**PATCH**

◆ The PATCH control statement allows a program in the job streamto be patched. Only the first load of a program may be patched. The PATCH statement can only follow the LOAD, SNAP, DUMP, or another PATCH statement.

The format of the PATCH statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | PATCH | AAAAAA, XXXXXX - - - |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - PATCH.

*Operand Fiela* -

AAAAAA - The hexadecimal address of the leftmost relative address (zero relative) where patches are to be applied.

XXXXXX - The information to be patched directly into memory. The information is in hexadecimal format, two characters forming one byte. Patching information may be placed in a statement up to and including column 71. A new statement must be created with a new relative address if more patches are needed.

**EQUATE**

◆ The EQUATE control statement equates previously assigned symbolic device to another symbolic device. SYSIPT, SYSOPT, and SYSLST may not be equated to another device. Up to six EQUATE statements may be active at any one time within a job.

The format of the EQUATE statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | EQUATE | Symbolic1,Symbolic1A[,Symbolicn, Symbolicna] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - EQUATE.

*Operand Field* -

Symbolic1 - Symbolic name of device previously defined.

Symbolic1A - New symbolic name by which the defined device is to be referenced.

A number of devices may be equated in one control statement up to and including column 71. An entry may not be split over two cards.

**DEALOC**

◆ The DEALOC control statement calls for the deallocation of a device allocated to the Monitor session. Deallocation occurs both in the device list and the EQUATE table. This statement may deallocate more than one device. The entries may be entered up to and including column 71. SYSIPT, SYSOPT, and SYSLST may not be deallocated.

The format of the DEALOC statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | DEALOC | Symbolic1 [, Symbolicn] |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - DEALOC.

*Operand Field* - Symbolic name of devices that are to be deallocated.

**COMM**

◆ The COMM control statement allows user comments to be written to SYSLST and the Console Typewriter. If a NOLOG statement is issued previously, the comments are not written to SYSLST, but only to the Console. Comments may be entered up to and including column 71.

The format of the COMM statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | COMM | Comments |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - COMM.

*Operand Field* - Any user comments.

**LOG**

◆ The LOG control statement requests Monitor to write all control statements to SYSLST. The LOG control statement is only required if it is desired to resume writing control statements to SYSLST after writing has been suspended by the NOLOG statement. LOG is assumed if NOLOG is not specified.

The format of the LOG statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | LOG | |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - LOG.

*Operand Field* - Blank.

**NOLOG**

◆ The NOLOG control statement indicates Monitor is not to write control statements to SYSLST. However, if any program under Monitor uses SYSLST, a request for its assignment will be made at execution time. If the second card in the input stream is not NOLOG, LOG is assumed.

The format of the NOLOG statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | NOLOG | |

*Name Field*-Two slashes in the first two positions.

*Operation Field* - NOLOG.

*Operand Field* - Blank.

**ENDMON**

◆ The ENDMON control statement terminates the Monitor session.

The format of the ENDMON statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | ENDMON | |

*Name Field*-Two slashes in the first two positions.

*Operation Field* - ENDMON.

*Operand Field* - Blank.

**PAUSE**

◆ The PAUSE control statement allows instructions to be issued to the operator via the Console Typewriter. This function differs from COMM only in that Monitor processing halts until the operator responds (MΔCONT) after completing the indicated instructions.

Other legitimate responses to the PAUSE statement are MΔHLTJ, MΔDUMPP, MΔDUMPJ, MΔSKIP, and MΔEND.

The format of the PAUSE statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| // | PAUSE | Comments |

*Name Field* - Two slashes in the first two positions.

*Operation Field* - PAUSE.

*Operand Field* - Any user instructions. A comment is required.

## MONITOR RUN-TIME PARAMETERS

◆ Monitor run-time parameters are a set of object time parameters that Monitor accepts and processes for one or more user programs and/or subprocessors. These parameters allow a program utilizing TOS FCP standard label procedures to be run under the control of Monitor. The run-time parameters which Monitor accepts are as follows:

```
// FILES
// VOL
// TPLAB
// VDC
```

A description of these run-time parameters can be found under the Executive Run-Time Parameters section (page 4-5) in this manual. When utilizing Monitor run-time parameters the user must take into consideration the following:

1. The Run-time parameter information is stored at the end of the Monitor allocated area. When a user program or subprocessor is loaded, no check is performed to ensure that the load does not overlay this information. Therefore, it is important that the E LOD operator command contain sufficient memory to hold the user program or subprocessor and its run-time parameters.

2. The sequence specified for these run-time parameters must be maintained or the parameters are rejected as out of sequence.

3. Insufficient or invalid information causes the parameter to be rejected and can cause a parameter sequence error.

4. These run-time parameters must be placed before the LOAD Monitor control statement (or EXEC control statement providing it implies a load and execute) in the job input stream for the user program or subprocessor.

5. Only one group of these run-time parameters can be specified for a given user program or subprocessor, however, any number of user programs and/or subprocessors can have run-time parameters in the job input stream. Any other group of run-time parameters, beyond the first group, for a given user program or subprocessor are rejected as out of sequence.

6. The Operation List entry for the user program or subprocessor reflects the presence of run-time parameters by having byte 1, bit 0 set to one.

7. The Executive Storage Area for the user program or subprocessor reflects the presence of run-time parameters through bytes 112-115 which contain the address of the run-time parameter area.

**MONITOR RUN-TIME PARAMETERS (Cont'd)**

8. If run-time parameters are desired during a compile and go job stream, an explicit call for Linkage Editor must be made preceding the RTP's and the LOAD or EXEC card for the programs.

   Example:

```
// JOB
        }
        } Various Assemblies and/or Compilations
        }

// LNKEDT
        }
        } Desired RTP's
        }

// EXEC
```

**MONITOR CONSOLE ROUTINES**

◆ The operator communicates with the Monitor by means of console request through the console typewriter. These routines allow the operator to initiate procedures that are not included as part of the normal job stream. All executive console routines may also be issued under Monitor control except the termination routines. All termination routines must be issued through the Monitor.

A functional description of each console routine is described in this section. Further information relating to the operating procedures and message formats for each of these routines are presented in the TDOS Operators' Guide. The Monitor console routines are:

| | |
|---|---|
| M HLTP | Terminate Program |
| M HLTJ | Terminate Job. |
| M DUMPP | Dump Program. |
| M DUMPJ | Dump Job. |
| M SKIP | Monitor Skip. |
| M END | Terminate Monitor. |
| M CONT | Continue Processing after PAUSE. |

**M HLTP (Terminate Program)**

◆ The Terminate Program console routine terminates the current program running in the Monitor job stream (// JOB,// EXEC, etc.). This request then initiates and executes the next program in the job stream unless the current program was the last one, and in this case Monitor terminates.

**M HLTJ (Terminate Job)**

◆ The Terminate Job console routine terminates the current Job running in the Monitor job stream (// JOB,// EXEC, etc.). This routine then initiates and executes the next Job in the job stream; however, if the current Job was the last one, Monitor terminates.

**M DUMPP (Dump Program)**

◆ The Dump Program console routine dumps the current program running in the Monitor job stream. This routine then initiates and executes the next program in the job stream (// JOB,// EXEC, etc.); however, if the current program was the last one, Monitor terminates. The program dump is in hexadecimal format with a grouping of four, unless another format and grouping factor had been indicated in a DUMP statement.

**M DUMPJ (Dump Job)**

◆ The DUMP Job console routine dumps the current program running in the Monitor job stream. This routine then initiates and executes the next Job in the job stream (// JOB, // EXEC, etc.); however, if the current Job was the last one, Monitor terminates. The program dump is in hexadecimal format with a grouping of four, unless another format and grouping factor had been indicated in a DUMP statement.

**M SKIP (Monitor Skip)**

◆ The Monitor Skip console routine terminates the current JOB running in the Monitor job stream and causes all statements in the input stream to be skipped until the indicated parameter is encountered in a // control card. The parameter may be up to 16 bytes long. The field that the parameter is compared to must begin in column 4 of the // control card. If a // ENDMON parameter is encountered, Monitor is terminated.

Example:

M SKIP JOBA would skip all statements in the input stream until the JOB Control parameter // JOBA is found. JOBA must start in column 4.

**M END (Terminate Monitor)**

◆ The Terminate Monitor console routine terminates the current Monitor session.

## MONITOR COMMUNICATION MACROS

◆ The program communicates with the Monitor by means of Monitor Communication macro statements. These macros are written in the program to request the Monitor to perform functions that the program cannot directly perform.

The Monitor Communication macros are:

| | | |
|---|---|---|
| TERMJ | Terminate Job. | SVC 63 |
| RDCRD | Read SYSIPT. | SVC 62 |
| WRTOT | Write SYSOPT. | SVC 61 |
| PROUT | Write SYSLST. | SVC 58 |
| ERFLG | Set Error Flag. | SVC 60 |
| MONTB | PARAM Table address. | SVC 59 |
| STUT1 | Set SYSUT1 Flag | SVC 57 |
| STUT2 | Set SYSUT2 Flag | SVC 56 |

**TERMJ**

◆ The TERMJ macro is issued in a program to terminate a job that is running under Monitor control. After the job is terminated the next job in the job stream is initiated and executed.

The format of the TERMJ statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | TERMJ | |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - TERMJ.

*Operand Field* - Blank.

**RDCRD**

◆ The RDCRD macro instructs the Monitor to read an input statement from the job stream on SYSIPT. All references to SYSIPT are controlled by the Monitor. The user may not directly read SYSIPT.

The format of the RDCRD statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | RDCRD | addr1, addr2 |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - RDCRD.

*Operand Field* -

addr1 - the symbolic name of the leftmost area in the program that is to receive the input from SYSIPT.

addr2 - the symbolic name of the return address if the record read is a Monitor control statement, a // record. If this entry is omitted, an address of all zeros is generated.

**WRTOT**

◆ The WRTOT macro instructs the Monitor to write an output record to the system output device SYSOPT. All references to SYSOPT while running under Monitor, are controlled by the Monitor. The user may not directly write to SYSOPT. A maximum of 80 bytes may be written at a time.

The format of the WRTOT statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | WRTOT | area, length |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - WRTOT.

*Operand Field* -

area - the symbolic name of the leftmost area in the program that is to be written to SYSOPT.

length - the decimal value of the length of the area to be written. If the length is not included, the length attribute of the area is taken as the length of the message.

**PROUT**

◆ The PROUT macro instructs the Monitor to write an output record to the system listing device SYSLST. All references to SYSLST while running under Monitor are controlled by the Monitor. The user may not directly write to SYSLST.

The format of the PROUT statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | PROUT | area, length |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - PROUT.

*Operand Field* -

area - the symbolic name of the leftmost area in the program that is to be written to SYSLST. The first character of the output area must be the printer control character.

length - the decimal value of the length of the area to be written. If the length is not included, the length attribute of the area is taken as the length of the message. The length of the area does include the one byte of printer control information.

**ERFLG**

◆ The ERFLG macro requests Monitor to set an error flag to prohibit the execution of the following functions in the job stream: LNKEDT, LLU, MLU, OMLU, LOAD, EXEC, DUMP, SNAP, and PATCH. The error flag is reset at the end of the Job.

The format of the ERFLG statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | ERFLG | |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - ERFLG.

*Operand Field* - Blank.

**MONTB**

◆ The MONTB macro places the address of the Monitor PARAM table in General Register 15. (See page 5-29 for explanation of Monitor PARAM Table.)

The format of the MONTB statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | MONTB | |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - MONTB.

*Operand Field* - Blank.

**STUTI**

◆ The STUTI macro sets a flag to indicate that there is an object module, an output of a language processor, on systems device SYSUT1.

The format of the STUT1 statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | STUT1 | |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - STUT1.

*Operand Field* - Blank.

**STUT2**

◆ The STUT2 macro sets a flag to indicate that there is a bound program, executable program, on systems device SYSUT2.

The format of the STUT2 statement is as follows:

| Name | Operation | Operand |
|---|---|---|
| name | STUT2 | |

*Name Field* - The symbolic name identifying the first byte of this statement.

*Operation Field* - STUT2.

*Operand Field* - Blank.

## MONITOR SUPERVISOR CALL

**PDUMP**

◆ The Program Dump (PDUMP) Supervisor Call causes the Monitor to call in and execute one of the Monitor memory dumps during the operation of the program. After the dump is completed, control is returned to the user program at the instruction following the parameter set for this supervisor call.

The Supervisor Call (SVC) is coded as 12 bytes in the following format:

Supervisor Call (two bytes) - SVC (1 byte)
64 - SVC code (1 byte)

Parameter Set (10 bytes) - Left-hand end address of the area to be dumped (absolute four-byte hexadecimal address).

Right-hand end address of the area to be dumped (absolute four-byte hexadecimal address).

Format type (one-byte):

8 - hexadecimal

7 - graphic (and literal, indicated with a grouping factor of zero).

6 - floating point (double precision).

2 - floating point (single precision).

5 - logical.

4 - halfword hex with its corresponding instruction mnemonic.

3 - hexadecimal with its graphic equivalent.

9 - integer.

Grouping Factor (one-byte):

0 - for literal dumps only.

1 - one byte per group.

2 - two bytes per group.

4 - four bytes per group.

## PARAM Table Contents

◆ The address of the Monitor PARAM Table is placed in General Register 15 (P1) by the MONTB macro. The Monitor Table outlined below reflects these options and how they are indicated. The Monitor Table is initialized to the standard usage of each option at JOB initialization.

**Monitor (PARAM) Table**

| Byte | Bit | Option | Setting |
|---|---|---|---|
| 0 | 0 | TAPE = YES | 1 |
| | | = NO | 0 |
| | 1 | CARD = YES | 1 |
| | | = NO | 0 |
| | 2 | LIST = YES | 1 |
| | | = NO | 0 |
| | 3 | MAP = YES | 1 |
| | | = NO | 0 |
| | 4 | OBJLST = YES | 1 |
| | | = NO | 0 |
| | 5 | DIAG = YES | 1 |
| | | = NO | 0 |
| | 6 | XREF = YES | 1 |
| | | = NO | 0 |
| | 7 | DUPL = YES | 1 |
| | | = NO | 0 |
| 1 | 0 | DEBUG = YES | 1 |
| | | = NO | 0 |
| | 1-2 | CODE = 1(EBCDIC) | 00 |
| | | = 2(7094) | 01 |
| | | = 3(3301) | 10 |
| | 3 | WORK = YES | 1 |
| | | = NO | 0 |
| | 4 | LIBRY = YES | 1 |
| | | = NO | 0 |
| | 5 | ERRLST = YES | 1 |
| | | = NO | 0 |
| | 6 | ASMLST = YES | 1 |
| | | NO | 0 |

**PARAM Table Contents** *(Cont'd)*

**Monitor (PARAM) Table** *(Cont'd)*

| Byte | Bit | Option | Setting |
|---|---|---|---|
| 1 (Cont'd) | 7 | INPUT = SYSIPT | 0 |
| | | = Symbolic | 1 |
| 2 | 0 | OUTPUT = SYSUT1 | 0 |
| | | = Symbolic | 1 |
| | 1 | SOURCE = SYSUT5 | 0 |
| | | = Symbolic | 1 |
| | 2 | SYSUT1 = No OMF | 0 |
| | | = OMF present | 1 |
| | 3 | SYSUT2 = No information | 0 |
| | | = Information | 1 |
| | 4 | Subprocessor error flag = No errors | 0 |
| | | = Errors | 1 |
| | 5 | Linkage Editor Call Indic. = User Call | 0 |
| | | = Monitor Call | 1 |
| | 6-7 | Reserved | |
| 3-8 | | | Symbolic entered with INPUT |
| 9-14 | | | Symbolic entered with OUTPUT |
| 15-20 | | | Bound program Load Name |
| 21-26 | | | Symbolic entered with SOURCE |

## EXAMPLE OF A MONITOR SESSION

◆ Shown below are control statements which constitute a sample job stream. The purpose of these statements is to illustrate the use and the options available within a given Monitor session.

| Statement No. | Name | Operation | Operand |
|---|---|---|---|
| 1 | // | STARTM | |
| 2 | // | ASSGN | SYSLST,05 |
| 3 | // | JOB | ONE |
| 4 | // | ASSGN | SYS002,01 |
| 5 | // | ASSGN | XYZ,02 |
| 6 | // | ASSGN | 13865,03 |
| 7 | // | ASSGN | 140,04 |
| 8 | // | EXEC | PAY |
| 9 | // | EQUATE | XYZ,ABC |
| 10 | // | DEALOC | 140,ABC |
| 11 | // | LOAD | FICA |
| 12 | // | EXEC | |
| 13 | // | PAUSE | Dismount tape on trunk 02. |
| 14 | // | JOB | TWO |
| 15 | // | EQUATE | SYS002,MAST |
| 16 | // | ASSGN | 13865,03 |
| 17 | // | LOAD | STATE |
| 18 | // | DUMP | 3 |
| 19 | // | EXEC | |
| 20 | // | ENDMON | |

1 - Identifies the start of the Monitor session job stream.

2 - Assigns SYSLST device.

3 - Identifies the start of Job ONE.

4 - Assigns symbolic device SYS002 to actual device 01.

5 - Assigns symbolic device XYZ to actual device 02.

6 - Assigns symbolic device 13865 to actual device 03.

7 - Assigns symbolic device 140 to actual device 04.

8 - Loads a program called PAY and transfers control to it for execution.

**EXAMPLE OF A MONITOR SESSION (Cont'd)**

9 - Renames the symbolic device called XYZ in program PAY to symbolic ABC. All references to actual device 02 from program FICA then uses symbolic ABC.

10 - Symbolic device 140 from program PAY is deallocated. It was not required in program FICA.

11 - Loads a program called FICA.

12 - Control is transferred to program FICA for execution.

13 - Instructs the operator to dismount the tape on trunk two.

14 - Identifies the start of Job TWO. All devices other than system devices (SYS) are deallocated.

15 - Renames symbolic device SYS002 to MAST. All references to actual device 01 from job TWO use the symbolic name MAST. SYS002 was not deallocated by the JOB statement.

16 - Assigns symbolic device 13865 to actual device 03. This device was deallocated by the JOB statement.

17 - Loads the program STATE.

18 - A dump at error termination is required.

19 - Control is transferred to program STATE for execution.

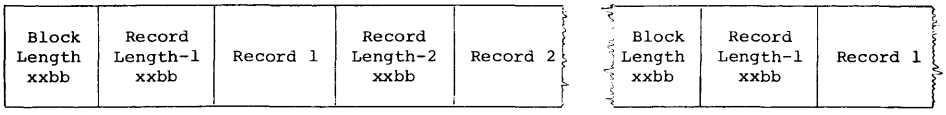
20 - The Monitor session is terminated.

**MONITOR SYSTEM OUTPUT**

◆ Job Stream output is written to the assigned output device (SYSOPT or SYSLST) each time a request is received by Monitor from the running program.

In addition to program generated outputs, certain Monitor generated outputs are written to the output medium. Upon request, job control statements are also recorded on the system output.

When SYSLST and/or SYSOPT is assigned to tape or direct/access, the records are blocked in the following format:




*Note:*

Number of records depends on maximum blocksize of 400 bytes.

At the beginning of each block is a four-character field specifying the block length. The first two characters (xx) specify the length, and the last two (bb) are blanks. The block length includes the four characters of the record length field.

## MONITOR SYSTEM OUTPUT (Cont'd)

A print record directed to tape or direct/access has the following out format:

| Record Length xxbb | Print Select Character P | Print Control Character y | Print Information |
|---|---|---|---|

The codes are:

xxbb = record length as specified above.

P = print select character. (Printer record.)

y = print control character as included in the print record (spacing control).

A punch record directed to tape or direct/access has the following out format:

| Record Length xxbb | Punch Select Character C | Punch Information |
|---|---|---|

The codes are:

xxbb = record length as specified above.

C = punch select character. (Punch record.)

# 6. FILE CONTROL PROCESSOR (FCP)

## GENERAL

◆ The File Control Processor (FCP) is a generalized input/output system that functions, in conjunction with the Executive, to control all input/output requirements of programs that are executed in a Tape-Disc Operating System environment. The subprocessing routines within the File Control Processor are designed in a modular fashion so only those routines required by individual programs are included as part of the generated object code. In order to provide the user complete flexibility for input/output programming, the File Control Processor may be used on either of the following levels:

*Logical FCP* is provided via input/output macros included within the Assembler. Utilizing this concept the programmer works at the logical record level and need not be concerned with the physical reading or writing of files.

*Physical FCP* is the actual transfer of data between memory and I/O devices. Executive communication macros are provided to enable the programmer to communicate directly with the Executive and provide information to initiate the physical I/O. The input/output control element of the Executive controls all physical transfers of data to or from peripheral devices.

## LOGICAL FCP

◆ Programs use logical FCP by issuing certain file definition and I/O control macro instructions that provide the following functions:

*File Definition Macros*

1. Describe characteristics of the logical file.
2. Describe the physical device on which the logical file resides.
3. Identify options to be utilized when certain predefined conditions exist.
4. Contain addresses of programmer-written routines.

*I/O Control Macros*

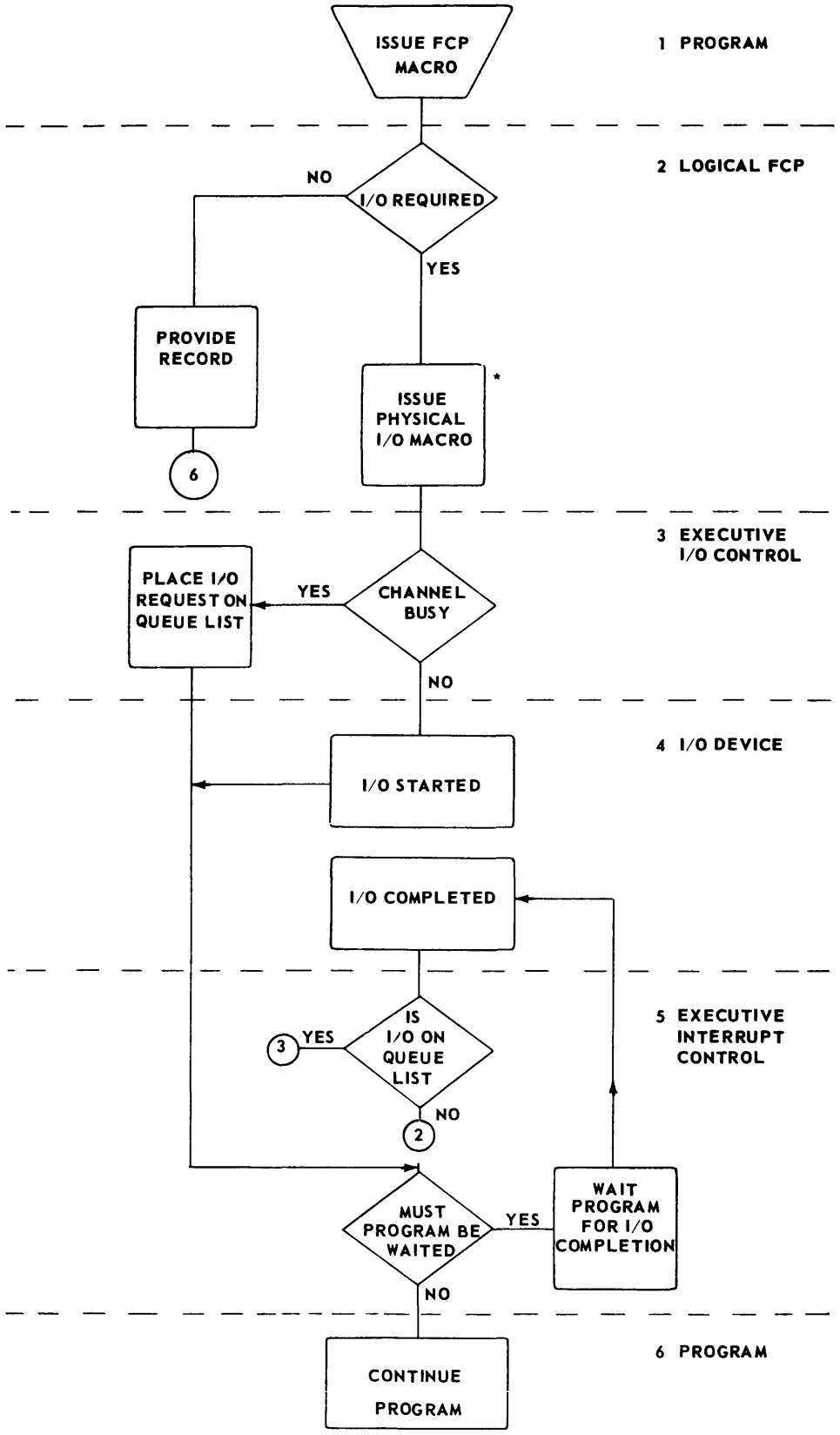
1. Make files available for processing (including label checking).
2. Alternate I/O areas (when two I/O areas are used).
3. Make logical records available for processing (blocking and deblocking).
4. Store logical records after processing.
5. Perform control functions such as rewind and paper advance.
6. Handle end-of-reel and end-of-file conditions.

## PHYSICAL FCP

◆ If the programmer desires to write his own logical data handling routines, physical FCP provides the necessary macro instructions to control I/O. These macros communicate directly with the Executive and provide for reading and writing of data, processing standard labels, nondata operations (rewind, line spacing, etc.), error processing, and checkpointing.

When utilizing Physical Level FCP, the programmer's data-handling routines are responsible for all record blocking or deblocking, data movement to work areas, and interrupt analysis.

The TOS/TDOS FCP and Executive Communication Macros Reference Manual (No. 70-00-608) describes the FCP in detail.




NOTE: Physical FCP would be initiated by the program issuing a Physical I/O macro in Block 2 in Chart.

**Logical/Physical I/O Processing**

# 7. COMMUNICATIONS INTERRUPT ANALYSIS

◆ Communication Interrupt Analysis (CIA) is considered a part of and an extension to the Tape-Disc Operating System Executive. It is executed in the same processor state (Program State 2) and contains the same protection key as the Tape-Disc Operating System Executive.

Communication Interrupt Analysis residency within the Tape-Disc Operating System is determined by the user. The user may generate one of two forms of an Executive, namely:

1. A Tape-Disc Operating System Executive with a communication capability; that is, the Communication Interrupt Analysis section is resident, or

2. A Tape-Disc Operating System Executive without a communications capability; that is, the Communication Interrupt Analysis section is omitted.

The communication-oriented Executive is capable of running non-communication user programs, a communication user program and the Multichannel Communication Program. The Tape-Disc Operating System Executive without the Communication Interrupt Analysis section is capable of running only non-communication user programs.

When the Communication Interrupt Analysis section is resident, it lays dormant until such time as the multichannel communications program is loaded, initialized, and available for communication interrupt servicing. Thus, during the period prior to the loading of the Multichannel Communications Program, no additional processor time is accrued because of the residency of the Communication Interrupt Analysis. Furthermore, after the Multichannel Communications Program has been loaded, the Communication Interrupt Analysis section is executed only during periods of communication activity.

The Communication Interrupt Analysis section is activated solely by the Tape-Disc Operating System Executive as a result of:

1. A Communication Initialization Supervisor Call macro issued by the Multichannel Communications Program in Program State 1;

2. A Wait with Nothing To Do Supervisor Call macro issued by the Multichannel Communications Program in Program State 1;

3. A Wait with Nothing To Do Supervisor Call macro issued by the communication user program issued in Program State 1;

4. Notification that an increment time value specified by the Multichannel Communications Program has expired;

5. A CMGET Supervisor Call macro issued by the communication user program;

## COMMUNICATIONS INTERRUPT ANALYSIS (Cont'd)

6. A CMPUT Supervisor Call macro issued by the communication user program;

7. A communication line interrupt notification; or

8. A conventional communication device (tape, disc, or typewriter) notification.

Hence, all communication activity is funneled into the Communication Interrupt Analysis section from the Tape-Disc Operating System Executive. Furthermore, the Communication Interrupt Analysis section will, after it has been activated:

1. Perform, in some instances, minor processing based upon the type of notification;

2. Determine the type of processing to be performed;

3. Indicate and table an item to be processed (where applicable) for the Communication Interrupt Dissemination section;

4. Effect the necessary procedures to direct and maintain control of the Multichannel Communications Program and the Communication User Program; and

5. Interact with the Tape-Disc Operating System Executive at the proper control points.

# 8. SYSTEM GENERATION

◆ System generation of TDOS is the process by which a TDOS system is created on random access equipment, tailored to the hardware and software requirements of a particular user installation. TDOS system generation also has the ability to produce a Call Library Tape (CLT).

## GENERAL DESCRIPTION

◆ The complete generation of a system is divided into five separate functions. These functions and the order in which they must be executed are:

1. Random Access Volume Initialization.
2. Random Access Storage Allocation.
3. Executive System Generation.
4. Call Library Generation.
5. Executive Library Transcription.

The five functions are executed by the Generation Controller as they are selected by user parameters. Within any one system generation process any number of these five functions may be requested within the following restriction: If more than one function is to be executed it must be requested in the order listed above, and to perform a function its input must be present on the indicated device.

The Program Library Transcriber and Call Library Transcriber do not run under the control of TDOS System Generator. They must be run by themselves outside of SYSGEN.

### Random Access Volume Initialization

◆ Random Access Volume Initialization prepares a volume for allocation as a TDOS Executive Library.

### Random Access Storage Allocation

◆ Random Access Storage Allocation is the process by which storage is allocated for a TDOS Executive Library.

### Executive System Generation

◆ Executive System Generation modifies the TDOS Executive according to the demands of a particular user installation. Optional functions are rejected or selected, and routines and tables are generated in the established locations.

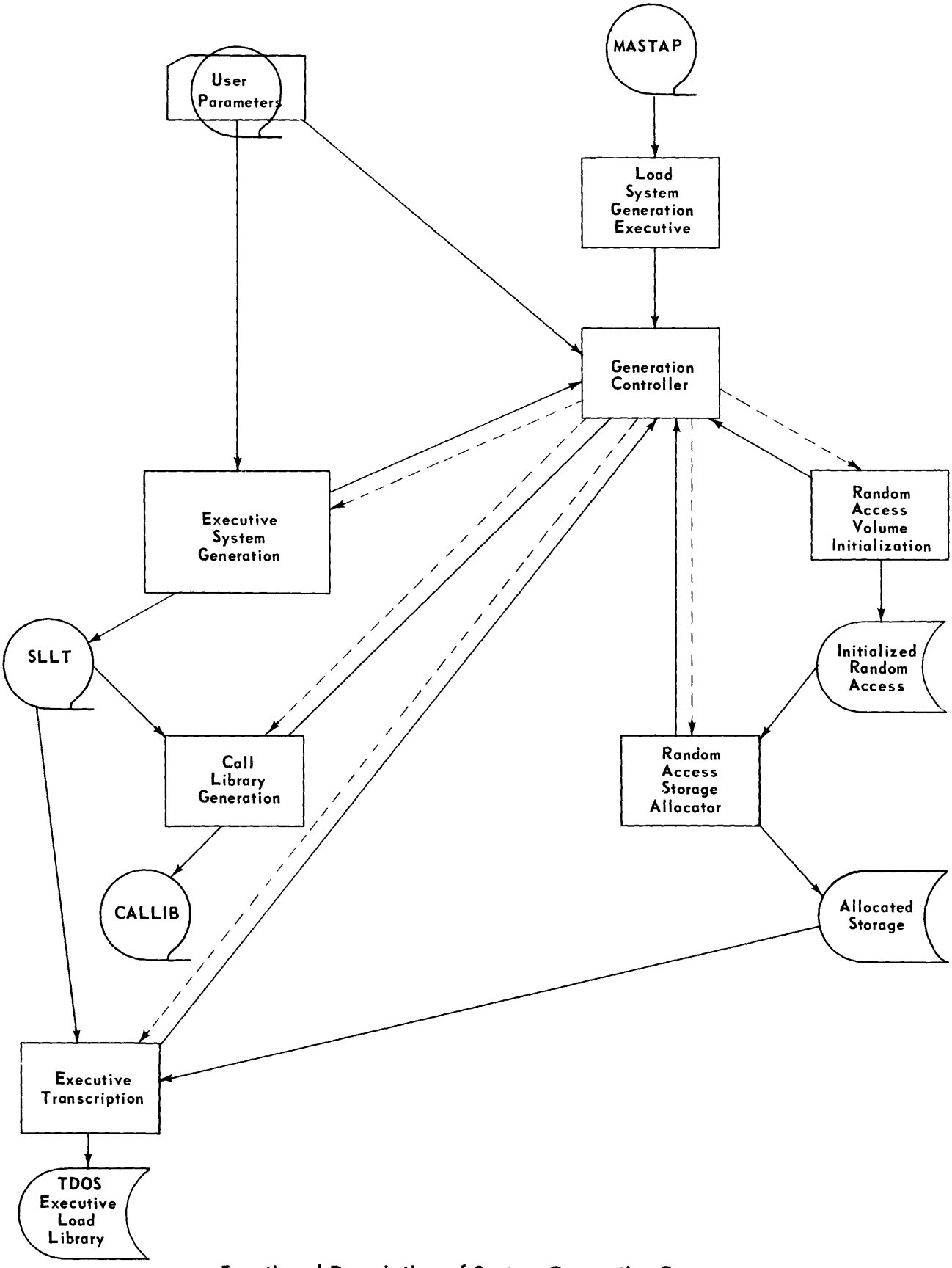
### Call Library Generation

◆ Call Library Generation transcribes system libraries from MASTAP and the TDOS Executive Library from SLLT to the TDOS Call Library tape.

### Executive Transcription

◆ Executive Transcription copies the TDOS Executive to its allocated storage on a random access volume.

Random Access Volume Initialization and Random Access Storage Allocation are described in this manual as to how they function within the System Generation process. For a detailed description on how they function and the parameters required for each, see the appropriate TOS/TDOS Utility Manuals.




**Functional Description of System Generation Process**

**Executive System Generation**

◆ Generates a non-executable system load library tape and/or a call library tape.

**PARAMETER DESCRIPTION**

◆ System generation is controlled by user parameters from the SYSIPT device (card reader or tape) assigned by the operator. These parameters include a description of the devices to be used for the system generation, the devices to be included in the generated Executive, optional Executive features required, system generation functions to be executed, and parameters for the system generation functions.

All // parameters are read by the Generation Controller. Other parameters are read by the programs which require them. If a program terminates before reading all of its input, the parameters are skipped by the Generation Controller until the next // parameter is encountered.

**// LOG**

◆ This parameter is permitted anywhere within the input stream except between a // EXEC and the // END parameter required by a generation function. It causes all subsequent parameters to the Generation Controller and SYSGEN to be listed on the typewriter.

**// ASSGN**

◆ The devices to be used by the generation process are identified by the // ASSGN parameter. A // ASSGN is permitted anywhere within the parameter input except between a // EXEC and the // END parameter required by a generation function. Devices required by a generation function must be assigned before the // EXEC for that function. If the devices are not assigned when the function is requested, the request is rejected. Device assignment is not permitted via the Console Typewriter.

**Device Assignment**

| Symbolic Device Name | Device Type | Remarks |
|---|---|---|
| SYSTAP | Magnetic Tape (Nine-Channel) | Output device which contains the generated System Load Library tape. |
| SYSLIB | Magnetic Tape (Nine- or Seven-Channel tape) | Output device which contains the generated Call Library tape. |
| SYSRES | Disc or Drum | Output device which contains the Executive Library. |
| SYSLST | Printer | Contains the listings of the routines. This is required for Random Access Storage Allocation and Executive Transcription. |

The following parameter identifies the SLLT assigned. It is required by the Executive System Generation, Call Library Generation and Executive Transcription.

//ΔASSGNΔSYSTAP,mn,cuu

where:

mn = two-character installation mnemonic for the device address.

cuu = hexadecimal channel and unit description for the device address.

**Device Assignment** *(Cont'd)*

The following parameter identifies the CALLIB assignment. It is required by Call Library Generation.

//ΔASSGNΔSYSLIB,mn,cuu [,t]

where:

mn and cuu are the same as defined on the previous page.

t = { blank - 9-channel tape.
or 7 - 7-channel tape.

The following parameter identifies the random access device to contain the transcribed Executive Library. It is required by Random Access Volume Initialization, Random Access Storage Allocation, and Executive Transcription.

//ΔASSGNΔSYSRES,mn,cuu,t

where:

mn and cuu are the same as defined on the previous page.

t = device type (see IODS parameter).

The following parameter identifies the printer device. It is required by Random Access Storage Allocator and Executive Transcription.

//ΔASSGNΔSYSLST,mn,cuu

where:

mn and cuu are the same as defined on the previous page.

**// DATE**

◆ This parameter is required to indicate the date of the system generation process. If a version number is to be used by Executive Generation it is also included.

A // DATE must be placed in the parameter stream before the first // EXEC.

//ΔDATEΔmm/dd/yyddd,vvv

where:

mm - two-digit month.

dd - two-digit day of month.

yy - two-digit year.

ddd - three-digit day of year.

vvv - the SLLT version number.

**// EXEC**

◆ Each generation program to be executed is indicated by EXEC parameters. These parameters are listed below in the order that they appear within the input stream. Any of these functions may be omitted in a given generation session.

### *Execute Random Access Volume Initialization*

This parameter calls for the Random Access Volume Initialization function.

//ΔEXECΔRAINIT

VOLIN and END parameters that are required by the Initializer follow in the input stream. See the Random Access Volume Initializer for a complete description of these parameters.

### *Execute Random Access Storage Allocator*

This parameter calls for the Random Access Storage Allocation function.

//ΔEXECΔRAALLR

Parameters that are required by the Allocator follow in the input stream. When generating libraries stored on the System Resident Random Access Device (SYSRES), and/or alternate devices, certain label identification keywords must be specified. They are:

'EXCLIB' - Generated Executive

'PGMLIB' - RCA Supplied routines and user programs.

'ASSEMBLYΔMACROSΔΔ' - Assembly call library.

'OBJECTΔMODULEΔLIB' - RCA and user supplied object modules.

'COBOLΔSOURCEΔLIBR' - User supplied COBOL Source statements.

See the Random Access Storage Allocator for a complete description of the parameter.

### *Execute Executive System Generation*

This parameter calls for the Executive System Generation function.

//ΔEXECΔSYSGEN

Parameters that are required by the Executive System Generator follow in the input stream. These parameters are outlined below.

**EXECP Parameter**

◆ An EXECP parameter is the first parameter expected by SYSGEN. It contains all information, excluding the I/O configuration parameters which are required to generate a resident Executive optimized per the user's specification.

EXECPΔC = nnnnnnnnnn [,0 = mmmm] [,A = pppp] [,M = 1]

*C = nnnnnnnnnn*

◆ This parameter defines the Executive Communication Region Configuration byte and optional Executive routines to be included in the resident Exec. The character n may have the values 0 or 1 and are identified from the left as follows:

$n_0 - n_2$ specifies memory size:

011 = 65K - 70/35-45-55.
100 = 131K - 70/45-55.
101 = 262K - 70/45-55.
110 = 524K - 70/55.

$n_3$ - specifies Memory Protect feature when $n_3 = 1$.

$n_4$ - specifies Elapsed Time Clock option and causes the Timer Package to be part of the resident Exec when $n_4 = 1$.

$n_5$ - specifies the Direct Control option and causes the Direct Control Package to be part of the resident Exec when $n_5 = 1$.

$n_6 - n_7$ - specifies the series of processor:

00 = 70/35.
01 = 70/45.
10 = 70/55.

$n_8$ - specifies Communications Interrupt Analysis option and causes the Communications Package to be part of the resident Executive when $n_8 = 1$.

$n_9$ - specifies Data Exchange Control option and causes the DXC Package to be part of the resident Exec when $n_9 = 1$.

*O = mmmm*

◆ This parameter specifies the size of the On-Line Catalog.

O is an alpha character.
mmmm = size (decimally) of the random access On-Line Catalog.

Formula for computing the On-Line Catalog size is:

9v + 1 = size

where:

v = number of volumes on line.

*A = pppp*

◆ This is an optional entry which specifies the size of the user common data area.

pppp = size (decimally) of the common data area.

*M = 1*

◆ This is an optional entry which requests the Monitor random access I/O function.

**IODS Parameter**

◆ The IODS parameter contains information about each input/output device except the Console Typewriter. This enables the Executive System Generator to construct the Device List and to include the proper error recovery packages in the generated Executive.

IODSΔc, [co] ,tt,mn,uu [,mn,uu . . . ]

[ ,, [ co ] ,tt,mn,uu [ ,mn,uu . . . ] . . . ]

where:

c = channel number 0,1,2,3,4,5,7 (0 = the Multiplexor Channel).

co = co-channel number 0,1,2,3,4,5,7 if none, a comma is required as a separator.

tt = Device type.

- 01 - 70/242 Printer.
- 02 - 70/243 Printer.
- 03 - 70/248 Printer.
- 04 - 70/234 Card Punch.
- 05 - 70/236 Card Punch (Models 10 and 11) - IBM 2540 and IBM 1402 Card Punches (70/293 Controller).
- 06 - 70/237 Card Reader.
- 07 - 70/221 Paper Tape Reader/Punch.
- 0B - 9-Channel Magnetic Tape.
- 0C - 70/564 Disc Storage Unit.
- 0D - 70/565 Drum Storage Unit: 32 Cylinders.
- 1D - 70/565 Drum Storage Unit: 64 Cylinders.
- 2D - 70/565 Drum Storage Unit: 96 Cylinders.
- 3D - 70/565 Drum Storage Unit: 128 Cylinders.
- 4D - 70/565 Drum Storage Unit: 160 Cylinders.
- 5D - 70/565 Drum Storage Unit: 192 Cylinders.
- 6D - 70/565 Drum Storage Unit: 224 Cylinders.
- 7D - 70/565 Drum Storage Unit: 256 Cylinders.
- 8D - 70/567 Drum Storage Unit: 100 Cylinders.
- 9D - 70/567 Drum Storage Unit: 200 Cylinders.
- 0F - 70/568 Mass Storage Unit.
- 10 - 70/627 Data Exchange Control.
- 11 - 70/668 Communication Control Multichannel.
- 12 - 70/350 Switch Controller.
- 15 - 70/236 Card Punch (Models 20 and 21).
- 1A - 7-channel Magnetic Tape (other than below).
- 1B - 1600 BPI Tape.
- 3A - 581 7-Channel Magnetic Tape.
- 5A - 70/441 7-Channel Magnetic Tape.

**IODS Parameter** *(Cont'd)*

mn - installation mnemonic. Although the following groupings are not mandatory, they are recommended.
- 00 - 99 Magnetic tape.
- L0 - L9 Printer.
- P0 - P9 Card punch.
- R0 - R9 Card reader.
- T0 - T9 Paper tape reader.
- U0 - U9 Paper tape punch.
- A0 - A9 Random access devices.

uu - device (physical) address; 00 through FF.

,, - separator between co-channel parameter groups on a single channel.

*Note:*

The first IODS parameter follows the EXECP; it must specify multiplexor device addresses, if there are any assignable devices connected to it. Other channels may be specified at random. Devices connected to two channels must be listed as a device of only one channel. Devices must be specified for at least one selector channel.

In addition, at least one random access device must be assigned.

If information exceeds the capacity of a card, the presence of a non-blank character in column 72 indicates that information is continued in the next parameter card starting in column 16 (col. 1 - 15 blank).

A separate parameter may be used for each device, or as many parameters as desired (within physical limitations) may be placed on one parameter. However, a new parameter must be used when a new channel is being specified.

**ENDP Parameter**

◆ The following parameter indicates the end of the Executive System Generation parameter set:

```
ENDP
```

**Execute Call Library Generation**

◆ The following parameter calls for the Call Library Generation function.

```
//ΔEXECΔGENCAL
```

No parameters are required by Call Library Generation.

**Execute Executive Transcriber**

◆ The following parameter calls for the Executive Transcriber function.

```
//ΔEXECΔLDISK
```

Parameters required by the Executive Transcriber follow in the input stream.

**Execute Executive Transcriber** *(Cont'd)*

The Executive Transcriber places the TDOS Executive routines on disc or drum in the correct system format. It also produces an edited listing of the Program Directory and Load Directories that are created for the Executive.

*Note:*

The Executive Transcriber must be executed under SYSGEN control.

*Preset Functions*

The preset functions of the Executive Transcriber are:

1. Ensure that a good area, large enough to store the Executive system, has been allocated on disc or drum.
2. Transcribe the Executive from a tape created by the TDOS System Generator routine to the disc or drum storage area.
3. Generate a Program Directory for the Executive and Load Directories for the Executive components.
4. Print a listing of the Program Directory and Load Directories.

*Optional Functions*

None.

*Input*

◆ Input to this routine is the Executive portion of a tape created by the TDOS System Generator.

*Output*

◆ The output of this routine is the TDOS Executive on disc or drum and a listing of the Program Directory and Load Directories.

The disc or drum output device must be initialized and allocated before the Executive is transcribed. The filename (entered in the VTOC) for the Executive storage area must be EXCLIB. The size of the area to be allocated will depend on the size of the Executive for the particular installation.

**Device Assignment**

◆ Under Executive or Monitor Control:

| SDN | Device Type | Remarks |
|---|---|---|
| SYSTAP | Magnetic tape. | Input device containing TDOS Executive. |
| SYSRES | Disc or drum. | Output device. Must be initialized and allocated before routine is run. |
| SYSLST | Printer. | Output listing device. |

## Output Examples

### *Executive Program Directory*

```
PROGRAM DIRECTORY
   (1)      (2)          (3)                (4)        (5)        (6)
PROGRAM    START    MAX/MIN MEMY       LD ADDR      DATE      VERSION    RESERVED
TDCSRE     000000   003E80  003E80     01 00 2      072766      001
FCPOVL     000000   0124F8  0124F8     01 00 3      072766      001
MONITB     000048   001800  001004     01 00 5      012667      001
```

(1) Program name of Executive part

(2) Initial load entry point (hexadecimal)

(3) Maximum and minimum memory requirement (hexadecimal)

(4) Disc address of load directory for Executive part (cylinder, head, record)

(5) Creation date

(6) Version number

### *Executive and Executive Error Recovery Overlay Load Directory*

```
EXECUTIVE LOAD DIRECTORY
   (1)        (2)
LOAD   DISK ADDRESS
 00      01 01 1
 01      02 00 1
 02      02 00 2
 03      02 01 1
 04      02 01 2
 05      02 02 1
 06      02 02 2
 07      02 03 1
```

(1) Load number

(2) Disc address of load (cylinder, head, record)

### *FCP and Monitor Load Directory*

FCP LOAD DIRECTORY

| (1) LOAD NAME | (2) DISK ADDRESS | (3) LOAD ADDRESS |
|---|---|---|
| VFCE1 | 03 06 1 | 000000 |
| VFCE2 | 03 07 1 | 000000 |
| VFCE3 | 03 08 1 | 000000 |
| VFCE4 | 03 09 1 | 000000 |
| VFCE5 | 04 00 1 | 000000 |
| VFCF1 | 04 01 1 | 000000 |
| VFCV1 | 04 02 1 | 000000 |
| VFCC1 | 04 03 1 | 000000 |

(1) Load name

(2) Disc address of load (cylinder, head, record)

(3) Program-relative load address of the first text byte (hexadecimal)

*Monitor Overlay Load Directory*

MONITOR LOAD DIRECTORY

| (1) LOAD NAME | (2) DISK ADDRESS | (3) LOAD ADDRESS |
|---|---|---|
| VP000E | 06 00 1 | 000000 |
| VP000C1 | 06 00 2 | 000000 |
| VP000D | 06 01 1 | 000000 |
| VP000E | 06 01 2 | 000000 |
| VP000F | 06 02 1 | 000000 |
| VP000I | 06 02 2 | 000000 |
| VP000J | 06 03 1 | 000000 |
| VP000K | 06 03 2 | 000000 |
| VP000L | 06 04 1 | 000000 |
| VP000M | 06 04 2 | 000000 |
| VP000P | 06 05 1 | 000000 |
| VP000G | 06 05 2 | 000000 |
| VP000R | 06 06 1 | 000000 |

(1) Load name

(2) Disc address of load (cylinder, head, record)

(3) Program relative load address of first text byte (hexadecimal)

## LIBRARY TRANSCRIPTION PROCEDURE

◆ After System Generation has been completed, the Executive has been transcribed to the System resident device and System Load Library (SLLT) and Call Library (CLT) tapes have been generated. Disc or drum storage for the program library (PGML1B) should be allocated under SYSGEN. These libraries must then be transcribed to the system resident device (SYSRES).* This is the procedure to be followed:

1. Load the TDOS executive from SYSRES.
2. Load the Program Library Transcriber (PRGTRN) from the TDOS MASTAP, using MASTAP as an alternate load library.

   Assign SYSUT2 to the SLLT generated by SYSGEN.

   Note: PRGTRN parameters and general operating instructions can be found in the TDOS Utility Manual and TDOS Operators' Guide.

**Note:*
The program library and object module library do not have to be disc-resident. They may remain on tape as alternate libraries.

3. Load the Call Library Transcriber (CLTR) from either the MASTAP as an alternate load library, or from SYSRES.
   a. If space has not been allocated for the libraries, the compute function (using the CDS entry in the parameter) can be used to determine how much disc storage is required. Then the storage allocator (RAALLR) must be run to allocate for the libraries.
   b. After allocation is completed, load the CLTR again to perform the actual transcription.

      Assign SYSUT2 to the previously generated Call Library tape.

      *Note:* CLLR parameters and general operating instructions can be found in the TDOS Utility Manual and the TDOS Operators' Guide.

**DEFINITION OF TERMS**

◆ Master Systems Tape (MASTAP) - The systems tape supplied by RCA to each installation to be used as input to the System Tape Generator. This tape contains the Common Executive (COMEX), Basic Executive (BASEX), RCA System programs in core image format, RCA macros, and RCA Object modules.

System Load Library Tape (SYSTAP) - The generated installation systems tape which contains the Executive, tailored to the installation's requirement, and RCA System programs.

Call Library Tape (SYSLIB) - The installation library tape which contains the RCA Macro Library, RCA Object Module Library, and a copy of the installation's tailored Executive.

Common Executive (COMEX) - A self-loading resident Executive without the Program Control component. It also includes tape read/write and card read error recovery routines as well as the System Tape Generator routine. This Executive not only contains the System Tape Generator routine but is used to control the execution of that routine.

Basic Executive (BASEX) - This Executive contains all of the Executive components in their unmodified form. It is from this Executive that the tailored installation Executive is generated.

Input to TDOS system generation process is a Master Release Tape (MASTAP). The data on this tape is a completely self-contained system and is the only requirement for generation of a TDOS file. The following are contained on the Master Release Tape:

Tape Bootstrap

Tape IPL

System Generation Executive

System Generation Routines
- Generation Controller
- Volume Initializer
- Storage Allocator
- System Generator
- Executive Transcriber

TDOS Executive Library

Systems Programs
- RA Volume Initializer
- RA Storage Allocator
- Executive Transcriber
- Program Transcriber
- etc.

Systems Libraries
- Macro Library
- COBOL Library
- OM Library

## MASTER SYSTEM TAPE FORMAT

◆ The format of the input (MASTAP) to the System is as follows:

Tape Bootstrap

Tape IPL

Program Descriptor Block (PDB)
Load Descriptor Block (LDB)
Tape Executive and Generation Controller Text and Modifier Blocks (T&MB)

PDB

PDB

LDB

Volume Initializer T&MB

LDB

Storage Allocator T&MB

LDB

System Generator T&MB

LDB

Executive Transcriber T&MB

PDB

Tape Mark (T/M)

Random Access Bootstrap

Random Access IPL

PDB Resident Exec. - TDOSRES I

LDB Resident Exec. - RESIEXEC

Resident Exec T&MB

LDB Communications Interrupt Analysis - CIARES I

CIA T&MB

LDB Direct Control - DCRES I

DC T&MB

LDB Timer Routine - TIMERES I

Timer T&MB

LDB Data Exchange Control - DXCRES I

DXC T&MB

PDB Resident Exec - TDOSRES I

T/M

PDB Dummy Program - 00000000

T/M

**MASTER SYSTEM TAPE FORMAT** *(Cont'd)*

PDB Executive Overlays - EXECOVLY

LDB Executive Overlay 1 - EXOVLY01

Executive Overlay 1 T&MB

LDB Executive Overlay 2 - EXOVLY02

Executive Overlay 2 T&MB

.
.
.

LDB Executive Overlay 21 - EXOVLY21

Executive Overlay 21 T&MB

PDB Executive Overlays - EXECOVLY

T/M

PDB FCP Overlay - FCPOVLAY

LDB FCP Overlay 1 - VFCE1

FCP Overlay 1 T&MB

LDB FCP Overlay 2 - VFCE2

FCP Overlay 2 T&MB

.
.
.

LDB FCP Pverlay n - WFRTRT

FCP Overlay n T&MB

PDB FCP Overlay - FCPOVLAY

T/M

PDB Monitor - MONITOR

LDB Monitor - MONITR

Monitor T&MB

LDB Monitor - Job Control Overlay - VP0000

Monitor - Job Control Overlay T&MB

LDB Monitor - SNAP Overlay - WP0000

Monitor - SNAP Overlay T&MB

LDB Monitor Overlay 1 - VP000D

Monitor Overlay 1 T&MB

LDB Monitor Overlay 2 VP000N

Monitor Overlay 2 T&MB

.
.
.

LDB Monitor Overlay n - WP000R

**MASTER SYSTEM TAPE FORMAT** *(Cont'd)*

Monitor Overlay n T&MB

PDB Monitor - MONITOR

T/M

PDB System Program 1

LDB System Program 1

Program 1 T&MB

PDB System Program 1

T/M

.
.
.

PDB System Program n

LDB System Program n

Program n T&MB

PDB System Program n

T/M

PDB Dummy Program - FFFFFFFF

T/M

Macro Library

COBOL Library

OM Library

T/M

T/M

## SYSTEM TAPE GENERATOR EXAMPLE

**GENERAL**

◆ The following example is used to illustrate some of the functions and options of the System Tape Generator.

**SYSTEM CONFIGURATION**

◆ The equipment requirements in the following example are as follows:

Processor

70/45 Model F (65K).

Memory Protect.

Elapsed Time Clock.

**SYSTEM CONFIGURATION** *(Cont'd)*

Selector Channels 1 and 2

1 Dual-Channel Tape Controller, Model 70/473-208.

4 Seven-Channel Tape Stations, Model 70/442.

4 Nine-Channel Tape Stations, Model 70/442.

Selector Channel 3

2 Disc Drives, Model 70/564.

1 Sixty-four Cylinder Drum, Model 70/565.

1 Mass Storage Unit, Model 70/568.

Multiplexor

1 Console Typewriter, Model 70/97.

1 Card Reader, Model 70/237.

1 Card Punch, Model 70/236.

1 Line Printer, Model 70/243.

1 Paper Tape Reader/Punch, Model 70/221.

**SELECTED PROGRAMMING OPTIONS**

◆ The programming options in the following example are as follows:

Memory Protect.

Communications Package (CIA).

Executive Timer Package.

Co-Channeling (Tapes).

On-Line Catalog (Random Access).

The generation of 1 System Load Library Tape (SLLT) and 1 Call Library Tape (CLT).

Common Data Area of 200 Bytes.

*Note:* Cards 13 through 16 in the example may be combined on the same card as all devices are on the Multiplexor. You do not specify communication devices. The entry for the communications is automatically made in the Device List when you indicate CIA is present.

DATA PREPARATION FORM

CHARGE NO. _____
PAGE _____ OF _____
DATE _____
PROGRAM _____
NAME _____
PUNCH ON CARDS ☐
PUNCH ON TAPE ☐
RCA _____ SYSTEM

```
NO. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1.  //  LOG
2.  //  ASSGN  SYSTAP,01,201
3.  //  ASSGN  SYSLIB,02,202
4.  //  ASSGN  SYSRES,A0,140,0C
5.  //  ASSGN  SYSLST,L1,006
6.  //  DATE  03/07/67066,001
7.  //  EXEC  RAINIT
8.  Insert Initializer Parameters Here
9.  //  EXEC  RAALLR
10. Insert Allocator Parameters Here
11. //  EXEC  SYSGEN
12. EXECP  C=011110011 0,0=0100,A=0200
13. IODS  0,,06,R1,01
14. IODS  0,,05,P1,02
15. IODS  0,,02,L1,03
16. IODS  0,,07,T1,04,U1,05
17. IODS  1,2,0A,01,00,02,01,03,02,04,03,,2,0B,05,04,06,05,07,06,08,07
18. IODS  3,,0C,A1,00,A2,01,,,,1D,A3,02,,,,0F,A4,03
19. ENDP
20. //  EXEC  GENCAL
21. //  EXEC  LDISK
22. Insert Executive Transcriber Patches Here if any.
23. //  END
```

28-00-083

**System Tape Generator Example**

## MEMORY LAYOUT TDOS EXECUTIVE

| | |
|---|---|
| Required Basic Executive | Key-In Area |
| | Communications Region |
| | Executive Tables |
| | Interrupt Control |
| | Device Control |
| | User Error Recovery Overlay Area |
| | Console Control |
| | Program Control |
| | Executive Overlay Area |
| Optional Executive Functions | MCP and CUP Error Recovery Overlay Area Lists (1) |
| | Direct Control (2) |
| | Data Exchange Control (2) |
| | Timer (3) |
| | Communications (3) |
| Created by Executive System Generator | MCP Error Recovery Overlay Area (4) |
| | CUP Error Recovery Overlay Area (4) |
| | Device List |
| | Device List Extensions |
| | Channel Queue |
| | Error Recovery Interface Queue |
| | Random Access On-Line Catalog (5) |
| | Common Data Area (6) |

*Notes:*

1. Assembled with the Basic Executive and overlayed by Systems Generator when CIA is not selected.
2. Optional elements of the Executive Assembled with the Basic Executive and floated by System Generator.
3. Optional elements assembled independently and floated by Systems Generator.
4. Generated when CIA is selected.
5. Optional.
6. Optional.

Title TDOS CONTROL SYSTEM REFERENCE MANUAL
Doc. No. 70-00-611
Date July 1969

RCA Information Systems

To ensure that the technical documentation published by RCA fulfills its user requirements, please provide the following information.

Check the category corresponding to the areas listed below. If your answer is "no", or you feel it needs further comment, use the space below or a separate piece of paper giving specific page and line references where appropriate.

| | YES | NO |
|---|---|---|
| Does this publication meet your requirements? | | |
| Is the material legible and understandable? | | |
| Is it organized for convenient use? | | |
| Is it complete? | | |

Comments:

Name _____

Title _____ Street or Box No. _____

Company _____ City _____ State _____ Zip _____

Staple

Staple

Cut Along Line

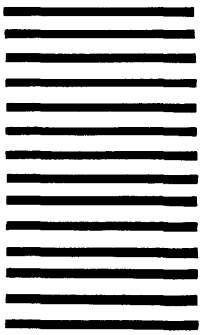
Fold

FIRST CLASS
PERMIT NO. 16
CAMDEN, N.J.

BUSINESS REPLY MAIL no postage necessary if mailed in the United States

Postage will be paid by addressee

RCA INFORMATION SYSTEMS DIVISION
CHERRY HILL, 204-2
CAMDEN, N.J. 08101
ATTN: Publication Services



Fold

70-00-611

RCA
Information
Systems

CAMDEN, N.J. 08101