AA61600-03C

GENERAL SPECIFICATION

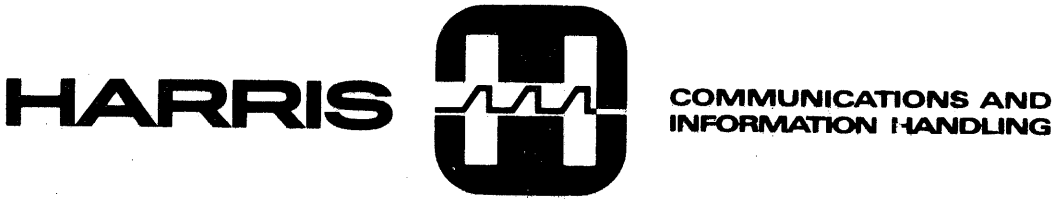
DISC MONITOR SYSTEM

September, 1973

Revision A
May, 1974

Revision B
March, 1975

Revision C
November, 1975



HARRIS CORPORATION Computer Systems Division
1200 Gateway Drive, Fort Lauderdale, Florida 33309 305/974-1700

LIST OF EFFECTIVE PAGES

TOTAL NUMBER OF PAGES IN THIS PUBLICATION IS: 377
CONSISTING OF THE FOLLOWING:

Page No.	Change	Page No.	Change	Page No.	Change
Title	C	5-35	B	9-12	C
A thru B	C	5-36, 5-36A	C	9-13	Original
i thru xiii	C	5-37	B	9-14	B
1-1, 1-2	Original	5-38, 5-39	A	9-15	C
1-3	C	5-40 thru 5-42	B	9-16	A
1-4 thru 1-6	Original	6-1 thru 6-2A	C	9-17 thru 9-19	Original
2-1 thru 2-8	Original	6-3, 6-4	C	9-20	A
3-1	C	6-5 thru 6-8	Original	9-21 thru 9-24	B
3-2	B	6-9 thru 6-18	C	10-1	Original
3-3 thru 3-6	Original	6-18A. 1 thru 6-18A. 7	C	10-2, 10-3	A
3-7 thru 3-9	C	6-19 thru 6-22	Original	10-4 thru 10-6	Original
4-1 thru 4-2A	C	6-23, 6-24	B	11-1	Original
4-3	Original	6-25	A	12-1, 12-2	Original
4-4, 4-4A	A	6-26, 6-26A	C	12-3 thru 12-4A	C
4-5	C	6-27 thru 6-32	Original	12-5, 12-6	Original
4-6, 4-7	Original	6-33	B	12-7	A
4-8	B	6-34 thru 6-40	Original	12-8, 12-9	Original
4-9	Original	6-41	A	12-10 thru 12-13	A
4-10	C	6-42, 6-43	B	12-14	B
4-11 thru 4-13	Original	6-44	A	12-15	A
4-14	B	6-45	Original	12-16	Original
4-15 thru 4-18	C	6-46	B	12-17	A
4-18A, 4-19	C	6-47 thru 6-49	Original	13-1 thru 13-3	Original
4-20, 4-21	Original	6-50, 6-51	A	14-1, 14-2	Original
4-22, 4-23	C	6-52	C	14-3	B
4-24	B	7-1 thru 7-6	Original	15-1	A
4-25, 4-26	C	7-7	A	15-2 thru 15-9	Original
4-27	B	7-8 thru 7-12	Original	15-10, 15-11	B
5-1	Original	8-1	C	15-11A. 1, A. 2	C
5-2 thru 5-5	B	8-2	A	15-12 thru 15-27	Original
5-6	C	8-3, 8-4	Original	15-28 thru 15-29	A
5-6A. 1, A. 2	C	8-5, 8-6	A	15-30 thru 15-40	B
5-7 thru 5-8A	C	8-7	Original	A-1	C
5-9	Original	8-8	C	A-2	B
5-10	A	9-1	Original	A-3	Original
5-10A. 1 thru 5-10A. 4	C	9-2, 9-2A	B	A-4	B
5-11	C	9-3 thru 9-6	Original	A-5, A-6	C
5-12 thru 5-16	Original	9-7	A	A-7	B
5-17 thru 5-22	A	9-8	C	A-8	Original
5-23 thru 5-26	Original	9-9	Original	A-9, A-10	C
5-27 thru 5-33	A	9-10, 9-10A	C	A-11	Original
5-34, 5-34A	B	9-11	A	A-12	C
				A-13	Original

Insert Latest Revision Pages. Destroy Superseded Pages.

LIST OF EFFECTIVE PAGES

Page No.	Change No.	Page No.	Change No.	Page No.	Change No.
A-14	C				
A-15	Original				
B-1, B-2	Original				
B-3	B				
C-1	Original				
C-2	C				
C-3	B				
C-4, C-5	C				
C-6 thru C-8	B				
C-9, C-10	C				
D-1	Original				
D-2 D-3	A				
D-4	Original				
E-1	Original				
E-2 thru E-27	B				
F-1	Original				
F-2 thru F-7	C				
G-1	Original				
G-2 thru G-9	B				
H-1	Original				
H-2	B				

Insert Latest Revision Pages. Destroy Superceded Pages.

CONTENTS

CONTENTS (CONT'D)

CONTENTS (CONT'D)

CONTENTS (CONT'D)

CONTENTS (CONT'D)

CONTENTS (CONT'D)

CONTENTS (CONT'D)

CONTENTS (CONT' D)

CONTENTS (CONT'D)

CONTENTS (CONT'D)

CONTENTS (CONT'D)

ILLUSTRATIONS

TABLES

SECTION I

GENERAL DESCRIPTION

1-1 SCOPE

This document contains the basic operational and programming considerations pertinent to the Series 6000 Disc Monitor System (DMS).

1-2 SYSTEM FEATURES

DMS is a real-time operating system that provides foreground multiprogramming concurrent with sequential batch processing in the background. The foreground is designed for application-related programs which could control a mill, process real-time data from an airborne missle, or interact with multiple terminal users; these programs receive highest priority and their requirements are met first. Batch processing is never time-critical, so the background is serviced when processing time and memory are available.

Salient features of the DMS are:

- Dynamic loading of foreground programs
- Dynamic memory allocation services
- Optional spooled I/O for any output devices
- Optional spooled job stream input from any input device
- Complete accounting system
- Full file security including read, write and delete protection
- Re-entrant foreground program capabilities
- Program priority structure that governs the allocation of memory, disc, and processing time
- Time slicing between programs of equal priority
- Program communications via System Common or parameter passing
- Timer scheduling of periodic foreground programs
- Automatic checkpoint and restore of the background memory area
- Re-entrant editor package for terminal users
- Complete memory protection of all inactive programs from currently active programs
- Concise job control language for batch processing
- Complete operator control over the system environment via the console typewriter or CRT

- FORTRAN IV interface routines for foreground services
- Sequential and direct access methods for data files on disc
- Automatic disc file compression and blocking
- Queued I/O methods for disc storage devices
- Link cataloger that prepares and outputs programs to disc in a format designed for rapid loading and relocation
- System file manager that maintains program and data files in source and object formats
- System generation procedure that adapts DMS to each configuration

The manner in which DMS operates and the services that it provides are described in the following paragraphs.

1-3 FOREGROUND OPERATION

A foreground program is a program executed under control of the DMS Dispatcher at a priority above the background. A foreground program can utilize the DMS I/O handlers and all other system services.

1-3.1 Program Initiation

Foreground program can be initiated by external interrupt, by timer schedule, by console keyboard, by remote terminals. and by other foreground programs. Selected foreground programs can be designated as core-resident;such programs remain in core even when they are inactive so they can be initiated quickly. Other foreground programs are designated as non-resident; they are allocated memory and loaded from disc as they are needed. Optionally, these non-resident foreground programs can make themselves temporarily resident. Foreground programs which do not modify any memory within themselves can be designated as re-entrant. If multiple requests are made for initiation of a re-entrant foreground program, then they will all share the same body of the program and execute concurrently within the priority structure.

1-3.2 Execution Priorities

When any foreground program is initiated, its execution priority is specified to DMS. This priority determines the order in which programs are serviced, governs the allocation of memory and processing time and determines the order in which entries are placed in the disc queue. The priority of the external interrupt that is used to initiate a foreground program has no influence on the execution priority of that program. In many cases, several foreground programs have equal importance and require similar system response. These programs can be initiated with equal priority values and memory processing time will be shared between them on a time slicing basis. Foreground program execution priorities range from 1 through 254 with 254 being the lowest and therefore serviced last, but immediately before background batch processing, whose priority is fixed at 255.

1-3.3 Program Cataloging

Any program that is to be executed under DMS must first be output onto a disc file. This means that it must be presented in object format (as output from the Assembler or Compiler) to the Link Cataloger which produces a load module of the program and saves it on disc. The

load module consists of the program linked with all necessary subroutines in a basic relocatable format that can be loaded very rapidly.

The Cataloger also saves several program attributes and characteristics with the load module on disc. The memory size needed to execute the program, the program type, the execution mode. and the starting address are preserved for all programs. For foreground programs, initial device and file assignments may be specified to the Cataloger and saved on disc.

The Cataloger technique thus provides DMS with all the information concerning each program such that, when a foreground program is initiated, the program can be allocated memory, loaded, relocated and executed quickly.

1-3.4 Background Checkpointing

Whenever the available memory space is not sufficient for a foreground program that is being initiated, the checkpoint procedures is automatically invoked. All I/O initiated by background is allowed to complete, background execution is suspended, and the entire content of the background memory area is saved on disc. That memory is then available for foreground programs and dynamic core requests by foreground programs. When foreground activity subsides, and the background memory area becomes available again, the background is reloaded and its processing resumed.

1-3.5 Program Communications

DMS provides several techniques through which data and status information can be transferred from one foreground program to another. Initially, all foreground programs are passed an initiation parameter which is loaded in the A-register when the program is started. This parameter is specified by the program making the initiation request or by the operator if the initiation was by console keyboard.

For small quantities of data. such as program status, the Program Switch Word is a convenient communication medium. Every active program has a switch word that it can set to any specified value using a system service. A corresponding service is available that allows a program to obtain the content of the Program Switch Word of another program.

For larger quantities of data, a reserved area of System COMMON storage is available. The block has the dedicated label SYSCOM, and that label cannot be used by any program for its local COMMON blocks. All programs declaring the COMMON block SYSCOM are linked to this system communication area. For still larger quantities of data, or where immediate access is not required, programs may itilize disc files for communications.

1-3.6 Program Restrict

A system integrity safeguard is provided by DMS in conjunction with the Program Restrict/Instruction Trap Option. Any program operating in the Restricted Mode will be trapped and aborted by DMS in the event that it attempts to transfer to, or alter a memory location outside its allocated space, or that it attempts to execute a privileged instruction, such as Halt. This feature protects DMS and all other programs from inadvertent destruction due to a program error. The restricted mode is the normal execution mode for all background and regular foreground programs. Re-entrant foreground programs running in the Restricted Mode are allowed to access any location except the resident system. Programs which must execute special instructions, such as input/output, or alter memory outside their allocated space, can be cataloged to execute in the Privileged Mode.

1-4 INPUT/OUTPUT FACILITIES

All input/output operations under DMS are originated by a system service call in which the calling program specifies a logical decive number, a function code, and any parameters necessary to define the operation. The Input/Output Controller resolves the logical device assignments, preforms some validity checks on each I/O request, and then transmits the descriptive information to the appropriate device handler. The handler determines whether the device is available and, if so, initiates the requested operation. Control is then returned to the user as the I/O operation will be completed using the device interrupts. While waiting for I/O operation to be completed, a program can either continue processing data which is not dependent on the I/O operation, or it can relinquish the use of the central processor to lower priority programs during the wait.

The Input/Output Controller maintains the allocation and status information for all devices. The card reader, line printer, paper tape devices, remote teletypes, and magnetic tape transports are considered as allocatable devices; i.e., they cannot be used in an orderly fashion by two programs simultaneously. Such devices are allocated to the first program requesting to "open" the device. That program then has exclusive use of the device until it "closes" the device, or until the program is terminated. The console teletypewriter is not allocated to any single program since it must always be available for operation communications functions. All disc drives are considered to be shared devices and receive special treatment. All disc input/output requests are placed in a queue, and the queue is serviced on a priority basis.

Symbolic output may be spooled out to any allocatable device providing that the DMS system and the device are configured as spooled. This means that I/O from the program is sent to a disc file, and when the "device" is closed by the program, the file is copied to the specified device by a re-entrant spooling program. The disc file used in this operation is dynamically created when the "device" is opened, and deleted when the file has been completely copied to the device.

1-5 FILE SYSTEM CONCEPTS

The File System allows the disc storage space in a Series 6000 configuration to be distributed among a large number of multi-sector files which can be referenced by symbolic name. A password facility allows the user to secure a file such that it cannot be read, written, or deleted by any program which does not supply the proper password. In addition, in accounting configurations, the individual user can maintain total control of his file through use of the user-number which is attached to the file. The file creator may allow at his discretion, other users to read, write or delete file, and each of these abilities can be specified separately. The proper password, if any, must of course, be supplied.

A single file can be established in any specified length on any specified disc pack in the system; and each file is always allocated contiguous sectors. The Master Disc Directory (MDD) contains complete information on file names, file extents, file types, passwords, and in accounting configurations, the file creator's user-number and file access bits. The Space Allocation Map (SAM) maintains storage allocation data for each disc pack. The directory resides on disc on the master system pack, and the individual maps reside on their respective packs, from which they are read into memory when needed.

Any program that needs to access a disc file must first "open" the file. When a file is opened, the entry for that file is located in the directory via a hash-coding technique and read into memory. The password, if any, is checked, and in accounting configurations, the user number and file access bits are checked to determine whether the file can be read

or written or neither or both by this user. The file extents are saved in memory and the disc pack on which the file resides is located. If the pack is not currently mounted on any drive, then the operator will be asked to mount the pack on any available disc unit. The file type is checked to determine the blocked/unblocked status of the file. The calling program can then access the data within the file or write new information into it. If the file is typed as blocked, multiple records are automatically blocked into a sector. In addition, symbolic data is compressed by removing strings of blank characters. In this mode, the calling program has no control over physical disc sectors and can only read/write single records. For unblocked files DMS starts the transfer of the beginning of a sector and transfers data until the user's word count is satisfied. In the direct access mode, the calling program must specify the record number within the file before making the data transfer request. For blocked files, this record number is located in the blocked data via an optimized search technique. For unblocked files, the record number is used as a relative sector number within the file.

Files can be created by the Link Cataloger, Library File Editor, File Manager, ACRONIM, and any user program. The Link Cataloger creates files when it catalogues load modules of system and user programs. The Library File Editor permits link module editing of the Link Library file by performing such functions as add, delete, replace, and list. The File Manager performs batch background file managing; provides a means of saving and restoring files using magnetic or paper tape; and prints a disc storage map indicating the current files and various parameters about each. ACRONIM provides the remote terminal user with many file utility operations such as creation, deletion, copying, etc.

Sector III of this manual describes the disc and file system in further detail.

1-6 BATCH PROCESSING CAPABILITIES

The batch processing capability of DMS allows the user to perform data reduction, scientific computation, and program preparation in the system background without interfering with the operation of interactive or real-time foreground programs. Batch input streams may be spooled in from any input terminal configured in the system. This allows the remote terminal user as well as the batch card reader to submit background job streams for execution. Background jobs are queued based on terminal priority and control card information.

Batch processing is performed as directed by job control statements expressed in a concise Job Control language. They are used to specify the program options and device assignments and to request the execution of system and user programs. Standard processors available for background execution are as follows.

- DMS Link Cataloger
- DMS File Manager
- DMS Library File Editor
- DMS Debug
- Utility Package
- Basic Assembler
- Cross Reference Program

- FORTRAN IV Compiler and Support Library
- RPG II
- SNOBOL

1-7 ACCOUNTING SYSTEM

The Accounting System is a functional subset of DMS which provides for device and system utilization accounting. This is a optional part of DMS and can be configured in or out whenever a system is generated.

In an Accounting System, DMS requires the use of a user-number to identify all activities. The use of this user-number within the file system has been discussed previously. Additionally, the user-number is associated with all records of usage of the system facilities. That is, whenever a program is run, the user-number is used to identify the resultant Accounting Records. Accounting Records are maintained for all devices within the system, including all peripheral devices, system disc drives, and CPU core and time usage. The format of these records is discussed in Appendix B.

Also provided with each DMS Accounting Configuration is the DMS Accounting Utility Program "ACCUP" which provides a convenient means of summarizing the accounting information and maintaining the accounting file. Section VII should be consulted for further information on Accounting.

1-8 DMS CONFIGURATIONS

Due to the extreme flexibility of DMS, and the extensive scope and power it possesses, the Disc Monitor System can be configured into a number of different systems. Of course, all DMS systems are tailored to the individual customer's Series 6000 hardware configuration, which provides the variety of peripheral devices, external interrupts, hardware options, and operation characteristics. However the DMS system itself can be supplied in various options. These options and their effects on this document are discussed below.

The major DMS option is whether the system is spooled or unspooled. In order to do any spooling operations, or to run interactive remote terminals, including using the operator communications device as an interactive terminal, the DMS system must be configured in the spooled mode. The basic unspooled DMS does not support the interactive terminals or spooling, but does constitute a powerful monitor system useful for multiple foreground and single batch background multiprogramming. The spooled option, which adds approximately 1200 locations to the size of the resident DMS system, makes DMS additionally a powerful time-sharing and high volume batch operating system.

Another option, available only with the spooled option, is the accounting system. The accounting system provides device utilization information, identifiable to individual users for all system devices including disc and CPU. User identity is established through use of a required user-number, which must be used for all operations on the DMS system. Additionally, the accounting system provides for an extremely flexible file security system, based on user-numbers, which allows the creator of a file to allow any combination of read, write, or delete access to be given to other users.

This manual covers the full spooled accounting configuration of DMS. However, where applicable, items which are part of only a particular configuration are noted in the text.

SECTION II

DMS OPERATION

2-1 SCOPE

This section is intended to give the user of DMS some general background knowledge about the operation of the monitor system. This includes the concept of foreground programs, batch background, and how DMS itself operates.

2-2 PROGRAMS

DMS can run any number of programs concurrently. That is not to say that they all can execute CPU instructions simultaneously, which would be impossible with only one CPU, but actually that DMS selects a program to run at a given instant based on needed resources, priority, etc. This interaction will be discussed in paragraph 2-4. Since the outcome appears to be multiple programs executing concurrently, DMS can be called a "multi-programming" system. What are actually running are any number of foreground programs, and one background or "batch" program.

A "program", whether the background or a foreground, always consists of an entry in an "Active Program List " (APL) and a block of storage called a "Program Service Area" (PSA). Associated with the Program Service Area is a block of instructions and data, which is a user program. Thus, as far as DMS itself is concerned, it is executing a Program Service Area, which in turn defines the instructions to be executed by pointing to the block of program code. The Program Service Area and Active Program List will be discussed later in this section.

2-2.1 Foreground Programs

A foreground program is a user program that has been catalogued (i.e., "linked") to produce a load module on disc. All foreground programs reside on disc, and may also become core resident when DMS is loaded. Foreground programs are those which perform a specific task or tasks when requested. For instance foreground programs might control processes within a factory; control various functions of a nuclear reactor; respond to external stimuli coming from an airborne missile; control interactive terminals; or be used to compute some repetitive calculations based on laboratory experiments.

Under DMS, foreground programs can be activated in a number of different ways. These include:

- Computer operator request
- Request from any other program
- External interrupt
- Time of day activation
- Periodic activation of a specified frequency
- Remote terminal user request (spooled version of DMS only)

Once a foreground program has been activated, it shares time with other programs as discussed below until it has completed its task. This could be anywhere from a few milliseconds to a several hours. A foreground program remains resident in memory from the time it is activated, until it has completed its execution and "EXITed" back to the system.

There are three distinct types of foreground programs under DMS. These are Resident, Non-resident, and Non-resident Re-entrant. Resident foreground programs are allocated memory and loaded into core when DMS is loaded from disc. They remain resident at all times, even when they are not being used. Resident foreground programs are used where rapid response time is necessary. That is, it saves the disc lookup and load time when the program is activated. However, they do use up resident memory.

Non-resident foreground programs are user tasks that reside on disc and are allocated memory and loaded from disc whenever their activation is requested. When the Non-resident foreground program "EXITs" to the system, its core is released and made available for other programs. This is the most common type of foreground program.

Multiple copies of Non-resident foreground programs are allowed when the second or subsequent request is made from another program, the operator console, or a remote terminal. Multiple copies cannot stackup in memory via the scheduler or external interrupt requests.

Non-resident foreground programs may, at their request, become temporarily resident. That is they remain in core after "EXITing" to the system similar to resident foreground programs. This is accomplished through use of the TERMIN System Service described in paragraph 4-5.

The third type of foreground program is the Non-resident Re-entrant foreground program. Re-Entrant foreground programs are those whose code is written such that it can be concurrently executed by more than one user. That is, the code does not modify itself, or use any fixed data storage cells within the program body. All variable storage must be done in some external block which is associated with the individual program. Under DMS, each user of a Re-entrant program has his own Program Service Area. If there is more than one such user of a Re-entrant foreground program at any given instant, then <u>all</u> of the respective PSA's are connected with the same program body.

Re-entrant foreground programs running under DMS must use either the Program Service Area or a Dynamic Core Block for all data storage to insure re-entrancy. Such foreground programs need not be concerned with the number of concurrent users; in fact they do not know such information. Re-Entrant foreground programs are written as any other foreground programs might be; with the restrictions that they must not use data storage within the program or modify the program text in any way. As an example, this excludes use of the BSL (Branch and Save Long) instruction, since it modifies the program body. If a foreground program meets these requirements, then it may be typed as a Re-entrant foreground program when cataloged (see paragraph 6-4) and the DMS system will handle the multiple users.

What actually happens with Re-entrant foreground programs under DMS is that the first request for such a program results in loading the program and a distinct Program Service Area. Another request for the same program results in only the creation of another PSA, which points to the same block of code as the other service

area. When a particular user of the program (PSA) exits, only the PSA memory is released. DMS maintains with the Re-entrant program body a count of the current number of users of the program. This count is incremented every time a new PSA comes in for the program and decremented when a user (PSA) exits. When this value drops to zero, the program body itself is removed from core. A subsequent request for the program will cause a reloading of the program from disc.

One additional feature provided for foreground programs is that whenever a foreground program exits (i.e., calls the EXIT or TERMIN System Service), DMS will Close/Deallocate all of the files that may have been opened by the program, and return all Dynamic Core Blocks that the program allocated back to the system. This feature insures system integrity and proper resource management, and relieves the programmer of the necessity of performing these tasks at the end of his program. This performance is slightly different than that for background programs, as will be shown in the following paragraph.

2-2.2 Background Programs

Background Programs are those user system programs which are designed to run in a "BATCH" environment, where execution is never time-critical. All of Datacraft's standard software processors are designed to run in background (excluding ACRONIM and BASIC which are Re-entrant foreground programs). Thus, all compilations, assemblies and link catalogings are run in the background.

The background memory area is a fixed size block of memory beginning at the high end of the resident DMS and continuing to some arbitrary location in available memory. This fixed value can be changed via the operator communications keyin "MB" (Modify Background) as discussed in Section IX. Whenever foreground programs are allocated memory that overlaps background, then the entire background memory area is written to disc (checkpointed). This will only happen, of course, after all current background I/O operations have been completed.

All background processors reside in this same area of core and start at the initial background location. Thus there is never more than one background program in core at any time.

A Background "Job" merely consists of a series of background program executions. For instance, consider the following background job, which is a Fortran compilation and execution.

```
$JOB
$FORTRAN
     (Fortran program)
$CATGO
$EOJ
```

Initially, the background processor Job Control ("JOBCTL") is executed to process the $JOB and $FORTRAN statements. Next the Fortran Compiler is executed to process the Fortran source program. When it EXITs, Job Control is again

executed to process the $CATGO statement. Next, the Link Cataloger is executed to produce a load module of the user program on disc. The Cataloger calls in the user program for execution. When the user program EXITs to DMS, Job Control is once again executed to process the $EOJ statement. This completes the execution of the job.

The reader will note from the above discussion, that whenever a background program EXITs to the system, the Job Control processor is activated in its place. This is an important difference from foreground operation. Note also that this is the only operation performed when a background processor exits. No files are closed and no Dynamic Core Blocks are deallocated. This allows information from one processor to be passed on to another processor during the execution of a background job. A useful application of this operation is in a job with multiple compilations and/or assemblies, the Binary Output Disc file will be left open and correctly positioned from one assembly to the next.

There is one instance, however, when background files are closed and Dynamic Core Blocks returned. This is when the Job Control program EXITs, which happens after processing a $EOJ statement. Thus the end of a background job produces the same cleaning up of system resources that is done when any foreground program EXITs.

2-3 MEMORY ALLOCATION

As stated in the preceding paragraph, the background memory area is fixed in size and location. However, memory for foreground programs and Dynamic Core Blocks is allocated as discussed in the following paragraphs. An example of the division of memory is shown in Figure 2-1.

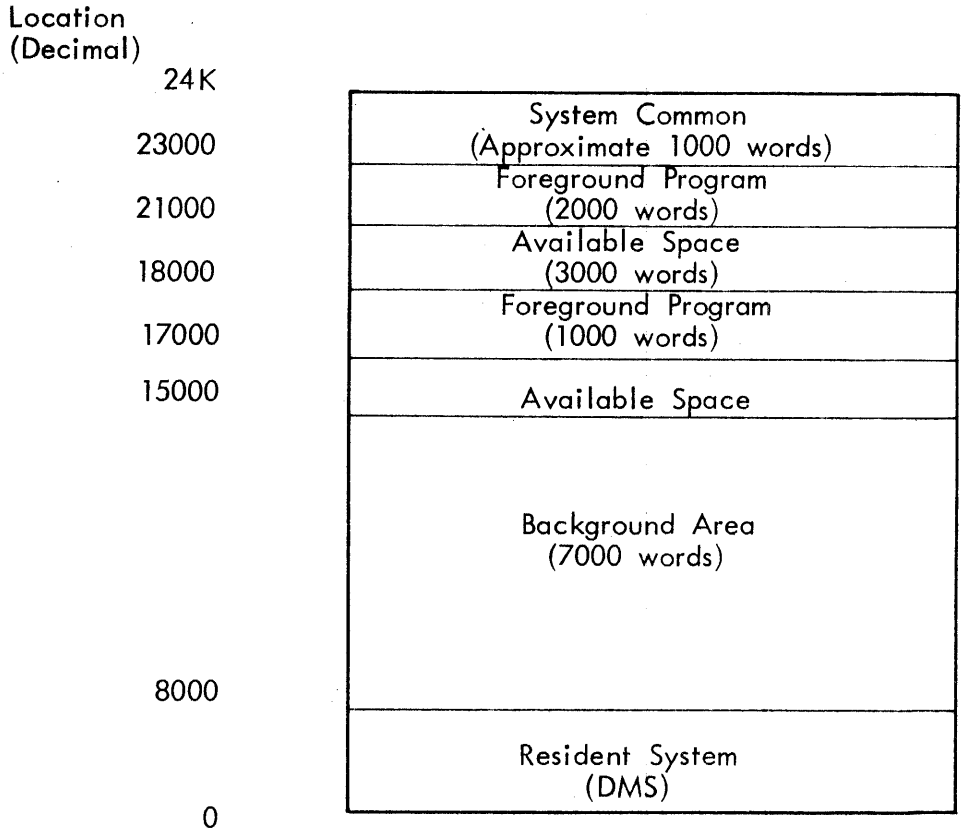
Whenever a request for memory comes to the memory allocator, whether it is for a foreground program, a Program Service Area, or a Dynamic Core Block, the allocator searches from the highest memory location (in the foreground area) down until it finds the first available block of memory that will contain the request. If the core block reaches down far enough, into the background area, then background must be checkpointed as discussed in paragraph 2-2.2. However, if the request that would cause background to be checkpointed is actually a request from background for a Dynamic Core Block, then the request is not processed and notification returned to the calling program that core is not now available.

Using Figure 2-1, if a request for a core block of less than or equal to 3000 words was received by the allocator, it would be allocated from the block at location 18000-21000 - in fact, if the request were less than 3000 words, the allocated block would be up against the location 21000 barrier and the remaining free space would begin at location 18000.

If a request for more than 3000 words were received, it would be satisfied by allocating a block of the required size immediately below location 17000. This also would necessitate checkpointing background, since the requested block would continue below location 15000.

Figure 2-1

Typical Memory Layout



2-4 PRIORITY STRUCTURE AND PROGRAM INTERACTION

As was memtioned previously, DMS maintains an entry in the Active Program List (APL) for each program in the system, including an entry for background, which handles whatever background processor is active. These APL entries are illustrated in Appendix A, but for purposes of discussion, here we need only be concerned that an entry contains the program's priority and status word. The Program Status Word is illustrated in Table 4-2.

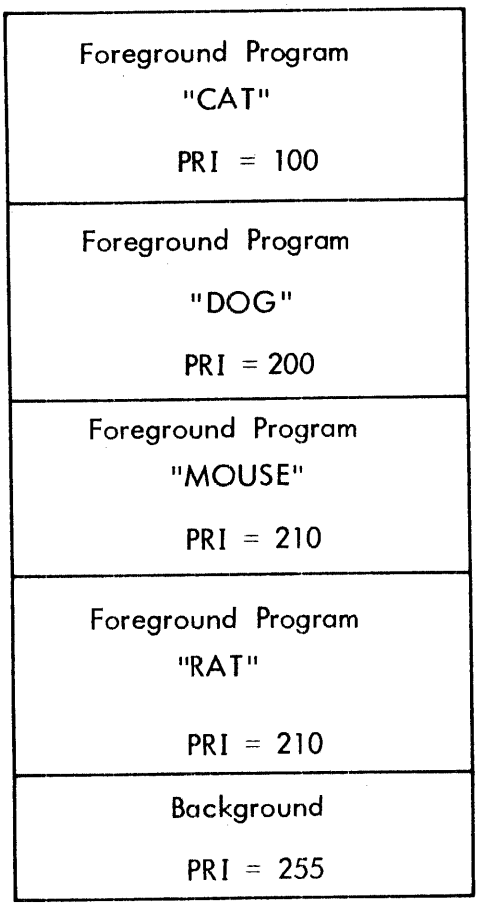
Active Program List entries are ordered by priority, with background always occupying the lowest slot, since it has the lowest priority (255). A sample Active Program List is shown in Figure 2-2. If two entries have equal priority, then their ordering is for the moment insignificant. This situation will be discussed in paragraph 2-4. 2.

2-4. 1 Context Switching

Before a discussion of how DMS selects programs can be made, we must first define a "significant event". A Significant Event is any operation whose result could cause a program whose execution was suspended for some reason to once again continue execution. For instance, if a program were waiting for a certain I/O operation to be completed, then the completion of that operation would be considered a Significant Event.

Figure 2-2

Sample Active Program List



When a Significant Event occurs under DMS, the routine performing the event executes special instructions which cause a Dispatcher Interrupt. This Dispatcher Interrupt is the lowest priority external interrupt in a DMS configuration. This is to insure that all interrupt processing is complete before the currently active program is interrupted by the Dispatcher. The Dispatcher Interrupt must interrupt program execution and must not interrupt an interrupt routine.

When a Dispatcher Interrupt occurs, as caused by a Significant Event, DMS is said to go through a Context Switch. During a Context Switch the following occur. Firstly, all register and program execution information (program counter and condition register) are stored in the currently active program's Program Service Area. Next, the Dispatcher scans the Active Program List from the top down, examining each program's status word until it finds a program that can be executed. This top down scanning maintains the priority structure of DMS. If no program can be run, then the dispatcher enters an idle loop awaiting another Significant Event which will produce a Dispatcher Interrupt. A Context Switch is also performed, of course, whenever the currently active program can no longer continue, such as when it must wait for the completion of an I/O operation.

Consider the following examples of Context Switching. Referring to Figure 2-2, program "CAT" is active and executing. Programs "DOG" and "MOUSE" are both in a WAIT state, waiting for the completion of an I/O operation. If at some time, an external interrupt occurs which signifies the completion of program "MOUSE's I/O operation, then the program "MOUSE" will become executable, although it will not actually be active yet. The particular interrupt routine will consider this a Significant Event and trigger the Dispatcher to perform a Context Switch. The Dispatcher will scan the Active Program List and first check program "CAT" and, finding it executable, will restart it. At a later time, program "CAT" starts an I/O operation and desires to wait for its completion. A Context Switch will occur and the Dispatcher will scan the Active Program List from the top. It will find both programs "CAT" and "DOG" waiting for I/O operations to complete and thus it will put program "MOUSE" into execution.

2-4.2 Timeslicing

Timeslicing is a feature of DMS wherein Foreground Programs of equal priority will share execution times so that they all have available about the same percentage of total execution time available to that priority level on the Active Program List. This is accomplished through use of the 120 Hz Clock. A timeslice period is defined as a set number of 120 Hz clock "ticks". This number can be modified via an operator communications keyin. (timeslicing can also be turned off by this keyin.) The default is two "ticks" or 1/60 of a second. Whenever this timeslice period is completed, DMS checks to see if there is another program which is at the same priority as the currently active one. If so, DMS effectively moves the currently active program to a position on the Active Program List below all other programs of that priority, declares a Significant Event, and triggers the Dispatcher to perform a Context Switch.

Consider a continuation of the previous example. If the timeslice period is over while program "MOUSE" is active, then DMS will interchange the entries for programs "MOUSE" and "RAT", and produce a list similar to that shown in Figure 2-3. At this point a Significant Event, and therefore a Context Switch, would occur, and providing program "RAT" had executable status, it would be made active.

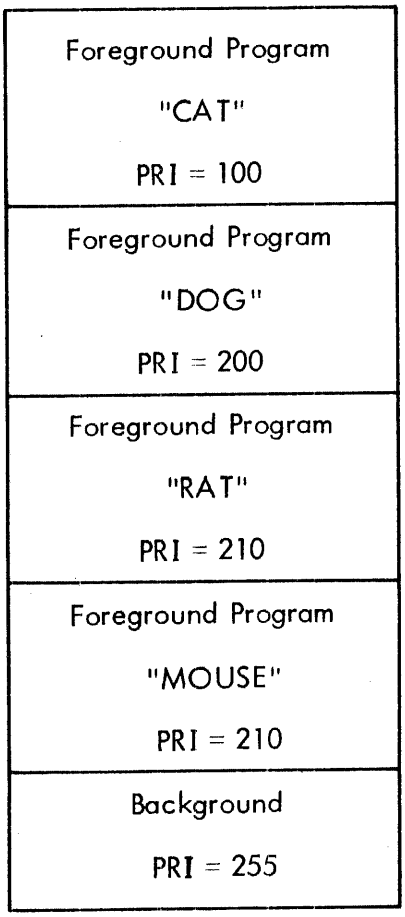


Figure 2-3. Sample Active Program List After Timeslice Switch

SECTION III

DISC AND FILE STRUCTURE

3-1 GENERAL

DMS will support up to 32 disc drives per system on which can be mounted a total number of 255 unique disc packs, of which one must remain mounted and is called the Master Pack. On single disc systems, only the Master Pack is used. However, if at least two disc drive units are present, then DMS supports up to 254 removable disc packs which can be mounted on the drive (s). It is important to note that Datacraft Series 5200 Cartridge Disc Units containing both a fixed and removable platter (Model 5204 and 5208) are considered as two disc units for DMS operation. The fixed platter is one unit and the cartridge is a second unit.

The space on each disc pack can be divided into variable length segments, called disc files. Disc files may then be referenced symbolically by name during program execution under DMS.

3-2 DMS MASTER PACK

All DMS configurations have a master disc pack. The master pack is pack number one. It contains all of the system information directories, etc., to maintain operation of the DMS system. The remaining space is allocated to user file storage.

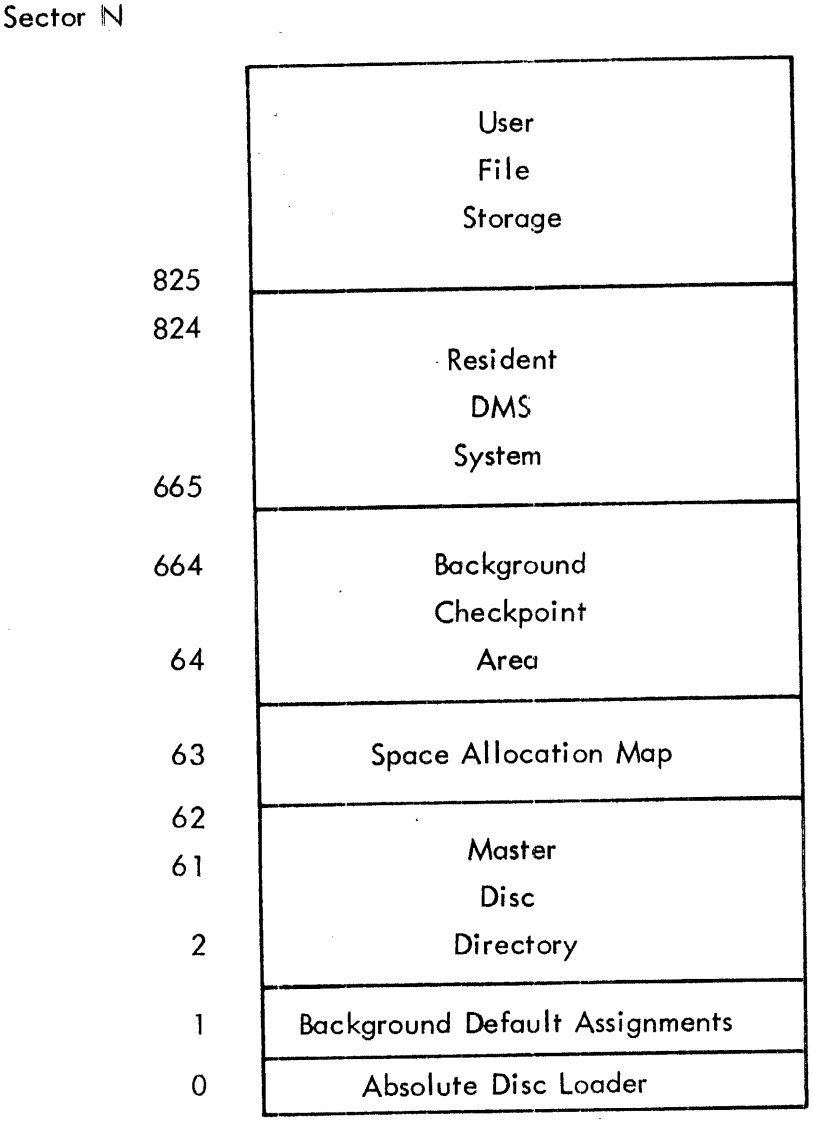
The master pack must remain mounted on the master disc drive at all times to insure proper DMS operation.

The master pack contains the following items (Refer to Figure 3-1 for sample layout giving relative sizes):

- Absolute disc loader, which resides on sector 0, used to load the resident DMS from disc to core.
- Background Default Assignment Storage, which is a one sector table which contains all of the default assignments for background jobs.
- Space Allocation Map (SAM) for the master pack, which maintains a record of which sectors are used and which sectors are available for user file storage. The SAM is always two sectors long.
- Master Disc Directory (MDD) which contains an entry for every user file on any pack within the system. The MDD entry gives file location, password, security, and type information.
- Background Checkpoint Area, which is the area on disc used to hold the copy of the background memory area when background is checkpointed.

  Optionally, this area can be on any other permanently mounted disc pack in the system.
- Resident DMS - This is the disc copy of the core resident DMS system, which is loaded by the absolute disc loader whenever DMS is loaded from disc.

Figure 3-1. Sample Master Pack Layout



3-3 MULTIPLE DISC UNITS

On configurations employing multiple disc units, disc packs containing user files may be interchanged on the "satellite" drives (all except master pack). Removable disc packs must all be identified by a Pack ID Number, which is between 2 and 255, inclusive. Pack ID Numbers are established on removable packs through use of the CLEAR statement of the File Manager (Refer to Paragraph 6-3.14).

The first four sectors of "satellite" packs are reserved for system information, and the remaining are available for user file storage. These first four sectors contain a two sector Space Allocation Map, a Pack ID Number, and a special bootstrap simulator.

The bootstrap simulator causes the Series 6000 Control Console lights to be flashed in a unique pattern if an attempt is made to bootstrap (or load) a program or operating system (such as DMS) from the pack.

All files created on "satellite" packs are identified by entries in the Master Disc Directory on the master pack. Whenever a file is referenced by a DMS program that is located on a removable pack, DMS determines whether or not that pack is currently mounted on a disc unit. If not, the computer operator is asked to mount the pack on any available drive. When this has been done, the operator notifies the system via a keyin that the pack has been mounted, and that program's execution can then continue.

3-4 MASTER DISC DIRECTORY

The Master Disc Directory (MDD) contains an entry for each user disc file in a DMS system. Each such entry contains the following information. (A complete MDD entry layout is given in Appendix A.)

- File Name
- Pack Number
- File Extents (First and last sector numbers of file)
- File Type
- File Access Bits (Accounting System only)
- File Password
- User-Number of user who created the file (Zero for non-accounting systems)
- Load Module Core Requirements
- Load Module Absolute Starting Address

The latter two items are described in the discussion of the Link Cataloger, paragraph 6-4. File Type, File Access Bits, File Password and User Number are discussed below in Paragraph 3-5.

The particular sector in the Master Disc Directory in which the entry for a given file is located is determined by a hash-coding algorithm. That is, by applying some arithmetic operations to the file name, a relative sector number within the directory is computed. This hash-coding allows rapid file access because it eliminates searching more than one sector of the directory to find the entry for a specified file.

The size of the Master Disc Directory is variable and is set at SYSGEN time. However, a single restriction is that one parameter, the maximum number of entries in the directory, must be a prime number. This topic is discussed further in Section XII. If the user finds particular sectors of his directory are filling up, then it is recommended that the size of the directory be increased, to allow the hashing algorithm to spread the entries further apart. This must be done via System Generation. Refer to paragraph 12-3.6.

3-5 DISC FILES

A disc file is a contiguous segment of disc storage, given a symbolic name, optionally a password, and in accounting systems, a user-number, all of which are used to identify the file. Disc files can be any size from one sector up to the maximum capacity of the disc pack. Once a file has been created, its size cannot be changed. The file can only be deleted and re-created.

When a disc file is to be created, the user must specify the name, password (if any), file type, size, disc pack number on which the file is to be located, and, in accounting configurations, the file access bits, if any. Initially, the file name, password, and user-number are checked against existing files, and if the name is not already in use, the DMS system finds a block of contiguous disc space of the required size by searching the Space Allocation Map of the specified pack. A first-fit technique is used. That is, the Space Allocation Map is scanned from the beginning until the first available slot is found. An entry in the Master Disc Directory for the file is then made, and file creation is complete. The allocated disc space is not cleared when the file is created. An error condition or message is returned to the user if, for any reason, the file cannot be created as specified.

As described above, DMS disc files are allocated in units of the Space Allocation Map. The SAM is always two sectors long and each bit is used to represent one or more sectors. Thus the SAM size is 2 x 112 x 24 or 5376 bits. When a disc pack has more than 5376 sectors, as most do, then each SAM bit is used to represent more than one sector. For example, on normal density (204 cylinder) Datacraft Series 5200 Cartridge Disc units, there are 8160 sectors. Thus for these discs, each SAM bit is used to represent two sectors. This limits disc storage allocation to units of two sectors. Thus if the user asks for a file of three sectors, actually four sectors of disc space are allocated; the user getting three with one wasted. On larger, moving head multi-layer disc units, each space allocation bit may represent more sectors.

3-5.1 File Security

DMS file security is maintained through use of a four character password, and additionally in accounting systems, a user-number and read, write and delete access bits. All of this information is stored in the Master Disc Directory entry for the file.

In non-accounting systems, file system integrity is maintained through use of the password. A password can be any string of one through four alphanumeric characters, provided the first is alphabetic. When a password is supplied with file creation, or is later added through a file Rename operation, then all references to the file must supply the correct password. That is, when the file is OPENed for I/O operations, or attempts are made to Delete or Rename the file, the correct password is required.

An additional feature of the DMS file system allows multiple files to have the same name, so long as they have unique passwords. That is, within one DMS system, there can exist file CAT with password X and another different file CAT with password ABCD. However, if there exists a version of the file without a password, which is considered a "public" file since anyone can access the file regardless of the specified password, then there can exist no other files with that name. For example, if the files CAT existed at stated above, then another file CAT with no password, (i.e., a "public" version) could not be created. Correspondingly, if there existed a file DOG without a password, then no other files named DOG could be created, even when a password was given.

In accounting versions of DMS, there exists a much more extensive file security system, in addition to that discussed for passwords in the previous paragraph. This involves the use of user-numbers and file access bits. When a file is created in an accounting system, the user-number of the program creating the file (refer to Section VII) is included in the directory entry along with any file access bits that may have been supplied by the creator. If no access bits are set when creating the file, then the file is totally private; that is, it can be Read, Written, or Deleted/Renamed only by programs executing with the creator's user-number. The creator may specify, however, thay anyone else be allowed to perform a particular operation of the file. For instance, Read, by setting the appropriate access bit(s). There is a distinct access bit for Reading, for Writing, and for Deleting/Renaming the file. Exact specifications of these bits is discussed in the appropriate paragraphs of Sections IV, VI and the ACRONIM General Specification (AA61681-00).

An extension of the use of multiple file names in addition to that discussed above in conjunction with passwords, is provided in an accounting system. First the difference between "public" and "private" files for accounting systems must be defined. A "public" file in this context is any file which has one or more of the read, write, or delete access bits set. A "private" file is one which has none of these set and can be referenced only by the original creator. Within the restrictions discussed above for passwords, if there exists a valid name and password combination, there can then exist any number of private files employing this file name and password, so long as the user numbers are different. If, however, there is a valid name/password combination that is "public" (has any access bit set) then there can be no other occurrences of this name and password together. For example, consider a file ABC with password X which is totally private (i.e. has no access bits set) with user-number N1. Then users N2, N3, etc. can all create files ABC with password X provided they are all totally private. However, no one would be allowed to create a file ABC having password X with any access bits set, which would be a public version. Correspondingly, if there existed file DEF with password Y having public read access set, then no one else could create a file DEF having password Y. Figure 3-2 gives a generalized description of what can and cannot exist.

3-5.2 Passwords on Load Modules

Whenever a load module (program) is created on disc by the Series 6000 Link Cataloger, it is given the special password "GORP". The purposes of this are two-fold. One is that it allows users to create other data and word files of the same name, i.e., it creates a non-accounting type private file instead of a public one. Second, it prevents accidental destruction of program files by requiring use of the password when doing writing or deleting operations. The System Loader uses the password "GORP" when reading any load module from disc. Hence, the user need not be concerned with its presence except when a purposeful attempt at deletion or renaming of the program is desired.

3-5.3 File Types

When a file is created, the user specifies a File Type. As far as DMS operation is concerned, file types 1, 4, 5 and 6 are all equivalent. This file type distinction is provided soley for purposes of the individual installation. DMS does, however, differentiate file types 2, 3, 7, 8 and 9. These distinctions are required to allow the system to determine the types of load modules and correctly load program files. The meaning of various File Types is given in Table 3-1.

Figure 3-2

File Name Uniqueness in Accounting Systems

Given the same file name if there exists → Then the following is: ↓	File w/o Password No Access Bits Set	File w/o Password, Some Access Bit(s) Set	File with Password No Access Bits Set	File with Password, Some Access Bit(s) Set
No password w/o access bits and different user	Valid	Invalid	Valid	Invalid
No password with access bits and different user	Invalid	Invalid	Invalid	Invalid
Same password w/o access bits and different user	---	---	Valid	Invalid
Same password with access bits and different user	---	---	Invalid	Invalid
Different password w/o access bits & different user	Valid	Invalid	Valid	Valid
Different password with access bits & different user	Invalid	Invalid	Valid	Valid
Different password w/o access bits & same user	Invalid	Invalid	Valid	Valid
Different password with access bits & same user	Invalid	Invalid	Valid	Valid

Table 3-1

File Types

1	User Data or Work File
2	Background Relocatable Load Module
3	Background Absolute Load Module
4	Object Library File
5	Source Library File
6	System Work File
7	Resident Foreground Load Module
8	Non-Resident Foreground Load Module
9	Re-entrant Foreground Load Module

An additional file type specification is provided through use of the blocked or unblocked types. Any of the file types 1, 4, 5 or 6 can additionally be specified as blocked. Once a file has been created as blocked, then all accesses to the file are through the resident DMS Blocked File Handler. Blocking of files provides increase disc storage density by packing multiple records per disc sector. Additionally, symbolic records have contiguous blanks replaced by a blank suppression character on the disc file, providing increased file capacity. For example, normal symbolic assembler source card images can be packed about 8-10 images per sector by blocking techniques, while, if the data were written in an unblocked file, one image per sector would be written. Further advantages of blocked files are given in Paragraph 3-5.4

A further distinction within blocked files is provided by a type designation of spool files. Spool files are those dynamically created by the system to hold input and output spool files. Input spool files providing background jobs are named S:nnnn where nnnn is an ascending decimal number. Output spool files generated by any program are named S#nnnn where nnnn is again a decimal sequence number. If a file is typed as spooled, then the Foreground Output Spooler will delete it when it has completed the dumping of the file. Correspondingly, input spool files are automatically deleted by background when the job has been completed.

Yet another file type designation available under DMS is the Core Resident Directory option. When a system is generated, a table of variable length is created in memory to hold those Master Disc Directory entries the user feels are accessed frequently enough that the extra disc access to the Master Disc Directory should be eliminated. For instance, if a particular configuration has a disc file (possibly a load module file) that was opened, say once a second, then it might be advantageous to have its directory entry reside in the in-core directory table to eliminate the disc access involved in the directory look-up. When a data or program file is created,

it can be specified that its directory entry is to be core resident. Refer to Section VI on data and program file creation. Note that this in-core directory table must be able to hold all directory entries that are created as "Core-Resident". The user should be careful not to overflow this table, which would cause a rejection of a file creation of this type.

Any of the above file types may also be classified as being Permanent files. A file which is typed Permanent is not moved during pack compression. A typical use of a Permanent type file would be to cover and flag a bad sector on a disc pack.

3-5.4 File Accessing Methods

Whenever a disc file is "OPENed", which is required before a program can access the file, the file password is checked. Additionally, in Accounting Systems, the user-number and file access bits are checked and saved to insure that the user is allowed only appropriate access to the file.

Records written to unblocked files are always written beginning at sector boundaries. Thus, if all records written are less than or equal to 112 words in length, one record will be written per sector. When unblocked files are read, the record length transferred is determined by the word count of the read request, not what was written to the file. Thus, for example, if two 30-word records were written to unblocked file, and the first was read with a word count of 224, the data transferred would be the first 30-word record, followed by 82 zeros, followed by the second 30-word record, followed finally, by another 82 zeros. End-of-file marks on unblocked files are written as hardware end-of-files, and occupy a full disc sector.

On blocked files, records are written continuously on sectors, with a software end-of-record mark between each record. Records may spill across sectors as necessary, but all 112 words of a sector are used. Additionally, if a record is written with a Symbolic Write request, then strings of multiple blanks will be replaced by a single blank suppression character and trailing blanks ignored. In contrast to unblocked files, which always reads the specified word count, when a record is read from a blocked file, at most one previously written record is transferred. On Symbolic Read requests, if more words are requested than were in the record written, the remaining positions are filled with blanks. End-of-file marks on blocked files are handled by software, and occupy one word of a sector.

File manipulative commands can be performed on both blocked and unblocked files. These include advance record, backspace record, advance file, backspace file, and rewind. One difference, however, is that for unblocked files, if the word count of the record(s) being bypassed is greater than 112 then this word count must be specified on advance and backspace record commands.

Random read/write capability is provided through use of the Set Current Record Address Function, which sets the file pointer to a specific record within the file. For unblocked files, this Record Number is actually just a relative sector number. For blocked files, an actual record number is used. Based on record numbers stored in each block of a blocked file, an optimized iterative search technique is employed to locate the specified record and set the file pointer to it.

Random access writing of blocked files is not normally valid. That is, blocked files function is similar to magnetic tape. However, an Edit Write function code is available for blocked binary records, allowing a specified record to be over-written provided the original word count and new word count are identical.

SECTION IV

SYSTEM SERVICES

4-1 GENERAL

System services perform those system functions that are required to serve all programs executed under DMS. These services are performed by resident system subroutines that can be accessed using the Branch and Link Unrestricted (BLU) instruction. A Service Linkage table for all services is contained in memory locations 0 through '37.

The first six services are provided in a manner compatible with the Series 6000 Disc Operating System (DOS) such that background processors can be executed interchangeably with either system.

Table 4-1 lists the system services and their functions.

Table 4-1
General System Services

Linkage Address (Octal)	Description**	System Services Function
0	BLU $ABORT	Abort calling program
1	BLU $I/O	Request I/O function
2	BLU $EXIT	Terminate calling program
3	BLU $HOLD	Output operator message and suspend calling program
4	BLU $CHAIN	Load program segment
5	BLU $INFO	Return system information
6	BLU $WAIT	Wait until flag is reset
7	BLU $SFUNC	Perform special system function
10	BLU $CONV	Execute number conversion
11	BLU $FROGS	Special foreground services
12	BLU $SYSTER	Restricted system services
13	BLU $TERMIN	Terminate without leaving memory
14	BLU $ASSIGN	Dynamic I/O device assignment
15	BLU $DCM	Dynamic Core Manager
16	BLU $UNTRAP	Background & card control
17	BLU $FPACK	Find specified disc pack
20	BLU $O/M	Output Operator Message
21	BLU $C/RTN	Contingency Return

**The Cataloger utilizes only the first three characters of the "BLU name", i.e., "BLU $ABO".

The contents of the registers are indeterminate upon return from any system service except where used to return the results of the service as indicated in this document. It is the responsibility of the calling program to save any essential data in the registers before calling a system service.

4-2 ABORT SERVICE

The ABORT service provides a means whereby the calling program can be abruptly terminated and an appropriate notification output to the operator. The ABORT function is invoked by DMS when a program error is detected (such as an I/O request with an invalid function code or logical file number), when an Operator Communications command to abort a program is received, or when this service is executed. The system message that is output to the program's output device or file is of the format:

XXXXXX: ABT cc @ aaaaaa

where: XXXXXX is a program name; aaaaaa is a relative program address; and cc is a unique error code.

After the message has been output, DMS then closes all logical files that were opened by the aborted program and otherwise proceeds with the termination process.

When any background program is aborted, the Job Abort flag is set such that remaining control statements associated with that job are ignored and Job Control searches for the corresponding $EOJ statement in spooled configurations or the next $JOB or $EOJ in unspooled systems. (If the Job Stream is assigned to the console teletypewriter, a $JOB is not required.) When a foreground program is aborted, only the current execution of the program is aborted, and nothing is done to prevent reinitiation of the same program. No interrupts are enabled as a result of an ABORT.

Assembly Language Call

```
BLU $ABORT
```

4-2A Contingency Return

The C/RTN service provides a means whereby a foreground or background program may regain control upon an SAU trap or system abort. The only requirement is that the programs must supply the service password in the D register when the service is called. The password consists of 6 ANSCII characters, and is denoted below by "KEY". The user program is allowed to be given control of up to 32 aborts, the 33rd abort is unconditionally fatal. Following an operator (eight OPCOM or terminal user) abort, any abort (including a second operator abort) is also unconditionally fatal. After the user program is given control of an abort, the next abort is fatal unless the user again makes a call to C/RTN.

The user program is allowed to be given control of an unlimited number of nonaborting SAU traps. C/RTN requests are made via the following calling sequence:

```
TMD         ="KEY"      D = service key
TLO         USRRTN      K = address of user routine to which
BLU         $C/RTN      control is to be given
DATA        PARAMETER
.
.
.
.
.
USRRTN      DATA 0
            DATA 0
```

The user may specify whether the system is to return control to the user only in the case of an abort, or in both cases of aborts and non-aborting SAU traps. The user may additionally specify whether system abort messages are to be printed by the system. Valid parameter values and their meaning are given below.

In case of an abort, the system will branch to USRRTN+2 with USRRTN containing the system abort code and USRRTN+1 containing the abort location in bits 15-0.

In case of a non-aborting SAU trap, the system will branch to USRRTN+2 with USRRTN containing the SAU error code with bit 23 set. USRRTN+1 will contain the trap location in bits 15-0 and the C register in bits 19-16. The user may return to the non-aborting SAU trap location by restarting all registers to the values contained upon entry to USRRTN and executing a BRL*USRRTN instruction.

Parameter	Definition
0	Remove previous C/RTN call entry.
1	Do not inhibit abort messages, branch to USRRTN+2 in case of an abort.
2	Do not inhibit abort messages, branch to USRRTN+2 for both aborts and non-aborting SAU traps.
3	Inhibit system abort messages, branch to USRRTN+2 only in case of an abort.
4	Inhibit system abort messages, branch to USRRTN+2 for both aborts and non-aborting SAU traps.

4-3 INPUT/OUTPUT REQUESTS

The I/O Control System performs all I/O functions on a priority interrupt basis, providing the ability to overlap I/O operations are internal processing. I/O requests are made with logical file numbers; physical devices and disc files may be assigned to logical file numbers through Job Control and the Cataloger for background and foreground programs respectively. I/O requests are made via the following sequence of instructions:

```
ALPHA       TLO         PARLIST     (K) = address of parameter list
            BLU         $I/O        Call I/O control
                                    Transfer has been initiated and will
                                    be performed on an interrupt basis
                                    concurrent with processing.
            ...
PARLIST     DATA        'XXYY       XX =logical file YY =function code
            DATA        n           Word count
            DAC         BUFFER      Buffer Address
BUFFER      BLOK        n           Reserve n words or storage
```

If a device is busy when a request is received, the calling program is placed in the "waiting" state until the device becomes available. When the device is available, the request is initiated such that it can be completed using the device interrupts. Control is then returned to the calling program (at ALPHA+2) with the condition register set to "not negative". A BON at ALPHA+2 is not necessary for DMS; it should be used, however for DOS.

When any program must determine if an outstanding I/O function has been completed, it should request a status test. The status test function will place the calling program in a waiting state until the I/O function being tested has been completed. When the I/O function has been completed, it next checks for device errors and, if necessary, performs complete error processing and recovery. Finally, when the I/O function has been completed, the status test function returns control to the calling program with pertinent data regarding the referenced function in the accumulator (End-of-File, word count not complete, etc.).

Functions not requiring a parameter list (word count and buffer address) may be executed via the following sequence:

```
ALPHA   TNK     'XXYY     (K) =logical file/function code
        BLU     $I/O      Call I/O control
```

This sequence may be employed to test status, open a file, etc. The acceptable function codes for each device and their precise definitions are provided in Section V of this document.

4-4 EXIT SERVICE

The EXIT service provides the normal means to terminate execution of a program. The EXIT function closes all logical files which have been opened by the calling program and awaits completion of all I/O initiated by the terminating program. When a background program is terminated, Job Control is reloaded and executed. When a non-resident foreground program is terminated, its allocated memory is released for use by other programs, and its entry is removed from the list of active programs. When a resident foreground program is terminated, its entry is merely removed from the active program list.

Assembly Language Call		FORTRAN Call
BLU	$EXIT	CALL EXIT

4-5 TERMIN SERVICE

The TERMIN service is identical to the EXIT service described above except that it results in the foreground program remaining in core in its allocated memory area. This call is redundant for background or resident foreground programs. Foreground programs using this service can later be reinitiated by normal means with the exception that they will not be loaded from disc. A program which has used this service can later do a normal EXIT when it is again active.

Assembly Language Call

```
BLU     $TERMIN
```

4-6 HOLD SERVICE

The HOLD service allows a program to output a message to the console typewriter and suspend execution pending an operator response (refer to Operator Communications, Section IX). The typewriter message has the following format:

program: message text

where: program consists of one to six ANSCII characters representing the program name for foreground programs or the characters "BAKGND" for any message originated by a background program, and message text is 1-15 ANSCII characters.

The following calling sequence is required to utilize the HOLD service:

Assembly Language Call			FORTRAN Call
	TMK	MESSAGE	PAUSE XXXXXX
	BLU	$HOLD	OR
	. . .		STOP XXXXXX
MESSAGE	'XX	="message text"	This causes HOLD with STOP or PAUSE, plus the six-character message.

where: XX is the octal word count of the text message (Maximum=5).

If XX is zero, there will be no message, but the execution of the program is suspended.

EXAMPLE:

```
        TMK     MOUNT1
        BLU     $HOLD
        . . .   . . . . .
MOUNT1  '05     ="MOUNT A3 TAPE"
```

Assuming the call is made by a background program, the background will be suspended and the following message output:

BAKGND: MOUNT A3 TAPE

4-6A O/M SERVICE

The O/M Service allows a program to output a message to the program's appropriate List Output File; or Device. For background programs this is the List Output Logical File (06), and for foreground programs this is the terminal at which the program was initiated. The typewriter message has the following format:

Program: message text

Where message text is 1-15 ANSCII characters. The following calling sequence is required to utilize the O/M service:

Assembly Language Call

```
TMK     MESSAGE
BLU     $O/M
. . .
MESSAGE 'xx = "Message text"
```

Where: xx is the octal word count of the text message (Maximum = 5).

4-7 CHAIN SERVICE

CHAIN allows a large program to be segmented into multiple load modules and "chain loaded" under program control for execution. Different Chain modules may reference a COMMON data pool and may call each other in any order. The last executable module should call EXIT. The CHAIN calling sequence is:

Assembly Language Call		FORTRAN Call
BLU	$CHAIN	CALL CHAIN (6HXXXXXX)
DAC	="XXXXXX"	

where: XXXXXX is the identification of the module being called. For additional information, refer to the Link Cataloger, Section VI.

4-8 INFO SERVICE

INFO provides any processor with system information such as date, lines per page, options, flags etc. An assembly language information request is as follows:

```
TOK     n
BLU     $INFO
```

Value of n	Result
1	The 24-bit Option word is returned in the A register. If the BLU $INFO request is from a background program, the word returned in the A register is the logical "OR" of the current background option word (see section 6-2.6) and any options cataloged with the program (see section 6-4.2). If the BLU $INFO request is from a foreground program the word returned in the A register consists of only those options cataloged with the program (see section 6-4.2) and the remaining bits are zeroed.
2	The nine-character date is returned in the E, A, and I register, respectively, three characters per register. See Paragraph 6-2.4.
3	The lines-per-page integer is returned in the A register. See Paragraph 6-2.5.
4	The 24-bit Flag word is returned in the A register. See Paragraph 6-2.7.
5	The current six-character job name is returned in the D register.
6	The current run time is returned in the D register as a double precision integer in Accounting Systems only. Time is expressed in milliseconds since start of job. In non-accounting systems, this call returns zero.
7	The background-low address is returned in the E register and background-high address is returned in the A register. The Memory-high address is returned in the J register.
8	The negative of program high for the current background processor is returned in register I.
9	Invalid. A program abort will result.
10	The contents of register D are saved as "Start-of-Job" time.
11	The previously saved "Start-of-Job" time is returned in register D.
12	The current time of day is returned in register A. It is expressed as an integer representing tenths of seconds since midnight.

4.9 SFUNC SERVICE

The SFUNC service performs several special system functions that can be requested by both foreground and background programs. A specific function is selected using the following general calling sequence.

Assembly Language Call

```
TME     p1
TMA     p2
BLU     $SFUNC
DATA    n
```

where: p1, p2 contain the parameters that are defined uniquely for each function and n specifies the required function as shown in the following table.

Value of n	Result
1	Sets the Program Switch Word of the calling program to the value of p2.
2	Fetches the Program Switch Word of the program identified by the six-character name contained in p1 and p2. If the program is active, the Program Switch Word of the specified program is returned in register A and the Program Status Word is returned in (E). The format of the Program Status Word and the corresponding system status conditions are defined in Table 4-2.
3	Disables the SAU overflow interrupt trap during the execution of the calling program overriding the option specified when the program was Link Cataloged. (TYPE=SAUT). The parameters p1 and p2 are meaningless for this function.
4	Re-enables the SAU overflow interrupt trap during the execution of the calling program. This function terminates the effect of the preceding function.
5	Determines the peripheral device number of the device assigned to the logical file core contained in bits 11-6 of p2. Upon return, (I) contains the address of the File/Device Control Block, which corresponds to the specified file code. The initial word of the F/DCB contains the peripheral device number in bits 5-0. If this device number is zero, then the file code is assigned to a disc file; the ANSCII name of the file can be obtained from words two and three (addressed as 2,I and 3,I). Register A contains the device type of the assigned device according to Table 4-3. If the logical file is not assigned to any device, then the service returns with (I) = 0.

Example:

```
TMD     ="TEST 1A"
BLU     $SFUNC
DATA    2

...     ...
```

When control is returned after executing this call, the E register will contain a -1 if the program TEST1A is inactive. If TEST1A is active, then E will contain a value other than -1 and A will contain its Program Switch Word.

Table 4-2
Program Status Word

Bit	Indication	Comments
B23 =1	Checkpointed	Program has been checkpointed on disc (BAKGND only).
B22 =1	Waiting	The program is discontinued until a specified flag becomes non-negative. The flag assress is contained in bits 15-0.
B21 =1	Suspended	The program has been discontinued indefinitely. An Operator Communications command is required to resume its execution.
B20 =1	Awaiting Initiation	An initiation request has been made for the program, but it has not yet been processed.
B19 =1	Awaiting Allocation	The program is a candidate for foreground memory allocation.
B18 =1	Abort Inhibited	The program is undergoing an initiation or termination sequence during which an abort cannot be processed.
B17 =1	Abort Pending	An abort request has been made for the program, but it has not been processed.
B16 =1	Terminated	Program has exited via TERMIN service.
B23-19=0 & B17-0=0	Executable	The program is contending for computer processor time which it will be allocated according to priority.

Table 4-3. Device Type Codes

Code (Decimal)	Type of Device
0	Disc
1	Console TTY
2	Remote TTY
3	Remote CRT
4	Paper Tape Reader
5	Paper Tape Punch
6	Line Printer
7	Card Reader
8	Card Punch
9	Mag Tape
10	Synchronous Interface
11	Real Time Peripheral Equipment
12	Incremental Plotter
13	Printer/Plotter

4-10 WAIT SERVICE

The WAIT service allows a program to relinquish all processing time until a flag word is set to a positive or zero value by some other program or interrupt routine. When this service is called by a foreground program, control is transferred to another foreground program or to the background as dictated by program priorities. When called by a background program, the system inters the null state until an interrupt occurs, enabling some program to resume execution.

The WAIT function that is associated with this service is used extensively by the I/O Control System and other DMS routines to provide multiprogramming. For example, when an I/O request cannot be initiated for a program because the required device is busy, then IOCS invokes the WAIT function and the program is discontinued until the busy flag associated with the required device is reset. The WAIT service allows this function to be adapted to more general applications. It can be used when a data processing program must wait until a data acquisition foreground program has input some quantam of data (and then resets the flag).

Assembly Language Call

```
TLO     FLAG
BLU     $WAIT
. . .
```

where: FLAG is the flag word and can reside in the calling program or in SYSCOM.

4-11 CONV SERVICE

The CONV service performs the number conversion functions. The service is utilized with the following calling sequence:

Assembly Language Call

```
        TMA     VALUE
        BLU     $CONV
        DATA    n
        ...
VALUE   DATA    X
```

where: n determines the type of conversion.
X is the value to be converted.

Value of n	Function
1	The value in (A) is converted to its decimal ANSCII equivalent and returned in register I, E and A. A negative is indicated by (-); a positive sign is indicated by a blank space (ƀ). (I) = Sign, ƀ 1st digit (E) = 2nd, 3rd, 4th digits (A) = 5th, 6th, 7th digits
2	The value in (A) is considered to be a 24-bit unsigned integer, and is converted to its octal ANSCII equivalent. The result is returned in registers I, E and A. (I) = ƀ 1st, 2nd digits (E) = 3rd, 4th, 5th digits (A) = 6th, 7th, 8th digits

4-12 SPECIAL FOREGROUND SERVICES

The foreground system services perform those functions required to coordinate foreground activity. These services can also be called by background programs. The general calling sequence for foreground services is as follows.

Assembly Language Call

```
        TMR     parameters
        BLU     $FROGS
        DATA    n
        ...
```

FORTRAN Call

```
Call FROGS (n, parameter1,parameter2,...)
```

where: n identifies the specific foreground system service parameters that are dependent on the specific service. These services and the corresponding value of n are described in the following paragraph, and summarized in Table 4-4.

4-12.1 Program Initiation

Any foreground program executing under DMS has the ability to initiate another foreground program which can be specified to operate at a higher, lower, or identical priority to the calling program. The calling sequence for this function is as follows:

Assembly Language Call		FORTRAN Call
TMD	="program"	Call FROGS (1,6HPROGRA,priority, parameter)
TOI	priority	
TMK	parameter	
BLU	$FROGS	
DATA	1	
...		

where: "program" or PROGRA identifies the program name in six ANSCII characters, priority specifies the execution priority and must range between 1 and 254, and "parameter" specifies a 24-bit value which is passed to the specified program and loaded in the A-register when it begins execution. Upon return, the A-register will contain the address of the APL entry for the program being initiated or -1 if the APL list was full.

Control is returned to the calling program once the initiation procedure has been started, but not necessarily completed. In accounting systems, the user number of the calling program is automatically used as the user-number for the program being initiated.

Example:

TOI	1	Call FROGS (1,6HDTALOG, 1,5)
TMD	="DTALOG"	
TOK	5	
BLU	$FROGS	
DATA	1	
...	...	

This example will initiate program DTALOG to execute at priority 1 with parameter of 5.

4-12.2 Foreground Interrupt Handling

Foreground system services are provided to allow program control of selected priority interrupts - - - those reserved for foreground program initiation during system generation. To utilize this type of interrupt, a program must be first "connected" to the interrupt level i.e., a software linkage must be established such that the program will be initiated when the interrupt becomes active. The CONNECT service used to make this linkage has the following sequence.

Assembly Language Call		FORTRAN Call
TMD	="program name"	Call FROGS (2,6HPROGRA, 'xyy,n,parameter)
TOI	'xyy	
TOK	n	
BLU	$FROGS	
DATA	2	
DATA	parameter	

where: "program name" - consists of six ANSCII characters that identify the program.

x - represents an octal digit specifying group number.

yy - represents two octal digit specifying level (within appropriate group).

n - represents the system priority at which the specified program will be executed; must range between 1 and 254.

parameter - represents a 24-bit value to be passed to specified program as parameter at initiation.

The CONNECT service disables and arms the specified level before returning to the calling program. When the connected program is initiated, the user-number for Accounting Systems will be the same as that of the program making the CONNECT call.

The ENABLE service can be used to enable the priority interrupt so that it can be activated by an external signal. The ENABLE function first verifies that the specified interrupt is valid and that a program has been connected to it. If these conditions are met, the ENABLE function enables the interrupt and returns to the calling program. The calling sequence to enable an external interrupt is as follows.

Assembly Language Call		FORTRAN Call
TOI	interrupt	CALL FROGS (3,interrupt)
BLU	$FROGS	
DATA	3	
...		

where: interrupt identifies the interrupt group and level as described for CONNECT.

The DISABLE service disables the specified priority interrupt and thereby prevents its activation. The DISABLE function first verifies that the designated interrupt is valid and then disables the interrupt. The calling sequence is as follows:

Assembly Language Call		FORTRAN Call
TOI	interrupt	CALL FROGS (4,interrupt)

```
BLU     $FROGS
DATA    4
...
```

The RELEASE service removes the software linkage extablished by CONNECT and disables as well as disarms the specified interrupt. A RELEASE must be issued before another CONNECT can ne made to the same interrupt level. The calling sequence for RELEASE is as follows:

Assembly Language Call		FORTRAN Call
TOI	interrupt	CALL FROGS (5,interrupt)
BLU	$FROGS	
DATA	5	

Example:

TMD	="ALARM1"	CALL FROGS (2,6HALARM1, '117,1)
TOI	'121	
TOK	1	
BLU	$FROGS	
DATA	2	
TOI	'121	
BLU	$FROGS	
DATA	3	
...	...	

This example connects program ALARM1 to the interrupt of group 1, level 17, and then enables that interrupt. When the interrupt becomes active, the program ALARM1 will be executed at DMS priority 1.

The user should consult Section XII, System Configuration, for information on specifying certain external interrupts as usable by these services.

4-12.3 Timer Scheduling

The timer scheduler provides the capability of repetitively executing foreground programs at fixed time intervals. The SCHEDULE service can be called by any program to add any foreground program to the timer schedule. The calling program must specify the program name, the activation intercal in 120 Hz Clock counts, and its execution priority as shown in the following calling sequence. The user number in Accounting Systems will be the same as that of the calling program.

Assembly Language Call		FORTRAN Call
TMD	="program name"	CALL FROGS (6,6HPROGRA,priority, timer counts, parameter)
TOI	priority	
TOK	timer counts	
BLU	$FROGS	
DATA	6	
DATA	user parameter to be passed to "program name"	

If this program has already been entered, the condition code is set to a negative (-) condition. If the timer schedule is full, the condition code register is set to a zero (0) condition before returning to the calling program and the request is ignored. Otherwise, the program is added to the schedule and the condition code register is set to a positive (+) condition.

```
TMD     ="SCANX"        CALL FROGS (6,5HSCANX,5,50,100)
TOI     5
TOK     40
BLU     $FROGS
DATA    6
DATA    100
```

In this example, the program SCANX will be executed at priority 5 once every 40 timer counts. There are 120 timer counts per second. The parameter passed to SCANX is 100.

A DELETE service is also provided that removes the specified program from the timer schedule. This service is called with the following instruction sequence.

Assembly Language Call		FORTRAN Call
TMD	="program name"	CALL FROGS (7, 6HPROGRA)
BLU	$FROGS	
DATA	7	
...	...	

An additional service is provided to cause a foreground program to be initiated after a specified delay. To use this service, the calling program must give the number of timer counts in the future at which time the program is to be initiated. By using this service in conjunction with the INFO time of day service, a program can cause another program to be initiated at a specific time of day.

Assembly Language Call		FORTRAN Call
TMD	="program name"	CALL FROGS (10,6HPROGRA,priority, timer, parameter)
TOI	priority	
TOK	timer counts	
BLU	$FROGS	
DATA	10	
DATA	parameter	

If this program has already been entered, the condition code is set to a negative (-) condition. If the timer schedule is full, the condition code register is set to a zero (0) condition before returning to the calling program; and the request is ignored. Otherwise, the program is added to the schedule and the condition code register is set to a positive (+) condition.

A WAIT service is also provided to enable the calling program to place itself in a suspended state for a specified interval. This interval is specified as a number of timer counts. This service is called with the following sequence:

Assembly Language Call		FORTRAN Call
TOK	timer-counts	CALL FROGS (11,timer)
BLU	$FROGS	
DATA	11	

4-12.4 System Common Accessing

The label SYSCOM is reserved for a special common storage area in upper memory by the DMS. This area provides a communications region for programs executing under DMS. The size of the SYSCOM area is established during system generation.

Any program that needs to reference this area must contain a common statement specifying the block name SYSCOM. The proper address for SYSCOM references will then be constructed by DMS when the program is loaded. Any program can then extract data from SYSCOM and process it. A program operating in the Restricted Mode cannot alter memory outside its allocated area and thus cannot normally modify the contents of SYSCOM. However, a pair of foreground system services are available that provide a restricted foreground program with temporary write-access to SYSCOM. The PROVIDE service modifies the contents of the upper limit register for the calling program such that is can alter SYSCOM. The RESTRICT service restores the upper limit register to its previous value.

	Assembly Language Call		FORTRAN Call
PROVIDE:	BLU	$FROGS	CALL FROGS (8)
	DATA	8	
	. . .	. . .	
RESTRICT:	BLU	$FROGS	CALL FROGS (9)
	DATA	9	
	. . .	. . .	

4-12.5 Change Limits Service

A program may alter its restricted status by the following call:

```
TME     NEWLO
TMA     NEWHI
BLU     $FROGS
DATA    12
```

Upon return from the service, the program limit registers are set to NEWLO and NEWHI respectively, and E and A contain the old values of the limits. If the new limits include part of the operating system, the program will be executed as a privileged program until the limits are again changed such that they no longer include any part of the system.

Table 4-4. Summary of FROGS Call

Value of n	Function
1	Initiate program.
2	Connect to External Interrupt
3	Enable External Interrupt
4	Disable External Interrupt
5	Release program connected to External Interrupt

Table 4-4. Summary of FROGS Call (Cont'd.)

Value of n	Function
6	Place program on timer schedule for periodic initiation.
7	Remove program from timer schedule.
8	Provide write-access to SYSCOM.
9	Restrict write-access to SYSCOM.
10	Initiate program at specified time.
11	Suspend program execution for specified period.
12	Change limits of restricted program.

4-13 ASSIGN SERVICE

The ASSIGN service provides dynamic I/O unit assignments for both foreground and background programs. It modifies the calling program's own File/Device Control Block area (Assign Table). The service is utilized with the following calling sequence for regular disc files:

```
TOI     LFN
TMD     FILENAM
BLU     $ASSIGN
```

The following sequence is used for physical devices:

```
TOI     LFN
TZE
TOA     PDN
BLU     $ASSIGN
BON     ERROR (PDN is not in system)
```

The following sequence is used for dynamic file creation of spooling files in spooled systems only:

```
TOI     LFN
TMA     SDN         (spooled device PDN in bits 23-18, file size in sectors
                     or zero in bits 17-0) (See text below)
TME     ='60000000  (for spooled file create/return to user on error) or
        ='70000000  (for spooled file create/abort on error) or
        ='40000000  (for regular file create/return to user on error) or
        ='50000000  (for regular file create/abort on error)
BLU     $ASSIGN
BON     ERROR       (PDN is not spooled PDN in system)
```

where LFN is a valid Logical File Number ('77 ≥ LFN ≥ 0); FILENAM is a 6 character ASCII file name; PDN is a valid Physical Device Number (76 ≥ PDN ≥ 0); and SDN is a valid PDN that is also a spooled device. (Physical device configured as terminal in spooled systems.)

a) the LFN is invalid
b) the PDN is invalid
c) the SDN is invalid (spooled file create.)

If the specified logical file is open, the ASSIGN service will close/deallocate it before making the assignment. An ABORT will result if the Assign Table is full or the Assignment 0 = 0 is made.

The spool mode of the Assign service is used for making assignments to spool output files. It is available only in spooled configurations of DMS and an abort will result if its use is attempted in unspooled systems.

The file will be dynamically created by the system using ascending ASCII numbers for file names. The file and its name are not actually generated until the file is initially opened. If the spool mode is selected, the file will automatically be spooled out to the device and deleted when output is complete. A second write operation to a spool file immediately following a write which returned EOT status to the user will be considered to be spool file overflow. From this point on, write requests to this spool file will be treated as null requests. An operator message will be output at the time the overflow occurs. When the file is closed and automatically spooled out, an overflow message will be output following the file. If the regular dynamic file create is used, the file is dynamically named and created, but is not spooled or deleted. The assigned file name may be referenced by the $SFUNC service (mode 5). The file size field may be zero if the default size is desired. Otherwise, a size as specified in sectors is placed here. If the size specified is not an exact multiple of the blocked file blocking factor, it will automatically be rounded up to the next exact multiple.

The following sequence is used to squeeze out unused entries from the File/Device Control Block Area.

```
TNI     1
BLU     $ASSIGN
```

The squeeze call is used to remove all assignments for Logical Files which are closed. In order to reuse these logical files after a squeeze, they must be re-assigned.

4-14 DYNAMIC CORE MANAGER (DCM)

The Dynamic Core Manager provides dynamic core allocation/deallocation services. These are accessible from both foreground and background with one exception: a background program will not be allocated a memory block so large that the background program must be checkpointed to permit allocation of the block, but instead will be returned a "memory space not available" condition. This problem may be remedied by modifying the size of background with an "MB" operator communications command.

The Dynamic Core Manager may be called via the following calling sequence:

```
TME     p1
TMK     p2
BLU     $DCM
DATA    n
```

where p1, p2 contain parameter(s) that are defined uniquely for each function; n specifies the particular function as follows.

Value of n	Function
1	Allocate a core block. The size in words is specified by p1. p2 contains a 24-bit identification value so that the calling program can locate the core block at a later time. (See following function). Upon return, register K contains the memory location. of the allocated core block. Error conditions are indicated by K=-1 insufficient memory available =-2 ID already exists (core is not allocated) =-3 insufficient memory exists to ever honor the request.
2	Find core block. p2 contains the identification parameter used when the core block was allocated. Upon return, register K contains the memory location of the specified core block.
3	Release Core Block. p2 contains the indentification parameter of the core block to be released.
4	Release all core blocks. All core blocks allocated to the given program are automatically released. This function is always called by the Executive EXIT logic when a program terminates.
5	Allocated 4-word cell. Returns in register K the location of a special 4-word cell from the available cell pool. These are used primarily by special system programs to pass spooled file information.
6	Release 4-word cell. p2 contains the location of a 4-word cell to be returned to the available pool.

Note that in order to access dynamic core blocks allocated by DCM, the calling program must be operating in the privileged mode and care must be taken so as not to destroy other programs in memory.

4-15 FIND PACK SERVICE

A service is provided to allow the calling program to locate the disc number on which a specific pack is mounted. This is called via the following call:

```
TOI     address of 120 word scratch buffer or zero
TOA     pack number being located
BLU     $FPACK
```

The disc drive numner on which the pack is mounted is returned in register A. If the I register is zero when called, then the 120 word buffer will be allocated from the Dynamic Core Manager.

If the specified pack is not currently mounted, then the operator will be asked to mount it. When mounted, the disc number will be returned as above.

4-16 UNTRAP SERVICE

UNTRAP provides the ability to have background processors read cards containing $ in column one from the assigned job stream device without having control taken away by Job Control.

Whenever a record is requested by a processor from job stream that has a dollar sign in column 1, IOCS will set the system TRAP flag and return END-OF-FILE status to the calling program. In addition, the record containing the dollar sign will be loaded into the user's buffer. If the calling program wishes to continue reading from this file without having control removed by Job Control, it must clear the TRAP flag by making a call to UNTRAP. Additional records may then be read.

UNTRAP will not clear the TRAP flag if the record just read was either $JOB or $EOJ.

Assembly Language Call		FORTRAN CALL
BLU	$UNTRAP	CALL UNTRAP

4-17 SYSTEM RESTRICTED SERVICES

These services provide any background or foreground program the ability to perfrom restricted system functions. The only requirement is that the programs must supply the service password in the D register when the service is called. The password consists of 6 ANSCII characters, and is denoted in the following paragraphs by "KEY".

4-17.1 File Creation

This service provides the ability to create a disc file of any size. The create service is called via the following sequence:

```
        TOI     address of 336 word scratch buffer or zero
        TLO     PARLIST
        TMD     "KEY"
        BLU     $SYSTER
        DATA    1
        ...
PARLIST DATA    pack # (on return contains disc number of pack)
        DATA    file size (words)
        DATA    2 word ASCII file name
        DATA    2 word ASCII file password or zeros
        DATA    file type (bits 23-16), memory requirement (bit 15-0)
        DATA    file protect bits (bits 18-16), absolute starting address (bits 15-0)
        DATA    absolute address for start of space allocation
```

Only the first four characters of the password, if present, will be used. In non-accounting systems, the file protect bits are ignored. For accounting versions, these bits provide protection of the specified type. That is, they are the reverse of the file access bits discussed in Section III. They are as follows:

Bit 18 set for read protection.
Bit 17 set for write protection.
Bit 16 set for delete protection.

If any of these bits are absent in accounting systems, then the corresponding accesses are provided for other users on this file.

The file type specification is defined as follows:

Bit 23	set for blocked file
Bit 22	set for spool file
Bit 21	set for core directory entry
Bit 20	set for permanent file
Bits 19-16	is the file type ('01 thru '11), see Table 3-1, FILE TYPES

If the content of the I register is zero when the call is made, a 336 word buffer is dynamically allocated by the system for use by the create service, and the condition register is set to indicate whether the request was honored, and if not, why it was not. The condition register is set as follows:

C = Positive	-File name already used.
C = Zero	-Request honored. File was created.
C = Negative	-Insufficient space on pack for file.
C = Overflow	-Insufficient core available for 336 word buffer.

The calling program can check the C register to determine what change or corrections are required to create the file.

If the content of I is non-zero when the call is made, any error condition (Positive, Negative, or Overflow) will cause the calling program to be aborted.

If the requested pack is not mounted, an operator message is output and the pack must be mounted prior to program continuation.

If it is desired to allocate the file at a particular sector number on disc, this value should be put into the absolute start for space allocation parameter and the pack number must be negated.

If, in the case of a blocked file creation request, the file size specified is not an exact multiple of the blocked file blocking factor, it will automatically be rounded up to the next exact multiple.

4-17.2 File Deletion

This service provides the ability to delete any disc file providing the password is known. The delete service is called with the following sequence.

```
            TOI     address of 336 word scratch buffer or zero
            TMD     "KEY"
            TLO     PARLIST
            BLU     $SYS
            DATA    2
            . . .
PARLIST     BLOK    2
            DATA    2 word ASCII file name
            DATA    2 word ASCII file password
```

The returns from this service function in the same manner as those of the create service, i.e., if the content of I is non-zero when the call is made and an error occurs, the calling program is aborted; if the content of I is zero the C register is set as follows:

```
C = Positive    -File was not present.
C = Overflow    -Insufficient core for scratch buffer.
C = Zero        -Request honored.  File deleted.
```

4-17.3 Absolute Sector Read

This service provides the ability to read data from any disc location. Input to the service is based on an absolute disc sector number (0-n, where n is number of sectors on disc) and word count. Absolute Read is called via the following sequence.

```
            TOI     address of buffer to read into
            TLO     PARLIST
            TMD     "KEY"
            BLU     $SYS
            DATA    3
            . . .
PARLIST     DATA    disc number
            DATA    absolute sector number
            DATA    word count
```

4-17.4 Rename File

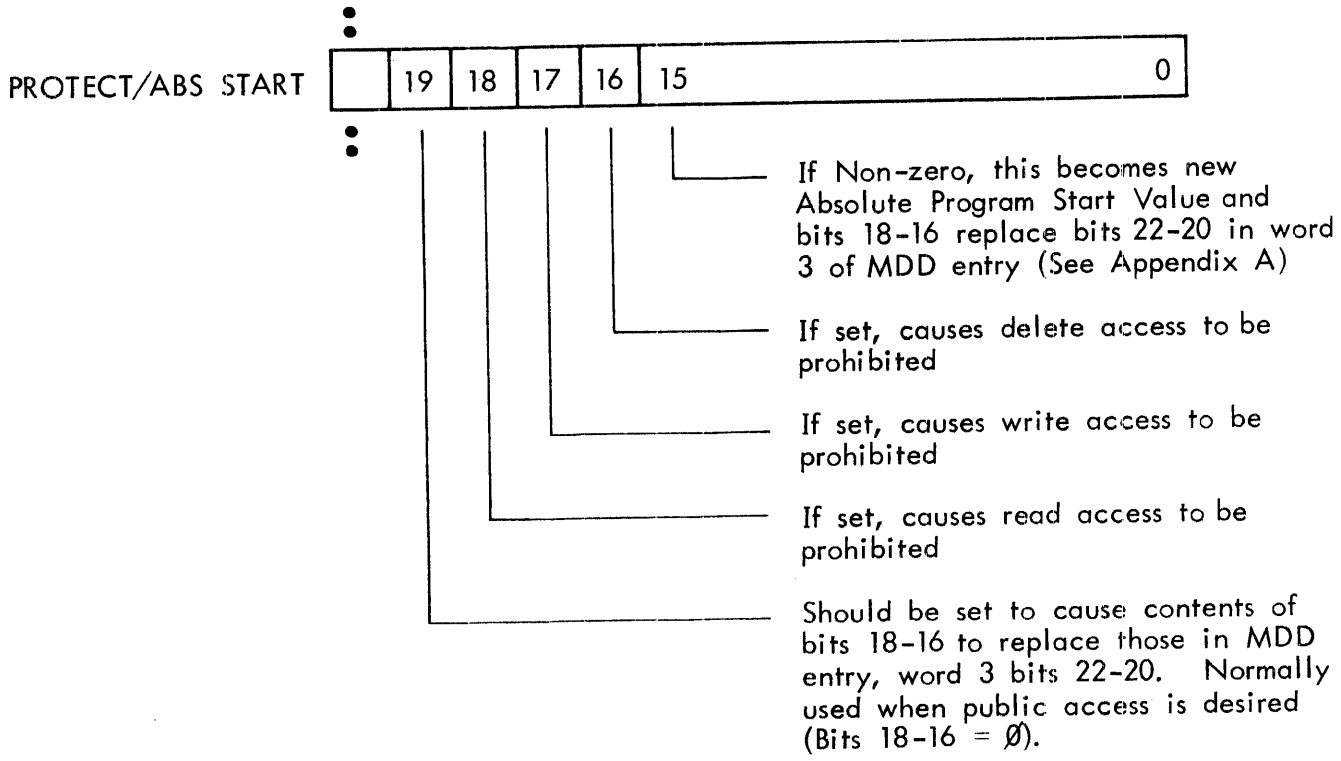
This service permits to changing the name or type of any disc file. The service is called as follows..

```
        TOI     address of 336 word scratch buffer or zero
        TMD     "KEY"
        TLO     PARLIST
        BLU     $SYS
        DATA    4
        ...
PARLIST DATA    2 word old file name
        DATA    2 word file password
        DATA    new type/memory requirement or zero (TYPE/MEM REQ)
        DATA    new protection/absolute start value or zero (PROTECT/ABS START)
        DATA    2 word new file name or zero (NEW NAME)
        DATA    2 word new file password or zero (NEW PASSWORD)
```

If the TYPE/MEM REQ word is non-zero, its contents replace the corresponding data in the Master Disc Directory (MDD) entry. If it is zero, the data is left unchanged.

If the PROTECT/ABS START word is non-zero, the protection bits in bits 18-16 replace those in the MDD entry and the contents of bits 15-0 replace the current value of ABS START in the directory. In order to change the file to complete access (bits 18-16 are zero) bit 19 of this word may be set to produce a non-zero value and hense set the corresponding information. The contents of bits 19-16 are ignored in non-accounting versions. See Figure 4-1.

Figure 4-1
Changing Protection with Rename Service



If the NEW NAME words are non-zero, then they will become the new file name. Otherwise the name will remain unchanged.

If the contents of both words of NEW PASSWD are zero, the old password will be left unchanged. If however, they are non-zero, then the first 4 ANSCII characters of these two words will become the new password. The password may thus be changed to zero by making both words zero except for some non-zero value in bits 15-0 of the second word of NEW PASSWD.

The return from this service function in the same manner as the create service. The condition codes are:

C = Overflow - Insufficient core available for scratch buffer.
C = Zero - Request honored - File renamed.

4-17.5 Special Assign

This service is used by certain system programs instead of Dynamic Assign to change assignment for LFN 77 (program file). This is not a user service and is listed for reference only. The calling sequence is as follows.

```
        TLO     PARLIST
        TMD     "KEY"
        BLU     $SYS
        DATA    5
        ...
PARLIST DATA    2 word ASCII file name to be assigned
```

The specified file name is assigned to the special program file LFN '77. This allows reference to the file via LFN 77.

4-17.6 Background Load Request

This service provides the linkage to the system loader to load background processors. It is called as follows.

```
        TLO     PARLIST
        TMD     "KEY"
        BLU     $SYS
        DATA    6
        ...
PARLIST DATA    2 word ASCII program name
```

The specified program name is loaded into background and executed. The service may be called only by background programs.

4-17.7 Write Absolute Sector

This service is the companion of Absolute Read by allowing data to be written at any specified disc sector. The service is called as follows.

```
        TOI     buffer address of data to be written
        TMD     "KEY"
```

```
            TLO     PARLIST
            BLU     $SYS
            DATA    7
            ...
PARLIST     DATA    disc number
            DATA    absolute sector number
            DATA    word-count
```

If the specified word-count is zero, a write end-of-file operation is performed. Caution should be observed with this service, since it allows a program to write over any sector of any disc, provided the service "KEY" is supplied.

4-17.8 Search Master Disc Directory

This service is a program to test if a given file name/password is present in the system. It is called as follows.

```
            TLO     PARLIST
            TMD     "KEY"
            BLU     $SYS
            DATA    8
            ...
PARLIST     RDAT    2 (0) (changed by system to non-zero)
            DATA    2 word ASCII file name
            DATA    2 word ASCII file password
            BLOK    2 (words for temporary usage by service)
```

The specified file name and password are located (if present) in the Master Disc Directory. If the entry is present, the J register returns the address of the directory entry which will have been placed in the calling program's PSA. The condition register reflects the results of the operation:

C = Positive -File name found and password is OK.
C = Negative -File name/password not found in directory.

Due to the arrangement of the PSA, subsequent system service calls may overwrite the contents of the MDD entry causing erroneous information to be utilized.

4-17.9 Trigger IOEXEC

This service permits programs to trigger the I/O EXEC wait flag, causing the I/O Executive to scan all terminals in order to perform required actions. This is needed only by special foreground interactive processors and it should not be necessary to use this in any user program. The service is called as follows:

```
TMD     "KEY"
BLU     $SYS
DATA    9
```

In an unspooled system, this call has no effect.

4-17.10 Place File on Spool-In Queue

This service permits any program to place a file on the background spool input queue. It is available only in spooled versions of DMS. The file to be placed on the queue must not have a password, and must consist of symbolic images representing a background job. The first image should be a valid $JOB (see Section VI) or the job will be ignored. The service is called as follows:

```
          TMD     "KEY"
          TLO     PARLIST
          BLU     $SYS
          DATA    10
          . . .
PARLIST   DATA    2 word ASCII file name
          DATA    Priority in queue (1-254)
```

An abort will result if the priority is not valid. In an unspooled system, this call has no effect.

4-17.11 Place File on Spool-Out Queue

This service permits programs to place a disc file name on the spool-out queue for any spooled or terminal device in spooled versions of DMS. It is non-existant in unspooled systems. The file to be placed on the queue must have no password and must consist of symbolic images suitable for the destination device. If the file is typed as a spool file, then it will be deleted when the output operation has been completed. The service is called as follows:

```
          TMD     "KEY"
          TLO     PARLIST
          BLU     $SYS
          DATA    11
          . . .
PARLIST   DATA    2 word ASCII file name
          DATA    priority in queue (1-254)
          DATA    device number to be spooled to
```

If the specified device is not a spooled or terminal device capable of output, the request is ignored. In an unspooled system, this call has no effect.

4-17.12 Overlay Load

This service provides the Overlay Cataloger an entry in the system to force a load of overlay segments. Its use is not intended for regular user programs.

4-17.13 Execute User Routine on Interval Timeout

This service permits a sophisticated usage of the 120 Hz clock in virtually any situation where some action after an elapsed timer interval is desired. Execution of the user routine is based on the following conditionals: (1) The timer interval specified has elapsed and (2) PROGRAM STATUS WORD .AND. CONDITIONAL WORD 1 .XOR. CONDITION WORD2=0. The calling sequence is:

```
          TMD     "KEY"
          TLO     PARLIST
          BLU     $SYS
          DATA    13
          ...
PARLIST   DAC     ADDRESS OF USER ROUTINE
          DATA    CONTROL WORD (SEE BELOW)
          DATA    TIMER INTERVAL IN 120 Hz COUNTS
          DATA    CONDITIONAL WORD 1
          DATA    CONDITIONAL WORD 2
```

If the control word is any positive value, conditions are tested for execution periodicall (as in the case of a scheduled program); if negative the special entry in the schedule tables for this service is removed upon interval timeout whether execution conditions are satisfied or not. The branch to the user routine is at clock interrupt level, and is a BSL. The user routine need not save registers and must not exit via a BRL*. If the program is restricted, use must be made of the service described in 4-12.5 as necessary to prevent a branch at interrupt level to an address within the limit register bounds.

4-17.14 Remove Execute Routine Entry

To remove an entry in the scheduler table made by the above call, the following must be done:

```
            TMD     "KEY"
            TLO     PARLIST
            BLU     $SYS
            DATA    14
            . . .
PARLIST     . . .
```

The parameters in PARLIST must have exactly the same values as when the entry described in 4-17.13 was made. This permits as many entries per program as desired provided no two are identical.

All entries associated with a given program are removed from system tables on an exit or abort.

EXAMPLES:

Suppose program A initiates some activity on an external piece of hardware, which in turn triggers an interrupt to initiate program B to signal completion of the activity. Program A might set a flag, do a BLU $WAIT and program B might reset the flag and just exit, at which point program A would continue execution. To prevent a hangup if the hardware fails to respond, Program A might use the following call to $SYS:

```
            TMD     "KEY"
            TLO     PARLIST
            BLU     $SYS
            DATA    13
            . . .
PARLIST     DAC     ERADR
            DATA    -1
            DATA    120
            '20     WFLAG
            '20     WFLAG
```

If after one second, program A is still waiting on a flag, release address is WFLAG (only bits 22 and 15-0 in the program status word are queried in this example). A BSL would be made to ERADR, and the entry would be removed from the table. If the hardware responds normally, and program B is initiated before one second elapses, program A could remove the entry it made as it resumes execution.

As an example, a program could make a permanent entry in the scheduler table, (PARLIST + 1) = +, and in effect have its own clock interrupt service routine; i. e., a branch to the user routine periodically. Conditional words 1 and 2 would both be set to zero to give an unconditional branch condition.

4-17.15 Special Delete

This service is used by certain system programs. Its use is not intended for regular user programs.

4-17.16 Special Absolute Sector Read

This service is identical to the Absolute Sector Read (4-17.3) except that a disc I/O error will not cause the calling program to be aborted. If a disc I/O error occurs bit 20 of the A register will be set and the actual word count transferred is returned in bits 0-15 of the A register. The Special Absolute Sector Read is called via the following sequence.

```
            TOI     address of buffer to read into
            TLO     PARLIST
            TMD     "KEY"
            BLU     $SYS
            DATA    16
            ...
PARLIST     DATA    disc number
            DATA    absolute sector number
            DATA    word count
```

4-17.17 Special Absolute Sector Write

This service is identical to the Write Absolute Sector (4-17.7) except that a disc I/O error will not cause the calling program to be aborted. If a disc I/O error occurs, bit 20 of register A will be set and the actual word count transferred is returned in bits 0-15 of the A register. The Special Absolute Sector Write is called via the following sequence:

```
            TOI     address of buffer to be written
            TLO     PARLIST
            TMD     "KEY"
            BLU     $SYS
            DATA    17
            ...
PARLIST     DATA    disc number
            DATA    absolute sector number
            DATA    word count
```

4-18 EXECUTIVE TRAPS

The executive trap routines are system modules of the resident portion of DMS. The standard interrupt assignments for executive traps are defined in Table 4-5.

Table 4-5. Executive Traps

Linkage Address (Octal)	Group, Level	Function
60	0,0	Power Fail
61	0,1	Power Restore
62	0,2	Memory Protect
63	0,3	Instruction Trap
64	0,4	Stall Alarm
65	0,5	Interval Timer
66	0,6	SAU Trap
67	0,7	Address Trap

4-18.1 Power Fail/Power Restore

When power fails, the power fail/restore routines saves registers and halt the machine. When power is restored the message "Power Fail" is output to the operator communications device and the machine again halted. Upon restart, the DMS system is rebooted from the disc.

4-18.2 Memory Protect/Instruction Trap

When the memory protect key switch is enabled, the DMS system has two modes of operation. The non-resident services (e.g., ACRONIM, Job Control and File Manager) operate in an unrestricted mode, that is, these services have access to any location in memory. Other non-resident services or background programs operate in the Restrict/Unprivileged mode, i.e., the programs may not reference any locations below the first location of the program nor above the highest location. Programs operating in the Restrict/Unprivileged mode are prevented from executing certain instructions as defined in the Computer Systems Reference Manual.

When a memory protect violation occurs, the abort service is called with an abort code of 04.

When a instruction trap occurs, the abort service is called with an abort code of 03.

4-18.3 Stall Alarm

When a stall alarm violation occurs, the abort service is called with an abort code of 5.

4-18.4 SAU Overflow/Underflow Trap

The SAU Trap routine processes overflow/underflow conditions as described in Table 4-6.

4-18.5 Address Trap

The address trap interrupt is triggered whenever the preset address is referenced in any manner (including ABC channel transfers). If a program is active when the interrupt occurs, the program is suspended and message giving the trapped address, program name, and the contents of the registers is output on the operator communications device. If there is no currently active program when the interrupt occurs the program name in the output message is "NONE", and no action other than the message is taken by the system.

4-18.6 Interval Timer

The Interval Timer option (T-register) is required by accounting versions of DMS to compute program execution CPU time. In non-accounting systems, the timer is not used and thus is available to user programs.

Table 4-6. SAU Trap Control

Error	Meaning	Result	Flag*
SAU 01	Square Root of a negative number	X=0.0	OA
SAU 02	Overflow during fix	A=F.S.P. or F.S.N. **	OA
SAU 03	Division by zero	X=F.S.P.	OA
SAU 04	Arithmetic Underflow	X=0.0	UA
SAU 05	Arithmetic Overflow	X=F.S.P. or F.S.N.	OA
SAU 00	Unrecognized SAU Trap	X=0.0	OA

Notes:

1. F.S.P. denotes full scale positive value (X='37777777, '37777577 or A= '37777777).
2. F.S.N. denotes full scale negative value (X= '40000000, '00000577 or A= '40000001).
3. When the specifies result is "F.S.P" or "F.S.N", the value selected will have the same sign as the result would have had, had there been no overflow.
4. SAU 00 is returned for any SAU traps that do not fall into the above catagories.

*Flag indicates the cataloger "TYPE" flag that is used to determine action following the error.

TYPE SPECIFICATION

Flag	Abort on error	No abort on error	Message on error	No message on error
OA	OA	NOA	OM	NOM
UA	UA	NUA	UM	NUM

**Because of the asynchronous nature of the SAU, this SAU trap normally occurs one instruction after the error causing instruction (FXA). Thus, if an error is possible during the execution of an FXA, care should be taken so that the A register is not accessed before the trap routine is able to adjust the result value. The standard FORTRAN Support libraries already do this.

SECTION V

LOGICAL FILES AND PHYSICAL DEVICES

5-1 GENERAL

A logical file is a series of data records transmitted to or from a specified physical device. A logical file number is a number (0-n) used by a program to reference a particular input or output file. A physical device number is a number (0-n) dedicated to a particular device or a named disc file.

In DMS, physical device numbers are established by the position a device handler's name in the peripheral device coordination table. By use of the Job Control statement $ASSIGN, a logical file number may be assigned to a particular device. A given logical file number may be assigned to only one physical device at a time. However, multiple logical file numbers may be assigned to a given physical device at the same time. By assigning a logical file number to physical device 0, the operator may nullify all operations to that logical file, which might be used to suppress all output to that file, for example.

The peripheral device coordination table is a resident table. Logical file assignments associated with a given program are contained within the program service area's file/device control table and resident only when that program is resident. There is only one program service area associated with background, common to Job Control and all background programs, since only one background program can be active at a given time. The background service area remains resident at all times.

Disc files are equivalent to physical devices. Disc files however, are referenced by name instead of numbers, e.g., $ASSIGN 10=SNOOPY. A physical device control block contains only two words of information. A disc file control block contains eight words including the file name, the first and last sector address, the current record address, the current End-of-File address, and the disc file protection status. An EOF within a disc file is functionally equivalent to an EOF mark on magnetic tape.

If a disc file is defined as "BLOCKED" when created, the disc file handler will automatically perform record blocking and symbolic data compression.

For consistency between background processors and related system service programs, certain logical file numbers are reserved. These logical file numbers (LFN) are defined in Table 5-1.

For convenience and consistency between various systems, a suggested standard table of physical device numbers (PDN) is given in Table 5-2.

5-2 INPUT/OUTPUT FUNCTIONS

Input/Output as discussed in Section IV, employes a logical file number and function code in requesting I/O operations. The following paragraphs define valid I/O function codes for each standard peripheral device. Table 5-3 provides a condensed list of I/O function codes and their relation to standard devices.

Table 5-1
Standard Logical File Numbers & Default Background Assignments

LFN (Octal)	Definition	Default Assignment
00	Job Stream	device 07
01	Operator Communications	device 01*
04	Binary Input	device 04
05	Binary Output	file LR
06	List Output	device 06
07	Symbolic Input	device 77 (job stream)
10	Symbolic Output	file W1
12	Link Library File	file LL
15	Link Ready File	file LR
16	Link Go File	file GO
* Not Reassignable		

Table 5-2
Standard Physical Device Numbers

PDN (Octal)	Device
01	Console Teletypewriter
04	High Speed Tape Reader
05	High Speed Tape Punch
06	Line Printer
07	Card Reader
10	Card Punch
11	Magnetic Tape #1
12	Magnetic Tape #2
20	TTY #0 (Console)
21	Remote TTY or CRT #1
22	Remote TTY or CRT #2

5-2.1 Common Device Function Codes

The two status checking function codes (00 and '77) provide flexibility in the processing of I/O calls.

If Function code 00 is specified in the call and the device is busy, the calling program is placed in a waiting state until the current I/O operation is completed and the ready status is set. Control is then returned to the calling program with the C register set not negative. This is the regular status check and is normally done after all I/O operations to guarentee completion.

If function code '77 is specified in the call and the device is busy, control is returned immediately to the calling program with the C register set to "negative", which is designed as a special purpose "busy check".

Table 5-3. I/O Function Codes

Function Code (Octal)	Definition of Function	ACTION CAUSED BY FUNCTION CODE							
		Disc	Magnetic Tape	Keyboard/Printer	Paper Tape Reader	Paper Tape Punch	Card Reader	Card Punch	Line Printer
00	Status	A Status C "Ø" E CRA J LSN I FSN K NRS	A Status C= "Ø" E CRA K EOF#	A Status C "Ø"	A Status C= "Ø"	A Status C "Ø"	A Status C "Ø"	A Status C "Ø"	A Status C "Ø"
01	Symbolic Read	Reads from CRA	Reads According to Tape Options	Pack Input 3 C/W ASCII	Reads 8-Bit Code Converts to 3 C/W ASCII		Reads Hollerith Converts to ASCII, 3 C/W		
02	Symbolic Write	Writes at CRA	Writes According to Tape Options	Prints 1 Line 3 C/W ASCII		Punches 8-Bit ASCII 3 C/W		Converts ASCII to Hollerith and Punches 1 Card	Prints 1 Line
03	Binary Read	Reads from CRA	Reads According to Tape Options	Reads 6-Bit Code 4 C/W Binary	Reads 6-Bit Code Packs 4 C/W Binary		Reads Formatted Binary 2 Columns Word		
04	Binary Write	Writes at CRA	Writes According to Tape Options	Punches 6-Bit Binary 4 C/W		Punches 6-Bit Binary 4 C/W		Punches Formatted Cards 2 Columns/ Word	
05	Special Action	Edit Write	Erases 3.5n Inches		Reads Unformatted 8-Bit Code and Packs 1 C/W	Punches Unformatted 8-Bit Binary 1 C/W	Reads 80 Columns 2 Columns/Word	Punches Unformatted 40 words/card	
06	Write EOF	Writes EOF Sector	Writes EOF Sector	Writes "EOT" and "EOF"		Punches EOF Record		Punches 9/8 Code Column 1	Print "EOF"
07	Open File	OPENS ASSIGNED FILE							
10	Close File	CLOSES ASSIGNED FILE							
11	Reposition File	Sets CRA CFA - 1	Repositions to 1st Record of Current File	If Tape mode "RPF" else null	"RPF TR" Hold Message		"RPF CR" Hold Message		
12	Backspace File	Sets CRA CFA	Repositions to Front of EOF	If Tape mode "BSF" else null	"BSF TR" Hold Message		"BSF CR" Hold Message		
13	Advance File	Advances to EOF and Sets CFA and CRA	Advances to EOF and Sets CFA, CRA and EOF#		Advances Past EOF Record		Advances Past 9/8 Punch		
14	Backspace Record	Sets CRA CRA - WC/WPS	Backspaces 1 Record	If Tape mode "BSR" else null	"BSR TR" Hold Message		"BSR CR" Hold Message		
15	Advance Record	Sets CRA CRA + WC/WPS	Advances 1 Record		Advances to End of Record		Advances Card Repeats if Column 1 9/7		
16	Rewind	Sets CRA Ø CFA Ø	Rewinds Transport and Sets CRA, CFA and EOF#						
17	Set Current Record Address	Sets CRA = Specified Sector Number	Positions Tape to Sepcified Record# within File						
20	Seek Current Record Address								
21	Set Current File Address	Sets CFA CRA	Sets CFA CRA						
22	Deallocate	DEALLOCATES FILES OR DEVICE							
23	Special Open	Open File and Return Status	Open File and Set Options	Open File	Open File		Open File		

CRA - Current Record Address
CFA - Current File Address
FSN - First Sector Number

LSN - Last Sector Number
WC - Word Count
WPS - Words Per Sector

EOF - End-of-File
C/W - Characters per word
NRS - Next Relative Sector Number

BSR - Backspace Record
BSF - Backspace File
RPF - Reposition File

No Action

Abort

In either case, the A register contains the following information.

A_{23}= 0 - Device not busy.

A_{22}= 1 - Word count not complete.

A_{21}= 1 - End-of-File detected.

A_{19}=1 - File Open.

A_{15}-A_0 = Number of words read on input.

Table 5-4. Standard Disc Files

File Name	Definition
LL	Link Library File (Blocked)
LR	Link Ready File (Blocked)
W1	Work One File (Blocked)
W2	Work Two File (Not Blocked)
GO	Link Go File (Not Blocked)

5-2.2 Disc

The DMS disc handler processes three types of disc access requests: absolute read/write, standard I/O non-blocked-file requests, and standard I/O blocked-file requests.

Absolute read/write requests are initiated by system routines and honored without restraint, bypassing normal IOCS file control logic.

Blocked and non-blocked IOCS requests are distinguished by the handler from information obtained from the master file directory when the file is opened.

A blocked data record consists of a one-word record gap followed by n-words of data. Each record gap contains a 12-bit backward and 12-bit forward word count. Blocked records are packed in a dynamic blocking buffer until it is filled. Buffer size is specified at system generation to be one or more sectors. Blocked records may spill from block to block, but may not be longer than 4096 words, in compressed form.

Symbolic data is compressed by converting all consecutive blanks into a one-byte blank count.

Each block contains two pointers. Word-0 contains the record number and block address pointer of the first record in the block, and word-1 references the last record of the block. Word-0 of block-0 contains the sector number of last recorded block. These pointers allow rapid random access of a blocked file.

Status (00 and 77) - The registers are returned as follows:

Register A

A23 = 1	If file is busy.
A22 = 1	If word count was not completed.
A21 = 1	If EOF record was detected on read or advance.
A19 = 1	File open.
A16 = A21 = 1	If file extents were exceeded on an input, output, or advance, or if the last recorded record of a blocked file plus one was requested on input, output, advance or set record address.
A15-0 =	Number of words transferred.

- Register C = F (A)
- Register I contains the first sector number of the referenced file.
- Register J contains the last sector number of the referenced file.
- Register K contains the next relative sector number of the referenced file.
- Register E contains the current relative record address (0-n) of the specified file. For non-blocked files (E) will be the same as (K).

Symbolic Read (01) - If the file extents or the last blocked record address are exceeded, input is suppressed and status bits 21 and 16 are set; otherwise, the transfer is performed. If blocked, the current compressed record is decompressed and transferred; and any remaining buffer is filled with blanks. If an EOF record is detected, status bit 21 is set.

Symbolic Write (02) - If the file extents will be exceeded by this transfer, output is suppressed and status bit 16 is set; otherwise, the specified word count is transferred to the current sector address of non-blocked files. For blocked files, the symbolic data is compressed (one or more consecutive blanks are compressed to a one-byte blank count) and blocked into a dynamic blocking buffer. When the buffer is filled it is written on disc.

Binary Read (03) - For non-blocked files, Binary Read is identical to Symbolic Read. For blocked files the current binary record is unblocked into the user buffer.

Binary Write (04) - The specified data record is transferred to the dynamic core block for blocked files and transferred directly to disc for non-blocked files.

Edit Write (05) - For non-blocked files function 05 is the same as binary write (04). For blocked files, this function sets an edit flag allowing an over-write of the current record. (NOTE: This function implies binary write and the same word count as the record being edited must be used.)

Write EOF (06) - An EOF sector is written on non-blocked files and an EOF record (zero word record) is written on blocked files.

Open File (07) - Verifies the existence of the referenced file name and sets the open status (Bit 18). For blocked files, not already open, a dynamic blocking buffer is allocated. The initial open of any disc file initiates a rewind of that file. On the initial open of the file, the password, if any, must be supplied left justified in the E & A registers.

On return from a successful open request, "E", will contain the blocking sector count (0 in unspooled systems) and "A" will contain the file type and memory requirement for the file:

B23	set for blocked file
B22	set for spool file
B21	set for core directory entry
B20	set for permanent file
B19-16	file type ('01 - '11) see Table 3-1
B15-0	program memory requirements (non-zero only for programs)

Close File (10) - For blocked files, the current block is written to disc if it is not empty.

Reposition File (11) - Record pointers are set to the CFA, or "current file address" (i.e., the record following the previously encountered EOF record), or to a rewind state if no EOF was encountered. CFA is set by reading an end-of-file, advancing to an end-of-file, or the "set current file address" function (21).

Backspace File (12) - The current record address is set to the record address of the previous EOF record, or to the rewind position if no EOF was encountered.

Advance File (13) - File pointers are advanced to the record following the next EOF record (blocked files) or sector (non-blocked files).

Backspace Record (14) - For non-blocked files, the specified word count (if supplied in the I/O call) is converted to a sector count (112 words/sector) and subtracted from the current record address. Where no word count is specified, one sector is assumed. For blocked files the current record pointers are moved back according to the blocked record gap which contains a backward and forward word count.

Advance Record (15) - File pointers are moved forward in the same manner as the backspace. In the non-blocked file, an EOF will not be detected by an advance reocrd.

Rewind File (16) - Current record address and current file address pointers are set to zero.

Set Current Record Address (17) - The current record address specified as parameter one of the user parameter list is stored as the CRA of the file. For blocked files this function is valid only for previously written files or for sequential write requests beginning immediately following the last recorded record, i.e. (SETCRA = 0, write, SETCRA =1, write, etc.) If the specified record is greater than the last recorded record, the file is left positioned to the end of recorded data and status bits 21 and 16 are set.

Seek Current Record Address (20) - Null.

Set Current File Address (21) - The current record address is saved as a "Current File Address" (i.e., start of a logical file). This function allows multi-pass processors such as the assembler to set CFA at the start of pass-one and issue a "reposition file" (11) for subsequent passes, precluding the requirement for a work file.

Close/Deallocate (22) - Causes the file to be closed (function 10), then deallocated. In the case of blocked files, the dynamic blocking buffer is deallocated to release the core.

Special Open (23) - Identical to Open File 07 except that if the referenced file is not there, control is returned with (C)=overflow & negative instead of aborting.

5-2.3 Magnetic Tape

One or more tape controllers are handled by a re-entrant mag tape handler. Each of these controllers may control one or more drives. A private service routine is used to interface with each drive.

The following is a list of valid I/O functions and their definitions for the mag tape handler.

Status (00-77) - Register A contains the common status word. The following additional status is returned:

- Bit 22 of register A is set to indicate that the previous I/O request was not completed as requested.
- Bit 16 of register A is set if software end-of-tape is detected. See function 20 for a definition of software end-of-tape.
- Register K contains the current file number of the specified transport, where the file beginning at load point is file number zero. This file number is incremented once for each end-of-file detected or written in the forward direction and decremented once for each end-of-file detected in the reverse direction.
- Register E contains the current record address, a relative record number (o-n) within the current file. A negative value indicates that the tape was backspaced over an end-of-file and a valid record number is no longer available.
- Register I contains a pointer to a seven-word block containing additional status information:

The contents of these 7 words are as follows:

0,1 hardware status word at completion of previous I/O request
1,1 hardware expanded status word at completion of previous I/O request
2,1 current hardware status word (status 00 only)
3,1 current hardware expanded status word (status 00 only)
4,1 number of retries on last I/O request
5,1 total number of retries since device was opened
6,1 net number of foreward requests processed since encountering hardware end-of-tape marker. A negative value indicates an end-of-tape marker not yet encountered.

Notes: A value of -2 as hardware status or hardware expanded status indicates that the particular status is not available due to hardware limitation or failures. Status 77 will not return current hardware status.

Symbolic Read (01)/Symbolic Write (02) - Internal symbolic data is assumed to be ANSC11, 3 C/W, and is transmitted to or from tape as 24-bit binary. Conversion to or from BCD is controlled via a tape-options table resident in the system or passed on via a special open request. Symbolic conversion is always 3 cpw, right justified in 8-bit bytes. Seven track tapes are converted to/from 6-bit BCD and nine track to/from 8-bit extended BCD. The transport type is specified at system generation. See Section 6-2.13, STAPEOP statement. For a description of error handling, see function 20 (set Error Options).

Binary Read (03) - The specified number of words are requested from the appropriate transport. Mode and density are taken from a tape option table in the resident system or passed on via a special open request. For a description of error handling, see function 20 (Set Error Options).

Binary Write (04) - The specified no. of words are transferred to the appropriate transport. Mode and density are taken from a tape option table in the resident system or passed on via a special open request. For a description of error handling, see function 20 (Set Error Options).

Erase (05) - Parameter two of the users parameter list specifies an erase repeat count (n) such that (n) (3.5 inches) of tape are erased. After reaching software end-of-tape (See function 20), a maximum of one 3.5 inch erase will be performed per function call. If the repeat count is not satisfied, WCNC status will be set and the word count returned on a status call will indicate the number of erases performed.

```
PARLIST     DATA    'XX05
            DATA    repeat count (n)
```

Write EOF (06) - An end-of-file record is written. The current file number is incremented and the current record address and current file address are set to zero.

Opend File (07) - An open bit is set in the user's device control block. The tape-options word is copied from the system tables on the initial open.

Close File (10) - Null

Reposition File (11) - The tape is repositioned to the first record of the current file (end of the previous End-of-File Record).

Backspace File (12) - The tape is positioned to the start of the previous End-of-File record. The current file number is decremented, the current record address is set to -1 and the current file address is set to zero.

Advance File (13) - The tape is positioned to the end of the next End-of-File record. The file number is incremented and the current record address and current file address are set to zero.

Backspace Record (14) - The tape is positioned to the start of the previous record.

Advance Record (15) - The tape is positioned to the end of the next record.

Rewind (16) - The tape is repositioned to load point. The file number, current record address and current file address are set to zero.

Set Current Record Address (17) - The tape is repositioned to the record number specified in the parameter list. The first record of each file is record zero. If the specified record does not exist in the current file, the tape is positioned at the beginning of the next file, the file number is incremented and the current record address and current file address are set to zero.

```
PARLIST     DATA    'XX17
            DATA    record number
```

Set Error Options (20) - The error options are set as specified. Any options not specified are reset to the system defaults. The number of options specified is indicated by word two of the parameter list. In other words, to change option word 4, the parameter list word 2 would be set to 4 and the first four option words must be specified. Option words 5-8 will be reset to the system defaults in this case. The parameter list is of the form:

```
PARLIST     DATA    'XX20
            DATA    N (Number of options specified)
            DAC     OPTNLST (Address of option list)

OPTNLIST    DATA    WORD 1
            DATA    WORD 2
            DATA    WORD N
```

The Error Options are defined as follows:

Word 1 - Read Error Positioning/Hold Message

= 1 - position before faulty record a fter retry limit is exceeded.
= 2 - position after faulty record after retry limit is exceeded

If bit 23 set - position as specified and output operator hold message repeat previous read operation if released
reset - position as specified and return to user program

Standard system defaults are positioning before faulty record and outputting operator hold message (1, bit 23 set).

Word 2 - Write Error Positioning/Hold Message

= 1 - position before faulty record and any erase attempts, after retry limit is exceeded
= 2 - position before faulty record but after any erase attempts after retry limit is exceeded
= 4 - position after faulty record after retry limit is exceeded

If bit 23 set - position as specified and output operator hold message-repeat previous write operation if released
reset - position as specified and return to user program

Standard system defaults are positioning before faulty record (but after erases) and out-putting operator hold message (2, bit 23 set).

Word 3 - Number of Retries Allowed on Read Error.
This is the number of re-read attempts which will be allowed upon detection of a read error. This number must be less than 256. The standard system default is 5.

Word 4 - Number of Retries Allowed on Write Error.
This is the number of re-write attempts allowed at one spot on the tape upon detection of a write error. This numbermust be less than 256. The standard system defualt is 2.

Word 5 - Number of Erase/Re-Writes Allowed on Write Error.
If the write retry limit is exceeded (word 4), 3.5 inches of tape will be erased and the write attempts repeated on the new spot on tape. This word specifies the number of erase/re-write attempts to be allowed and must be less than 256. The standard system default is 3.

Word 6 - Software EOT location/Hold Message.
This word defines where the "software EOT" is located as well as the action taken when it is reached. "Software EOT" is defined as the net number of forward tape operations allowed after detecting hardware EOT. The number specified must be less than 256. If bit 23 is set, an attempt to position past software EOT will output an operator hold message. If the program is released the handler will assume a new

tape has been mounted and the operation will be performed. If bit 23 is not set, the operation will be performed and status bit 16(EOT) will be set. No hold message is output if bit 23 is not set. The standard system default is not forward operations allowed = 0 and output hold message at EOT (0, bit 23 set).

Word 7 - Close/Deallocate Special Action.
This word defines any special action to be taken upon close/deallocate (function 22).

= 0 - no special action
= 1 - rewind tape upon clode/deallocate
= 2 - output operator message "UNLOAD TAPE T#) upon close/deallocate. This is only a message and does not suspend the program.
= 3 - rewind tape and output message
If bit 23 set - program will be suspended until tape drive is switched off-line
reset - program does not wait for drive to go off-line
The standard system default is no special action (0).

Word 8 - Software Write Protect.

= 0 - allow writes to be processed normally
= 1 - all write functions (02, 04 05 and 06) are treated as null functions.

If bit 23 set - output hold message "WP VIOLCATION" upon write attempt - if released, the write operation will be performed or ignored as specified

The standard system default is no special action (0).

Set Current File Address (21) - This function causes the current record address to be saved as a fake "start of file" so that a subsequent Reposition file will reposition the tape to this record address. This permits the assembler, for example, to assemble a string of programs without End-of-File separators or an auxiliary work file for pass two.

Close/Deallocate (22) - The tape drive is deallocated from the calling program. Any special action previously requested with function 20 is performed.

Special Open (23) - The open bit is set in the user's device control block and if the call is of the long form the second parameter contains the tape options word to be used. The form of this word must be identical to that in the Default Tape Option Table given in Appendix A.

```
PARLIST     DATA     'XX23
            DATA     tape options word
```

5-2.4 Teletype Terminals

One or more teletypes are handled by a reentrant teletype handler. Each teletype is interfaced by a private service routine, allowing local or long distance communications via any standard hardware interface.

Anyone of the teletypes may be specified at the operators terminal at system generation. The operators terminal may be alternately specified at "system-boot" time by setting sense-switch one along with the desired terminal device number in control switches 0-5, before activating the bootstrap.

The specified operators terminal functions as a normal remote terminal in addition to being the Operator Communications console. The operator communications service (OPCOM) is not accessable from any terminal other than the operators terminal. The operator may relinquish operator communications to another terminal via the command:

/OT,n

where n is the desired terminal device number.

The reentrant teletype handler honors standard IOCS requests for all teletypes, including the assigned operator's terminal. Additionally the following special action keys are acknowledged from all terminals:

Control "X-OFF"	Abort terminal job - applicable only in spooled systems, the "X-OFF" key causes the I/O Executive to abort the current terminal program, if any, and the message "ABT.." to be typed.

Control "RUB-OUT"	Line Delete - If input is active, the "RUBOUT" key is considered to be a line delete; in which case the handler outputs a "!" character followed by a carriage-return and line-feed, then reinitiates the input request, unless the input being cancelled was an OPCOM request, in which case input is not reinitialized.
Control "BELL"	Terminal Activation - In spooled systems, an input request bit is set causing the I/O Executive to initiate an input request for the interactive terminal editor (ACRONIM).
"/"	OPCOM Request - A "/" as character one of an input record defines an operator communications statement, and is ignored from all except the operator's terminal. If an input request is active at the operator's terminal, a "/" causes the input parameters to be saved for restart, and input to be initiated into the opcom input buffer. This buffer is passed to OPCOM by the handler upon detecting an end-of-record. The original input is then reinitiated by the handler. If input is not pending when an operator communications statement is issued, it is passed to OPCOM and the handler reset to an idle state.
"←"	Character Delete - Each "←" causes the handlers input column pointer to be decremented. If decremented to zero, a line delete is effected (i.e., a "!", carriage-return, and line-feed).
Control "TAPE"	Tape-On - The control key "TAPE" causes a tape reader flag to be set so that subsequent input requests for the terminal will be accepted from the paper tape reader.
Control "TAPE"	Tape-Off - The control key "TAPE" resets the tape reader flag so that subsequent input requests are accepted from the keyboard. Note that a properly prepared input tape should be terminated by a "TAPE". If not, the operator may type a "TAPE" during trailer, because reader and keyboard data are ORed. Removing the tape from the reader will also disable the reader.
Note:	"TAPE" and "TAPE" are not relevant for unmodified teletype transmission over MODEMS, because tape input is enabled by a manual switch.
Control "FORM"	Feed Tape - The control key "FORM" causes 6 inches of tape to be punched as leader.
"Line-Feed"	Start of Record - The 8-bit linefeed code ('212) is defined as a binary start-of-record. Linefeeds are ignored for symbolic input.

"Carriage-Return"	End of Record - The 8-bit carriage return code ('215) is defined as a binary end-of-record. For symbolic input, carriage returns are ignored prior to the first input character; afterwards it is an end-of-record code (seven bit ASCII '015).
Control "EOT"	Binary EOF - The 8-bit code '204 is defined as a binary EOF for standard paper tape handlers, therefore the teletype handler honors this code as an input EOF.

Standard IOCS functions are listed below with a description of their meaning to the teletype handler.

Status (00) - Normal status is returned in register A and register C is set as a function of A.

Symbolic Read (01) - The user buffer is blanked and up to 72 characters of input are accepted, terminated by a carriage return. A carriage return is defined as an end-of-record code, so that a blank line is input if no characters precede the carriage return. Line feed codes are ignored.

A line feed is issued as an input ready signal each time an input request is made.

A "$EOF" statement is defined as an EOF input record.

Symbolic Write (02) - A leading line feed followed by up to 72 characters (trailing blanks are suppressed), and a carriage return are output. C_0 is assumed to be a forms control character and is not output. If C_0 = "1" a triple line feed precedes the output and if C_0 = "0" a double line feed precedes it.

Binary Read (03) - The tape mode is set and the tape reader enabled. Following the start-of-record code ('212) 6-bit binary bytes are accepted and packed four per word until the user's word count is complete or an end-of-record code ('215) is detected. The 8-bit code ('204) is detected as an end-of-file condition.

Binary Write (04) - A start-ot-record code ('212) is output, followed by the user's data unpacked as four 6-bit bytes per word, until the user's word count is complete. The record is terminated with an 8-bit end-of-record code ('215).

Unformatted Write (05) - The lower eight bits (7-0) of every word are output with no conversions. The bits 23-8 are ignored.

Write End-of-File (06) - The message "EOF.. " is output, preceded by a line feed and followed by an EOT and carriage return. The EOT ('204) is a non-printing character, defined as a binary EOF code.

Open (07) - An open flag is set, causing 6 inches of tape leader to be output before the first record if the output is binary.

Close (10) - Null.

Reposition File (11) - The message "RPF.. " is printed and the user placed in wait until the "TAPE" key is pressed by the operator.

Backspace File (12) - Null.

Advance File (13)- The message "ADF.." is printed and the user placed in a wait until the "TAPE" key is pressed.

Backspace Record (14)- The message "BSR.." is printed and the user placed in a wait until the "TAPE" key is pressed.

Advance Record (15)- The message "ADR.." is printed and the user placed in a wait until the "TAPE" key is pressed.

Rewind (16)- Null.

Unformatted Read (17) - Data is input from the terminal and stored in bits 7-0 in the user's buffer, with no data or format conversions. Bits 23-8 are set to zero.

Seek Current Record Address (20)- Null.

Set Current File Address (21)- Null.

Close/Deallocate (22)- Closes and deallocates the device from the calling program.

Special Open (23)- Null.

5-2.4A TI Silent-700 Terminals with Cassettes

One or more Silent-700's are handled by a re-entrant Silent-700 handler. Each Silent-700 is interfaced by a private service routine, allowing local or long distance communications via any standard hardware interface.

Note: Silent-700's without cassettes (KSR models) should be regarded in all respects as teletypes (see section 5-2.4). This section only pertains to ASR models with cassettes.

All functions using the keyboard printer of the Silent-700 are similar to those for a teletype with the following differences:

Special Action Keys:

"RUB-OUT" is not used. To obtain a Line Delete either the "DEL" or the "ESCAPE" key may be used.

"← " is not used. To obtain a Character Delete the underscore key may be used ("US" or "_").

"Control TAPE", "Control NOT TAPE", and "Control FORM" are not used.

"Line Feed" is ignored on all input.

Standard IOCS functions are listed below with a description of their meaning to the keyboard/printer section of the Silent-700.

Status (00) - Normal status is returned in register A and register C is set as a function of A.

Symbolic Read(01) - The user buffer is blanked and a line-feed is issued to indicate an input request is pending. A carriage return is used to terminate input.

A "$EOF" or a "Control EOT" is recognized as an End-of-File.

Symbolic Write (02) - Up to 80 characters from the user's buffer are output (trailing blanks are suppressed). The first character is the user's buffer is not printed but is used as forms control:

"1" causes a triple line feed
"0" causes a double line feed
"+" causes no line feed.

all others cause a single line feed.

Binary Read (03) - Null.

Binary Write (04) - Null

Unformatted Write (05) - The lower eight bits (0 thru 7) of each word are output with no conversion, no line-feed or carriage return. This continues until the word count is complete.

Note: Several characters are processed by the Silent-700 as commands. Because of this, these characters will not be output for any Write function code. Instead a "NUL" character (octal 000) will be output. These characters are:

DLE - octal 020
DC1 - octal 021
DC2 - octal 022
DC3 - octal 023
DC4 - octal 024

Write End-of-File (06) - The message "EOF.." is output.

Open (07) - Null

Close (10) - Null

Reposition File (11) - The message "RPF..." is output.

Backspace File (12) - The message "BSF.." is output.

Advance File (13) - The message "ADF..." is output.

Backspace Record (14) - The message "BSR.." is output.

Advance Record (15) - The message "ADR.." is output.

Rewind (16) - Null

Unformatted Read (17) - Data is input from the terminal and stored in bits 7 thru 0 of the user's buffer (1 character per word). Input data is not echoed to the terminal. Bits 23 thru 8 are set to zero.

Seek Current Record Address (20) - Null

Set Current File Address (21) - Null

Close/Deallocate (22) - Closes and deallocates the device from the calling program.

All other function codes are null.

Cassette I/O - All cassette data is recorded in 86 character blocks. The Silent-700 handler will automaticcaly pack records into a block to avoid waste. Therefore, records may be wholly contained within blocks or they may be spread across 1 or more blocks. The formats of records as processed by the handler are as follows:

Symbolic Records - Leading line feed, followed by the symbolic record without translation, followed by a carriage return.

Binary Records - Leading line feed, followed by the binary record encoded at 4 characters per word. Each character consists of 6 bits from the binary word (most significant bits first) padded by adding octal '40 to the 6 bit character. On a binary read, this padding is removed. The record is terminated by a carriage return.

End-of-File Record - An EOF record is a two character record consisting of an EOT (octal '004) and a carriage return (octal' 015). On input an EOF record is any record which begins with an EOT.

Standard IOCS functions are listed below with a description of their meaning to the cassette section of the Silent - 700.

Status (00) - Normal status is returned in register A and register C is set as a function of A.

Symbolic Read (01) - Input is read from the cassette until a carriage return is received. The remainder of the user's buffer is set to blanks. If the user's buffer is too small, trailing data is lost.

Symbolic Write (02) - Data is written to the cassette from the user's buffer. Trailing blanks are suppressed. A carriage return is output to terminate the record.

Binary Input (03) - Data is transferred from the cassette to the user's buffer according to the format discussed above. Input is terminated by a carriage return.

Binary Output (04) - Data is transferred from the user's buffer to the cassette according to the format discussed above. The record is terminated by a carriage return.

Unformatted Write (05) - The lower eight bits of each word in the user's buffer is written to the cassette without any conversion (see Note under Keyboard/Printer Unformatted Write). Bits 23 thru 8 are ignored. A carriage return is not output.

Write End-of-File (06) - An EOF record (EOT, carriage return) is written to the cassette.

Open (07) - Null

Close (10) - Any partially written cassette blocks are dumped to the cassette. This also occurs whenever a physical tape movement is required.

Reposition File (11) - A backspace file followed by an advance record is performed.

Backspace File (12) - Record are backspaced over until an EOF record is encountered. A subsequent read will input the EOF record.

Advance File (13) - Records are advanced over until an EOF record has been skipped.

Backspace Record (14) - Characters are examined backwards until a carriage return is read. The resultant position is immediately after the carriage return.

Advance Record (15) - Characters are input and ignored until a carriage return has been processed.

Rewind (16) - The cassette is rewound to its "clear leader". It will be "loaded" to its load point upon any subsequent function request except 00-Status, 11-Reposition File, 12-Backspace File or 14-Backspace Record.

Unformatted Read (17) - Data is input from the cassette and stored without conversion i bits 7 thru 0 of the user's buffer. Bits 23 thru 8 are set to zero. Transfer continues unti the user word count is complete.

Seek Current Record Address (20) - Null

Set Current File Address (21) - Null

Close/Deallocate (22) - Closes and deallocates the device from the calling program.

All other function codes are null.

Due to interactions between the keyboard and the cassettes and due to internal bufferin of cassette blocks, the following rules must be observed.

1. When using the Silent-700 as the operator communications device, the cassettes should not be used.

2. The terminal "Control X-OFF" should not be used when cassettes are actually in us (motion).

3. The first operation on a cassette after inserting it into the Silent-700, after any manual positioning of the cassette or after a reboot should be a rewind.

4. The record control tape format switch must be in the CONTinous mode.

5-2.5 Paper Tape Reader

The following is a list of valid I/O functions and their definitions for the paper tape reader.

Symbolic Read (01)- Data is accepted on an interrupt basis (one character per interrupt) until a carriage return character is detected. If a carriage return is detected before the word count is complete, the user's buffer is filled with blanks. If the word count is satisfied before receiving the carriage return, the trailing data within the record is ignored. The data is converted to ANSCII and packed three characters per word as it is received. If an up-arrow (↑) is detected during input, the record is ignored and the input operation is reinitialized. A delete code ('377) is ignored. Leading line feed and carriage returns serve as record delimiters and are ignored.

Symbolic Write (02)- Invalid function code.

Binary Read (03)- The binary tape format is four 6-bit frames per word. Each binary record is preceded by an 8-bit line feed ('212) and terminated by an 8-bit carriage return ('215). Data is accepted on an interrupt basis (one character per interrupt). After detection of a line feed, data is packed in the user's buffer, four frames per word. On detection of a carriage return, input is terminated. If the user's word count is satisfied before detecting a carriage return, the status word (B15-B0) will indicate the actual number of words transferred and B22 is set.

Binary Write (04)- Invalid function code

Unformatted Read (05) - Eight-bit bytes are input on an interrupt basis and stored, one byte per word into bits 7-0, without conversion until the specified word count (byte count) is satisfied.

Write EOF (06) - Invalid function code.

Open (07)- Null.

Reposition File (11)- Operator hold message "RPF TR" is typed. The operator should reposition the tape to the end of the previous End-of-file record and release.

Backspace File (12)- Operator hold message "BSF TR" is typed. The operator should reposition the tape to the start of the previous End-of-file record and release.

Advance File (13)- Tape is advanced past the next End-of-File record.

Backspace Record (14)- Operator hold message "BSR TR" is typed.

Advance Record (15)- Tape is advanced past the next carriage return.

Rewind (16)- Null.

Set Current Record Address (17)- Invalid function code.

Seek Current Record Address (20)- Invalid function code.

Set Current File Address (21)- Invalid function code.

Close/Deallocate (22)- Null.

Special Open (23)- Null.

5-2.6 Paper Tape Punch

The following is a list of the I/O functions and their definitions for the paper tape punch.

Symbolic Read (01)- Invalid function code.

Symbolic Write (02)- The handler outputs three 8-bit frames for each word of the user's buffer. A leading line feed ('212) and a trailing carriage return ('215) are generated by the handler. Data is unpacked and transferred on an interrupt basis until the word count is complete.

Binary Read (03)- Invalid function code.

Binary Write (04)- The handler outputs four 6-bit frames for each word of the user's buffer. An 8-bit line heed ('212) is punched as a start of record and a carriage return ('215) as an End-of-Record.

Special Action (05)- Unformatted Write. The handler outputs one 8-bit frame for each word (7-0) in the user's buffer. Transfer continues until the word count is satisfied. Start and End-of-Record codes are suppressed, permitting the user to generate any type of formatted type.

Write EOF (06)- Punches End-of-File code (EOT = '204) as the start code, followed by a carriage return. This generates a zero word record with EOT as the start-of-record.

Open File (07)- Punch power is turned on and 18 inches of leader (blank tape) is generated.

Close File (10)- Punches End-of-File code (EOT = '204), generates 18 inches of trailer (blank tape), and turns off punch power.

Reposition File (11)- Invalid function code.

Backspace File (12)- Invalid function code.

Advance File (13)- Invalid function code.

Backspace Record (14)- Invalid function code.

Advance Record (15)- Invalid function code.

Rewind (16)- Null.

Set Current Record Address (17)- Invalid function code.

Seek Current Record Address (20)- Invalid function code.

Set Current File Address (21)- Invalid function code.

Close/Deallocate (22)- Closes the device as with function 10 above, if not already done, and deallocates the device from the calling program.

Special Open (23)- Invalid function code.

5-2.7 Card Reader

Card input is buffered and converted according to the IOCS input request. Some special checks are made by the handler before allowing a transfer to the user buffer.

A card image of "$26" in columns 1-3 is thrown away and the HO26 code conversion flag is set. A "$29" card sets the HO29 flag. HO29 is assumed as default. A 9/8 multi-punch in column one is defined as an EOF record.

All other records are passed to IOCS and tested for "$" job control statements. If they pass these tests, data is passed to the user buffer according to the following function requests.

Symbolic Read (01)- A single 80 column card is converted from 026 or 029 code to internal ANSCII, three characters per word, until the user's buffer is filled (byte 81, and all other words greater than 27 (if any) will be blank).

Symbolic Write (02)- Invalid function code.

Binary Read (03)- Consecutive cards are accepted and passed to the user's buffer, until the specified word count is satisfied or until an End-of-Record card is detected. Binary cards are formatted as follows:

- Column 1 - Reserved for a special action code: A 9/8 multi-punch signifies an End-of-File record, and a 9/7 multi-punch is a partial code signifying that this card does not complete the binary record.
- Column 2 - blank.
- Columns 3 through 6 - Reserved for a user sequence number.

  The handler ignores 2-6.
- Columns 7 through 80 - Contain 12-bit binary data allowing 37 data words per card, at 2 columns per word.

Binary Write (04)- Invalid function code.

Hollerith Read (05)- A single 80-column card is transferred to the user's buffer, without conversion, and packed two columns per word.

Write EOF (06)- Invalid function code

Reposition File (11)- Operator hold message "RPF CR" is typed.

Backspace File (12)- Operator hold message "BSF CR" is typed.

Advance File (13)- Cards are input until an End-of-File card (9/8 multi-punch in column 1) is detected.

Backspace Record (14)- Operator hold message "BSR CR" is typed.

Advance Record (15)- Cards are input and ignored until a card not containing a 9/7 multi-punch in column 1 is detected.

Rewind (16)- Null.

Set Current Record Address (17)- Null.

Seek Current Record Address (20)- Null.

Set Current File Address (21)- Null.

Close/Deallocate (22)- The device is deallocated from the calling program.

Special Open (23)- Null.

5-2.8 Card Punch

The following is a list of the I/O functions and their definitions for the card punch.

Symbolic Read (01)- Invalid Function Code.

Symbolic Write (02)- Converts ANSCII to Hollerith core and punches 1 card. Conversion code is assumed to be HO29 unless system option 23 is set, requesting HO26 conversion

Binary Read (03)- Invalid function code.

Binary Write (04)- Punches one or more cards as a binary record. Column 1 of each card is reserved as a special action code. A 9/8 Multi-punch in column 1 indicates an End-of-File record (one card). A 9/7 multi-punch indicates a partial record. Columns 2 through 6 are open for a sequence number (the handler does not generate a sequence number).

Special Action (05)- Punches 80 columns, 2 columns per word, unformatted.

Write EOF (06)- A 9/8 multi-punch is punched in column 1.

Open (07)- The device is allocated to the calling program.

Close (10)- A blank card is punched.

Reposition File (11)- Invalid function code.

Backspace File (12)- Invalid function code.

Advance File (13)- Invalid function code.

Backspace Record (14)- Invalid function code.

Advance Record (15) - Invalid function code.

Rewind (16)- Null.

Set Current Record Address (17)- Invalid function code.

Seek Current Record Address (20)- Invalid function code.

Set Current File Address (21)- Invalid function code.

Close/Deallocate (22)- The device is closed and deallocated from the user's program.

Special Open (23)- Invalid function code.

5-2.9 Line Printer

The following is a list of valid I/O functions and their definitions for the line printer.

Symbolic Read (01)- Invalid function code.

Symbolic Write (02)- Data is assumed to be ANSCII, 3 C/W. The first column is assumed to be a carriage control character and the printer responds in the manner defined in Table 5-5. Special characters "1", "0", " ", or " + " in this position are replaced by the appropriate control character as indicated in Table 5-5.

Binary Read (03)- Invalid function code.

Binary Write (04)- Invalid function code.

Special Action (05)- Invalid function code.

Write EOF (06)- The message "EOF.." is printed.

Reposition File (11)- Invalid function code.

Backspace File (12)- Invalid function code.

Advance File (13)- Invalid function code.

Backspace Record (14)- Invalid function code.

Advance Record (15)- Invalid function code.

Rewind (16)- Null.

Set Current Record Address (17)- Invalid function code.

Seek Current Record Address (20)- Invalid function code.

Set Current File Address (21)- Invalid function code.

Close/Deallocate (22)- The device is deallocated from the calling program.

Special Open (23)- Invalid function code.

Table 5-5
Line Printer Carriage Control

Carriage Control Character	Action		
"@" or " + "	0 line advance	Accepted by all printers	All spacing is done before printing.
"A" or " "	1 line advance		
"B" or "0"	2 line advance		
"C"	3 line advance		
"D"	4 line advance	Accepted only by obsolete printers	
.	.		
.	.		
"N"	14 line advance		
"O"	15 line advance		
"P"	Channel 1 (top of form)	Accepted by all printers	
"Q"	Channel 2		
"R"	Channel 3		
"S"	Channel 4		
.	.	Accepted by only some printers	
.	.		
"W"	Channel 8		
: (colon)	Causes columns 4-18 to be output to the operator as a Hold Message, upon release, program execution continues.		
Other	1 line advance		

5-2.10 Real-Time Peripheral Equipment

The RTP Handler will handle the RTP I/O Expander and up to 30 RTP devices, each containing up to 16 channels, for a total of 480 controlled channels. Each RTP "Device", or set of 16 channels on a common controller, is assigned a particular DMS physical device number, and can be open to, at most, one program at a time. Thus, when a program opens a particular RTP device, it has sole control of the 16 channels on the device.

DMS physical device numbers are associated with RTP device numbers through a resident DMS table which is defined at system generation (see Paragraph 12-3.17). A second system table contains a bit for each channel on each RTP device which is used to determine whether the particular channel is interrupt or non-interrupt oriented.

I/O is performed to the RTP devices through standard I/O calling sequences utilizing function codes 01 through 04. For each call, two buffers are transferred to the handler.

As can be noted, there are two distinct methods of programming RTP equipment: Sequential and Random. In sequential operation the I/O transfer is begun at a particular card or channel on the RTP equipment and RTP equipment automatically advances through the channels until the entire word count has been transferred. In this case the user supplied only the initial channel address. In random mode, a channel address is supplied for each data word transfer, and automatic selection is not done by the equipment.

When communicating through the RTP Handler, the following I/O calling sequence is used:

```
          TLO       PARLIST
          BLU       $I/O
          .
          .
          .
PARLIST   DATA      XXYY
          DAC       word count
          DAC       buffer address
          DAC       connection into buffer address
```

The first two parameters are as in all other DMS I/O operations. The third parameter (buffer address) is the address of the buffer from/to which the data is transferred. It must be at least "word count" words long. The fourth parameter (connection info buffer address) tells which channels/slots the data transfer is to be done from/to. In the case of sequential mode transfers, only the first word of this buffer is referenced. In random mode transfers, each channel/slot number in the connection buffer corresponds to a word in the data buffer, requiring the connection buffer - to be at least "word count" words in length. The actual format of the contents of both of these buffers is variable depending on the device involved, and is described in the following sections.

Common information is described below:

1. Data Buffer (First buffer address)

   Bits 15-0 contain the data in/out. These correspond to RTP bits 0-15 respectively.

   Bits 23-18 on input operations contain the card slot address 7420/20, 7420/30 only) from which the data was input. These bits are undefined in all other operations.

2. Connection Buffer

Bits 10-0 contain the card slot address, channel number, gain information, etc., as required by the particular device. These formats are given in the following sections Bits 10-0 correspond to RTP Bits 5-15.

Bit 17 if set indicates that this transfer, and all others following it in I/O operation, are to be done in interrupt mode regardless of the settings in RTPTB2.

Special Note: When mixing interrupt and non-interrupt I/O in the same I/O call to the handler - all non-interrupt I/O must precede interrupt I/O in terms of connection buffer ordering.

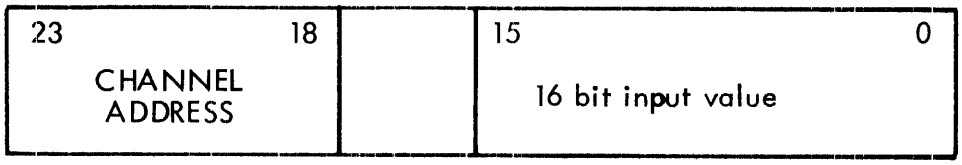
Device Programming Considerations

The following sections define particular bit configurations and considerations for all RTP modules.

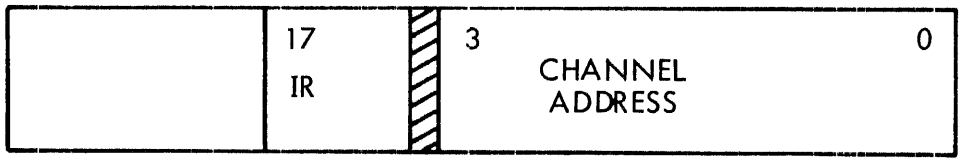
9472 7435/20 Digital Input

Interrupt: Optional - external device supplied

Data Buffer:



Connection Buffer:

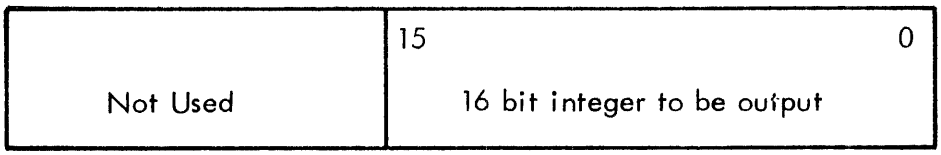


Special Notes: None

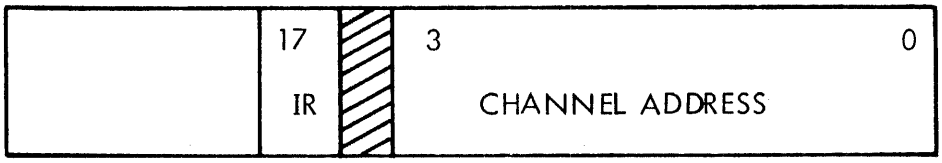
9473 7435/22 Digital Output

Interrupt: Optional - external device supplied

Data Buffer:



Connection Buffer:

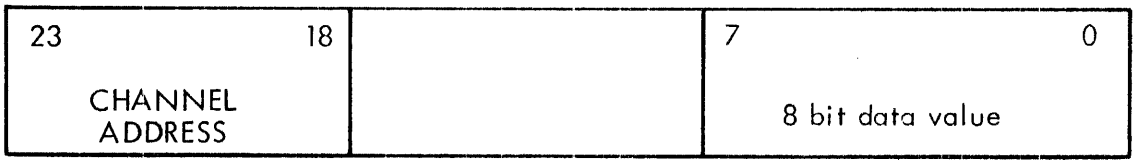


Special Notes: None

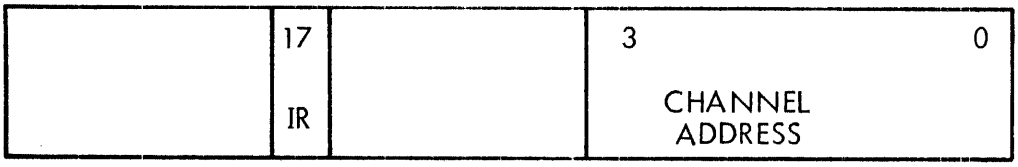
9474 7435/24 AC Input

Interrupt: Optional - configuration item.

Data Buffer:



Connection Buffer:

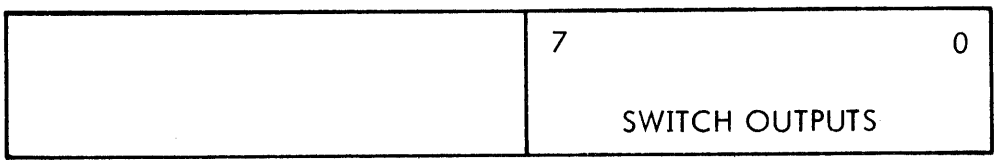


Special Notes: None

9475 7435/25 AC Output

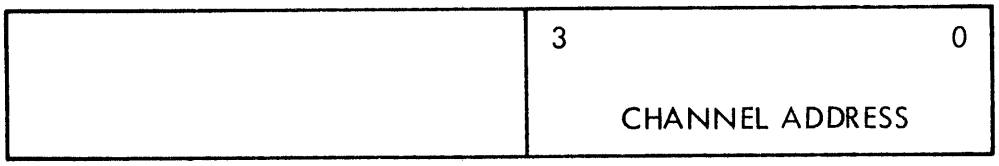
Interrupt: None

Data Buffer:



1 = Switch Closed - Conducting
0 = Switch Open

Connection Buffer:

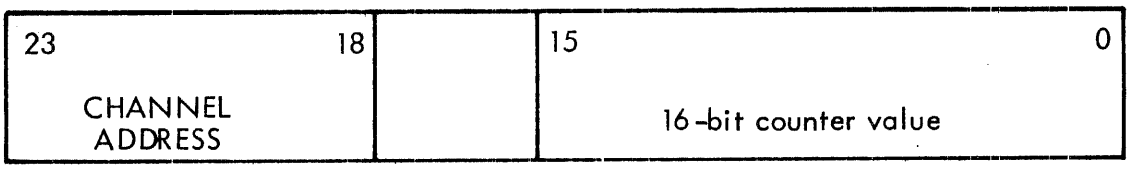


Special Notes: None

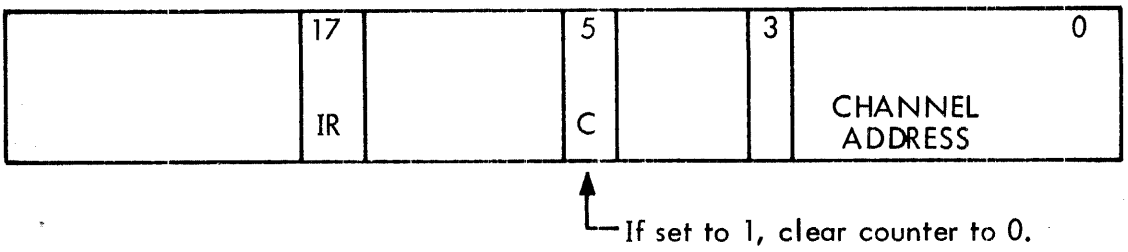
9476 7435/33 Pulse Counter

Interrupt: On reaching total count of 65536 if detection of interrupt is desired, place one-word input request to device with B17 of connection buffer to wait for input interrupt.

Data Buffer:



Connection Buffer:

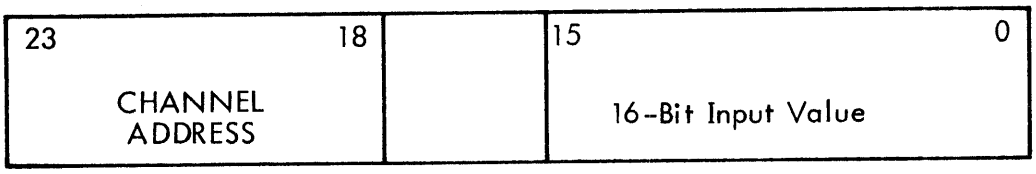


Special Notes: To clear counter, do a one-word (non-interrupt) with Bit 5 of connection Buffer Set and ignore input.

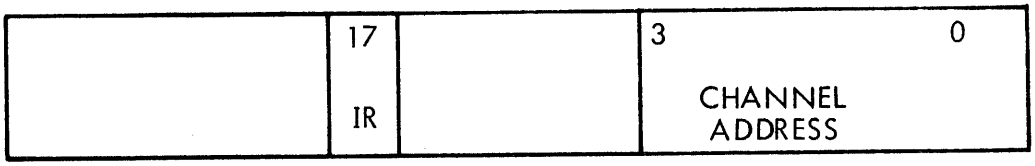
9478 7435/37 Optically Isolated Input

Interrupt: Optional - externally supplied

Data Buffer:



Connection Buffer:



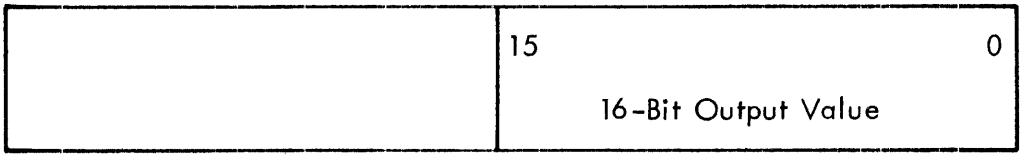
Special Notes: None

9479 7435/38 Dry Relay Output

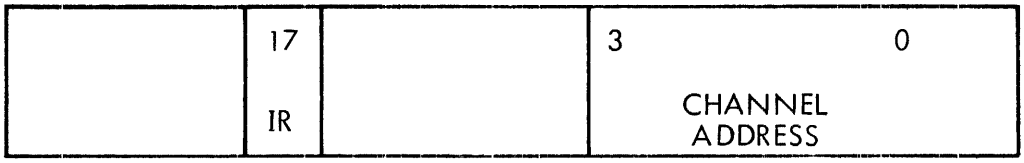
9480 7435/39 Mercury Relay Output

Interrupt: Optional - externally supplied

Data Buffer:



Connection Buffer:

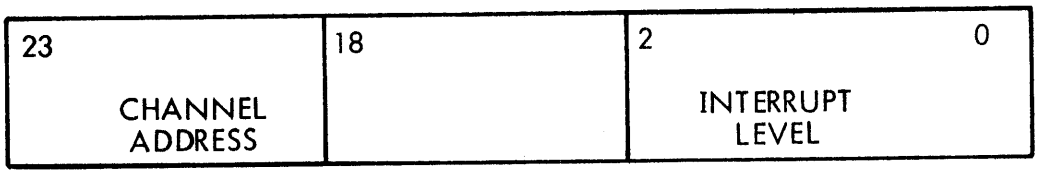


Special Notes: None

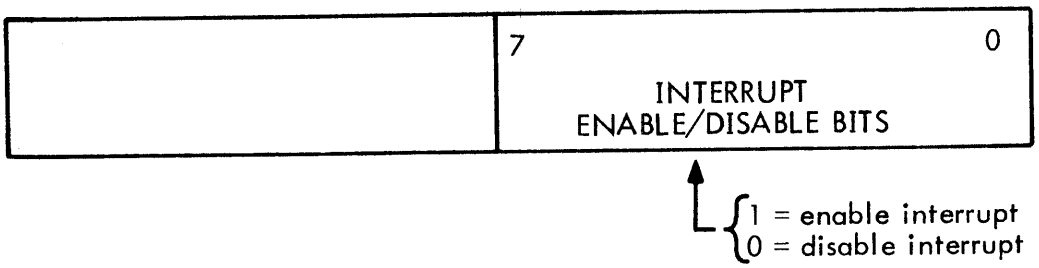
9481 7435/42 Interrupt Expander

Interrupt: Yes, use Bit 17 of Connection Buffer only.

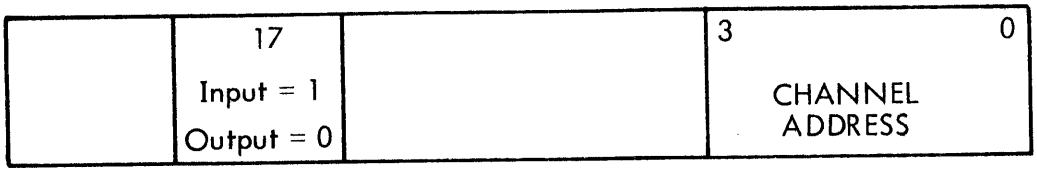
Data Buffer (Input):



Data Buffer (Output):



Connection Buffer: (I and 0)



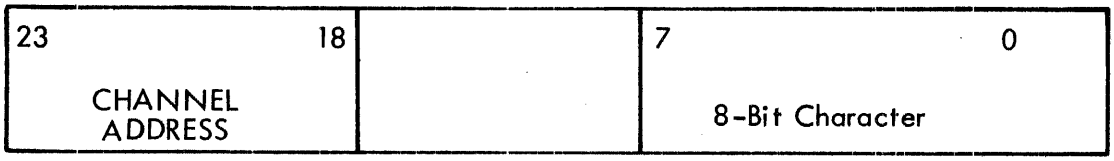
Special Notes:

1. Normal programming of Interrupt Expanders through handler is as follows:
   A. Do output with B17 of connection buffer off to enable selected interrupts.
   B. Place one word input request on the card slot with B17 of connection buffer set.
   C. When an interrupt occurs, determine location from buffer and repeat step B for next interrupt.
2. Bit in RTPTN2 should be 0 for Interrupt Expanders.

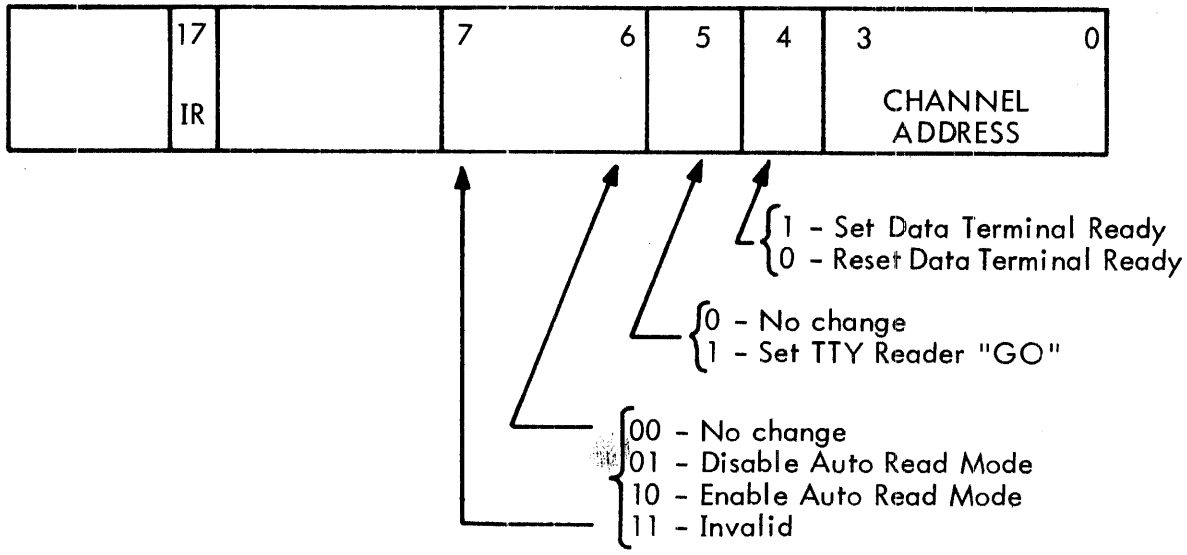
9482 7435/45 Serial Input Module

Interrupt: On each input character

Data Buffer:



Connection Buffer:



Special Notes:

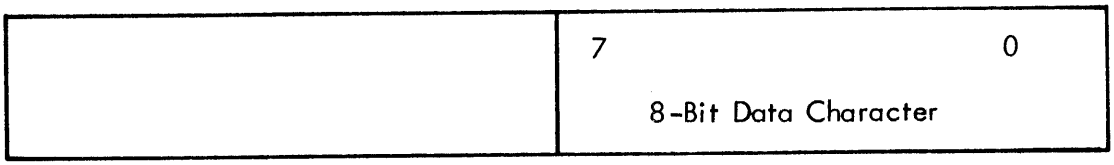
1. The "TTY" Reader "GO" and "Auto Read" Modes are used when a TTY is directly connected to the Serial Input Module. The Auto Read Mode, when enabled causes "TTY Reader GO" to be set after each character is input. The "TTY Reader GO" signal is normally applied to the Teletype Tape Reader Mechanism to control paper tape input. When using a Data Set, these signals are not used.

2. Start and Stop bits on the serial character are not passed into the input character and parity is not checked.

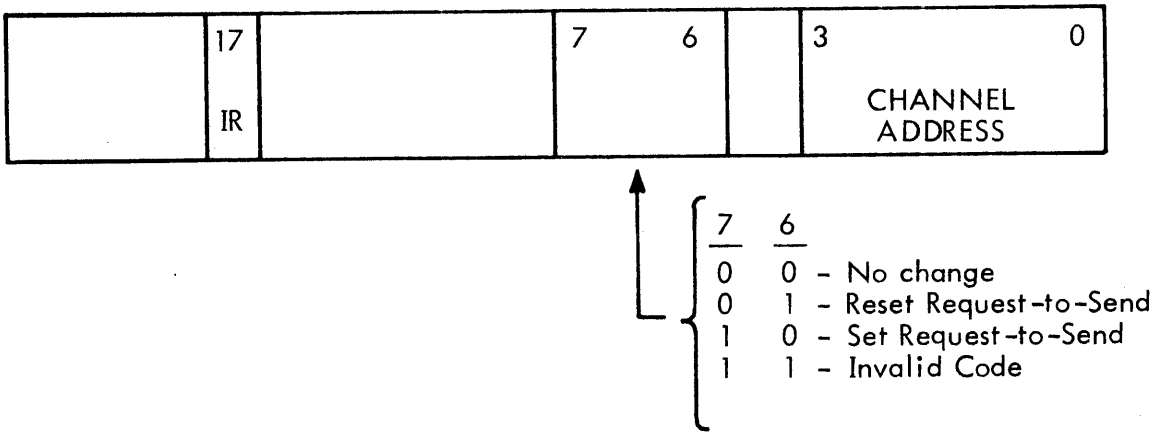
9483 7435/46 Serial Output Module

Interrupt: On each character to indicate ready

Data Buffer:



Connection Buffer:



Special Notes: None

9484 7435/47 Analog Input Module

9485 7435/49 Analog Input Module

Interrupt: Yes

Data Buffer:

23 18		15 12	11	10 0
CARD SLOT		EXTENDED SIGN	SIGN	DATA VALUE

Connection Buffer:

	17		7 4	3 0
	IR		CHANNEL NO. WITHIN CARD	CARD SLOT #

Special Notes:

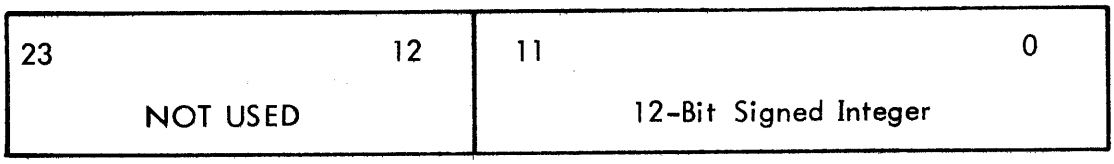
1. Use of this card requires a bit in the appropriate word in RTPTB3.
2. "Card Slot" on this sheet refers to the channel address on the 9471 - 7430/30 - Common Equipment Chassis.

9462 7445/20 Analog Output - Sample and Hold

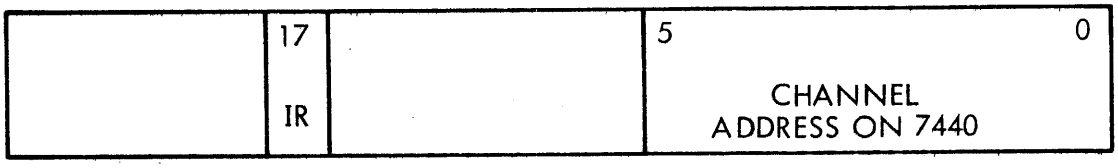
9463 7445/21 Analog Output - Sample and Hold

Interrupt: Yes

Data Buffer:



Connection Buffer:

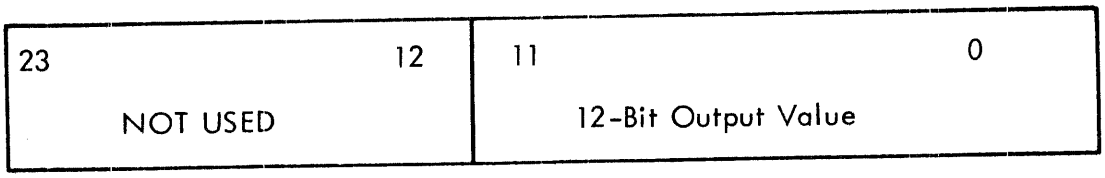


Special Notes: None

9451	7455/20	Analog Output Modules
9452	7455/21	
9453	7455/22	
9454	7455/23	
9455	7455/24	

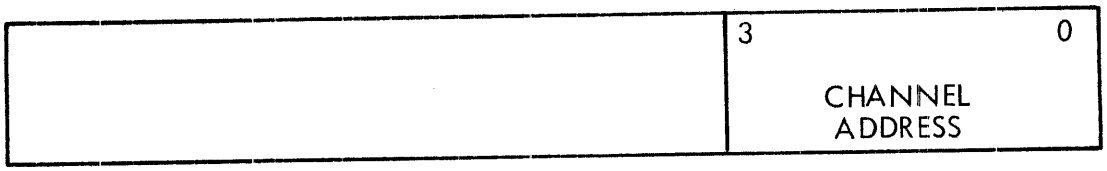
Interrupt: None

Data Buffer:



This is optionally a 12-Bit Signed Integer depending on Card Model.

Connection Buffer:



Special Notes:

1. These cards can be used on both of these devices:

   9450 7450/20 Analog Output Common

   9471 7430/30 Universal Common

9443 7465/20 High Level Analog Input

9443 7465/21 High Level Analog Input

Interrupt: Yes

Data Buffer:

23 16	15 12	11 0
UNDEFINED	SIGN EXTENDED	12-Bit Signed Integer Data Value

Connection Buffer:

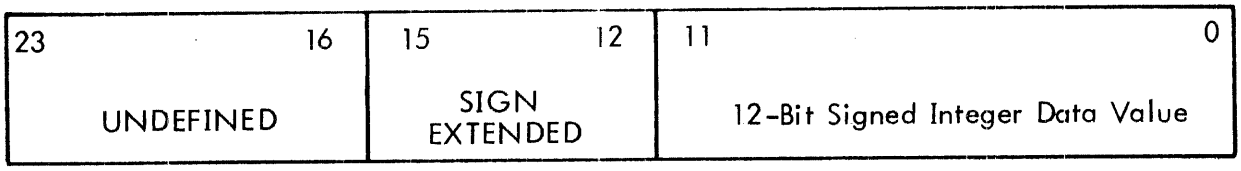
	17		10 4	3 0
	IR		CHANNEL ADDRESS	NOT USED

Special Notes: For devices containing these modules (7460/XX), the entire word in RTPTB3 should be set to -1 (77777777_8).

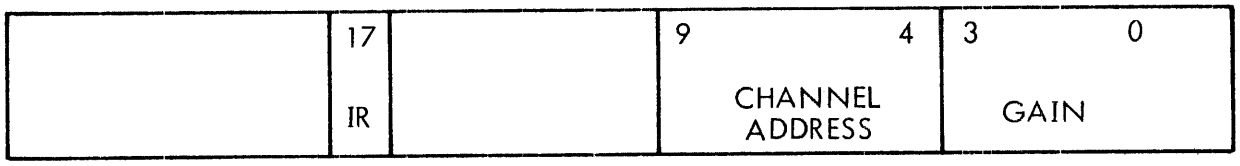
9431 7476/20 Low Level Analog Input

Interrupt: Yes

Data Buffer:



Connection Buffer:



Special Notes:

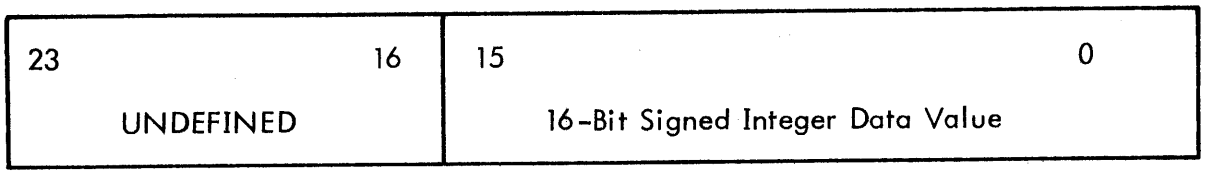
1. The Sequential Input Function will not operate on these modules. Random must be used.
2. Consult the RTP device manual (PM070-006/037) for information or construction of the gain information.
3. The entire RTPTB3 word in the DMS SYSDAT must be set to -1 (77777777_8) for proper operation.

9421 7485/30 Wide Range Analog Input

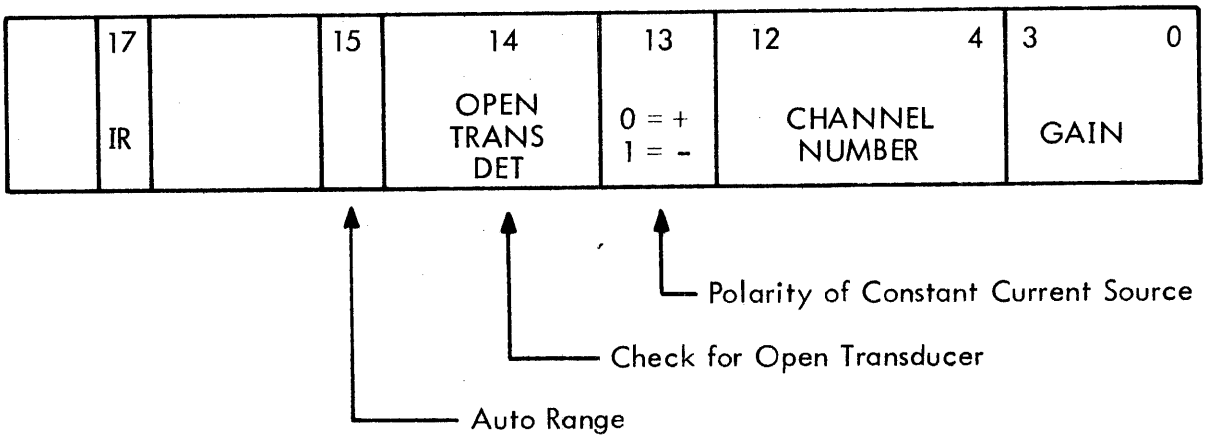
9422 7485/31 Wide Range Analog Input

Interrupt: Yes

Data Buffer:



Connection Buffer:



Special Notes:

1. The Sequential Input Function will not operate on those modules. Random must be used.
2. Consult the RTP device manual (PM 070-027) for information on the items in the connection buffer.
3. The entire RTPTB3 word in the DMS SYSDAT must be set to -1 (77777777_8) for proper operation.

Function Codes

Sequential Read (01) - Performs an input operation to the specified device, setting the sequential (automatic channel increment) mode, starting with the channel number in the first element of the connection buffer.

Sequential Write (02) - Performs an output operation to the specified device, setting the sequential (automatic channel increment) mode, and starting with the channel number specified in the first element of the connection buffer.

Random Read (03) - Performs an input operation on the specified device, setting the random access mode. Individual channel numbers are taken from each corresponding position of the connection buffer for each data word input.

Random Write (04) - Performs an output operation on the specified device, setting the random access mode. Individual channel numbers for each output data word are taken from the corresponding position of the connection buffer.

Set Completion Variable (05) - The first buffer address of the standard parameter list is the address of a variable in the users program to be decremented by one whenever an input or output operation is complete. Once this variable has been set into the handler, it is decremented by one each time the device completes an operation. This variable parameter may be cleared at any time by passing a zero address for the variable. This service allows user programs to monitor the status of an operation by checking a variable within the program.

Open (07) - The particular RTP device is allocated to the calling program and the device storage area is allocated and initialized.

Close/Deallocate (22) - The particular RTP device is closed and released, allowing other programs to open it. The device storage area is deallocated.

5-2.11 Alphanumeric CRT's

One or more CRT's are handled by a re-entrant Display Handler. Each handler is interfaced by a private service routine, allowing local or long distance communications via any standard hardware interface. Any one of the CRT's may be designated as the operator's terminal at system generation (see Section XII). The operator's terminal may be alternately specified at system boot time by setting sense switch one along with the desired terminal device number in control switches zero through five, before activating the bootstrap.

The specified operator's terminal functions as a normal remote terminal in addition to being the operator communication console. The Operator Communication Service (OPCOM) is not accessible from any terminal other than the operator's console. The operator may relinquish operator communications to another terminal via the command/OT,n where "n" is the desired terminal device number.

The re-entrant display handler honors standard IOCs requests for all CRT's including the assigned operator's terminal. Three special characters are acknowledged when in the first column of any input line, "/", "\", and "⊥". The "/" signifies an OPCOM input statement,

"\" signifies a terminal abort request, and "⊥" signifies a handler mode command. The handler treats line zero as an input media, and lines 1 through 23 as an output page. A symbolic write below line 23 is handled as determined by the current handler mode. Three handler modes are available. In the "scroll" mode (which is the default mode), a symbolic write below line 23 is written on line 23 after line 1 has been deleted and lines 2-23 have been scrolled up one line. In the "page" mode, the screen is erased and the line is written on line 1. In the "wait" mode, an uparrow (^) character is output in the home position and the handler waits for the user to enter a transmit. When the transmit is received, the screen is erased and the line is written on line 1.

The handler modes are selectable from the terminal by entering a "⊥S" for scroll, "⊥P" for page, and "⊥W" for wait mode. The modes may also be set by the use of function code 17* (Set Handler Mode).

When an OPCOM statement is detected, the display handler makes sure that the terminal is the currently allocated console device, and then passes the statement to Operator Communications. If a regular IOCS input request was pending, it is re-initialized after the OPCOM statement has been passed.

A terminal abort request causes the I/O Executive to abort the foreground program currently connected to the terminal, and clears the screen of the display.

If an input statement is transmitted and no request is pending the statement will remain displayed and a handler flag set until an input request is made, at which time the statement is accepted and the line cleared. If an input request is made by a program and no data has been transmitted, the request is saved but the device remains "not busy" to permit output requests. When a transmit is detected, the pending input is processed.

Standard IOCS functions are listed below with a description of their operation.

Status (00) - A normal status word is returned in Register A and the condition register is set as a function of A.

Symbolic Read (01) - The user's buffer is blanked and up to 80 input characters, packed 3 characters per word, are accepted. A "$EOF" statement is defined as an input end-of-file. The parameter "n" specifies the word count to be input.

Symbolic Write (02) - Up to 81 characters are transmitted to the next available line of the output page. Column zero (carriage control) is ignored. Data is assumed to be packed three characters per word ASCII. The parameter "n" specifies the word count to be output.

Edit Read (03) - "n" characters are accepted from the specified cursor position and packed in the user's buffer three characters per word. The calling sequence for this request is:

```
DAC     'XX03
DATA    n
DAC     Buffer address
DATA    /0,row,col/     where 8 bit bytes per field are used.
```

The cursor position is saved and restored to its original position at the end of the operation.

Edit Write (04) - Outputs "n" characters at the specified cursor position. The calling sequence is:

```
DAC     'XX04
DATA    n
DAC     Buffer address
DATA    /0,row,col/     where 8 bit bytes per field are used.
```

The current cursor position is saved and restored to its previous state at the end of the output operation.

Operator Message (05) - Outputs six character message on line one in column 75 through 80. The calling sequence is as follows:

```
DAC     'XX05
DATA    n
DAC     Buffer address
```

The character count "n" must be adequate to include "blink" and "unblink" codes, if present, but should not exceed six.

Input Wait (05*) - If a parameter one (character count) of a function 05 call is zero, the calling program is placed in a wait until a transmit code is detected from the keyboard. No transfer takes place.

Write End-of-File (06) - Writes the output message "EOF. . " on the next available output line.

Open (07) - This function sets the output line number for top-of-page (first output line).

Rewind (16) - This function erases the current display page and sets the output line number for top-of-page (first output line).

Set Line Number (17) - Sets the output line pointer to the specified line number. The calling sequence is:

```
DAC     'XX17
DATA    line number (1 for first output line)
```

Set Handler Mode (17*) - If the parameter of a function 17 call is negative, the handler mode is set as specified by the parameter. Valid parameters are: -1 for scroll mode, -2 for page mode, and -3 for wait mode. All other negative values are invalid. The calling sequence is:

```
DAC     'XX17
DATA    parameter (-1,-2, or -3)
```

Set/Reset Data - Panel Lights (17**) - This function turns the data-panel lights on or off as specified by the parameter. On CRT's not equipped with data-panel lights, this is a null function. The lights to be turned on or off are specified, one bit per light, in bits 0-15 of the parameter word. Lights not specified remain unchanged. Bits 23-21 of the parameter word determine whether the specified lights are to be turned on or off. A 6 in this position (B23 B22 set, B21 reset) will turn off the specified lights, a 4 in this position (B23 set, B22 B21 reset) will turn on the specified lights. The calling sequence is:

```
DAC     'XX17
DATA    parameter
```

Get Cursor Address (20) - Returns Row/Column in register E, bytes 2 and 1 respectively (high order bits of E are zero) of the current cursor position.

Set Cursor Address (21) - The CRT cursor is position to the specified row and column. The calling sequence is:

```
DAC     'XX21
DATA    /0,row,column/      where 8 bit bytes are used for fields.
```

NOTE

"row" specifies a horizontal row between 0 and 23 where row 0 is the input line and rows 1-23 are the output page; and "column" specified the vertical column number between 1 and 80.

Functions greater than 22 are invalid and all others are "null".

5-2.12 Reader/Punch

The reader/punch may be considered to be logically two separate devices, either a card reader or card punch. Depending on the entry point referenced, the unit will perform the specified operation for that logical device. A word in the SYSTEM DATA MODULE, RPAMOD, allows access to one or both of the logical devices. The "mode word" is defined as follows:

"RPA MOD"	DEVICE
Negative	Reader Only
Zero	Reader or Punch
Positive	Punch Only

A reference to the punch when in "reader only" mode will generate an abort, ABT 10.

In the "reader and punch" mode, the device will be used as a card reader until a $CLOSD card is read. At that time, the device is allocated to the first program needing the device. The logical device opened first will determine whether the device will read or punch.

For example, to spool in a job which generates punch output the following procedure could be followed:

Spooling in Jobstream

$JOB...

.
.
.

$EOJ

Deallocate the spool in card reader

$CLOSD

.
.
.

BLANK CARDS

.
.
.

The program will produce punch output on cards. Upon a close to the punch, the extra blank cards will be flushed and offset until the first non-blank card is detected by the reader.

5-2.12.1 Card Reader

Card input is buffered and converted according to the IOCS input request. Some special checks are made by the handler before allowing a transfer to the user buffer.

A card image of "$26" in columns 1-3 is thrown away and the HO26 code conversion flag is set. A "$29" card sets the HO29 flag. HO29 is assumed as default. A 9/8 multi-punch in column one is defined as an EOF record.

All other records are passed to IOCS and tested for "$" job control statements. If they pass these tests, data is passed to the user buffer according to the following function requests.

Symbolic Read (01) - A single 80 column card is converted from 026 or 029 code to internal ANSCII, three characters per word, until the user's buffer is filled (byte 81, and all other words greater than 27 (if any) will be blank).

Symbolic Write (02) - Invalid function code.

Binary Read (03) - Consecutive cards are accepted and passed to the user's buffer, until the specified word count is satisfied or until an End-of-Record card is detected. Binary cards are formatted as follows:

- Column 1 - Reserved for a special action code: A 9/8 multi-punch signifies an End-of-File record, and a 9/7 multi-punch is a partial code signifying that this card does not complete the binary record.
- Column 2 - Blank.
- Columns 3 through 6 - Reserved for a user sequence number.

  The handler ignores 2-6.
- Columns 7 through 80 - Contain 12-bit binary data allowing 37 data words per card, at 2 columns per word.

Binary Write (04) - Invalid function code.

Hollerith Read (05) - A single 80-column card is transferred to the user's buffer, without conversion, and packed two columns per word.

Write EOF (06) - Invalid function code.

Reposition File (11) - Operator hold message "RPF CR" is typed.

Backspace File (12) - Operator hold message "BSF CR" is typed.

Advance File (13) - Cards are input until an End-of-File card (9/8 multi-punch in column 1) is detected.

Backspace Record (14) - Operator hold message "BSR CR" is typed.

Advance Record (15) - Cards are input and ignored until a card not containing a 9/7 multi-punch in column 1 is detected.

Rewind (16) - Null.

Set Current Record Address (17) - Null.

Seek Current Record Address (20) - Null.

Set Current File Address (21) - Null.

Close/Deallocate (22) - The device is deallocated from the calling program.

Special Open (23) - Null.

5-2.12.2 Card Punch

The following is a list of the I/O functions and their definitions for the card punch.

Symbolic Read (01) - Invalid Function Code.

Symbolic Write (02) - Converts ANSCII to Hollerith core and punches 1 card. Conversion code is assumed to be HO29 unless system option 23 is set, requesting HO26 conversion.

Binary Read (03) - Invalid function code.

Binary Write (04) - Punches one or more cards as a binary record. Column 1 of each card is reserved as a special action code. A 9/8 multi-punch in column 1 indicates an End-of-File record (one card). A 9/7 multi-punch indicates a partial record. Columns 2 through 6 are open for a sequence number (the handler does not generate a sequence number).

Special Action (05) - Punches 80 columns, 2 columns per word, unformatted.

Write EOF (06) - A 9/8 multi-punch is punched in column 1.

Open (07) - The device is allocated to the calling program.

Close (10) - A blank card is punched.

Reposition File (11) - Invalid function code.

Backspace File (12) - Invalid function code.

Advance File (13) - Invalid function code.

Backspace Record (14) - Invalid function code.

Advance Record (15) - Invalid function code.

Rewind (16) - Null.

Set Current Record Address (17) - Invalid function code.

Seek Current Record Address (20) - Invalid function code.

Set Current File Address (21) - Invalid function code.

Close/Deallocate (22) - The device is closed and deallocated from the user's program.

Special Open (23) - Invalid function code.

5-2.13 Printer/Plotter

The printer/plotter allows two modes of operation. A "symbolic write" request instructs the device to function as a normal list output device. A "binary write" request instructs the device to perform as an electrostatic plotter. The device handler is non-re-entrant and is executed via the standard IOCS calling sequence. The following is a list of valid function codes and the action associated with each code.

Symbolic Write (02) - Write a symbolic logical record in the print mode. The buffer is assumed to be ASCII symbolic code packed three characters per word, preceded by a carriage control character in the first position in the buffer. Although the printer/plotter does not perform as a standard hard copy print device, it is physically unable to allow the same degree of control exhibited by most "line printers". Notably, there is no VFC or "carriage tape" nor is there a facility for remaining on the same line and "overprinting". Thus, only a restricted set of carriage control characters is honored (all others are effectively treated as a blank): The valid carriage control characters are:

" " (blank) - advance to next line and print.

" 1 " - Advance to next page - A "form feed" is issued.

On a device with the "fan-fold" option a physical top-of-form is issued. A device without the fan-fold option will advance the paper 2 1/2 inches.

"0" - Double Space - Leave a blank line before printing.

" : " - Operator HOLD message. The following line is printed at the OPCOM device and the user program is placed in the HOLD state.

If the word count specified is greater than the physical line length then the record will be truncated. All "unprintable" characters are effectively "null" and thus ignored.

Binary Write (04) - The buffer is considered to be a packed bit string. Since the physical plot buffer is an integral multiple of bytes (8 bits) and not words, a single request to plot multiple lines would require that the plot records span across words (the second and subsequent records are not guaranteed to be word alligned). A "1" bit designates that the nib in that position should print, a " 0 " bit designates absence of print. The word count is software truncated to an integral multiple of plot lines.

Write EOF (06) - Included for system wide compatibility. Effectively a null operation.

Open (07) - Logically opens the device. An EOT action is taken which will advance the paper eight inches, clearing it from the ink reservoir. An open must be issued prior to any other logical I/O operations.

Rewind (16) - Included only for system wide compatability. Effectively a null operation.

Close (1 0) - Logically closes the device.

Close/Deallocate (22) - Logically close the device and allow it to be assigned to another user. An EOT action will be taken to insure that the paper does not stand in the ink reservoir.

All other logical input/output operations are invalid and the program issuing such a request will be aborted.

As an additional note, care should be taken to drive the printer/plotter at an even rate near its maximum speed; otherwise, inertial errors may occur (which may cause an uneven appearance in the quality of the print).

5-2.14 Incremental Plotter

The plotter may be driven directly by a user program or the special foreground program "S. PLOT". The initial open after a reboot will output the message "POSITION PLOT" to the operator console and will suspend the program. Once the operator has positioned the pen to 1/2 inch from the Z-fold, the program may be released. Thereafter, the Z-fold position is maintained by the handler. This enables direct plotting with or without use of the plot support library.

The following is a list of the I/O functions and their definitions. All others are null.

Symbolic Write (02) - The handler outputs three 8-bit plot commands for each word in the user's buffer. The data is unpacked and transferred on an interrupt basic until the word count is complete.

Binary Write (04) - Identical to the symbolic write (02).

Special Action (05) - The handler outputs one 8-bit (7-0) plot command for each word in the user's buffer. Transfer continues until the word count is complete.

Open File (07) - The handler will perform a page restore if the pen is not correctly positioned at a Z-fold. The initial open after a reboot will output an operator message requesting the pen to be positioned.

Close File (10) - The handler will perform a page restore if the pen is not correctly positioned at a Z-fold.

Close/Deallocate (22) - Closes the device as with function 10 above, if not already done, and deallocates the device from the calling program.

SECTION VI
BACKGROUND BATCH PROCESSING

6-1 JOB STREAM

Under DMS, the background input stream is considered the "Job Stream Input File". This is defined as the file or device to which Logical File 00 (Job Stream) is assigned. A number of special features effect the Job Stream Input File. These are discussed below.

6-1.1 Special Assignment (Spooled Systems only)

Under DMS, one does not always know to what file or device his job stream is assigned. This is particularly true in the case of spooled input, since under spooling job stream is assigned to a particular file on disc, the name of which is not immediately available to the programmer. To assign any logical file to use the same file or device as job stream, one merely assigns the LFN to Physical Device 77_8. The I/O control supervisor will automatically transfer the request to Physical Device 77 to the Job Stream File. Thus the assignment

```
$ASSIGN 7,77
```

will assign the Source Input (LFN 7) to use the Job Stream Input File. This feature is not available in unspooled DMS configurations.

6-1.2 Reading "$" Cards

Whenever a card containing a dollar sign ($) in column 1 is read from the Job Stream file or device by any processor other than Job Control, special action occurs. Under these circumstances, DMS will: 1) Copy the "$" card to the resident Job Control buffer, 2) Set the system "trap" flag so that Job Control can process this record, 3) Return the card image containing the "$" to the user's buffer, and 4) Return the end-of-file status to the calling program.

If the processor performs another read request to the file without clearing the "trap" flag via the UNTRAP service, the processor exits and Job Control will be loaded and will attempt to process the record. If the "trap" flag is cleared, (see paragraph 4-16) then further reads can be made.

6-1.3 File Insertion Feature

The File Insertion feature, available in spooled DMS systems only, allows the insertion of a specified disc file in the Input Job Stream at any point. This is accomplished by using the $ADD card which is written as

```
$ADD FILNAM
```

This card effectively causes the contents of "FILNAM" to be inserted in the job stream in place of the $ADD card. The $ADD card may be placed anywhere in the job stream, in particular, in the middle of a data deck, etc. The $ADD card is transparent to all background processors. If "FILNAM" is non-existent, the message ">>$ADD FILNAM NON-EXISTENT" will be output to the list output device and the job will continue.

The file which is being ADDed must be terminated with a $RETURN control card. When DMS reads this record, the next image returned to the calling program will be the next line following the previously encountered $ADD card. No password may be used on ADD files, and $ADD cards may not be nested.

The $ADD card may not be used in jobs read in the unspooled mode (see paragraph 8-6. 1).

6-2 JOB CONTROL PROGRAM

Job Control (JOBCTL) is a special system program that is loaded and executed in the DMS background whenever the system is initialized and whenever any background program is terminated. Job Control executes in the unrestricted mode such that it can modify resident DMS tables as necessary to direct background processing. JOBCTL accepts control statements from the input device assigned to job Stream (logical file 0), outputs each statement to logical file 6, and then processes valid statements as described in subsequent paragraphs of this section. Invalid statements cause an abort with the message "JCL ERROR" output to logical file 6. Invalid statements entered from the operator communication device (LFN 0 assigned to device 1) cause an operator hold message. A release command is required to continue.

If job stream is assigned to some device other than the console typewriter, the $JOB statement is required following an ABORT condition; all other statements are accepted and ignored until a $JOB is received.

Job stream can be reassigned to the console teletypewriter at any time via the operator communications statement (JC). This does, however, abort the execution of the current background job, as well as slow down the execution of other jobs by holding background operation to the operator console.

6-2. 1 $JOB Statement

This statement transfers the job name to the Background Information Area, resets all background flags, options, and lines and sets logical device numbers to their default assignments (as specified during SYSGEN). Additionally, the job name is printed on the console teletype. The format is as follows.

$JOB name

where: name consists of zero to six ANSCII characters.

On spooled systems, additional parameters may be used to specify other than default parameters. The general form of the spooled job card is

$JOB name Unnn Pnnn Dnn Lnnn Innn Tnn Mnnn

where: name consists of one to six ANSCII characters.

Unnn is the user number and is valid and required only in accounting versions of DMS (see Section VII).

Pnnn is the priority; nnn must be between 1 and 254 and must conform to the established input terminal priority (see Section VIII). The priority value specifies the relative priority in the background spool input queue. It does not effect the priority of background execution, which is fixed at 255.

Tnn is the time limit in seconds for the execution of this job (accounting systems only). If not used, the system default time limit is used.

6-2.2 $EOJ Statement

This statement signifies the end of the current job and must be present to terminate the background job. The message "EOJ" is output on the console teletype. In accounting systems, the CPU execution time of the job is printed on the assigned List Output File. The format is:

```
$EOJ
```

6-2.3 $ASSIGN Statement

This statement causes the specified Logical File Number (LFN) to be assigned to a Physical Device Number (PDN), a Disc File Name (DFN), or a Spooled Device number (SDN). Thereafter, when any background program executes an I/O request with the specified logical file number, DMS will utilize the device or disc file that was assigned to the logical file number with this statement. The format is as follows:

(NOTE: An equal (=) sign or comma (,) may be used interchangeably between LFN and PDN, DFN or SDN):

```
$ASSIGN LFN, PDN, LFN, PDN, ...
$ASSIGN LFN=PDN, LFN=PDN, ...
```

or

```
$ASSIGN LFN, DFN, LFN, DFN, ...
$ASSIGN LFN=DFN, LFN=DFN,...
```

or

```
$ASSIGN LFN, #SDN, LFN, #SDN(file size)
$ASSIGN LFN=#SDN, LFN=#SDN(file size)
```

Invalid values for the LFN, PDN, or SDN will result in a "JCL error" and termination of the Job. (See Section 4-13, ASSIGN Service for a definition of valid values for LFN, PDN & SDN.)

The latter format is used to assign an output Logical File to a Spooled File. A spool file will be dynamically created and will be spooled out to the specified device when the job is completed. If the optional "(file size)" field is present, this size in sectors will be used as the size of the file. Otherwise the default size will be used. The specified device number must be a spooled terminal device. Some examples of assignments are given below.

1) $ASSIGN 5,10

   This assignment specifies that Logical File 5 (Binary Output) is to be assigned to Physical Device 10_8.

2) $ASSIGN 6,0

   This assignment specifies that all output to Logical File 6 is to be ignored

3) $ASSIGN 16,W1

   This statement specifies that Logical File 16_8 is to be assigned to disc file "W1".

4) $ASSIGN 3,#10(50)

   This statement specifies that output to Logical File 3 is to be spooled out to Physical Device 10_8 using a disc file of 50 sectors.

6-2.4 $DATE Statement

In non-accounting versions of DMS, this statement transfers the specified nine ANSCII characters or less to the background information area date storage.

The format is as follows.

$DATE xxxxxxxxx

In accounting systems, the statement is ignored as the system date is established when DMS is loaded from disc.

6-2.5 $LINES Statement

This statement alters the default number of lines per page for listed output. The format is a follows.

$LINES n

The default is returned at the beginning of the following job.

6-2.6 $OPTIONS Statement

This statement sets the specified bits of the Options word; i.e., places 1's in the selected bit positions (n is a decimal integer less than or equal to 23). The leading dot (.), if present, enters Zeros (0) in all bits of the word prior to setting the specified bits. The Options word is acquired by the service INFO. The format is as follows.

$OPTIONS .1,2,3,4,...,n

The usage of OPTIONS settings by standard DMS routines and processors is given in Appendix D.

6-2.7 $FLAGS Statement

This command functions in the same manner as the $OPTIONS statement, except that a different memory location is used. The format is as follows.

$FLAGS .0,1,2,...,n

The FLAGS values used in assembling DMS modules and processors are listed in Appendix D.

6-2.8 $INCLUDE Statement

This statement is used to copy program elements in link module form as produced by the Fortran Compiler or Macro Assembler, from one device to another. The basic format is as follows.

$INCLUDE (Filename, password)

If no file name is specified, input is from Logical File 04 (Binary Input). If a file name is specified, this file is rewound and used as the input source. JOBCTL copies link modules from the input file to Logical File 05 (Binary Output), which is normally assigned to the Link Ready disc file ("LR"), until an end-of-file is encountered. At that time an end-of-file is written to Logical File 05 and this file is backspaced a record in anticipation of another $INCLUDE statement, an assembly or a compilation.

6-2.9 $CATALOG Statement

This statement causes the Link Cataloger to be loaded and executed. The Link Cataloger accepts specification cards that follow the $CATALOG, builds a load module from the routines on the Link Ready file (together with required library subroutines), and outputs the load module to a disc file. The resultant load module is not executed as a direct result of this statement. The format is as follows.

$CATALOG

Refer to paragraph 6-4 for additional information.

6-2.10 $CATGO Statement

This statement sets the GOFLAG, then executes the Link Cataloger which catalogs onto the system "GO" file for immediate execution. The format is as follows.

$CATGO

Refer to paragraph 6-4 for additional information.

6-2.11 $LOADGO Statement

This statement causes the specified background program to be loaded and executed in the DMS background. The specified program must have been previously cataloged via $CATALOG. The format is as follows.

$LOADGO programname

When initiated, the A register is loaded with the address of the background job control buffer, which contains the image which caused the program to be started, allowing user programs to interograte the line for options, parameters, etc.

6-2.12 $HOLD Statement

This statement causes the first 15 characters of the specified text to be output to the operator communication device and the background enters a wait condition until the operator intervenes (refer to Section IX, Paragraph 9-1). The format is as follows.

$HOLD text

6-2.13 $TAPEOP Statement

This statement is used to select the tape transport mode, density, and characters per word specification for a specific tape transport for the duration of the background job, or until another $TAPEOP is encountered. Execution of this statement will not effect any foreground magnetic tape operation (see paragraph 5-2.3). The format is as follows.

$TAPEOP transport, mode, density, cpw.

where the ordering of the parameters is insignificant and transport is of the form T0, T1, T2, or T7 specifying the transport number;

mode is one of "BINARY" or "ASCII" for no conversion or "BCD" for IBM

compatible BCD conversion (7-track only) or "EBCDIC" for ebcdic conversion (9-track only);

density is of the form { 200 / 556 / 800 / 1600 } (BPI) where "BPI" is optional and the number represents tape density;

Cpw is of the form { 1 / 2 / 3 / 4 } (cpw) where "cpw" is optional and the value represents the number of characters per word. Note that for valid BCD or EBCDIC conversion, 3 characters per word must be specified. Characters will be right justified in the user's buffer, with unused portions filled with zeroes; the absence of any parameter specifies that the system default will be used. For example, the statement

$TAPEOP T0,1600,2CPW

specifies that transport 0 is to be used for default (normally binary) conversion, but use 2 characters per word at a density of 1600 BPI.

6-2.14 $ADUMP Statement

This statement is used to produce a memory dump of a portion of the background memory area when a background program aborts. The format is as follows.

$ADUMP a,b

where a is the initial relative memory location (octal) at which the dump is to start, and

where b is the ending relative memory location which defines the end of the area to be dumped.

Both memory addresses are relative to the start of background. If either or both are absent, the start or end of background, respectively, are used as the bounds values for the dump. Whenever a background processor aborts with this statement active, the dump is written to logical file 6. If a processor exit normally, the $ADUMP request is cancelled. Thus, this statement must be used immediately prior to the execution of each processor which may cause an abort.

6-2.15 $OPEN Statement

This statement opens logical file xx (octal) using the specified password yyyyyy. The format is as follows.

$OPEN xx,yyyyyy

The password field is required only if the file has a password.

6-2.16 $CLOSE Statement

This statement closes logical file xx (octal). The format is as follows.

$CLOSE xx

6-2.17 $REW Statement

This statement rewinds logical file xx (octal). The format is as follows.

$REW xx

6-2.18 $BSF Statement

This statement backspaces one file on logical file xx (octal). The format is as follows.

$BSF xx

6-2.19 $ADF Statement

This statement advances one file on logical file xx (octal). The format is as follows.

$ADF xx

6-2.20 $RPF Statement

This statement repositions the current file on logical file xx (octal). The format is as follows.

$RPF xx

6-2.21 $WEF Statement

This statement writes an End-of-File on logical file xx (octal). The format is as follows.

$WEF xx

6-2.22 $XXYY Statement

This statement performs function yy (octal) on logical file xx (octal). The format is as follows.

$XXYY xxyy

A Table of I/O function codes is given in Section V, Table 5-3.

6-2.23 $EDITLF Statement

This statement loads and executes the Library File Editor. The format is as follows.

$EDITLF

For additional information, refer to paragraph 15-3.

6-2.24 $FILEMA Statement

This statement loads and executes the File Manager. The format is as follows.

$FILEMA

For additional information, refer to paragraph 6-3.

6-2.25 $NAME Statement

This statement is equivalent to a $LOADGO name statement. The format is as follows.

$NAME

Refer to paragraph 6-2.11. For example, the statement $FORTRAN is equivalent to $LOADGO FORTRAN.

6-2.26 $ Comments Statement

This statement causes no action to be performed. It is used only for introducing comments onto the List Output File. The format is:

$ comments

where the blank character in column 2 is required.

6-3 FILE MANAGER

The File Manager performs functions that are associated with the utilization of the DMS file system. The user should be familiar with the material discussed in Section III of this manual prior to using this program. The File Manager is initiated via the following job control statement:

$FILEMA

Once loaded, the File Manager reads directive statements from the Job Stream Logical File. The general format for these commands consists of an alphabetic function starting in the first character position, followed by multiple parameters, as shown below.

FUNCTION param1,param2,param3,...

Parameters are separated by a single blank or a comma. A double comma is used to indicate that a parameter is missing, and not needed. Two consecutive blanks may also be used in place of a double comma to indicate a missing parameter. The format sequence "comma, blank, comma" is functionally the same as three successive commas.

The format of the Master Disc Directory (MDD) may be found in Appendix C. The SAVE/RESTORE formats may be found in section 6-3.29.

The File Manager will output to LFN 6 pertinent data for commands which may reference a group of file names. The following information may be provided for each file, depending on the command:

1. Data and Time
2. File name
3. Pack number
4. First sector number
5. Last sector number
6. File size in sectors
7. Type
8. User access
9. Memory requirements
10. Absolute load origin
11. User number
12. Password

A new title will be output on every page. The page size may be varied with the job control $LINES command. To obtain a list with only one title line, a very large value on the lines per page job control statement should be used.

The following is a summary of the functions available within the File Manager. Additional information and the specific formats are discussed in the following paragraphs.

1. CREATE-	Generates a file
2. ESTAB-	Generates a file from an LFN
3. DELETE-	Removes a file
4. RENAME-	Change the file name only
5. DMAP-	Disc directory of all files
6. DMAPS-	Disc directory (sequentially)
7. DUMP-	Generates an absolute load module
8. DUMPBF-	Generates an absolute load module (no parameter record)
9. SAVE-	Produce a backup of a file
10. SAVEP-	Produce a backup of a file (no protection retained)
11. SAVPAK-	Produces a backup of all files on a pack
12. SAVUSR-	Produces a backup of all files by a user number
13. LIST-	Verifies and list all files on a 'save' tape.
14. RESTORE	Recreates a file previously 'saved'.
15. RESPAK-	Recreates all files for a pack
16. RESUSR-	Recreates all files for a user number
17. SAVMDD-	Saves MDD entries for pack during sysgens.
18. REPSYS-	Rewrites the resident DMS on the disc.
19. CLEAR-	Identifies a new pack to DMS
20. ZEROPK-	Performs a "clear" and then zero's entire pack.
21. READPK-	Reads a pack and outputs error messages upon read errors.
22. FLAGPK-	Flags unreadable sectors of a pack with "permanent" files.
23. ADVANCE-	Position to next header record (no error checking)
24. REW-	Rewinds specified LFN
25. WEF-	Write an end-of-file on specified LFN
26. ADF-	Advance one file mark on specified LFN
27. BSF-	Backspace one file mark on specified LFN
28. HOLD-	Suspends background with a message.
29. EXIT-	Returns to the system.

6-3.1 CREATE Statement

This statement directs the File Manager to allocate sequential disc space and create an entry in the Master Disc Directory. The file name, password, and in accounting systems, the user-number combinations must be unique within the directory as discussed in Section III. Sufficient contiguous disc space must be available for the file to be created. In accounting versions of DMS, the current background user number is stored with the file entry as the creator. The format for this statement is shown below.

CREATE name,password,pack#,size,type,read,write,delete,sector

name - consists of one to six characters, of which the first must be alphabetic.

password - consists of one to four characters, of which the first must be alphabetic. A zero should be used to indicate that the file is "public" and no password is required. Once a file has been created with a non-zero password, any program desiring to access the file must supply a matching password when the file is opened.

pack# - consists of one to three numeric decimal digits specifying the pack ID number of the disc pack on which the file is to reside. The main disc on which the resident system resides is considered to be pack one. If the requested pack is not currently mounted, the operator will be informed.

size - consists of a decimal number specifying the desired file size in sectors.

type - consists of a single numeric character optionally preceded by "B" or "C" or both defining the file type as follows.

Value	Type of File
1	User data or work file.
4	Object Library file.
5	Source Library file.
6	System work file.
Bx	A "B" preceding any of the above specifies that the file is to be typed as a "blocked file".
Cx or CBx	A "C" preceding any of the above combinations specifies that the Master Disc Directory for the file will be stored in the core resident directory buffer.

read - write delete - An "R", "W", or "D", respectively, as additional parameters, valid only in accounting systems, specifies that the associated access bit is to be set to allow users other than the creator to perform the specific functions on the file. All, none, or any combination of these characters may be present, each as distinct parameters separated by commas.

Sector - This optional parameter is used to specify a lower bound when allocating disc space for a file, i.e., the file will not be created if there is not room above this location.

6-3.2 ESTABLISH Statement

The ESTABLISH statement is used to build a permanent file from the specified work file.

Prior to establishing a named file, the temporary work file must control recorded data. This recorded data can be placed on the work file in many ways.

For example, a link module can be placed on the work file by assembly with the binary output assigned to the work file. Also a module can be transferred from a selected binary input device to the work file using the job control service ($INCLUDE).

ESTABLISH assumes the current record address (CRA) of the specified work file to be positioned at the end of the recorded data, and uses the (CRA) to determine the size of the file to be built. A call to the DMS system service ($SYS) is made to allocate the required disc space, then data is transferred from the work file and copied into successive sectors of the named disc file. After all records are copied, the procedure is terminated by writing an end-of-file sector.

Statement formats are defined as follows.

ESTAB, XX,name,password,pack#,size,type,read,write,delete,sector

XX - Decimal (or octal if preceded by apostrophe ('')), Logical File code assigned to the work file which is to provide the data to be established.

name - One-to-six character identifier of the file to be established, of which the first must be alphabetic.

size - should be entered as zero because it is computed from the CRA.

password, pack#, type, read, write, delete - should be entered as defined for the create statement, with the exception that the absence or presence of the blocking specifications "B" is ignored in the file type designation, and the new file is automatically given the same blocked/unblocked status as the work file.

Sector - This optional parameter is used to specify a lower bound when allocating disc space for a file, i. e., the file will not be created if there is not room above this location.

6-3. 3 DELETE Statement

The DELETE statement removes a specified file name from the directory and releases its allocated disc space. The format of this statement is as follows:

DELETE name,password

where "name" and "password" are as defined for the CREATE statement.

6-3. 4 RENAME Statement

This statement permits the user to change the name, password, file type or, in accounting systems, the protection status of any specified disc file. A monitor service is employed which finds and loads the Master Disc Directory entry, validates the parameters, replaces effected information, and restores the directory entry. The format is as follows:

RENAME oldname,newname,oldpassword,newpassword,type,read,write,delete

oldname - required to identify existing file.

newname - optional new file name. If the field is omitted, the name remains unchanged.

oldpassword - required to identify existing file if it has a password. The parameter may be omitted if the file has no password.

newpassword - if a password is specified here, it will be the new file password. If a zero is entered in this field, the current password is removed and the file will not have a password. If the field is omitted, by entering consecutive commas, the current password remains unchanged.

type - A "C" or "NC" may be entered here to indicate that the Core Resident Directory entry status is to be added ("C") or removed ("NC") from the type designation. No other file type changes are permitted.

read
write
delete - These parameters are valid only in accounting systems and are used to indicate the new file access status. An "R", "W", or "D" correspondingly, or any combination of them, is used to indicate the respective file access. If no parameters follow the type field, then the protection status remains unchanged. The file may be changed to totally private (no access bits sets) by entering a single parameter "P" to indicate private.

6-3.5 DMAP Statement

The Disc Map Statement directs the File Manager to produce a file system disc allocation map on the list output logical device. The file names are sorted by alphanumeric sequence prior to output file.

i)	DMAP n	where n is a disc pack number and causes only the disc pack named to be mapped.
ii)	DMAP Unnn	where nnn is a user- number and only programs with the given user-number are mapped.
iii)	DMAP name	where name is a file name and only programs with the specified name are mapped.
iv)	DMAP	with no parameter specifies that the entire directory is listed.

6-3.6 DMAPS Statement

This statement performs identically to the DMAP statement defined above, except that the map is ordered by sector number, rather than alphabetically.

6-3.7 DUMP and DUMPBF Statements

These statements cause a specified cataloged program file to be loaded into the background memory area and dumped in an absolute load format on the assigned binary output file. Address relocation is performed according to specified parameters.

The DUMP output consists of two binary records. Record one is a 6-word parameter record containing the program name, low, high, start and checksum. Record two consists of a n word data record, program low through program high.

The DUMPBF output consists of only the n word absolute data record, producing a bootstrap format record. (The user should take care in coding a program for paper tape or card bootstrap loading, to insure that there are no zero instruction or data words. The bootstrap loaders terminate input upon detection of a zero word.)

Statement formats are defined as follows.

DUMP, name, relative program origin, absolute dump bias
DUMPBF, name, relative program origin, absolute dump bias

where: name is a one-to-six character identifier, of which the first must be alphabetic.

relative program origin consists of a numeric value and is the relative program origin specified in the source program (0 if not specified).

where: absolute dump bias consists of a numeric value specifying the absolute memory address at which the program is intended for execution.

Address relocation is accomplished by adding the dump bias minus the program origin.

6-3.8 SAVE Statement

The SAVE statement directs the File Manager to output the entire contents of a specified disc file to the binary output logical device (Logical File number 5). This data consists of a header record that preserves the file definition (name, password, type, size, etc.) and a checksum for the record. Each data record consists of one sector of disc data plus a checksum for the record producing records of 113 words. The header record contains a sector count for the file which defines how many data records follow.

The specified file is not deleted by the process of saving it on the binary output device. The format of the statement is as follows:

SAVE name, password, Unnn, newname, newpass

name - file name to be saved.

password - file password is required if the file has a password.

Unnn - is the user-number and is valid only in accounting systems. This number is required only if the file is totally private (has no access bits specified) to select the proper copy of the file.

newname- is an optional parameter which becomes the file name stored in the Save Module. This allows a file to be saved under a different name than on the current system. If this parameter is not present, the original file name is used.

newpass- is an optional parameter which becomes the password for the file stored in the Save Module. This allows a file to be saved under a different password than on the current system. If this parameter is not present, the original password is used.

6-3.9 SAVEP Statement

The SAVEP statement performs identically to the SAVE statement discussed above in non-accounting systems. In accounting versions of DMS, the SAVEP statement performs the same function as the SAVE statement, except that all file protection bits and the user-number are cleared to zeros. This allows the resulting SAVE module to be reloaded on a non-accounting system. The format is as follows:

SAVEP name, password,user-number,newname,newpass

6-3.10 SAVPAK Statement

The SAVPAK statement enables all of the disc files on the specified disc pack to be saved on the binary output logical device. The statement is equivalent to a string of SAVE statements for each disc file on the given disc pack. The files are saved in alphabetic order and the format is as for the SAVE statement. The name of each file that is saved is listed on the list output logical device. Upon completion of the entire group save, an end-of-file is written to binary output, and backspaced over. The format of the statement is:

SAVPAK n

where n is the disc pack number and the default n is pack one.

6-3.11 SAVUSR Statement

The SAVUSR statement saves all of a user's files on all packs to the binary output device. The files are saved in alphabetical order and in the same format as the SAVE statement. The name and other information is output to list output device as each file is saved. Upon completion of the entire group save, an end-of-file is written to binary output and backspaced over. This assures proper termination of the data. The format of the statement is:

SAVUSR Unnn

SAVUSR nnn

where nnn is the user number specification.

6-3.12 LIST Statement

The LIST statement is used to list and verify previously saved files. The binary input logical file is scanned and information pertaining to each saved file is listed on the list output device. If a data checksum error is detected, a warning message is output, and the remainder of the files data records will be skipped. If a checksum error is detected in the header record after three retries, a fatal message is output and the job is aborted. The format of the statement is defined as:

LIST

No parameters are required.

6-3.13 RESTORE Statement

The RESTORE statement is utilized to restore a disc file that has been previously saved and deleted. The binary input logical device (Logical File 04) is searched from the current position, bypassing other "SAVE" format modules, until the file with the specified name and password is located. The file definition information is then obtained from the header record, and the file is re-defined in the directory and the appropriate disc space is allocated. The data

is then copied onto the disc. It should be noted that when a file is restored, it is allocated the same quantity of disc space that it originally occupied, but not necessarily the same physical area. The format for this statement is as follows:

RESTORE name, password, pack#, sector, newname, newpass

name-
password
sector
pack# - defined as in CREATE statement
newname
newpass-

6-3.14 RESPAK Statement

The RESPAK statement enables all files previously save to be recreated on the disc. The binary input device starts from its current position and reads the next "SAVE" header record for file name and password. The file will be recreated automatically and data copied into the disc area. If the file name is already used, a message is output to the list output device and the file is skipped. This sequence repeats until an end-of-file is detected while attempting to read a header record. Pertinent information is output as each file is restored. The statement is defined as follows:

RESPAK n,m
where n is the pack number from which the files were saved.
m is the pack where the files will be restored to. If not specified, n is assumed.

6-3.15 RESUSR Statement

The RESUSR is similar to the RESPAK statement. Binary input device is scanned for all files saved with the specified user number. Upon finding a match, the file will be restored if not already present. The statement is defined as follows:

RESUSR Unnn, pack#
or
RESUSR nnn, pack#

where the nnn is the user number specification.

pack# is an optional parameter which defines the disc pack to which the files are to be restored. If the parameter is not present, then the pack number from which the SAVUSR was generated is used.

6-3.16 SAVMDD Statement

The SAVMDD statement saves the Master Disc Directory entries for all disc files on all packs other than the master pack(pack 1), onto the binary output logical device. The entries are saved in 112-word blocks plus a checksum word. The last block is filled with zeros following the last saved entry. The format of the statement is:

SAVMDD

and no parameters are required.

6-3.17 REPSYS Statement

The REPSYS statement is used to replace the resident DMS on disc without going through the original DMS generation procedure of initializing the entire disc and restoring the various DMS files.

During the REPSYS procedure, several holds occur. Each must be released with the OPCOM command /RB.

Logical files 4 and 5 are dynamically assigned during the copy process. Each read or write will be exactly 1 sector or 112 words.

the format of the command is defined as follows:

REPSYS ffffff, tttttt

ffffff is the file name containing the new DMS system catalogued as TYPE=SYSGEN. If the parameter is missing, the file name 'DMS' is assumed.

tttttt is an output file name or physical device number where the current DMS ('D. M. S.') will be copied to. If no parameter is given, W2 is assumed.

The generation of a second copy of the current DMS allows recovery if the new DMS does not function properly.

The following is an example of a REPSYS from a file 'NEWDMS'.

```
$JOB .....
.
.
.
(generation of SYSDAT)
.
.
.
(Inclusion of DMS modules)
.
.
.
(Inclusion of OPCOM modules)
.
.
.
$CATALOG
DNAME=NEWDMS,,R
TYPE=SYSGEN
BEGIN
.
.
.
(Cataloging of OPCOM)
.
.
.
```

```
$FILEMA
REPSYS    NEWDMS

          FILEMA: REPLACE DMS?          (OP message & hold)

          FILEMA: COPY TO W2            (OP message & hold)

          FILEMA: COPY FROM NEWDMS      (OP message & hold)

          FILEMA: DMS REPLACED          (OP message)

RENAME OPCOM, OPX, GORP, GORP
RENAME OPCOM1, OPX1, GORP, GORP
RENAME OPCOM2, OPX2, GORP, GORP
RENAME OPCOM3, OPX3, GORP, GORP
RENAME OPCOM4, OPX4, GORP, GORP
RENAME OPCOM5, OPX5, GORP, GORP
RENAME OPC, OPCOM, GORP, GORP
RENAME OPC1, OPCOM1, GORP, GORP
RENAME OPC2, OPCOM2, GORP, GORP
RENAME OPC3, OPCOM3, GORP, GORP
RENAME OPC4, OPCOM4, GORP, GORP
RENAME OPC5, OPCOM5, GORP, GORP
$EOJ
```

Error messages associated with the REPSYS command are listed in Table 6-1.

6-3.18 CLEAR Statement

The CLEAR statement is used to initialize a new disc pack on multi-drive DMS configurations. It has no meaning and cannot be used single disc systems. The format of the CLEAR statement is as follows:

CLEAR pack#, disc#

where: pack# is the pack ID number (2-255) of the pack being initialized and disc# is the disc number (2-64) on which the pack is now mounted.

The message

```
FILEMA    CLEAR
PK# xxx DSC #yyy?
```

is output to the operator to allow verification of the input parameters. If valid, a release background command must be given. If invalid, background should be aborted. When released, the specified pack in initialized and all MDD entries giving reference to that pack are removed. When the initialization is completed, the message

```
PACK CLEARED
```

is output.

6-3.19 ZEROPK Statement

The ZEROPK statement is used to initialize a new disc pack on multi-drive DMS configurations. It has no meaning and cannot be used on single disc systems. The ZEROPK statement first performs a CLEAR (see section 6-3.14) and then writes zeros in every sector of the pack from sector 4 to the end. The number of sectors zeroed per write operation is dependent upon background size, the disc being zeroed fastest when background size is large. The format of the ZEROPK statement is as follows:

ZEROPK pack#, disc#

where pack# and disc# are defined as in section 6-3.14 (CLEAR statement)

the message:

FILEMA ZERO
PK# xxx DSC#yyy?

is output to the operator to allow verification of the input parameters. If valid, a release background command must be given. If invalid, background should be aborted. When the pack is completely zeroed, the following message is output:

PACK ZEROED

6-3.20 READPK Statement

The READPK statement is used to read a complete disc pack to find any sectors which are unreadable. When a sector is found which cannot be read, information is output describing the read attempt, the number of words received and the sector which caused the error. The reads then continue with the sector following the bad one. The format of the READPK statement is as follows:

READPK pack#, wordcount

where: pack# is the pack ID number (2-255) of the pack to be read.

and: wordcount is the number of words to be read per read operation. This parameter is optional and if not supplied, the wordcount will be determined by the size of background.

6-3.21 FLAGPK Statement

The FLAGPK statement is used to create permanent files over any sectors on a disc pack which cannot be read. The FLAGPK statement functions identically to the READPK statement (section 6-3.14B), but it automatically creates permanent type files over the sectors determined to be bad. The file names are dynamically generated and are of the form Z#NNNN with a password of B.S.. The format of the FLAGPK statement is as follows:

FLAGPK pack#, wordcount

where pack# and wordcount are defined as in section 6-3.14B (READPK statement).

If a file cannot be created over a bad sector (if another file already included that sector for example), a message will be output to that effect. If a file is successfully created the file name created will be output.

6-3.22 ADVANCE Statement

The ADVANCE statement is used to position the binary input logical file in front of the next SAVE header record. No data error checking is performed. The statament is defined as follows:

ADVANCE

No parameters are required.

6-3.23 REW Statement

The REW statement perform a rewind function on the specified logical file number. The statement is defined as follows:

REW n

where n is the logical file specifications.

6-3.24 WEF Statement

The WEF statement will write an end-of-file on the specified logical file number. The statement is defined as follows:

WEF n

where n is the logical file number specification.

6-3.25 ADF Statement

The ADF statement will advance the specified logical file number to after the next end-of-file. The statement is defined as:

ADF n

where n is the logical file number specification

6-3.26 BSF Statement

The BSF statement will backspace the specified logical file number to before the previous end-of-file. The statement is defined as follows:

BSF n

where n is the logical file number specification.

6-3.27 HOLD Statement

The HOLD statement outputs a 15 character message to the OPCOM terminal and then suspends background. The statement is defined as follows:

HOLD MESSAGE TEXT!!!!

where "MESSAGE TEXT!!!" is the message to be output. Note that the format assumes two blank characters between HOLD and the message text. This command requires an OPCOM "/RB" to continue.

6-3.28 EXIT Statement

The execution of the file Manager is terminated by the statement:

EXIT

6-3.29 SAVE/RESTORE Formats

The following is the SAVE/RESTORE format:

SAVE/RESTORE HEADER

WORD									
0	FIRST THREE ASCII CHAR OF FILE NAME								
1	SECOND THREE ASCII CHAR OF FILE NAME								
2	23 PRIVT	22 PRO	21 WPRO	20 DPRO		19-0 RECORD WORD COUNT ZERO = 113			
3	FILE SIZE IN SECTORS								
4	23 BLOCK	22 SPOOL	21 CORE	20 PERM	19-16 TYPE	15-0 MEMORY REQUIREMENTS			
5	23-16 PACK NO					15-0 ABSOLUTE PROGRAM ORIGIN			
6	FIRST THREE ASCII CHAR OF PASSWORD								
7	23-16 4TH CHAR OF PASS					15-0 USER NUMBER			
8	CHECKSUM = -(SUMMATION OF WORDS 0-7)								

DATA RECORD LAYOUT

WORDS	
0-111	DATA FOR SECTOR ONE
112-223	DATA FOR SECTOR TWO
224-335	DATA FOR SECTOR THREE
	* * *
784-895	DATA FOR SECTOR EIGHT
896	23 22 21 20 19 18 17 16 / SEC# 1 2 3 4 5 6 7 8 / BIT SET = EOF SECTOR — 15-0 / NOT USED
897	CHECKSUM = -(SUMMATION OF WORDS 0-896

6-3.30 ERROR MESSAGES

Error checking is performed on file system functions to safeguard the integrity of the disc. When an error is detected, an error message is printed on the program's list output device and the current job is aborted. The error messages and their complete definitions are listed in Table 6-1.

Table 6-1
Error Message Definitions

Message	Error Condition
CORE NOT AVAIL	A checksum error has been encountered from the binary input logical device. *
DISC# ERROR	A disc number was specified that was above the maximum for this configuration.
DISC READ ERROR	A disc failure was detected following a read operation involving the Master Disc Directory or a space allocation map.
DISC WRITE ERROR	A disc failure was detected following a write operation involving the Master Disc Directory or a space allocation map.
FILE NAME ERROR	The file name from a RESTORE statement does not match the one contained in a header record. This message is also used when any file name specification does not start with an alphabetic character.
FILE NAME USED	The file name specified for a CREATE or RESTORE function is already in the directory. **

Table 6-1 (Cont)
Error Message Definitions

Message	Error Condition
FILE NON-EXIST	The user attempted to delete a file which is not defined in the Master Disc Directory.**
FILE NOT THERE	The user attempted to save a file which was not present in the Master Disc Director.**
FILE SIZE ERROR	A zero-negative or non-numeric file size parameter was specified.
FILE TYPE ERROR	A file-type parameter was specified incorrectly on a CREATE statement.
INVAL FUNCTION	A directive statement was encountered with a function other than CREATE, DELETE, SAVE, RESTORE, DMAP, EXIT, CLEAR, REPSYS, SAVEP, SAVPAK, SAVMDD or DMAPS.
PACK # ERROR	The pack number requested was not between 1 and 255 inclusive. Or was not specified when more than one drive is available.
ABS BIAS ERROR	An illegal (negative) value was detected in a Dump statement.
DMS NOT ON FILE	The file name containing the new DMS for the REPSYS command is not available.
INVALID PDN	The output device specification for a REPSYS does not exist.
DMS REPLACE	The REPSYS command has successfully been completed.
FATAL CKSUM ERROR	Three unsuccessful attempts have been made to read a SAVE/RESTORE record.
ALLOC SEC # ERR	An invalid 'start of allocation' sector number has been detected in the Create statement.
NEW DMS INVALID	The file containing the new DMS is incorrect during the REPSYS command.
REPLACE DMS?	Message to signify that a request is being made to over write the disc resident DMS.*
COPY TO:.....	The back up copy of the disc resident DMS is about to begin.*
COPY FROM:.....	The new DMS is ready to be copied onto the current disc resident DMS.*
PACK ZEROED	The pack has been successfully cleared
INVALID LFN	Only LFN's 4 & 5 are allowed on REW, ADV, BSF, WEF, commands.
ERROR IN USER#	An invalid user number has been detected.

Table 6-1 (Cont)
Error Messages Definitions

Message	Error Condition
EOT DETECTED	The end-of-output-area has been detected during a SAVE to a disc file.
MDD SECTOR FULL	The Master Disc Directory sector to which a specified file has been assigned by the CREATE routine is already full. The user must either change file name such that it will be assigned to another sector, or delete files such that the required sector is no longer full.
FILE IN USE	An attempt was made to delete a file which was in use by another program.
PASSWORD ERROR	The user attempted to save or delete a file with an incorrect password specified.
FORMAT ERROR	A request has been made supplying an improper sequence of parameters.
MEM REQ ERROR	The size of a program to be dumped was found to be greater than the allocated background area.
REL ORG ERROR	An illegal (negative) value was detected in a DUMP command.
LF ASSIGN ERROR	An assignment to a device rather than a file was made prior to an Establish command.
DMS TOO LARGE	The DMS being replaced with a REPSYS command is too large to be written in the disc space where DMS currently resides.

* This command requires an OPCOM" /RB" to continue.
** This command generates a non-fatal error.

6-4 LINK CATALOGER

The DMS Link Cataloger is a restricted background processor that is supplied with DMS. Its purpose is to link separately assembled and compiled program modules, subprogram modules, and library routines. During the linking process, common addresses are assigned, externals and external equivalence requests are satisfied, and stringing is performed. The resulting output of the linking processes is a complete program module, that is stored in a format that enables the program to be quickly relocated in memory by the DMS loader prior to program execution.

The Link Cataloger provides several methods of generating and executing a program that is too large for available core. One of these methods is through the use of CHAIN segments. A program that can be broken into several independent parts may use this method. Each part is cataloged as a distinct program, called a CHAIN segment. All data that is transferred between CHAIN segments is passed through logical files or through common. The Link Cataloger has provisions to enable all of these CHAIN segments to share the same common area. Each program unit (or CHAIN segment) is executed separately, and when complete, it passes control to the next CHAIN segment by use of the System Service CHAIN (see Paragraph 4-7) or by use of the FORTRAN CALL CHAIN.

Another method of conserving core is through the use of overlays. This method is often called segmenting, but should not be confused with CHAIN segments. An overlay program is made up of several overlay segments and a main segment which is always in core and is known as the root. Each overlay segment contains one or more subprograms. These segments are loaded into core only when a reference is made to one of the subprograms contained within the segment. Thus, one or more of these segments may be loaded into the same area of core. The only restriction being that the two segments may not be needed at the same time (i.e., they may not call each other).

6-4.1 Overlay Type Programs

An overlay type program consists of a main segment (called a root) which is permanently core resident during execution and one or more additional segments which are loaded into core by a FLIPER routine when they are needed. The FLIPER routine is part of the root. This routine is automatically included into an overlay program.

The structure of an overlay program is determined by the layout of overlay areas within the total program area. The simplest meaningful example consists of a main program which calls two subprograms. This example is indicated by the following diagram.

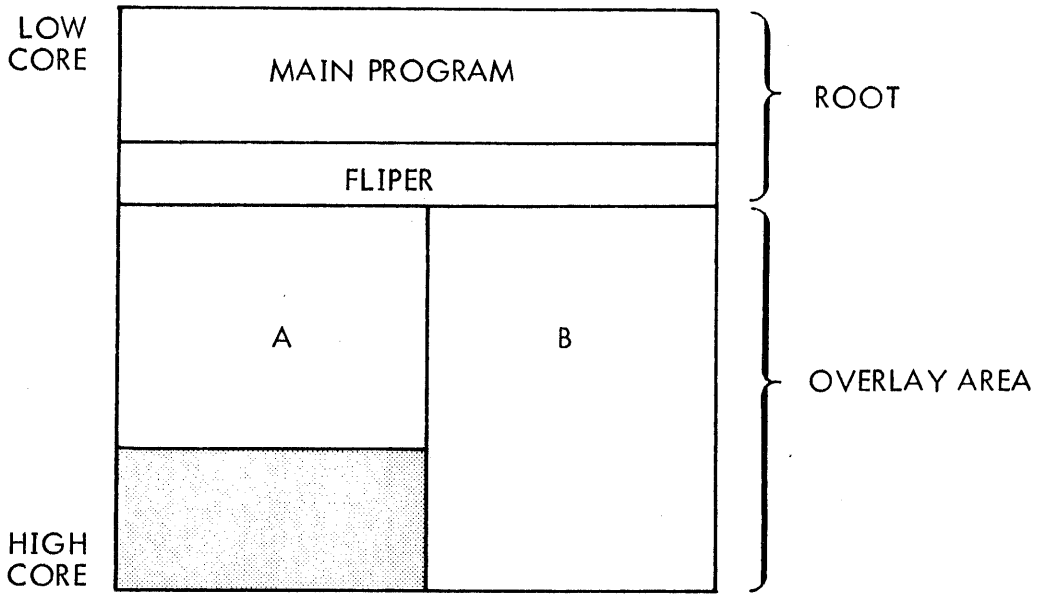


Figure 6-1. Overlay Program

In this example, the main program contains calls to each of the subprograms A and B. The root of this program consists of the main program and the FLIPER routine. There are two segments which may be called A and B. Segment A consists of subprogram A and segment B consists of subprogram B. Note that segments A and B share the same area of core. This area is known as an overlay area. In this example it is assumed that segment A does not fill up the entire overlay area. It is normally true that all segments that share an overlay area are not the same length. Because of this, the length of an overlay area is determined by the length of the longest segment that is to be loaded into the area.

The main program in this example defines the base address for the overlay area. Because of this, all segments in the overlay area are known as children of the main program. And correspondingly, the main program is known as a parent of each of its children.

A program may consist of more than one overlay area. When more than one overlay area exists, each area is identified by a phase number. Phase zero is that overlay area closest to the parent segment. Phase one is the next overlay area; it resides just above phase zero in core. Phase two is the next area, etc. The following diagram is an example of a three phase overlay program.

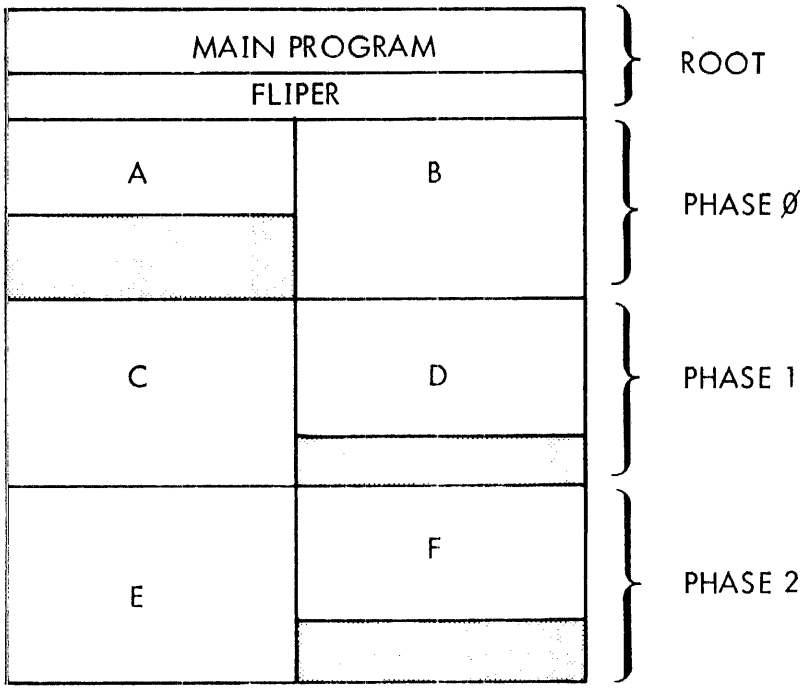


Figure 6-2. Multi-Phase Overlay Program

In this example, all of the segments A, B, C, D, E, and F are referred to as children of the root. Because of this multiphase structure, it is possible for more than one segment to be present in core at one time. For example, the segments A, D and E may all be present at one time. Therefore, it is also possible for each of these segments to reference another segment not on the same phase level. It is not possible for segments on the same phase level to reference each other. Care should be taken that this is not attempted. If such a reference is done indirectly, (i.e., A calls E, which calls B) the results are undefined.

In addition to multiphase overlays, it is also possible for a single segment to be broken up into several overlays. This is referred to as multilevel overlays. This is illustrated by the following example.

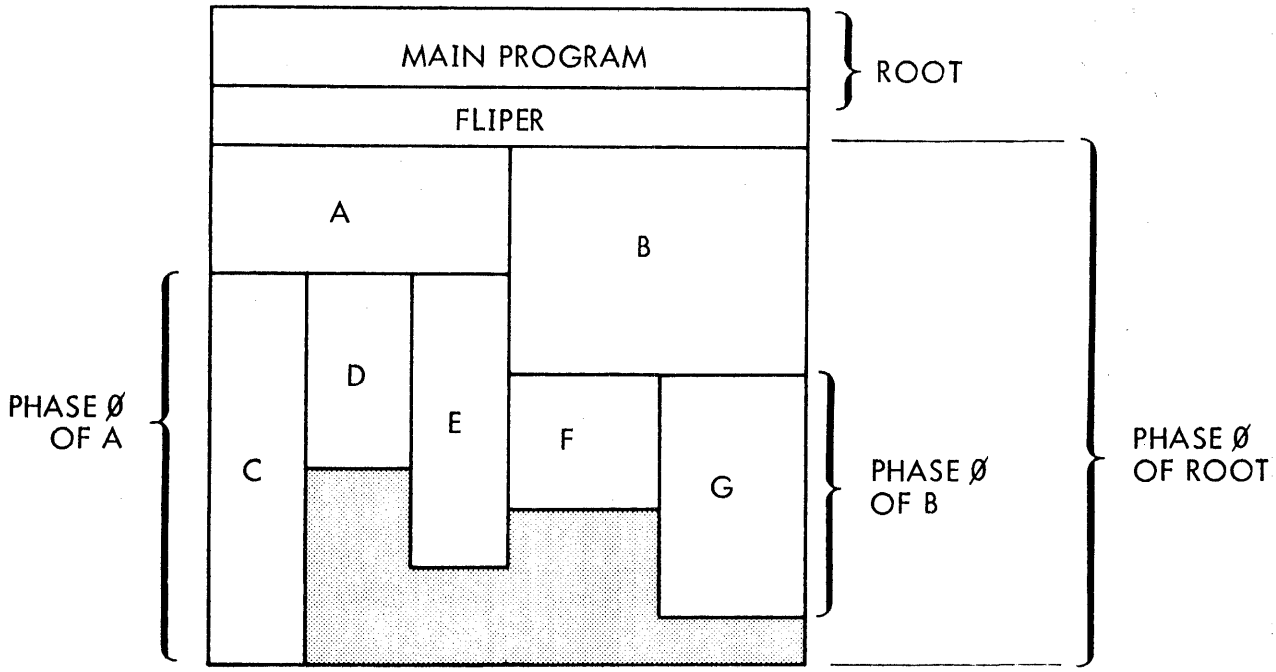


Figure 6-3. Multi-Level Overlay Program

In this example, there is one overlay area associated with the root, this overlay area extends from the initial location of segment A to the final location of segment C. This means that all of the segments (A, B, C, D, E, F, G, and H) are children of the root, and they are all in phase zero of the root. There is also one overlay area associated with segment A. This overlay area is phase zero of segment A and extends from the first location of segment C to the final location of segment C. Similarly, phase zero of segment B extends from the first location of segment G to the last location of segment G. Note, that the definition of phase zero of the root includes segments A and B and all of the phases of A and B. Segments C, D and E are children of both A (since they are in a phase of segment A) and the root (since they are in a phase of the root).

References between parents and children are allowed. References between two segments that are not related as parent and child are allowed only if the two segments are in different phases of the same parent segment. Thus in the above example, segments F and G may not reference each other, segments C, D and E may not reference each other, and none of the segments A, C, D, or E may reference or be referenced by any of the segments B, F or G.

These two structures, multi-phase and multi-level overlays, may be continued to any desired extent. They are limited only by available core. The following example is presented as a possible structure available through the Link Cataloger. It is presented without explanation only to demonstrate the possible complexity of structure that can be obtained.

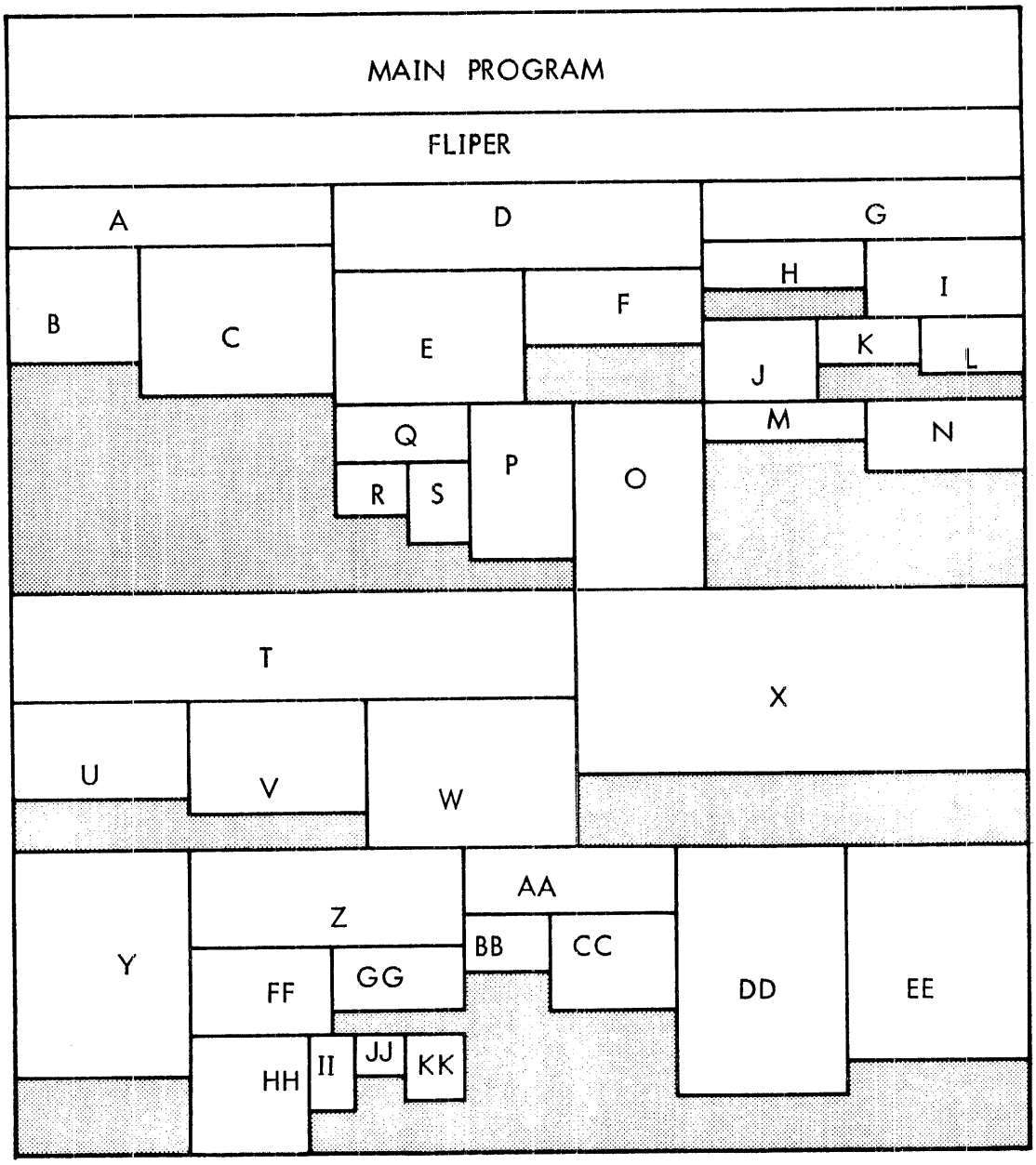


Figure 6-4. Multi-Phase, Multi-Level Overlay Program

6-4.2 Link Cataloger Control Statements

The Link Cataloger is activated in the background area by issuing one of the following Job Control Statements.

```
$CATALOG
$CATGO
```

When activated by the $CATALOG Job Control Statement, a series of input requests are made from the Job Control logical device for a set of control statements. Each control statement is output to the list output logical device unless both Job Control and List Output are assigned to physical device one. These control statements describe the various parameters to be associated with the program to be link cataloged and the condition under which the link process is to take place.

The various control statements, their format, and meanings are discussed in the following paragraphs.

TYPE

The TYPE control statement has the following format:

```
        BG
TYPE = FG, PRIV, SAUT, OA, OM, UA, UM, IOA, SEGMNT, LM, DICT, SYSGEN
       RFG STRICT NSAUT NOA NOM NUA NUM NIOA
       RENFG
```

The meanings of the various parameters are as follows.

a. BG — Specifies that the program to be cataloged is to be typed as a background program.

b. FG — Specifies that the program to be cataloged is to be typed as a foreground program.

c. RFG — Specifies that the program to be cataloged is to be typed as a resident foreground program.

d. RENFG — Specifies that the program to be cataloged is to be typed as a re-entrant foreground program.

e. PRIV — Specifies that the program is to be run in a privileged mode at execution time. (Bits 23 of P/MODE of PSA is set to one.) **

f. STRICT — Specifies that the program is to be run in a restricted mode at execution time. (Bits 23 of P/MODE of PSA is set to zero.)**

g. SAUT — Specifies that the program is to be executed with the SAU executive interrupt trap enabled. (Bits 22 of P/MODE in PSA is set to zero.)*

h. NSAUT — Specifies that the program is to be executed with theSAU executive interrupt trap disabled. (Bits 22 of P/MODE in PSA is set to one.)*

i. OA — Specifies that the program is to be aborted if an SAU overflow occurs. (Bit 18 of P/MODE in PSA is set to zero.)*

j. NOA — Specifies that the program is not to be aborted if an SAU overflow occurs. (Bit 18 of P/MODE in PSA is set to one.)*

k. OM — Specifies that a message is to be output to the operator communications device if an SAU overflow occurs. (Bit 19 of P/MODE in PSA is set to zero.)*

**Refer to Paragraph 4-18.2

l.	NOM	Specifies that no message is to be output if an SAU overflow occurs. (Bit 19 of P/MODE in PSA is set to one.)*
m.	UA	Specifies that the program is to be aborted if an SAU underflow occurs. (Bit 20 of P/MODE in PSA is set to zero.)*
n.	NUA	Specifies that the program is not to be aborted if an SAU underflow occurs. (Bit 20 of P/MODE in PSA is set to one.)*
o.	UM	Specifies that a message is to be output to the operator communications device if an SAU underflow occurs. (Bit 21 of P/MODE in PSA is set to zero.)*
p.	NUM	Specifies that no message is to be output if an SAU underflow occurs. (Bit 21 of P/MODE in PSA is set to one.)*
q.	IOA	Specifies that the program is to be aborted if an I/O error occurs. (Bit 17 of P/MODE in PSA is set to zero.)*
r.	NIOA	Specifies that the program is not to be aborted is an I/O error occurs. (Bit 17 of P/MODE in PSA is set to one.)*
s.	SEGMNT	Specifies that the program being cataloged is a program CHAIN segment.
t.	LM	Specifies that the program being cataloged is a "large" module" to be dumped relative to some other location than background low and therefore the check for crossing the map boundary is to be disabled.
u.	DICT	Specifies that the cataloged program is to be typed such that its Master Disc Directory entry is to be loaded in the core resident directory table. Refer to Paragraph 3-5. 3.
v.	SYSGEN	Specifies that the two header words normally output by the cataloging process are not to be produced. The resulting module contains the first program word as the first word on disc. This parameter is normally used only in the System Generation procedure to produce the DMS load module. Any other situation should not use this parameter unless a special circumstance warrants its use.
w.	NONACC	Specifies that the foreground program being cataloged is not to generate accounting records when executing. If the program is a background program, this specification is ignored. (Bit 16 of P/MODE in PSA is set to one.)

* Refer to Paragraph 4-18. 4

All of the preceding parameters are optional and may be specified in any order within the statement. If any of the parameters are omitted, then the default values which are established are as if the following TYPE control statement were issued.

TYPE = BG,STRICT,SAUT,OA,OM,UA,UM,AFER

The equal sign (=) in the format given above is optional any may be replaced with a blank.

Any number of TYPE control statements may be given. If any of the program type or mode parameters are respecified, either within the same statement or on separate statements, then the last encountered specification will be used.

NAME

The NAME control statement has the following format:

NAME = XXXXXX,p,R,W,D

The XXXXXX specification is a one to six characters program name that must begin with an alphabetic character and must not contain an equal sign (=), comma (,), or blank. Issuing a NAME control statement indicates that a permanent file is to be created with the name XXXXXX. If no NAME control statement is received, then the output of the Link Cataloger will reside on the Go File ('16). The p specification is an optional parameter that specifies the pack number on which the program is to be permanently cataloged.

If an "R", "W", or "D" is entered as an additional parameter, then the corresponding read, write, or delete access bits are entered in the file directory entry. These access bits are useful only in accounting systems. See Section III. More than one of these bits may be specified. If no access bits are specified, then the program can only be referenced by its creator.

NOTE

If the program is an overlay program, then both "R" and "W" access bits must be set in order for the program to be executed by other than its creator.

DNAME

The DNAME control statement is identical in format and function with the NAME statement with the following difference. Immediately before creating a permanent program file, the specified name is used to delete the previous version of the program. The previous version of the program is not deleted (and a new program is not created) if any cataloger errors occur during the cataloging process. If the previous program does not exist or if the program is not accessible by the current user, then the message, FILE NON-EXIST, will appear on the list-output device. In either case, the cataloger will attempt to create a permanent file. If the program file cannot be created due to a conflict with another user's program, then the message, FILE NAME USED, will appear on the list-output device and the job will abort. Otherwise, the catalog process will run to completion.

ASSIGN

The ASSIGN control statement has the following format:

ASSIGN xp=y,xp=y,xp=#y (s)

The x parameter is an octal logical device number which is assigned to file name y or physical device y, or spooled device y. In the case of spooled assignments, the file size may optionally be specified. (See Section 6-2. 3). The p specification following the logical device

number is optional and, if present, indicates that the assignment is to be permanent and therefore cannot be changed by program or operator action. A comma (,) may be substituted for the equal sign (=) in the above format.

The Link Cataloger accepts ASSIGN control statements only if the program type has been previously specified as a resident or standard foreground program via a TYPE control statement. Assignments for background programs must be made via the $ASSIGN Job Control statement. (See Paragraph 6-2. 3). Assignments for re-entrant foreground programs must be made through the system service ASSIGN. (See Paragraph 4-13).

Any number of ASSIGN statements may be issued and any number of assignments may be made on any given statement. However, once an assignment has been made for a logical device number, another assignment referencing that same logical device number may not be made.

OPTION

The OPTION control statement has the following format:

OPTION=N

where N may be anyone or more (separated by commas) of the options 16, 17, 18, 20, 22 or 23. The equal sign (=) in the format given above is optional and may be replaced with a blank.

Through the use of this statement, options may be cataloged with a program. In the execution of foreground programs, the options used at execution time are those which were cataloged with the program and all others are zeroed. In the execution of background programs, the options used at execution time consist of the logical "OR" of the current background option word and the options which were cataloged with the program.

The usage of OPTION settings by standard DMS routines and processors is given in Appendix D.

NOMAP

The NOMAP control statement specifies that an external table listing is not to be output at the completion of the linking process.

CBASE

The CBASE control statement has the following format.

CBASE =x

The x specification of the statement is an octal address of a common base specification. A blank may be substituted for the equal sign in the above format. Refer to Paragraphs 6-4.9 and 6-4.10 for usage of this statement.

LIBRARY

The LIBRARY control statement has the following format.

LIBRARY=XXXXXX,YYYYYY,ZZZZZZ

The XXXXXX, YYYYYY and ZZZZZZ specifications are library file names. If the library file specified has a password, then the password should follow the file name in parentheses (e.g., CAT(DOG)). When this statement is encountered, any previously encountered library specifications are lost and the new library list is stored as the current library list. If no LIBRARY statement is used, then the current library list consists of the file to which LFN 12 is assigned during the cataloging process.

During the cataloging process, each library is processed in order according to the order of their specification on the LIBRARY control statement. Each library is processed completely until no more undefined externals can be satisfied, before the next library is processed.

The word LIBRARY may be abbreviated to LIB and the equal sign may be replaced by a blank in the above format.

NOLIB

The NOLIB control statement specifies that the current library list is not to be scanned to find undefined externals for the program being cataloged. When this statement is used for an overlay program, the statement refers only to the segment currently being defined.

This statement would normally be used to save catalog time when it is known that the undefined externals will not be satisfied during the library processing.

INCLUDE

The INCLUDE control statement has the following format.

INCLUDE=XXXXXX,AAAAAA,BBBBBB,CCCCCC . . .

The XXXXXX specification is an include file name. If the include file specified has a password, then the password should follow the file name in parentheses (e.g., CAT(DOG)). The AAAAAA, BBBBBB and CCCCCC specifications are module identifiers. These identifiers may be external definitions or names (i.e., generated by NAME statements in FORTRAN or Assembly Language). If it is desired to specifically allow only an external definition or a name, then the specification should be preceeded by a dollar sign ($) or a period (.), identifying an external definition or name respectively.

The function of this statement is to cause the inclusion of the specified modules from the include file into the program being catalogued. If no module names are specified, then the entire file is included; in this case, the INCLUDE control statement functions similar to the Job Control Statement $INCLUDE.

If the program being catalogued is an overlay program, then the module or modules specified are included only in the segment currently being defined.

The word INCLUDE may be abbreviated to INC and the equal sign may be replaced by a blank in the above format.

MODULE

The MODULE control statement has the following format.

```
MODULE=AAAAAA,BBBBBB,CCCCCC . . .
```

The AAAAAA, BBBBBB and CCCCCC specifications are module identifiers. These identifiers may be external definitions or names and should be specified similarly to the identifiers on the INCLUDE control statement.

The function of this statement is to identify selected modules from the Link Ready File (LFN 15). When this statement is used, the normal Link Ready processing is modified. Normally all modules in the Link Ready File are included in the program; however, when this statement is used, only those modules specified will be included in the program being catalogued.

If the program being catalogued is an overlay program, then the module or modules specified are included only in the segment currently being defined. If this statement is not used during the definition of the main segment (or root), then the root will contain all modules from the Link Ready File that are not specified as part of a segment.

The word MODULE may be abbreviated to MOD and the equal sign may be replaced by a blank in the above format.

SEGMENT

The SEGMENT control statement has the following format.

```
SEGMENT=AAAAAA,BBBBBB,CCCCCC
```

This statement defines a specific segment in an overlay program. If this is the first SEGMENT statement, then it also defines the program to be an overlay type program. This statement terminates specifications for the previous segment. The library list that is

current when this statement is encountered is used for the previous segment (unless a NOLIB control statement was entered for that segment). If this is the first SEGMENT control statement, then the previous specifications refer to the main segment (or root).

Each segment must contain within its defining specifications at least one MODULE or INCLUDE control statement. This does not apply to the main segment (or root) since the absence of a MODULE control statement for the main segment implies that all unspecified modules in the Link Ready File are to be included in that segment.

The AAAAAA specification is used as the segment name. This specification is optional. If it is omitted, then the segment name will appear as blanks on the output listing and the segment may not be referenced in another SEGMENT control statement.

The BBBBBB specification is the name of the parent segment being defined. If this specification is omitted, then the parent segment is assumed to be the main segment or root. The main segment may also be identified explicitly by ROOT. Except for ROOT, BBBBBB must have already been identified on a previous SEGMENT control statement. The BBBBBB specification may also be optionally preceeded by one or more asterisks (*). The number of asterisks indicate the phase level of the segment. None indicates phase zero, one asterisk indicates phase one, etc. If the parent segment is the main segment, the BBBBBB specification may consist of just one or more asterisks.

The CCCCCC specification is the overlay type of the segment. CCCCCC may be omitted or it may be one of the following words: STNDRD, NEW or UPDATE. If CCCCCC is omitted, it is assumed to be STNDRD. The overlay type of the segment defines the loading and unloading procedures for the segment. For more explanation, see Paragraph 6-4.5.

The word SEGMENT may be abbreviated to SEG and the equal sign may be replaced by a blank in the above format.

Figures 6-5, 6-6, 6-7 and 6-8 shown examples of the use of the SEGMENT control statement to define several overlay structures. These structures are the same structures that were shown in Figures 6-1, 6-2, 6-3 and 6-4.

NOTE

The assumption is made that each segment contains only one module. This module is present on the Link Ready File and has the same name as the segment.

```
$CATALOG
SEGMENT A
MODULE A
SEGMENT B
MODULE B
BEGIN
```

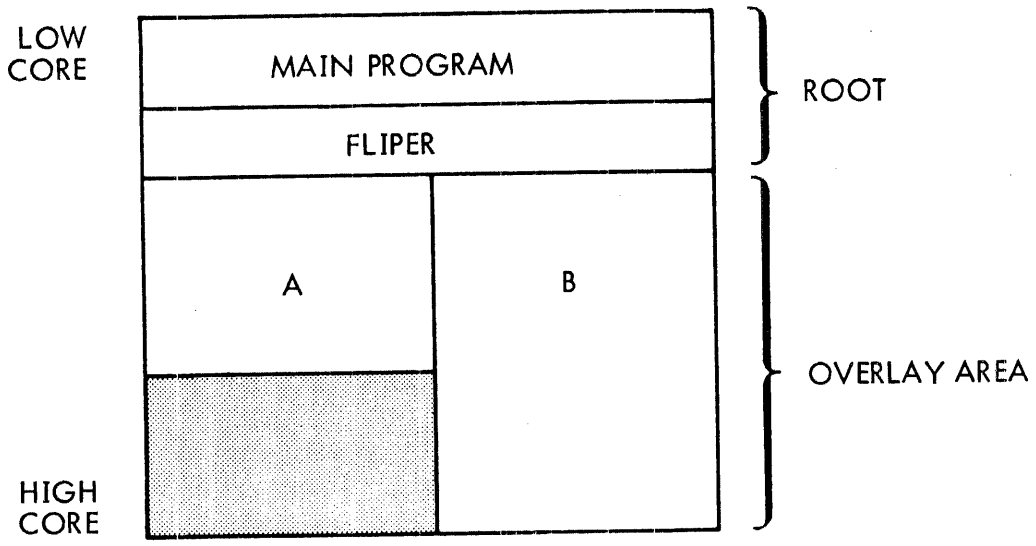


Figure 6-5. Overlay Program - Control Statements

```
$CATALOG
SEGMENT A
MODULE A
SEGMENT B
MODULE B
SEGMENT C,*
MODULE C
SEGMENT D,*
MODULE D
SEGMENT E,**
MODULE E
SEGMENT F,**
MODULE F
BEGIN
```

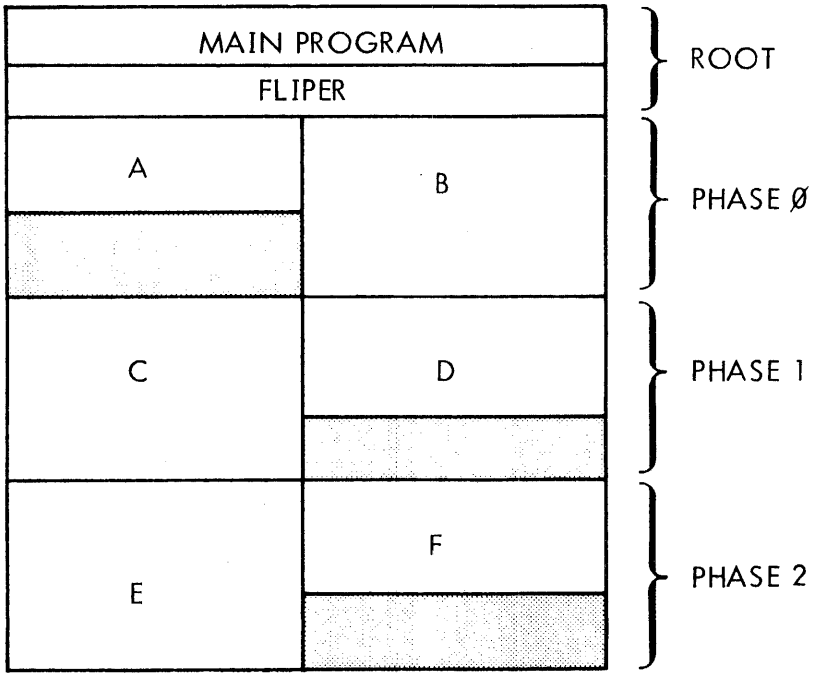


Figure 6-6. Multi-Phase Overlay Program - Control Statements

```
$CATALOG
SEGMENT A, ROOT
MODULE A
SEGMENT B,ROOT
MODULE B
SEGMENT C,A
MODULE C
SEGMENT D,A
MODULE D
SEGMENT E,A
MODULE E
SEGMENT F,B
MODULE F
SEGMENT G,B
MODULE G
BEGIN
```

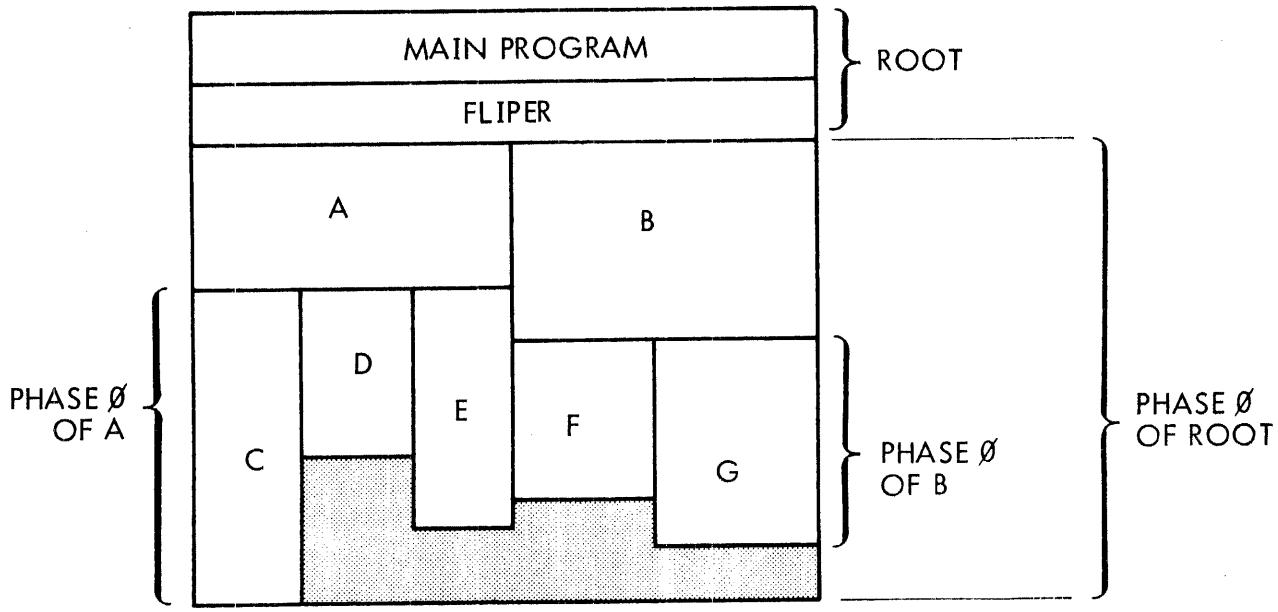


Figure 6-7. Multi-Level Overlay Program - Control Statements

```
SEG A
SEG B,A
SEG C,A
SEG D
SEG E,D
SEG F,D
SEG G
SEG H,G
SEG I,G
SEG J,*G
SEG K,*G
SEG L,*G
SEG M,**G
SEG N,**G
SEG O,*D
SEG P,*D
SEG Q,*D
SEG R,Q
SEG S,Q
SEG T,*
SEG U,T
SEG V,T
SEG W,T
SEG X,*
SEG Y,**
SEG Z,**
SEG AA,**
SEG BB,AA
SEG CC,AA
SEG DD,**
SEG EE,**
SEG FF,Z
SEG GG,Z
SEG HH,*Z
SEG II,*Z
SEG JJ,*Z
SEG KK,*Z
```

NOTE

Only the SEGMENT control statements were shown. The appropriate MODULE or INCLUDE control statements would need to be added to actually catalog this program.

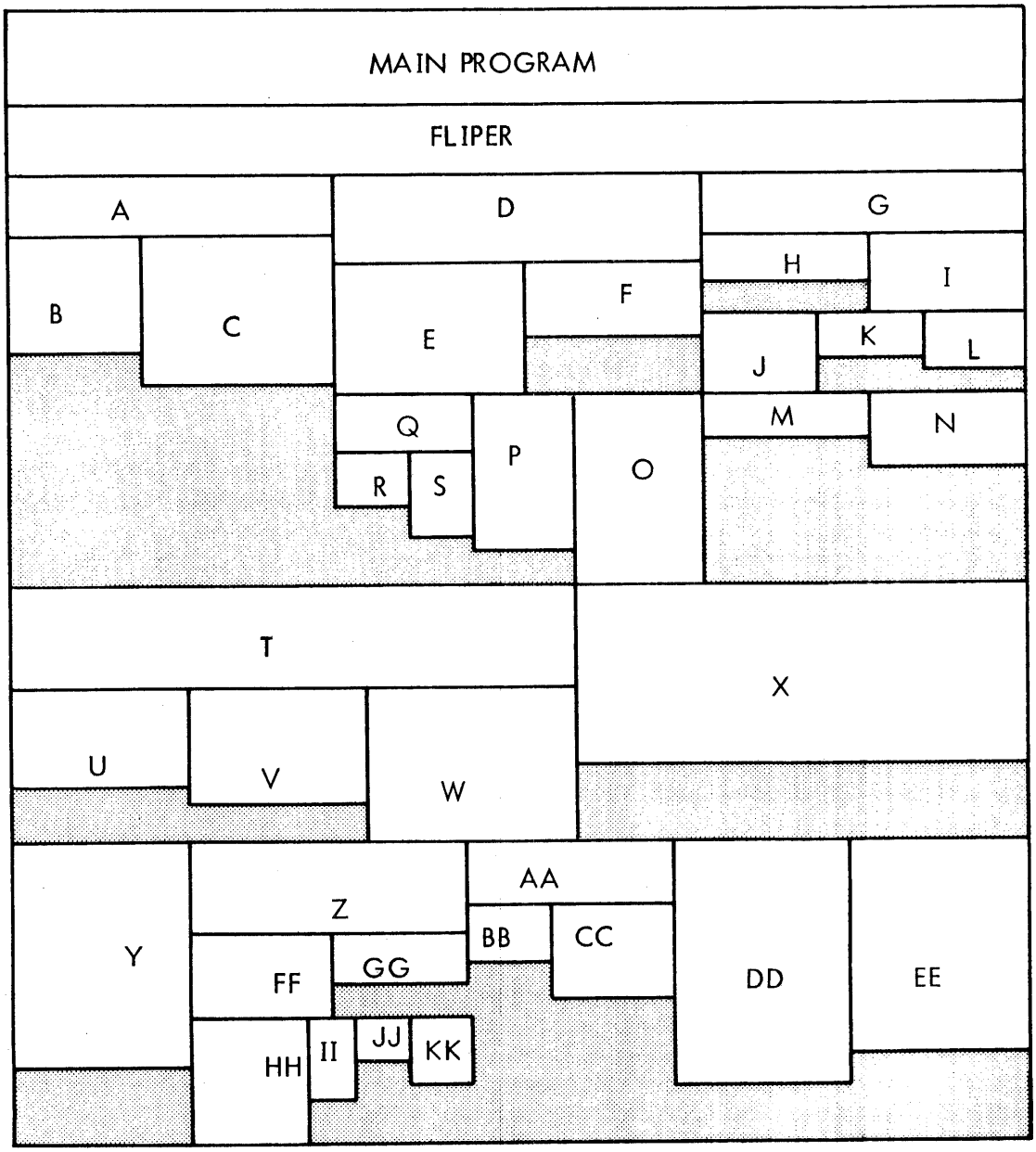


Figure 6-8. Multi-Phase, Multi-Level Overlay Program - Control Statements

FLIPER

The FLIPER control statement is useful only during the cataloging of an overlay program. It may appear anywhere before the BEGIN control statement. In the absence of this statement, the only external references between segments are through the following instructions: BLL, BSL, BLI, BLJ, and BLK. The exception is from child to parent, since this external reference will never require the loading of a segment. If the FLIPER control statement is used, then all references between segments are allowed, and the assumption is made that the user has made allowances for non-standard external references.

RESTRT

The RESTRT control statement is a special purpose statement designed for system generations. Its effect is to cause the link cataloger to input additional control parameters and restart the cataloging process at the completion of the cataloging of the current module. When this restart is performed, the logical file 15 position is not changed and all previously encountered external definitions and external equivalence values are saved to be available to the next module. The effect is to allow more than one program to reference the same set of external definitions and equivalences. The programs to be cataloged via this process must all be on logical file 15; separated only by a single end-of-file mark between each program. Any number of programs may be cataloged together via use of multiple RESTRT parameters.

BEGIN

The BEGIN control statement indicates that no more Link Cataloger control statements follow and that the linking and cataloging process is to begin. This statement may be omitted since the cataloging process will also begin on encountering an EOF on command input. (Reading the next JOB CONTROL statement will return an EOF status and initiate cataloging.)

6-4.3 Link Cataloging Process

The link cataloging process consists of rewinding the Link Ready File (LFN 15) and linking all programs and subprograms residing on this file until an End-of-File is encountered. If unsatisfied externals remain, then the library file will be rewound and searched for the unsatisfied external requests. If, while passing through the library file, all requested externals are satisfied, the linking process terminates and cataloging begins. If an End-of-File is detected on the library file when unsatisfied externals still exist than a determination is made as to whether or not at least one external was satisfied during the most recent pass over the library file. If at least one external was satisfied, the library file is rewound and the process is repeated. If no externals were satisfied at the completion of a pass over the library file, then the second library file (as specified on the LIBRARY control statement) is processed in a similar manner. When the last library has been similarly processed, the linking process terminates and the cataloging procedure begins. Any unsatisfied external requests are replaced by a BLU $ABORT.

An exception to this is the external "PG::HI". If this external is referenced in the module being cataloged and is left undefined at the termination of library processing; then the program high location is used as the definition of $PG::HI. This location is the first location beyond the entire program, including common and overlay segments.

The cataloging procedure consists of determining whether a permanent or non-permanent program file is to be created. If a NAME control statement was issued, then the output of the link cataloger which resides on the System GO file (LFN 16) is copied to a permanent file having the name specified on the NAME control statement and a password of GORP. If a NAME control statement was not issued, then the type and memory requirement specification of the System GO File is redefined in order that the linked program may be loaded and executed when the Link Cataloger relinquishes control of the system.

If the Link Cataloger is initiated by the $CATGO (CATalog and GO) Job Control statement, then the link cataloging process begins immediately as if the following control statements were issued.

```
TYPE=BG,STRICT,SAUT,OA,OM,UA,UM,AFER
BEGIN
```

Upon completing the link cataloging process, control will be passed to the System Loader. The resultant load module which was produced by the Link Cataloger will then be loaded and executed.

6-4.4 Link Cataloging Overlay Type Programs

The link cataloging process for overlay programs consists of several steps. The first step consists of rewinding and processing all programs and subprograms on the Link Ready File and on each of any specified Include files. In addition, the entire file FLIPER with the password SYSTEM is included into the root; this file contains the FLIPER routine. The processing of each module consists of determining which segment each module belongs to. If the module is part of the main segment (or root), then it is linked immediately. If the module is part of an overlay segment, then the module is processed only to determine its length and any externals which are referenced or defined within it. The length of the module is then added to a length counter for the appropriate segment.

When all Include files have been processed, then the external table is examined to find all undefined externals. If an undefined external is found, an attempt is made to define it through a definition in another segment. If no definition is found or the reference between segments is not legal, then the external is left undefined, with the assumption that it will be defined within the library.

During the library processing, a determination is again made as to which segment a library module belongs. If a library module defines an unsatisfied external in two or more segments, then this module will be included in <u>each</u> of the segments. If this is not desired, then the library should be explicitly included in the parent of these segments (see INCLUDE). Again, only length and external information is obtained for modules that are within a segment.

When library processing is complete, the main root module is finished up. Common is allocated and segment locations are allocated. A link map of the root is output if requested.

Then, each segment is cataloged separately. Each module within the segment is accessed randomly by a relative record address and the segment is cataloged. After each segment, the link map for the segment is output if requested.

6-4.5 Execution of Overlay Type Programs

The program file of an overlay program consists of several relocatable absolute modules. There is one such module for each segment (root and overlay) in the program. In addition, there is a reserved area for each overlay segment. This reserved area is used to save a copy of the segment after it has been relocated so that on subsequent loads of the segment, it can be reloaded without relocation and thus it can be reloaded faster.

When the program begins execution, logical file 77 is assigned to the program file. This assignment must not be changed during execution of the overlay program, since the loading and unloading of segments takes place through LFN 77.

The execution of an overlay program when an overlay is necessary takes place as follows. When a reference is made from one segment to an external in another segment (which is not a parent), the address actually referenced is in the FLIPER table. This table is automatically created by the Link Cataloger when it is determined that an overlay type program is being catalogued. An entry exists in the FLIPER table for each external which is referenced in another segment from that in which it is defined (except for child to parent references). This entry appears as follows.

```
AAAAAA    BSL    $FLIP:1
          DAC    BBBBBB
          DAC    SEGMENT
```

AAAAAA is the external address that is actually referenced. BBBBBB is the actual entry address of the routine, when the segment is loaded. FLIP:1 is an entry point in the FLIPER routine that loads the segment. And SEGMENT is the address of an entry in the FLIPER table that contains information needed to load the segment. If AAAAAA is referenced by a BSL instruction, then the entry appears as follows. (FLIP:2 is the FLIPER routine entry point used to process BSL calls.)

```
AAAAAA    ***
          BSL    $FLIP:2
          DAC    BBBBBB
          DAC    SEGMENT
```

When called, the FLIPER routine will load the segment in which the requested external is defined. The FLIPER routine will also load all of the parents of the requested segment. If any of these segments that are to be loaded are already present in core, then the load process is bypassed. If the segment is not present, then all segments that are present and are partly or wholly within the area to be overlaid will be unloaded. When this process is complete, all registers (including the condition register) are restored, and the segment is entered at the actual external address requested. In the special case of a call by a BSL, the return address and condition are stored at the actual external address requested; therefore, making the FLIPER processing transparent to the segment.

In some special cases, it may be desirable to cause a segment to be loaded without directly calling it. This function may be performed by the following calling sequence.

```
          TLO    SEGMENT
          BSL    $FLIP:3
```

This call will result in the loading of the specified segment, and all parents of the specified segment. Because of this and other system requirements, the following list of externals may not be used when cataloging an overlay program: FLIP:0, FLIP:1, FLIP:2, and FLIP:3. FLIP:0 is the first location of the FLIPER table and is used by the FLIPER routine to enable loading of all parents of a specified segment.

The actual process of loading and unloading a segment is dependent upon the type of segment. In all cases, the initial load of a segment is done through the System Loader. The System Loader loads and relocates the segment into the appropriate overlay area. Subsequent loads are done with a single read from the reserved area of the program file for the requested segment.

For a STNDRD (see SEGMENT control statement) overlay segment, the first unload request for the segment results in a single write of the segment to the reserved area of the program file for the segment. Subsequent unload requests are ignored, and the segment status is flagged so that following load requests cause the segment to be read in.

For a NEW (see SEGMENT control statement) overlay segment, the initial load of the segment is immediately followed by an unload to the reserved area of the program file. All standard unload requests result only in flagging the segment status so that the segment will be read in when next requested. The distinction between NEW and STNDRD segments is that data modified during the initial load of the segment will be re-initialized during a reload of a NEW segment, and will remain in the modified state during all reloads of a STNDRD segment.

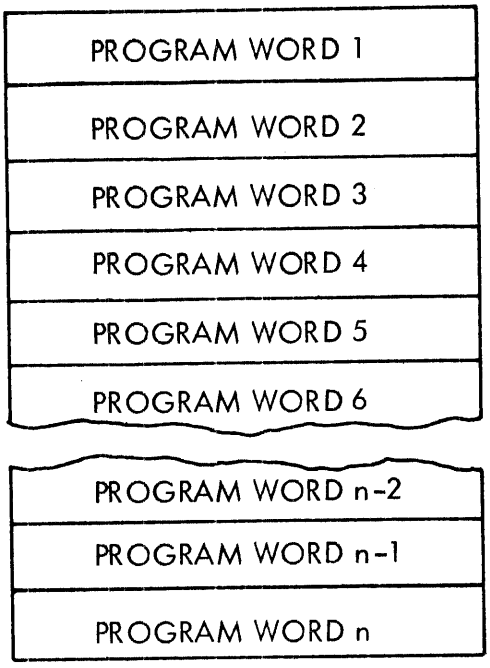
For an UPDATE (see SEGMENT control statement) overlay segment, each unload request for the segment will result in a single write of the current segment to the reserved area of the program file for the segment. Thus data modified during any execution of a segment module will remain modified for the next reload. It should be noted, that for any significant number of overlays, an UPDATE segment will result in approximately twice as many disc accesses as a STNDRD or NEW segment.

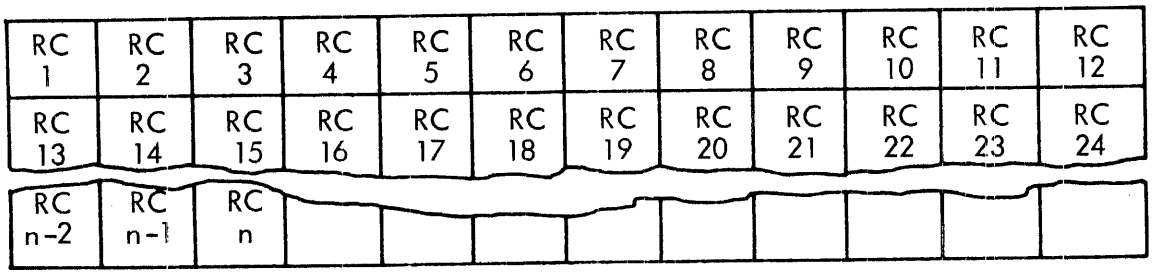
6-4.6 Link Cataloger Output Format

As the linking process proceeds, an internal buffer is used to accumulate the program being processed. Whenever this buffer is filled, it is output to logical device 16 which must have been assigned to a disc file. As each program word is placed in this buffer, a corresponding relocation code is placed in the relocation vector array. This code specifies to the DMS system loader whether the word is to be relocated or not, and if so, whether the relocation is to be within a 15 or 16 bit address field (refer to Figure 6-9).

When all linking functions have been performed, the relocation vector array will be output behind the program in fixed word length records consisting of 112 words.

For background programs the DMS System Loader reads the program into memory as a single record. The relocation vectors are then read, sector by sector, and the program relocated. Foreground programs are loaded with one disc access; the program body is read into the area allocated for the program, and using the restart feature of the disc ABC channel, all of the relocation vectors are read into a dynamic core buffer. After relocating the program the dynamic core buffer is deallocated. The size of the program body read into core is the program plus 2 additional words for background and re-entrant foreground program and the program plus 114 additional words for foreground programs. The relocation bias used by the DMS System Loader will correspond to the address of the first word of the program minus 2 for 114 for background and foreground programs, respectively.





NOTES: RC=two-bit relocation code:

IF RC=00, then the corresponding program word is not to be relocated.

RC=01, then the corresponding program word is to be relocated in a 15-bit address field.

RC=10, then the corresponding program word is to be relocated in a 16-bit address field.

Figure 6-9. Link Cataloger Output Format

6-4.7 Link Cataloging Background Programs Accross the 32K Boundary

Provision is made in the Link Cataloger for automatically creating background load modules that will cross the 32K memory partition. If during the linking process it is determined that a subroutine will cross the 32K boundary when loaded by the System Loader, then the link cataloging process will be restarted for that subroutine. The restart procedure is such that the subroutine will reside at '100000 when loaded. (Note that the Link Cataloger is able to determine boundary crossing since all background programs are loaded at the same memory location.) When the final load module file is created by the Link Cataloger, the program will be typed as "ABSOLUTE BACKGROUND". The significant difference between non-absolute and absolute background programs is that it will be necessary to re-catalog absolute background programs if a new System Generation is performed. This is because the background low memory address will generally change; therefore, the absolute address will not be valid under a new system. This checking is disabled by entering the "LM" option on the "TYPE" statement, allowing the user to catalog large background programs and later dumping them for execution at another address.

6-4.8 Link Cataloger Table and Buffer Allocation

In order to build the program and its associated relocation vector array on the System GO File, it is necessary that the Link Cataloger be provided with working storage. The entire area between the end of the Link Cataloger (which is a background processor) and background high is utilized. This available space is partitioned into two regions, one in which the program or portion of the program is built, and the other where the external table and relocation vector array are built (refer to Figure 6-10). A determination of where this partition is to be located in the available background area is made according to the following rules.

A. An initial relocation vector array and external table area consisting of 2000 locations is assigned if at least 112 locations for the program work buffer are available. If there are not 112 locations for the work buffer when the table area is 2000 locations, then a work buffer area of 112 locations is assigned and the remaining area is assigned to the table region.

B. If during the Link Cataloging process, it is determined that the relocation vector array is about to overlap the external table, then the following occurs. The current work buffer is output to the System GO File. The table area is increased by 1000 locations if there will be at least 112 locations left for the work buffer, otherwise the table area is set to the entire available area less 112 locations. Then appropriate internal parameters are recalculated, the external table is moved down to the new beginning location of the table area, and the cataloging process continues.

The formula for determining the amount of storage required to contain the relocation vector array is as follows.

Array Size = program size/12

The formula for determining the amount of storage required to contain the external table is as follows.

External table size - 12+E+C+Q+I

where: E = (number of external entries. * (average size of entry (=4)))

C = (numbers of common block names)* 3+2 * (distinct number of common variables)

Q = (number of external equivalence definitions) *4

I = (number of internal string requests) *2

The minimum size of the program work buffer is 112 locations, however, the time required for the link cataloging procedure will be considerably lessened if larger amounts of storage are available.

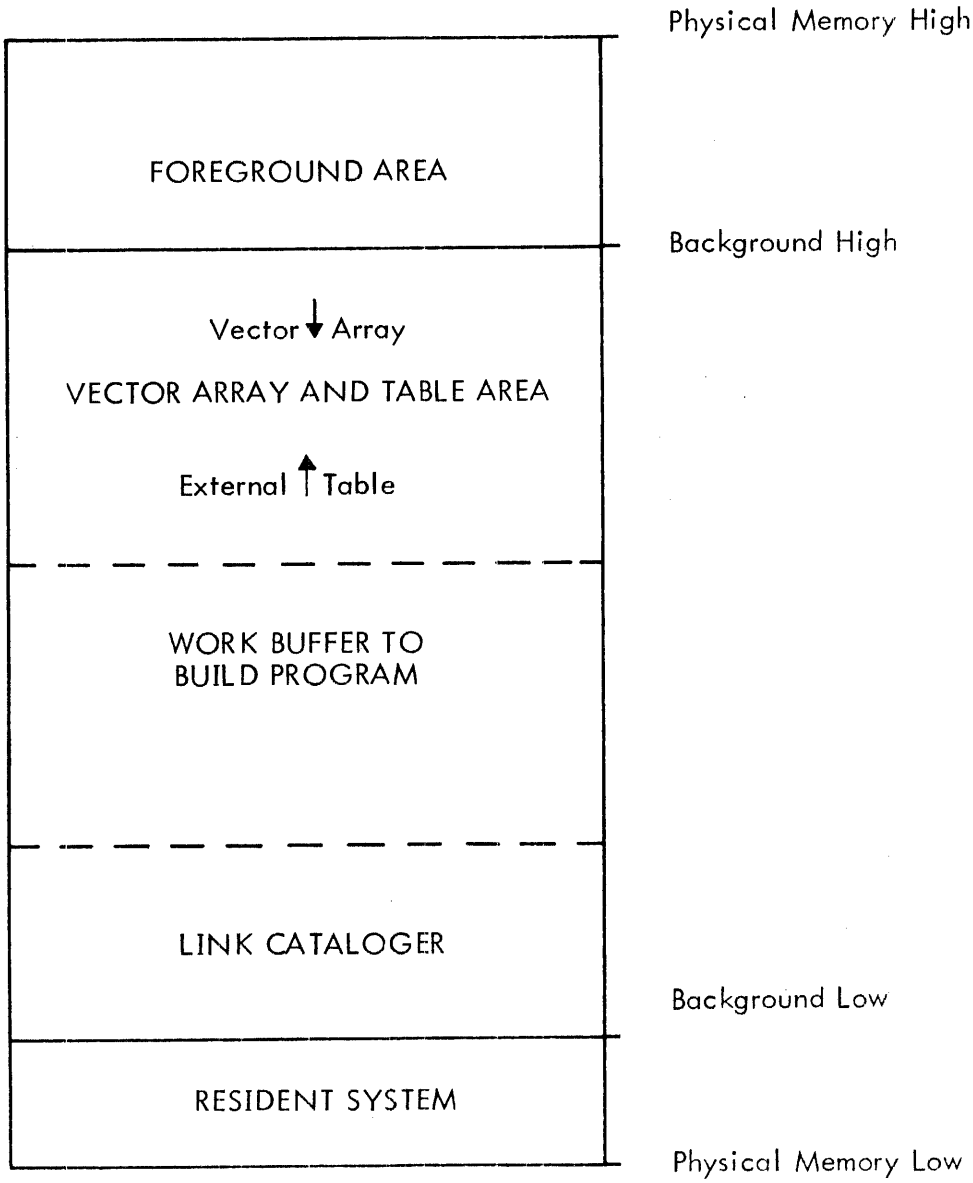


Figure 6-10. Link Cataloger Table and Buffer Allocation

6-4.9 Common Memory Allocation

One of the design features of the Link Cataloger is that the common region (which is always assigned above the program) is automatically compressed to the first available location following the user's program. This permits the Link Cataloger to define (for the system Loader) the smallest possible program memory requirements, since the common region is normally considered as being part of the program. If however, a CBASE control statement has been issued, then the start of the common region will be assigned to the relative address on the control statement and be built upwards. The address on the CBASE control statement is always relative to the actual program and is independent of the program type (background or foreground).

In overlay type programs, the common is normally compressed to the first available location following the main segment (or root) and all overlay segments. If however, a CBASE control statement has been issued, then the start of the common region will be assigned to the relative address on the control statement and be built upwards as for a non-overlay type program. The segments will be allocated as much as possible between the program and the common area, with the segment overlay areas which are too large allocated above the common area.

6-4.10 Link Cataloging Program CHAIN Segments

Size and common allocation are the main considerations in Link Cataloging program CHAIN segments. Since the primary method of argument passing between CHAIN segments is through common, it is important that the first CHAIN segment specifies the common region as being part of the program. (This occurs automatically unless otherwise specified.) This is necessary if the program is a foreground program, so that the proper memory requirements are defined; or if the program contains BLOCK DATA, so that the common area is initialized properly. It is equally important that all other CHAIN segments not include the common region as part of the cataloged module since to do so would cause the common region to be destroyed when the CHAIN segment is loaded for execution. To prevent the inclusion of common as part of the CHAIN segment, a SEGMNT parameter would be specified on the TYPE control statements. Because CHAIN segments are always loaded at the address of the first CHAIN segment, provision must be made to insure that the common region is at the same address relative to each program CHAIN segment. This is done by issuing a CBASE control statement as each program CHAIN segment is cataloged. The relative address specified on the control statement should be identical for each CHAIN segment. If some or all program CHAIN segments do not use common storage, then care must be taken to ensure that the first program CHAIN segment is the largest one of the set.

NOTE

The function of program CHAIN segments to provide program overlay capabilities is completely independent of the overlay type program discussed in Paragraph 6-4.1. Similarly, the TYPE=SEGMNT control statement is unrelated to the SEGMENT control statement. However, it should be noted, that a program CHAIN segment may be an overlay type program.

6-4.11 Link Catalog Map

Upon successful completion of the linking process, a listing of the Link Cataloger external table will be output to the List Output logical device (06) unless a NOMAP control statement was previously issued. The map will consist of the program name (if any) followed by two side-by-side columns of external, external equivalence, common, and system names and values which will appear in alphabetic and numeric sequence, respectively.

The format of the Link Map is as follows.

P . NAME = LLLLLL

XXXXXX	YYYYYY	ZA	XXXXXX	YYYYYY	ZA
XXXXXX	YYYYYY	ZA	XXXXXX	YYYYYY	ZA

where: LLLLLL is the name of the program (as defined by the parameter on the NAME control statement).

XXXXXX is the external, external equivalence, common block name, or catalog parameter.

YYYYYY is the assigned octal memory location or value.

Z is a single character identifier which may be one of the following:

1) (blank) - indicates that the XXXXXX name is a Link Catalog parameter.
2) C - indicates that the XXXXXX name is a common block.
3) $ - indicates that the XXXXXX name is an external.
4) # - indicates that the XXXXXX name is an external equivalence definition.

A is a single character identifier which may be one of the following:

1) (blank) - indicates that the XXXXXX name was singly defined.
2) M - indicates that the XXXXXX name was encountered more than once, i.e., has a multiple definition.
3) U - indicates that the XXXXXX name was undefined. In the case of external equivalences, the "U" indicates that at the time of the first external equivalence request, the name XXXXXX was not defined. If this occurs, the value of the external equivalence is set to zero, and subsequent definitions (if any) of XXXXXX are ignored.

The Link parameters that are passed back to the operating system also appear as an XXXXXX name with a special format. The first character of the XXXXXX name of a link parameter is a special character ("*" or ". "). These names and their meanings are as follows:

*LOW The YYYYYY value associated with this name is the relative low memory address of the program.

*HIGH The YYYYYY value associated with this name is the relative high memory address of the program, If the program contains references to common, then this address will be that of the highest element within the common region. (This is not true when a SEGMNT parameter is given on a TYPE control statement.)

*START The YYYYYY value associated with this name is the relative starting address of the program.

. BCOMM — The YYYYYY value associated with this name is the lowest address of blank common if defined in the program.

. PASS — The YYYYYY value associated with this name is the number of passes through the Library File necessary to satisfy external requests. If this value is greater than one, then the modules on the library file are not ordered or more than one library was processed.

*P. END — The YYYYYY value associated with this name is the relative high memory address of the program. This address does not include common blocks, and is provided only when there are common blocks, and TYPE=SEGMNT is not used.

*C. END — The YYYYYY value associated with this name is the relative high memory address of the common area of a program. It is provided only when there are common blocks and TYPE=SEGMNT is used.

*STRT n — The YYYYYY value associated with this name is the nth encountered "starting address" after the first. The starting address actually used is " * START " and is the first such address encountered. The presence of these entries indicates additional "main" programs. If this occurs, care should be taken to ensure proper functioning. More than 9 additional "starting addresses" are ignored.

If the program being catalogued is an overlay type program, then the memory map printed for the program is split into several parts. The first part corresponds to the main segment (or root), and appears as a single program. However, only externals which are defined or referenced in the main segment (or root) are printed in the first part of the map. Each additional part of the map corresponds to a particular segment. Each of these parts is headed by a line with "SEGMENT" and the segment name on it. The map for each part lists only these externals which are defined or referenced in the segment. Only two link parameters appear in these sections. These are the high and low address in the segment, identified by *SG. HI and *SG. LO respectively.

NOTE

Some of the externals defined in the segments may appear with addresses that are not within the segment. This is because the externals are referenced outside of the segment and a reference to them may cause an overlay to take place. Thus, the entry to these externals is made through the FLIPER table.

6-4. 12 Code Processing

The following paragraphs describe some of the more important codes that are processed by the Link Cataloger. The processor(s) that produces the code is identified along with usage consideration and the action taken by the Link Cataloger.

External Definition

The Macro Assembler generates an external definition for each XDEF pseudo-operation encountered. The FORTRAN IV compiler generates external requests when the SUBROUTINE or FUNCTION statements are processed. The code generated defines an address that is to be associated with an external name. The name may be identical to an external equivalence or Common Block name without conflict. If external definitions of the same name are encountered in subsequent modules, then linkage will be made to the definition that was encountered first. The module that contains the duplicate external definitions will not be loaded if it is encountered on the Library File unless it also contains another definition that was previously requested but not satisfied. If the module that contains the duplicate external definition resides on the Link Ready file, then it will be unconditionally loaded. If the module is loaded, then any external names that are defined more than once will be noted on the Link Map.

External Request

An external request is generated by the Macro Assembler when an operand is encountered that is preceded by a dollar sign ($). External requests are also generated by the FORTRAN IV Compiler any time subroutines or functions are determined to be external to the program. External requests are also identified as to whether they are to be considered unconditional and conditional requests. Conditional external requests (denoted in assembly language by preceding the operand by two consecutive dollar signs) are satisfied only if the requested name was previously unconditionally requested. If the name was not unconditionally requested prior to the conditional requests, then a BLU instruction to the system service routine $ABORT will be substituted for the requesting instruction.

System Service Request

A system service request is generated by the Macro Assembler when the instruction BLU $XXX is encountered.

If the requested external name is found in the Link Cataloger's External Name table (indicating that an external definition having that name has already been loaded), then a BLL instruction is inserted and the linkage is satisfied. If the requested external name is not found in the Link Cataloger's Name table and is in the DMS System Service table, a BLU instruction to the associated dedicated memory location is inserted and the linkage is satisfied. If the requested external name is not found in either table, a BLL instruction is loaded and the external name is entered in the Link Cataloger's External Name table. For the linkage to be satisfied, the requested external name must follow in a module residing on the Link Ready file or the library file.

This function permits linkage to DMS System Services or user defined routines irrespective of whether a particular service is resident within the operating system or on the Library File. Services may also be added or deleted from the resident portion of the operating system without re-assembling or compiling the requesting program.

External Equivalence Definition

External equivalence definitions are generated by the Macro Assembler when the pseudo-operation "XEQV" is encountered. This definition defines a 24-bit constant which is to be merged with the corresponding external equivalence request when encountered. This function is useful in externally defining data constants, channel/unit numbers or input/output instructions, etc. The external equivalence name associated with the definition may be identical

to an external or common block name without conflict. If multiple equivalence definitions of the same name are encountered, the first one will be used as the definition and all subsequent definitions of the same name are encountered, the first one will be used as the definition and all subsequent definitions of the same name will be ignored. However, the fact that the name was defined more than once will be noted on the Link Map. External equivalence definitions are similar to Common definitions in that the external equivalence definition must precede any external equivalence requests.

External Equivalence Request

External equivalence requests are generated by the Macro Assembler when an operand is encountered that is preceded by a number sign (#). This request indicates that the value associated with the previously encountered external equivalence definition of the same name is to merged (24-bit OR) with the requesting frame. If this is not done, the external equivalence label will be flagged as undefined, whether or not a subsequent definition is encountered.

Common Definition

A common definition is generated by the Macro Assembler when the pseudo-operation COMM is encountered, or the FORTRAN IV compiler when a COMMON statement is received. The common definition specifies the size of the common area to the Link Cataloger so that subsequent common requests may be assigned an address. The definition of the overall size of a labeled common block must be identical in all programs and subprograms in which it is defined. However, blank common areas defined in a set of programs and subprograms that are to be linked do not have to correspond in size. The common block name associated with the common definition may be identical to an external name or external equivalence name without conflict.

Common Request

A common request is generated by the Macro Assembler or FORTRAN IV Compiler whenever a reference is made to a variable which has been declared to be in common. The request carries the displacement from the common block name with which it is associated. For example, if variables A, B, and C are declared to be in common (labeled or blank), then their displacement from the block name is 0, 1, and 2 respectively. Common requests are distinguished as to whether the address size is 15 or 16 bits. Hence, 15-bit common requests must reference only those variables defined in the same memory bank as the request.

Name Definition

A name definition is generated by the Macro Assembler when the pseudo-operation "NAME" is encountered, or by the FORTRAN IV Compiler when a "NAME" statement is received.

Source Program Error

A source program error code is generated by the Macro Assembler and the FORTRAN IV Compiler when an irrecoverable error is detected. This code will immediately terminate the link cataloging process and an error message will be output.

Common Origin

A common origin code is generated by the FORTRAN IV Compiler when a BLOCK DATA subprogram is encountered. It defines a displacement from a common block name where data is to be loaded.

END Code

An END Code is generated by the Macro Assembler and the FORTRAN IV Compiler when an END statement is received. This code defines the end of the link module currently being processed and causes the Link Cataloger to prepare to accept another module.

END-Jump Relative Code

An END-Jump relative code is generated by the Macro Assembler when an END pseudo-operation is received which contains a relative operand expression. An address is associated with this code which defines to the Link Cataloger the relative starting address of the program being link catalogued. If more than one END-Jump relative code is received, then the last one encountered will be used.

END-Jump Absolute Code

An END-Jump absolute code is generated by the Macro Assembler when an END pseudo-operation is received which contains an absolute operand expression. This code must not be presented to the Link Cataloger since only relocatable programs are processed.

Relative Origin

A relative origin code is generated by the Macro Assembler when the pseudo-operations RORG and BLOK are encountered and by the FORTRAN IV Compiler when DIMENSION statements are received. This code resets the relative location counter within the Link Cataloger.

Absolute Origin

An absolute origin code is generated by the Macro Assembler when the pseudo-operation AORG or a BLOK pseudo-operation which was preceded by an AORG. This code must never be presented to the Link Cataloger since only relocatable programs are processed.

END$ Record

An END$ record is generated by the Macro Assembler or the FORTRAN IV Compiler when an END$ statement is received. This record is ignored if option bit 6 is not set.

6-4.13 Input and Code Placement

An input record to the Link Loader consists of 55 words: six 9-word subfields; and a one-word, hash-total checksum. The first word of each 9-word subfield contains eight 3-bit loader codes that determine the action to be taken for each of the following eight words in the subfield (refer to Figure 6-11). Some codes require the use of multiple words to describe a particular function in which case the codes corresponding to the extra words are set to zero. If word 1 and word 55 of the input record are set to a -1 and all other words within the record are set to zero, then the record is considered to be an END$ record.

Table 6-2 lists the various codes which are accepted by the Link Cataloger and Table 6-3 lists the special action codes.

Table 6-2. Link Loader Input Codes

Code Bit Configuration	Identification and Placement
000	Direct Load.
001	Memory Reference 15-bit.
010	External Definition - the first word contains the address to be associated with the name which follows in the next two words.
011	External Request - the first word is the request frame with the following bit settings: B0 = 0: 15-bit request. B0 = 1: 16-bit request B1 = 0: Unconditional request. B1 = 1: Conditional request. The request name follows within the next two words.
100	Memory Reference 16-bit.
101	Common Request 15-bit - the address filed in the first word contains the displacement from the block name specified in the next two words.
110	Special Action bits 16 through 20 determine the action to be taken (Refer to Table 6-3).
111	Common Request 16-bit - the address field in the first word contains the displacement from the block name specified in the next two words.

Table 6-3. Link Loader Special Action Codes

Special Action Bit Configuration	Identification and Placement
00000	ORG Absolute - bits 0-15 of the word contains the absolute address.
00001	ORG Relative - bits 0-15 of the word contains the relative address.
00010	END
00011	END-Jump Absolute - bits 0-15 contains the absolute address to be passed as the starting address.
00101	Internal String Back - bits 0-15 contain the address of the first link in the chain to be strung.
00110	External String Back - bits 0-15 contain the address of the first link in the chain; the next two words contain the external name.
00111	Name Definition - the next two words contain the name to be associated with the program.
01000	Common Definition - bits 0-15 contain the size of the block; the next two words contain the block name.
01001	Common Origin - bits 0-15 contain the displacement from the block name contained in the next two words into which data is to be loaded.
01010	Source Program Error.
01011	System Service Request - the next two words contain the requested external name.
01101	External Equivalence Definition - the next two words contain the name and the following contains the equivalence value.
01110	External Equivalence Request - the next two words contain the name and the following contains the instruction into which the request value is "OR"ed.

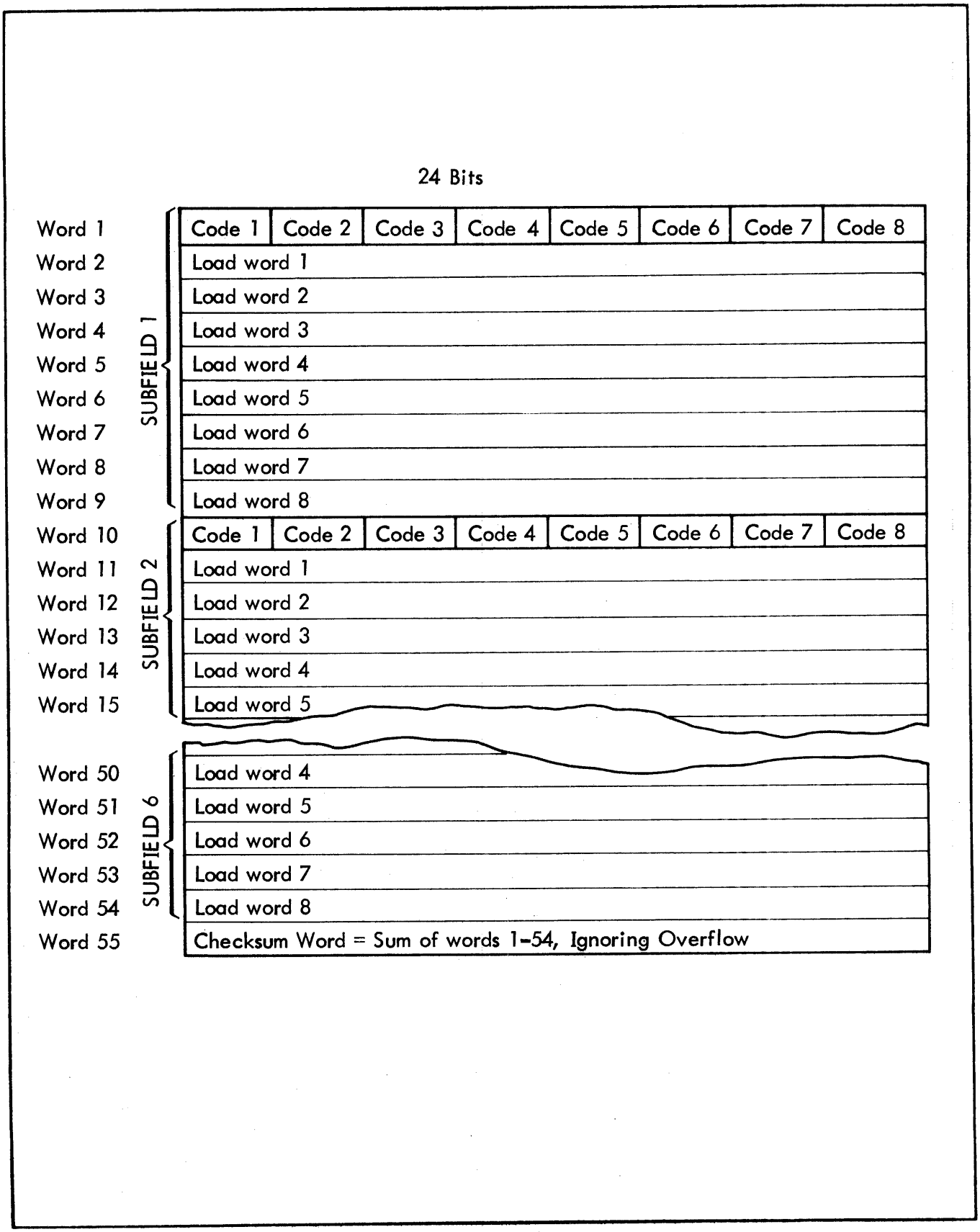


Figure 6-11. Code Placement Format

6-4.14 Link Cataloger Error Message Codes

During the linking process, extensive checking is performed to ensure a proper program load. If an error condition is detected, a message will be output to the list output logical device (06). The format of the error message is as follows.

LCT XX @ABSOLUTE YYYYYY, RELATIVE ZZZZZZ IN MODULE AAAAAA**LINK ABORTED**

where: XX is a two-decimal digit error code. (The meanings of the error codes are given in Table 6-4.)

YYYYYY is the memory address relative to the program (main plus subroutines) in which the error occurred.

ZZZZZZ is the address relative to the program module being linked in which the error occurred.

AAAAAA is the last encountered external definition or name definition. If no external or name definitions were encountered, "**MAIN" will be output.

For some of the error messages listed in Table 6-4, the module name AAAAAA could possibly be deceptive; hence, caution should be exercised.

Table 6-4. Error Messages

Number	Meaning
1	Invalid control statement.
2	A "TYPE" control statement was encountered after an ASSIGN control statement.
3	An invalid parameter was encountered in the "TYPE" control statement.
4	Invalid delimiter following a parameter on a control statement.
5	An end-jump-relative code was encountered within an overlay segment.
6	File access specification on a "NAME" control statement is not "R", "W", or "D".
7	The first character of the specified name on a "NAME" control statement was not alphabetic.
8	An invalid common base specification was made on a "CBASE" control statement.
9	An invalid pack number specification was made on a "NAME" control statement.
10	An "ASSIGN" control statement was encountered with either an implied or stated program type of background or re-entrant foreground.
11	An invalid logical device number was encountered on an "ASSIGN" statement.

Table 6-4. Error Messages (Cont'd.)

Number	Meaning
12	A logical device number was assigned to more than one physical device or file name.
13	A physical device number of file name was encountered on an "ASSIGN" control statement that was un-recognizable.
14	A physical device number of zero or greater than '77 was encountered on an "ASSIGN" control statement.
15	The number of assignments made has exceeded the available room in the File/Device control block.
16	Character position 80 has been reached on a control statement without having encountered a statement terminator.
17	Invalid link cataloger input code from the binary input stream.
18	An absolute origin code was encountered on the input stream.
19	An end-jump absolute code was encountered on the input stream.
20	A source program error code was encountered on the input stream.
21	Logical device '16 is assigned to a blocked disc file. The GO file must be unblocked.
22	A labeled common block name was encountered that was not the same size as previously encountered block of the same name.
23	Logical File 12 or 15 is not assigned to a disc file and overlay type program specifications were entered.
24	The indicated segment has a zero length. None of the modules specified for the indicated segment were found.
25	A common base was specified on a "CBASE" control statement such that the program will overlap the common area.
26	String boundary violation.
27	Less than 336 locations are available in buffer area.
28	An external in the FLIPER table was referenced through an instruction which was not a BLL, BSL, BLI, BLJ, or BLK.
29	There is insufficient background area to catalog the program.
30	The word count was not complete when a binary input record was requested.
31	A checksum error was encountered on an input record which was requested from the Link Ready File.
32	A checksum error was encountered on an input record which was requested from the Library File.
33	A BLOCK DATA subprogram was encountered in a program which was explicitly typed as being segmented (i.e., TYPE=SEGMNT).
34	A BLOCK DATA subprogram was encountered on the Library File.

Table 6-4. Error Messages (Cont'd.)

Number	Meaning
35	More than ten BLOCK DATA subprograms were encountered.
36	An attempt is being made to catalog a foreground program without a name.
37	An attempt is being made to load data into the system common region via a BLOCK DATA subprogram.
38	An end-of-file was encountered at an improper position on the Link Ready file ('15).
39	An end-of-file was encountered at an improper position on the Library file ('12).
40	An end-of-file is present on start of the Link Ready file. (i.e., no program is on the Link Ready file.)
41	Logical device '16 (the cataloged output of the Link Cataloger) is not assigned to disc file.
42	A BLOCK DATA subprogram was encountered on non-disc device.
43	Assignment statements were encountered when the program being cataloged was specified as being a CHAIN segment.
44	Logical device '15 and '16 are assigned to the same disc file.
45	A 15 bit common request in a background program was encountered such that an invalid reference across 32K memory partition would be made.
46	A 15 bit external request in a background program was encountered such that an invalid reference across the 32K memory partition would be made. (The external name is in one memory bank and the request is in the other memory bank).
47	A foreground program exceeds 32K in size.
48	A resident foreground program was specified as being a CHAIN statement.
49	A "SYSCOM" definition was encountered that is greater in size than the DMS System Common Area.
50	No modules are specified on a "MODULE" control statement.
51	"Read access" must be specified for Resident foreground programs.
52	Foreground programs cannot have a type "SYSGEN".
53	Overlay type is not "STNDRD", "NEW" or "UPDATE" on a "SEGMENT" control statement.
54	Undefined parent segment specification on a "SEGMENT" control statement.
55	Segment name already specified on a previous "SEGMENT" control statement.
56	Neither an "INCLUDE" or a "MODULE" control statement has been specified for an overlay segment.
57	Option specified is not one which may be cataloged with program.

Table 6-4. Error Messages (Cont'd)

Number	Meaning
58	Program exceeds 65K.
59	Include or Library file does not exist. The missing file is identified after "MODULE" in the error message.

SECTION VII
ACCOUNTING SUBSYSTEM

7-1 SCOPE

This section describes the accounting subsystem, an optional part of a DMS configuration. Included is a summary of the accounting considerations described in other parts of this document along with a description of the use and operation of the DMS Accounting Utility Program "ACCUP".

7-2 USER NUMBERS

User numbers are used to identify activities within a DMS system and to provide an identifying number to which system usage can be allocated.

7-2.1 Establishing User Numbers

User numbers can be added to a functional DMS system at any time and must be entered through the operator communication device, via the "AU" (Add User) command. This command provides for user number, account number and name identification (refer to Section IX for the command format). The user number must be a 1-5 decimal digit number and as in all other occurrences, it must be preceded by the letter "U", i.e., U123. User number limits are set at system generation time for the individual installation. DMS provides for a maximum range of 1-65535.

Account numbers can be any integer valid on the Series 6000 computer and are used by the Accounting Utility Program to provide another, larger grouping for summary purposes. For instance, this might be a department number, which then allows the utility program to summarize by department.

Name can be any 15 character string, the first of which, of course, is non-blank. Individual DMS installations should establish a uniform naming convention. It is recommended that the last name be entered first, since the alpha-sort feature of the Accounting Utility Program assumes this format.

7-2.2 Changing and Removing Users

User numbers can be removed and entries changed via other operator communication entries. The "CU" (Change User) command provides the ability to specify a new account number, a new name, or both. The "RU" (Remove User) command removes a specified user from the files. Refer to Table 9-1 for further information.

7-2.3 Listing Users

The Accounting Utility Program provides the ability to list users either by number, by account, or alphabetically by name, via use of the LIST command. See Paragraph 7-7 for further information.

7-2.4 Specifying User Numbers on Program Initiation

As stated earlier, a user number must be associated with the execution of every program in the accounting configuration of DMS. User numbers are associated with programs in a variety of ways as discussed in the following paragraphs.

7-2.4.1 Programs Initiated via Operator Communications

Any foreground program initiated via operator communications obtains its user number from the user-number parameter that is required in accounting systems. The user number must be the decimal number preceded by the letter "U". For example:

IP,XYZ,U123,56

is used to initiate program XYZ with user number 123 at priority 56_8.

Other commands which require a user-number are:

IP	Initiate Program
SP	Schedule Program for periodic execution
BP	Begin Program at specific time
CP	Connect Program to external interrupt

Additionally, the "SO" (Spool Output) operator communication command (refer to Table 9-1) requires a user number for the execution of the Foreground-output-spooler program which this command causes to be initiated. Its format is:

SO,FILENM,U123,terminal

7-2.4.2 Terminal Foreground Programs

Any foreground program initiated from a terminal under ACRONIM is automatically given the user number of the current terminal user. This was entered via the $ON command which is required to be the first command entered from a terminal in a DMS Accounting System.

7-2.4.3 Foreground Programs Initiated by other Programs

Any foreground program which is initiated via a System Service Call from another foreground program is automatically given the user number of the program which requested the initiation.

7-2.4.4 Background Batch Programs

Background batch programs are given the user number of the job in which they are executing. All jobs in an accounting system must be preceded by a valid $JOB card as discussed in Section VI. This $JOB card must contain a valid user number which is used for all program executions for the duration of that job. The $JOB card must contain somewhere on it, in the first 60 columns, a valid user number which is a decimal number preceded by the letter "U". For example:

$JOB CAT U123 D6

is used for a job named CAT run by user 123 with list output spooled to device 6.

7-3 TERMINAL OPERATION

Under an Accounting DMS System, a user number identifying the particular terminal user is required at all times. ACRONIM executions are charged to this number and access is allowed to the particular user's files via the user number.

The user number is initially established via a $ON command whose format is:

$ON Unnn

Where: Unnn is the decimal user number preceded by the letter "U". This user number remains in effect until one of the following occurs:

a) A $OFF command is entered. This causes the terminal to be turned off and a $ON is required to turn it back on.

b) A $JOB image is entered directly from the terminal with a user number different from that of the current user. When the $EOJ (end-of-job) image is received for the spooled job, the terminal is turned off just as if a $OFF had been entered. If the user number on the job card was the same as the current one, the terminal retains its identity after the $EOJ is encountered.

c) When another $ON command is entered. The first such effectively becomes just a $OFF. Another $ON is then required to establish the identity of the user and turn on the terminal.

It is important to nore that most accounting information for the terminal is not written to disc until the terminal is terminated via the $OFF or similar commands as discussed above. Thus care should be taken that remote terminals are turned "off" before system operation is terminated.

7-4 DISC FILES

The use of user numbers within the file security system of DMS has been discussed in Section III. However, the important points are repeated below.

Each disc file created under the DMS Accounting System has associated with it a user number. This is the user number of the creator of the file, i.e., the user number associated with the program whose execution created the file. The user number is stored in the Master Disc Directory entry in the 5th and 6th character positions of the password. (See Master Disc Directory entry layout in Appendix A). Optionally, if the file was created without a user number, such as if it were created under a non-accounting DMS system and then transferred to an accounting version of DMS, then the first time a rename command is given for the particular file, it will obtain the user number of the program doing the rename.

The user number provides access to the file. Any program running under the file's user number is allowed to Read, Write, or Delete (including Rename) the file. If no access keys were given when the file was created, then no other user is allowed to access the file. If any of the keys were specified, then any user providing the correct password, if any, can perform the specified operation on the file. For example, if file CAT was created by user 123 with no password and the read access given, then all users can read the file but only programs running with user number 123 can write on or delete the file.

Thus it is important to supply the correct user number so that the correct files can be accessed.

7-5 ACCOUNTING RECORDS

Accounting Records are 8-word blocks of information that summarize the usage of a particular facility within the computer system. All accounting records have within them a user number to which the usage is allocated.

Accounting records are collected in an in-core buffer and the buffer is written to disc whenever it fills. The buffer is in dynamically allocated core and the size is determined at system generation. For efficient packing on disc, this should be a multiple of 112 words.

A variety of types of accounting records are generated. These are listed below.

1) CPU Record, containing CPU execution time in milliseconds and core usage.

2) Disc Record, containing number of disc accesses and hundreds of words transferred.

3) Console TTY Record, containing number of lines transferred from/ to operator communications device.

4) Individual I/O Device Records, one for each device used by a program, containing length of time device was allocated, and number of I/O units transferred.

5) System Idle Time Records, containing the amount of CPU time that the DMS system spent in the Executive Idle loop.

The exact layout and informational content of these records is given in Appendix B of this document.

The in-core buffer is cleared of all information when a "DA" (Dump Accounting) command is given via operator communications. This should be done prior to closing down the system to ensure that all information is stored on disc.

7-6 ACCOUNTING FILES

A number of special disc files are associated with the Accounting Subsystem. These are a user number file, a user name file, and individual accounting record files to collect accounting records.

7-6.1 User Number File

The user number file, called "AC$USR" with a password of "SYST" contains all valid user numbers in a special format. The file has only public read access and only the system can write to the file. The file contains one word per user number, sequentially from the lowest user number in the system through the highest, packed 112 words per disc sector. See Figure 7-1. A zero in the position for a user indicates an invalid number, while if the number itself is stored in the correct slot, then the user number is valid.

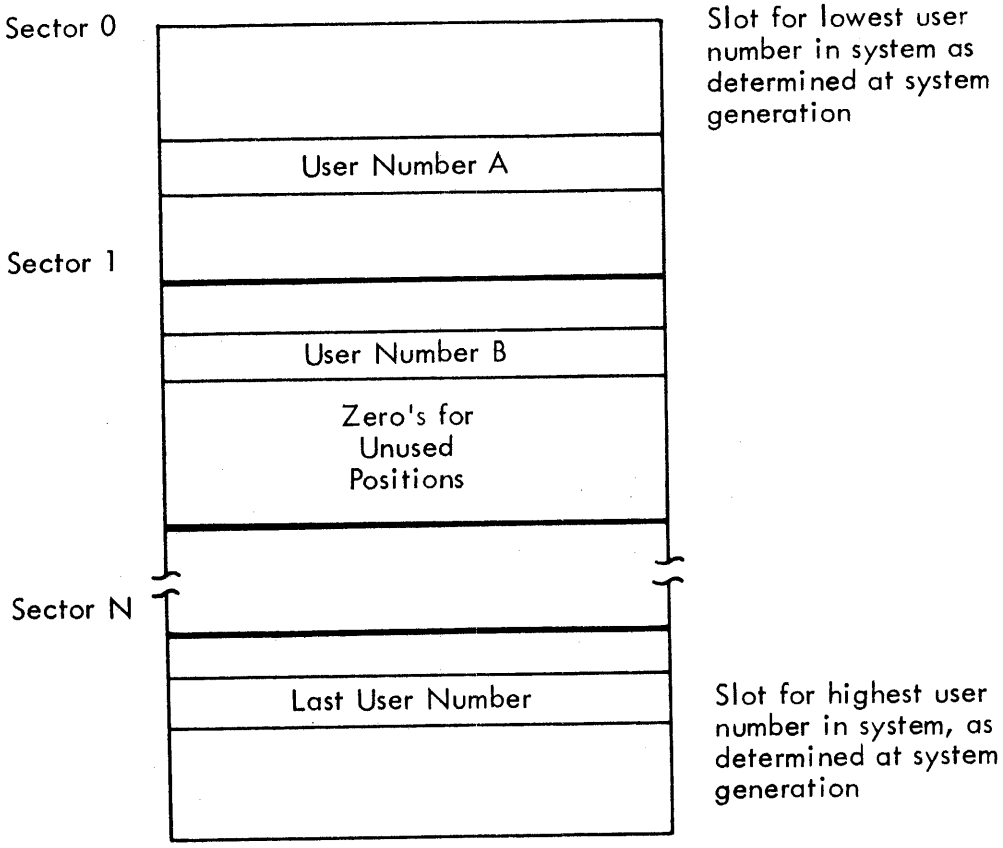


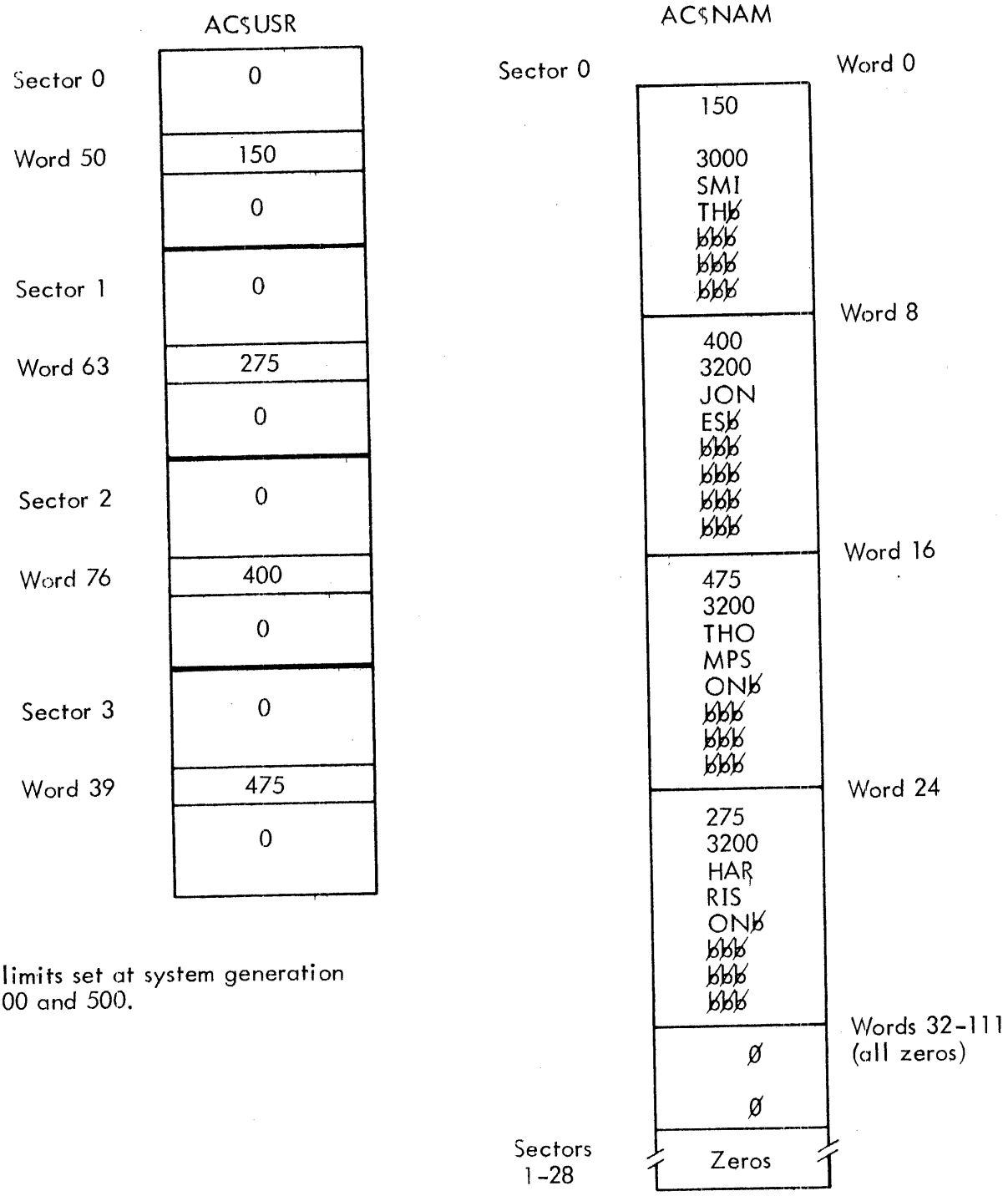
Figure 7-1. User File Layout

7-6.2 User Name File

The user name file, called "AC$NAM" with a password of "SYST" contains the user number, account number, and name of all users in the system. It is used for reports and summaries. The file has public read access but can be written only by the system. This file is organized into 8-word segments, with one segment for each user. The file is not ordered and empty slots are denoted by a 8-word block of zeros. The file is created by the system when first initialized after a system generation if not already there, and made sufficiently large enough to hold all of the users in the system.

Figure 7-2 shows an example of the two files, AC$SUR and AC$NAM when only four users are in the system, as created by the following statements:

```
AU,U150,. 3000,SMITH
AU,U400,. 3200,JONES
AU,U475,. 3200,THOMPSON
AU,U275,. 3200,HARRISON
```



User limits set at system generation are 100 and 500.

Figure 7-2. Sample AC$NAM and AC$USR Files

7-6.3 Accounting Record Files

Accounting records are accumulated on disc in a series of disc files. These are name "AC$nnn", where nnn is a decimal number from 001 through 999. The files are not necessarily in any order although the system does go sequentially when assigning numbers. The first 8-word block of all files is a date/time record whose layout is described in Appendix B, thus defining the earliest record on the file. Thus all of the files can be ordered just by looking at the first record. Files are either full (all sectors have information) or they are terminated with an end-of-file mark. Eight-word records containing all zeros are fillers and should be bypassed when processing the files.

An end-of-file record is always written following any accounting block. This is done every time an in-core buffer is written to disc. The end-of-file mark allows the system to find the end of the current file whenever the DMS system is loaded from disc. Thus in the event of a catastrophic system failure, only the information in core is lost, and all information written to disc is preserved.

When DMS is loaded from disc, the system initialize program finds the latest disc file by looking at the date/time record at the beginning of every file and then scans to the EOF mark in this file. This is where the first accounting records will be written as computing operations begin. The date/time entered by the Operator is compared with that on the disc file and if found to be earlier than the disc file, a warning message is output.

If DMS cannot create an accounting file due to a lack of sufficient available space on the accounting disc pack, it will output the message "ACCTG DISC FULL" to OPCOM. From this point until room becomes available on disc, all accounting records will be lost. The first accounting record generated after a 20 second interval will initiate another attempt to create the file. These attempts and messages will continue until a file is successfully created.

7-7 ACCOUNTING UTILITY PROGRAM

The DMS Accounting Utility Program "ACCUP" provides a convenient means of preparing simple summaries of accounting information and of maintaining the accounting record files. It is a background batch utility program which can be run at any time as needed. ACCUP reads a set of input parameter records from the job stream file, and then proceeds to execute based on this input. The control records are discussed in the following paragraphs. Thus a basic deck structure for running ACCUP would be:

```
$JOB NAME Unnn etc
$ACCUP
        control records
$EOJ
```

Of course, other job control records could be inserted in the job as required.

7-7.1 RUN Command

The RUN command causes execution of the previous input to begin and signals the end of the control statements. When processing has been completed, ACCUP will input more control records from the job stream if any are present. And end-of-file on the input job stream file, as caused by a $ control card, effectively generates a RUN command. However, ACCUP exits when processing is complete.

7-7.2 Accounting Period Designation Statements

The START and END statements provide information to define the extents of accounting record processing. The basic format is:

```
START
END     MM-DD-YY (HH:MM:SS)
```

where:

MM is the numeric month
DD is the numeric day
YY is the last two digits of the year
HH is the 24-hour clock hours value
MM is the minutes value
SS is the seconds value

The time-of-day field is optional but if present, is input as a 24-hour clock value.

If no START command is given, then the earliest accounting information present will define the starting position. If a starting date is specified, but no time, then the earliest record on or after the specified day will determine the start time.

If no END command is given then the accounting information to the end of the last file will be processed. If an ending date is specified, but no time, then all records on the specified ending date will be included.

If neither the START nor the END parameters are input, then all available accounting records will be included.

Multiple START/END combinations may be entered to provide a summary containing portions of a period of time. For instance, the following START/END pairs will summarize all activity from Noon to 1:00 p.m. for three consecutive days.

```
START   05-01-73   12:00:00
END     05-01-73   13:00:00
START   05-02-73   12:00:00
END     05-02-73   13:00:00
START   05-03-73   12:00:00
END     05-03-73   13:00:00
```

There may be at most seven pairs of START/END cards in a particular run. All pairs must be in consecutive ascending order to provide valid output.

7-7.3 BLOCK Statement

The BLOCK statement is used to specify the blocking factor to be used (if any) when SAVing accounting records as discussed in the following paragraph. Its format is:

```
BLOCK  n
```

where n is a decimal integer specifying the number of 8-word accounting records to be put in one block before being written to the SAVE file.

An error condition exists if insufficient work space is available to build a blocking buffer of the specified size.

If no BLOCK statement is entered, then the records will be output unblocked. (Blocking factor = 1).

7-7.4 SAVE Statement

The SAVE statement indicates that all records processed in the period defined by the START/END parameters are to be copied out to the specified logical file as they are processed. Its format is:

SAVE n

Where n is a decimal logical file number to which the 8-word records are output. If a BLOCK statement was entered they are blocked by the specified factor as they are output. The unused portion of the last block is filled with zeros.

The 8-word records are output unmodified, and are exactly as they appear on the accounting files.

7-7.5 DELETE Statement

The DELETE statement causes all 8-word records processed in the period defined by the START/END parameters to be removed from the accounting files. The DELETE is performed after all processing including the SAVES. The delete is accomplished for the most part by over-writing the affected records with zeros. If however, this results in a particular file having no valid data remaining, the file itself is deleted from the disc.

For example, if no START or END statements were given, then all of the files on disc will be processed and all will be deleted.

7-7.6 UTILIZATIONS Statement

This command causes the Device Utilization Summary to be output. This is a summary, by device, of how much time it was used, and how many records were processed. In addition, the amount and percentage of CPU time used for execution, and the amount and percentage of time that background was active are also output in the summary.

7-7.7 Usage Summary Statements

The three statements, USER, USER/ACCOUNT, and ACCOUNT are used to select which form of output is to be used in generating the accounting summary report. This report breaks down by user or account, as specified below, all individual charges and totals and, optionally, provides costs in conjunction with the RATES statement discussed later. Only one of the above three forms may be used. If none are input, then no accounting summary report is generated.

7-7.7.1 USER Statement

This causes the accounting summary report to be output by user only, ignoring account numbers. Optionally, user numbers may be specified on the statement as shown below.

USER n1,n2,n3-n4,...

to specify that only certain user numbers are to be output in the report. If no parameters are present on the user statement, then all users will be output. If users are specified, only they will be output. User numbers are separated by commas, except that a range of user numbers is denoted by placing a dash immediately between two user numbers without intervening blanks. For example, the command

USER 100,105-109,111

causes the report to be generated for users 100, 111 and any users 105, 106, 107, 108 or 109. The user numbers specified need not exist on the accounting files or in the system and if they do not, then no output will be produced for them.

The specification of user numbers is limited to one input record. That is, only one USER card may be present in a particular run.

7-7.7.2 USER/ACCOUNT Statement

The user/account statement causes the accounting summary report to be output by user within account number. That is, the basic sequence is by account, and within account the output is by user. No additional parameters may be specified on this statement.

7-7.7.3 ACCOUNT Statement

This command causes the accounting summary report to be grouped by account number for output. That is, all charges for a particular account are added together and output as one information block. This statement, like the user statement, can specify particular account numbers on it. The basic format is:

ACCOUNT n1,n2,n3-n4

As with the USER statement, the ACCOUNT statement can specify individual account numbers as well as ranges of numbers, when two account numbers are separated by a dash. If no account numbers are specified on the statement, then information for all accounts will be output.

7-7.8 ALPHA Statement

This statement, when used in conjunction with the USER or USER/ACCOUNT statement, causes the individual users to be stored alphabetically by the user name, instead of coming out sequentially by user number. If no user or user/account statement is entered, then the ALPHA statement is ignored.

7-7.9 LIST Statement

This statement should be used in conjunction with those listed in Paragraphs 7-7.7 and 7-7.8 to produce listings of account numbers or user numbers and names. It functions by producing a normal accounting summary report except that the chargeable information is suppressed. Its effect is to produce a listing of all user and/or accounts in the system sorted as specified by the commands described above.

7-7.10 TIMES Statement

This statement is used in conjunction with the accounting summary report described in Paragraph 7-7.8. Its format is:

TIMES d1,d2,d3,...

The effect is to cause the device allocated time (the amount of time that the device was "OPEN" to a particular user) to be included as a chargeable item. d1, d2, d3 are octal physical device numbers. For example, the statement

TIMES 21,22

indicates that device allocation time is desired to be included in the accounting summary report for devices 21 and 22. Devices 0 (disc), 1 (operator communication device) and CPU may not be specified since they are not allocatable devices.

7-7.11 RATES Statement

The RATES statement is used to assign monetary values to chargeable items for output on the accounting summary report, described in Paragraph 7-7.7. The basic format is

RATES ITEM1 CHARGE1 ITEM2 CHARGE2 ...

where items and their respective charges are paired as shown. Multiple RATES cards may be input in a particular run as desired.

The "charge" values are decimal values representing the dollar cost for a unit of the specified item. For example, the charge .001 indicates that one-tenth cent is to be charged for each unit of the particular item. Dollar signs should not be included in the rate value.

The "items" are described by a device type, intormation type, and optional device number. The format of an "item" entry is:

$$DT,(d^{\#}),\begin{matrix}T\\R\\W\end{matrix}$$

where DT is a device type code as shown in Table 7-1, $d^{\#}$ is a physical device number of the specified type, and T,R,W specifies whether the charge is to be for allocated time (T), records processed (R) or hundreds of words transferred (W) where applicable.

If the device number is included, then the charge applies only to that device. Some examples of RATES are shown below.

RATES CU .001 RT,21,T .003

This statement indicates the core is to be charged at the rate of one hundredth cent per word (100 words per cent), and that "connect time" for remote teletype number 21 costs three tenths cent per tenth second or 3 cents per second.

Table 7-1. RATES Parameters

Device Type	Device	Records Measured in units of: **	Hundreds of Words (W) valid?
DI	Disc	I/O requests	Yes
CU*	Core Usage	Words	No
TI*	CPU Time	Milliseconds	No
RT	Remote TTY	Lines	No
RC	Remote CRT	Lines	No
TR	Paper Tape Reader	Characters	No
TP	Paper Tape Punch	Characters	No
LP	Line Printer	Lines	No
CR	Card Reader	Cards	No
CP	Card Punch	Cards	No
MT	Magnetic Tape	Records	Yes
SC	Synchronous Communications	Characters	No
AD	Analog-Digital I/O	Words	No

* No T R W should be specified on CU or TI parameters as this is done by system.

** Allocated Time (T) always measured in tenths of seconds.

SECTION VIII
TERMINAL OPERATION

8-1 INPUT TERMINALS

For spooled DMS systems, any input or input/output device which is created as a spooled terminal at system generation (SYSGEN) has a special operating procedure as discussed in the following paragraphs. Batch input terminals (such as card readers) are treated differently from interactive terminals (such as CRT's and teletypes).

8-1.1 Theory of Operation

All spooled terminals are controlled by the I/O executive module of DMS which can communicate with multiple terminals, permitting several functions to be performed by each terminal device. For instance, all terminals can communicate with ACRONIM, the re-entrant editor package.

8-2 SPOOLED BACKGROUND JOBS

Spooled background jobs may be entered from any input terminal on the system. The presence of such a job is denoted by the $JOB card as the first line of the job. All input images following the $JOB statement until the $EOJ statement is encountered are written to the spool input file. When the $EOJ is read, the file just created is placed on the background spool queue on a priority basis. Jobs on this queue are processed as time permits.

8-2.1 $JOB Statement

The $JOB statement is described in detail in Section 6-2.1. The basic format is repeated below.

$JOB Name Unnn Pnnn Dnn Lnnn Innn Mnnn

where:

Unnn is the user number designation in accounting systems only.

Pnnn is the priority.

Dnn is the list output file or device.

Lnnn is the list output spool file size.

Innn is the input spool file size.

Mnnn is the background memory size.

Items of particular importance to the terminal user are priority, list output device, and input file size.

8-2.2 Priority

The priority value determines where in the background input spool queue the job is placed. The priority range is 1-254, with 1 being the highest and 254 the lowest. If no number is placed in this field on the $JOB card, the terminal's default priority is used. The number placed here must conform to the terminal's priority limits set at SYSGEN time (see Paragraph 8-5).

8-2.3 List Output Device

This field specifies the device or file to which list output is to be assigned. If a zero is entered here, all the list output will be suppressed. If a filename is entered here, then list output is assigned to that disc file. If a non-zero number is entered, then list output will be spooled out to that physical device through a spooled disc file. If no value is entered here, the output device associated with the input terminal is used and the spooling system actually enters this value on the statement for later processing by background. Output device associations are controlled by the SYSGEN procedure when the input terminal is defined.

8-2.4 Input File Size

This field is used to specify the size of the input spool file in sectors if the default size is not desired. The default size is set at System Generation.

8-2.5 Error Messages

Error messages produced by the input spool system are listed below. In the case of batch terminals, input lines are bypassed until a $EOJ is read. For interactive devices, the message "RE-ENTER SPOOL JOB" is typed after the error and no input is bypassed. All error message output is written to the associated output device for the terminal. Error messages are:

INPUT SPOOL FILE OVERFLOW:	The input spool file size is too small to contain the job being input. The job should be re-entered with a larger input file size specified. Or, user "X-OFF"ed the spooled input
INVALID PRIORITY FIELD:	The priority field is invalid in format (not in range 1-254, etc.) or does not conform to the terminal's priority limit. Re-enter the job with a valid priority.
INVALID $JOB CARD SYNTAX:	The syntax of the $JOB is not correct. Correct the $JOB statement and re-enter the job.
INVALID OUTPUT DEVICE:	The output PDN field contains a number or disc file name which is not valid. Correct the $JOB card and re-enter the job.
INVALID USER NUMBER:	The user number where required in accounting systems is not present or is not a valid user number.

8-3 ACRONIM

The re-entrant interactive editor ACRONIM can be run from any spooled input terminal under DMS. The general instructions for the various commands are described in the ACRONIM General Specification AA61681-00.

ACRONIM is active whenever the terminal is on-line. In fact, the $JOB statement for a spooled job is actually processed by ACRONIM.

Error messages from ACRONIM are output to the associated output device for the input terminal and are of the format

CD nn

where nn is the error code as listed in Table 8-1.

Note that the concept of a disc "area" is what is commonly referred to as a disc file under normal DMS usage.

8-4 FOREGROUND PROGRAMS

Foreground programs may be initiated from any terminal via the special ACRONIM command $name. For example, to initiate foreground program CAT from a terminal, the command "$CAT" should be entered.

Once the foreground program has been initiated, the terminal is no longer available for ACRONIM operation. The terminal will, however, still respond to the Program Abort keyin (see Paragraphs 8-7. 3 and 8-8. 3), which, if actuated, will cause the foreground program to be aborted. If the terminal was the operator communication device, this function is still available also.

The foreground program may communicate with the terminal by assigning the required logical files to the physical device number of the terminal. When the program is initiated, the parameter passed (loaded in the A-register at initiation) is the address of the Terminal Control Block for the terminal. (See Appendix A). The physical device number is located in bits 5-0 of the first word of this block.

When the foreground program EXITs, the terminal is again available for ACRONIM operation.

8-5 TERMINAL PRIORITIES

Each input terminal as established for it a priority limit and a variable amount of priority control specified at System Generation. There are one of three types of control available for the terminal. These are described in the following paragraphs.

8-5. 1 Default Priority

If the terminal is set only with a default priority, any priority may be input in the full range of 1-254, but if none is specified, this default is used.

8-5.2 Priority Limit

If the terminal has a priority limit, then if an attempt is made to use a priority value higher than the limit, (that is, a smaller numeric value) it will be ignored, and the default, or limit, value will be used.

8-5.3 Fixed Priority

If the terminal has a fixed priority, then programs initiated by it will always run at that priority, and any other value input will be ignored.

8-6 BATCH INPUT TERMINAL OPERATION

Two additional features are available to spooled input batch terminals. These are "unspool" mode, and "off-line" mode.

8-6.1 Non-Spooled Mode

If it is desired to run a background job in a direct, i.e., unspooled mode, without going through an intermediate disc file, it is necessary to issue a special job statement. The format is

$JOB* name Unnn

The $JOB* statement causes an entry to be placed in the background spool input queue to indicate an unspooled job. When this entry becomes active, job stream is assigned directly to the background default list output device and processing continues normally. This mode is necessary for special I/O mode transfers, such as reading binary cards, etc. The Unnn user number field is required only in accounting systems.

8-6.2 Off-Line Mode

Normally, high speed batch terminals such as the card reader are allocated to the I/O Executive and cannot be referenced directly by background or foreground programs. However, the device can be placed off-line to I/O Executive by placing a $CLOSD statement at the beginning of the data. This allows any DMS program to open the device and communicate directly with it. When the requesting program closes/deallocates the device, the I/O Executive again regains control. This operation is used to allow foreground programs to input data cards directly from the reader.

8-7 TELETYPE TERMINAL OPERATION

A number of special keys and functions are used to control interactive teletype terminals. These functions are discussed in the following paragraphs. Note that if the Operator Communication Device is a teletype, it can interact with ACRONIM just as any other teletype, and is operated as discussed below.

8-7.1 Initiating Communications

When DMS is loaded from the disc, all terminals are left in the "off-line" mode. To "turn-on" or initiate communications, the Control key "BELL" is used. When the system initializes a terminal, the message "HELLO" is output, and a carriage return line feed combination will be issued, and input may be entered. The first line input on accounting systems must be a $ON command, specifying the user number.

8-7.2 Line Cancel

If an error is made in a line being entered at a remote teletype it may be corrected by performing a "character cancel" or a "line-cancel" and re-entering the character or line. The "character cancel" is achieved by entering a "←" key. Successive "←" characters cancel previous characters enetered in the input record. If the whole record is cancelled it becomes equivalent to the "line cancel". The "line-cancel" is actuated by entering the RUBOUT key. When this key is entered, the terminal responds with a "! " and a carriage-return line-feed is issued. The line should then be re-entered.

8-7.3 Program Abort

The operator of a teletype terminal may abort the foreground program which he last initiated by use of the X-off key. The terminal will respond with the message "ABT.. " and the foreground program will be aborted with an abort code 37 (terminal user requested abort). This abort feature may also be used to cancel spooled output being transmitted to the terminal, or to cancel spooled input being transmitted from the terminal.

8-7.4 Terminating Communications

To place the terminal in the off-line state, the command $OFF is entered. This causes all storage allocated by the terminal to be released and all file pointers maintained by ACRONIM will be lost. To re-establish communication, it is necessary to use the Communication Initiation Key. The terminal is automatically placed off-line whenever the end of a file being spooled out to the terminal is reached.

8-8 ALPHANUMERIC CRT TERMINAL OPERATION

The same functions available to the teletype user are also available on the CRT, however different keys are used. These same instructions apply to an alphanumeric CRT used as the Operator Communication Device and terminal combination.

8-8.1 Initiating Communications

Alphanumeric CRT terminals are automatically "turned-on" or placed on-line to DMS when the first input line is made. That is, to initiate communications, the first line should be entered on the first line of the screen and the TRANSMIT key depressed. This will cause terminal initiation and subsequent processing of the first line. Note that in accounting systems, this first line must be a $ON image containing a valid user-number in order that subsequent input will be processed.

8-8.2 Line Cancel and Line Editing

Because the input from the CRT is not accepted until the TRANSMIT key is depressed, the line being entered may be changed, edited or erased as desired through the use of local keys on the CRT. Once the line is satisfactory, the TRANSMIT key should be depressed and the line will be processed by DMS.

8-8.3 Program Abort

The operator of a CRT terminal may abort the foreground program which he last initiated by placing a backslash (" \ ") character as the first character in an input line. The remainder of the line will be ignored. The CRT will display the message "ABT. . " and the foreground will be aborted with abort code 37. Spooled output or input to or from the CRT may also be aborted in this way.

8-8.4 Terminating Communications

As with any ACRONIM terminal, the CRT may be placed in an off-line state by entering a $OFF command. All storage and file pointers for the terminal will be lost.

Table 8-1. ACRONIM Error Codes

nn	Meaning
1	Syntax error
2	Dynamic core unavailable
3	Too many concurrently active disc areas
4	Cannot copy directly to CRT terminal
5	First output device or file name invalid
6	Second output device or file name invalid
7	Third output device or file name invalid
8	Invalid command code or no $ON has been given
9	BASIC statement number error
10	No area is in edit mode
11	Not in BASIC mode
12	Cannot do I/O to another interactive terminal
13	Cannot copy an "area" fron non-disc device
14	Initialization request must be on first input (after $ON)
15	Unexpected end-of-area read
16	Disc area overflow on destination file
17	Invalid standard scratch area name
18	More than 8 numbers on TV statement
19	Cannot change processor modes while editing

Table 8-1. ACRONIM Error Codes (Cont'd.)

nn	meaning
20	Invalid variable on "mode value" statement
21	BASIC work area "C1" not present
22	Disc area name already exists, or not enough space on disc
23	BASIC statement no. not in 1-9999 range area
24	Edit area full: update needed
25	RC area full: update needed
26	Cannot specify RC size as zero
27	No edit area currently defined
28	Edit area too small -- system error
29	Edit area not on disc
30	Insufficient disc space to create scratch area
31	Output disc area overflow on update command
32	Core allocation problems: delete edit area name then rename TP area to old edit area name
33	RUN name not same as edit area name
34	Function not allowed in BASIC mode
35	Requested disc area(s) not on disc
36	New disc area name already in use
37	Cannot rename or delete standard terminal scratch areas
38	Disc area is non-existent
39	No disc area name supplied where required
40	No list or view area defined
41	Non-BASIC record found in disc area
42	Request to list or view beyond end of disc area
43	Cannot delete "output area" disc area
44	Key (password) must be alphabetic
45	Edit requests for records past end of disc area-ignored
46	Cannot delete or change line number 0.
47	Too many (>7) arguments on CR request
48	Specified text not found on search request
49	Column number not in 1-80 range
50	Specified text cannot be found: not enough room in search field
51	Cannot edit an end-of-file record

Table 8-1. ACRONIM Error Codes (Cont'd.)

nn	Meaning
52	Requested text not found in edit record
53	No "output area" defined
54	Edit line not defined
55	Cannot insert End-of-file line in edit record mode
56	Requested word in line does not exist
57	Cannot delete across tab field
58	Non-existent register(s)
59	Register(s) not large enough for given text
60	Command only valid when ACRONIM control is from disc area
61	Non-numeric values in number register
62	Requested deconcatenated text not found
63	More than 80 characters building text
64	List values or view values out of range
65	Name not in disc area table
66	Register(s) requested are beyond maximum
67	Disc area does not contain valid $JOB card as first record
68	Invalid $JOB card parameter
69	Invalid or non-existent user number
70	Invalid output device on $JOB statement
71	Terminal already on. $ON treated on $OFF. Re-enter $ON
72	Invalid user number
73	Edit record number greater than 32767
74	User number on $ON statement has not been entered
75	Invalid password
76	Rename failure because either old area gone or new area already exists
77	File being updated does not have proper access bits. Rename TP to new name
78	Cannot delete edit area file (delete prohibited)
79	Register Setting command is invalid
80	Buffer size too large (> 999 characters), too small (< 10 characters) or from a non-interactive device.
81	Illegal PDN is first parameter on "SM" card
82	Illegal optional parameter on "SM" card
83	Transport was not specified on "SM" card
84	Non-existent disc area to be spooled
85	Invalid PDN as spool out device

SECTION IX
OPERATOR COMMUNICATIONS

9-1 OPERATOR COMMUNICATIONS FACILITIES

The Operator Communications provides the computer operator with complete, immediate control of DMS, including both foreground and background processing. The operator can obtain the attention of Operator Communications of any time by striking the slash ("/") key on whichever teletype is designated as the operator communication device. Following any output line which may be in process, or immediately in the absence of any output, DMS will respond by outputting a "line feed" character, followed by the slash entered by the operator. The operator should then enter the Operator Communications command, followed by a "carriage return" character. If the console device is a CRT alphanumeric display, then the operator should enter a slash ("/"), followed by the command and terminate by depressing the "transmit" key, on the first line of the CRT. Whenever that device finishes its current output, the command will be accepted and the first line of the CRT will be cleared.

The general format consists of a two character abbreviated alphabetic command, followed by multiple parameters as required; all are separated by a single comma, or one or more blanks. Numeric entries preceded by a decimal point (period) are considered to be decimal. Otherwise, numbers are considered to be octal.

The Operator Communications modules will process the command as soon as foreground memory becomes available for loading the required module. The valid operator communications commands are discussed in the following paragraphs, and summarized in Table 9-1.

9-1.1 Program Control Commands

These commands are used to initiate, terminate, abort, suspend, release and otherwise control the execution of programs running under DMS. When programs are initiated, they are passed an initiation parameter which is a 24-bit value placed in the program's A register when loaded. This may be specified on the initiation commands as an octal or decimal integer. In accounting systems, a user number is required when initiating any program to allow the proper account to be charged for the execution of the program. This parameter is not used in non-accounting systems.

Because DMS allows multiple copies of the same program to be running concurrently, it is often necessary to distinguish between these copies when referencing the program via Operator Communications commands. This is done by use of the "terminal" parameter, which is the physical device number of the terminal or device which initiated the program (see Section XII for further information on terminals). This parameter is required on some commands only when multiple copies of the specified program are active. The program control commands are as follows:

The "IP" (Initiate Program) command is used to initiate the specified program for execution. The format is:

IP,program,(Unnn,)priority,parameter

where: program is a 1-6 character program name;

Unnn	is the user number in accounting systems;
priority	is the execution priority of the program in the range of 1-254;
parameter	is the optional initiation parameter discussed above. If no parameter is specified here, zero is used.

The "SP" (Schedule Program) command is used to place a program on the timer schedule for periodic initiation. Until the programis removed from the timer schedule (see "TP" below) it will be initiated once every time the specified interval has elapsed, unless the previous copy is still active, in which case the initiation is ignored. The format is:

SP,program,(Unnn,)interval,priority,parameter

where:

program Unnn priority parameter	are as discussed above for Initiate Program;
interval	is the number of 120 or 100 Hz clock cycles between initiations. Thus with a 120 Hz clock, if this value is 240, the program will be initiated once every 2 seconds.

The "BP" (Begin Program) command is used to initiate a program at a specific time of day. It is initiated only once similar to the "IP" command. The format is:

BP,program,(Unnn,)time-of-day,priority,parameter

where:

program Unnn priority parameter	are as discussed above under Initiate Program;
time-of-day	is the time of day at which the program is to be initiated. The format is HH:MM:SS where HH is the hours, MM the minutes, and SS is the seconds on a 24-hour clock.

The "TP" (Terminate Program) is used to remove a program from the timer schedule. This will cancel the effect of the Schedule Program command discussed above, and also will inhibit the initiation of a program waiting for initiation from a Begin Program ("BP") command. The format is:

TP,program

where: program is a 1-6 character program name.

Program termination is accomplished via an Abort Program ("AP") key in. As discussed above, the terminal designation is used to resolve ambiguity if more than one program with the specified name is active. The format is:

AP,program,terminal

where: program is a 1-6 character program name;

where: terminal is an optional physical device number of the interactive terminal which initiated the program. The parameter "ALL" may be specified here in place of a terminal number. If "ALL" is specified, all programs of the specified name, except those associated with a terminal, will be aborted. "ALL" is a valid parameter only on the AP command and is not accepted on other OPCOM commands.

The background execution may be aborted via an Abort Background ("AB") key in, which is equivalent to an AP,BAKGND command.

Program execution may be temporarily suspended via an "HP" (Hold Program) command. This command will suspend execution until released by the operator, if active, or will set the suspend bit in the timer schedule table if the program is currently set for periodic execution (see "SP") command. The format is:

HP,program,terminal

where: program, terminal are as discussed above under Abort Program.

Background execution may be temporarily suspended via the "HB" (Hold Background) command, which is functionally equivalent to the command HP,BAKGND.

In spooled systems, execution of the Foreground Output Spooler (S. FGSP), the program which DMS uses to copy all spooled output files to their respective devices, may be suspended via the "HS" (Hold Spooler) command. The format is:

HS,terminal

where: terminal is the number of the physical device to which the output is being spooled.

This command is equivalent to the command: HP,S. FGSP,terminal.

Programs which have been suspended may be allowed to continue via the "RP" (Rlease Program) command. This command is used to continue execution whenever the Program has been suspended by using the HOLD service, or by a system or operator action. The Release command resets the suspend bit if the program was suspended on the timer schedule via an "HP" command, or changes the status of a previously suspended active program to allow it to continue. The format is:

RP,program,terminal

where: program, terminal are as discussed above under Abort Program.

Background execution may be released via the "RB" (Release Background) command, which is identical to the RP,BAKGND statement.

In spooled systems, execution of the Foreground Output Spooler (S. FGSP) as discussed above may be continued via the "RS" (Release Spooler) statement. The format is:

RS,terminal

where: terminal is as discussed above under the "HS" command.

Operation of this command is identical to that of: RP,S. FGSP,terminal.

9-1.2 Foreground Interrupt Control

Foreground interrupt control is accomplished by the following commands. The reader should consult Paragraph 4-12.2 for a discussion of the operation and control of foreground interrupts. The commands discussed here merely duplicate the services discussed in Section IV.

A foreground program may be "connected" to an interrupt via the "CP" (Connect Program) command. The format is:

CP,program,(Unnn,)interrupt,priority,parameter

where:
- program, Unnn, priority, parameter — are as discussed in the preceding paragraphs under Initiate Program.
- interrupt — is an integer constant giving the interrupt level (0-23) in bits 5-0 and the interrupt group (1-3) in bits 7-6.

A foreground program may be disconnected from a foreground interrupt via the "DP" (Disconnect Program) statement. The format is:

DP,interrupt

where: interrupt is as discussed under Connect Program above.

A foreground interrupt may be enabled (not done by Connect command) once a program has been connected to an interrupt via the "EI" (Enable Interrupt) command. The format is:

EI,interrupt

where: interrupt is as discussed under Connect Program above.

A foreground interrupt may be correspondingly disabled via the "DI" (Disable Interrupt) command. The interrupt must be connected with a program. The format is:

DI,interrupt

where: interrupt is as discussed under Connect Program above.

9-1.3 Program Assignment Commands

Two commands are available for making logical file to physical device assignments for programs. These are the "AL" (Alter Assignment) and the "AS" (Assign) commands. The "AL" command is used to alter an assignment made to a previously cataloged non-resident or resident foreground program on disc. This command does not effect any programs currently active in memory, but will be used in subsequent executions of the program.

The "AS" command is used to make an assignment for a currently active program (background or foreground) but does not effect the information stored on disc and thus does not effect subsequent executions of the program. The active program must not have the specified logical file "open" or the assignment cannot be accepted.

The format of these commands are as follows:

AL
AS ,program,logical file,physical device

where: program is a 1-6 character name.

logical file is a valid logical file number between 0 and '76.

physical device is either a valid physical device number between 0 and '77, or a disc file name.

Paragraph 6-2. 3, $ASSIGN statement, should be consulted for further information.

9-1. 4 Spooling Control Commands

Eight commands are available to control spooling functions on spooled/interactive implementations of DMS. In all of these commands, the designation "terminal" is used to indicate the physical device number of any device which is configured as a terminal in the DMS configuration (see Section XII).

The "TO" (Terminate Output) command is used to terminate the current spooled output file on the specified device. The file will be deleted unless a "KO" command (see below) has been given for the file. The format is as follows:

TO,terminal

The "KO" (Keep Output) command is used to specify that the current output spool file on the specified device is not to be deleted at the end of the output operation, but instead to be left in the system for future use. When this command is given, the name of the current spool output file is output to the Operator Console device.

If the file is later terminated via a "TO" key in, the relative position (record number) within the file will be output also. The format is as follows:

KO,terminal

The "AO" and "BO" (Advance and Backspace Output) are used to move the current position with the current output spool file on the specified device. An Advance causes the specified number of lines (records) to be skipped, and a Backspace causes the specified number of lines (records) to be repeated. If no line count is given for a Backspace function, the entire file is repeated. The format of these commands is as follows:

AO,terminal,lines

BO,terminal,lines

The "MO" (Multiple Outputs) command is used to produce multiple copies of the file currently being output on the specified device. The number of additional copies to be output, including the current one is specified and passed to the Foreground Output Spooler. The format is as follows:

MO,terminal,copies

where: copies is an integer specifying the number ot copies to be output, including the one currently being output.

The "SO" (Spool Output) command is used to place files on the spool-out queue for the specified device. The user number specification is required in accounting systems to allocate the execution charges. If the specified file is a type "spool" file, then it will be deleted at the end of the output operation. The format is as follows:

SO,filename,(Unnn,)terminal

Background job stream files may be placed on the background spool-in queue via the "IJ" (Initiate Job) command. The specified priority determines the placement of the file within the current background queue. If no priority is specified, 255 is used (see Paragraph 8-5). The format is as follows:

IJ,filename,priority

where: priority is an integer between 1 and 255.

The "BQ" (Background Queue) command is available to display the contents of the current background job spool-in queue. When this command is processed, the filename, priority, and optionally user number of each job in the queue, including the currently active background job, are output to the Operator Communications Device.

In addition to the commands listed above, the user should note that the "HS" and "RS" commands described previously in Paragraph 9-1.1 are also applicable as spooling control commands.

9-1.5 Accounting System Commands

Four commands are available in accounting systems only to control those special functions. In accounting commands, the user number is a decimal integer preceded by the letter "U" specifying the applicable user number. The "account" is any integer whose use is determined by the individual customer. The "name" is a 1-15 ASCII character string giving the user name in some manner as determined by the individual site. Refer to Section VII for additional information.

The "AU" (Add User) command is used to add new user-numbers to the system. The format is as follows:

AU,Unnn,account,name

The "RU" (Remove User) command is used to remove an existing user from the system. The format is as follows:

RU,Unnn

The "CU" (Change User) command is used to change either the "account" or "name" or both for an already existant user. The format is as follows:

CU,Unnn,account,new name

If either the "account" or "new name" are omitted, then the previous information is left intact.

The "DA" (Dump Accounting) command is used to clear the contents of the in-core accounting record buffer and dump this information to disc. Its use is recommended prior to system shutdown. Refer to Section VII for additional information.

9-1.6 System Status Commands

The following commands are used to display the operational status of programs or functions within DMS.

The "QP" (Query Program) command is used to test the status of a specified program. As discussed in Paragraph 9-1.1, a terminal number may be required to distinguish multiple copies of the program. However, if multiple copies do exist, and no terminal number is specified, then information relevant to all copies of that program is output. The format is as follows:

QP,program,terminal

For individual programs, the information output consists of the following three lines:

```
STATUS    xxxxxxxx
MEMLO       xxxxxx
MEMHI       xxxxxx
```

where: STATUS is the contents of the Program Status Word (see Table 4-2),

MEMLO is the location of the first word of the program, and

MEMHI is the location of the last word of the program.

In addition, if the program is waiting for a flag to go non-negative (status bit 22=1), the value of the flag (whose address is given in bits 15-0 of the status word) will also be output following the status as:

(FLAG) XXXXXXXX

The background status may also be displayed via a "QB" (Query Background) key in, which is equivalent to a QP,BAKGND command.

For multiple copies of a given program, when no terminal number was given, the output is of the form:

```
MEMLO       xxxxxx
MEMHI       xxxxxx

TERMINAL        xx
STATUS    xxxxxxxx

TERMINAL        xx
STATUS    xxxxxxxx
```

Where: MEMLO and MEMHI give the memory bounds on the re-entrant program, and each TERMINAL/STATUS pair gives the terminal number and program status word of each copy of the program.

The "PL" (Program List) command is used to display the contents of the Active Program List (see Section II). The output consists of the program names, terminal numbers it any, and execution priorities.

The "MM" (Memory Map) statement is used to output a memory map to a specified device (or the standard list output device if none was specified). The data output consists of all current foreground memory allocations, programs, and sizes. The format is as follows:

MM,device

where: device is a physical device on which the map is to be printed. If a "#" precedes the device number, then the map will be spooled out to the device, providing it is a spooled terminal.

9-1.7 Time and Date Control Commands

The following commands are used to test and set the system time and date. The time is always of the form HH:MM:SS where HH is hours, MM is minutes, and SS is seconds using a 24-hour clock. Dates are always of the form MM-DD-YY, where MM is month, DD is day, and YY is year. All numbers are decimal.

The date is set via the "SD" (Set Date) command. The format is as follows:

SD,date

The time is set via the "ST" (Set Time) command. The format is as follows:

ST,time

The date and time may be displayed by the commands "PD" (Print Date) and "PT" (Print Time) respectively.

9-1.8 Miscellaneous Commands

The "JC" (Job Control) command causes the current background operation to be terminated (effectively aborted) and assigns job stream to the console device. If background was checkpointed when the command was processed, then the size of background is reduced to that available.

The "MB" (Modify Background) command is used to change the background boundary to provide a background area of the specified size. The actual modification is done when the next $JOB card is processed by background. The format is as follows:

MB,size

where: size is the desired background size.

The "CD" (Clear Device) command is used to reset the specified device. The I/O handler is entered with a RESET code, and any applicable hardware is cleared and software flags reset. The format is as follows:

CD,device

where: device is any valid physical device number in the DMS configuration.

The "OT" (Operator Terminal) command is used in spooled systems only to change the operator communication device. The specified device must be an interactive terminal (TTY or CRT). If no terminal number is specified, then operator terminal control is returned to the system default operator device. The format is as follows:

OT,device

The "NP" (New Pack) command is used to indicate to DMS that the operator has mounted a new disc pack on the specified disc number. The format is as follows:

NP,pack#,disc#

where: pack# is a valid disc pack number (2-255) which has been written on a DMS pack via the CLEAR statement of the File Manager (Paragraph 6-3. 14).

disc# is a valid disc number (2-n) in the DMS configuration.

The "IR" (Initiate RJE) command is used to provide an initialization parameter to the applicable RJE system on the particular configuration. This command is not valid if no RJE system exists. Refer to Section X for an example. The format is as follows:

IR,nnnn

where: nnnn is the numeric parameter passed to the RJE system.

The "TA" (Trap Address) statement is used to set the address trap (where available) to the specified address. When this trap location is accessed, DMS outputs the following messages and program execution continues.

ATRAP... address
PROGRAM... name
I REG... data
J REG... data
K REG... data
E REG... data
A REG... data

The "DD" (Disc Dump) command is used to output the contents of one or more disc sectors to a specified device. The sectors can either be absolute (no file name supplied) or relative to the start of a specified disc file. Either one sector (single sector specified) or multiple sectors (starting and ending sector numbers separated by dash) may be output. The password field is omitted completely if the file has no password. If no device number is specified, the data is output to the standard list output device. Otherwise, the data is output to the specified device directly, or may be spooled out to it if a "#" precedes the device number. The data output consists of both octal and ASCII words. The various formats include the following:

DD,filename,password,sector1-sector2,device

DD,filename,sector1-sector2,#device

DD,sector1-sector2

DD,sector

DD,filename,sector

DD,filename,password,sector

DD,filename,password,sector,device

The "MS" (Modify Sector) command is used to modify the contents of a specified disc sector. The sector number may either be absolute or relative to the start of a specified file. The word number is the word within sector and must be between 0 and 111. The sector is read from disc and if the specified "fromdata" matches the contents of the specified word, the "todata" replaces this word and the sector is rewritten. The various formats are as follows:

MS,filename,password,sector,word,fromdata,todata
MS,filename,sector,word,fromdata,todata
MS,sector,word,fromdata,todata

The "TS" (Time-Share) command is used to specify the time-slice given to programs qual priority. It is specified as a multiple of 120 Hz clock counts. The format is:

TS,n

Where n is the number of 120 Hz clock counts to be used as the time-slice interval. ı is zero, time-slicing is turned off.

The "BS" command is used to create a permanent type file at a specified sector number ɜ specified pack. One use of this command is to create a file to flag a bad sector on a disc :k. The file created will not be moved during a pack compression operation. The file name lynamically generated of the form Z#NNNN with a password of B.S.. The format of the tement is as follows:

BS,pack#,sector

where: pack# is a valid pack ID number (1-255), and sector is the sector number which ɔad.

Table 9-1. Operator Communications Commands

Command Format	Function
IP, program, (Unnn,) priority, parameter	Initiates a foreground program at a specified priority with an optional specified parameter.
HP, program, terminal	Sets suspend bit of timer scheduler table is scheduled, and suspends execution of program if currently active.
HB	Suspends execution of background.
HS, terminal	Suspends execution of foreground output spooler (S. FGSP) when outputting to specified terminal.
RP, program, terminal	Reset suspend bit of timer scheduler table if set, and resume execution of program if active.
RB	Resumes execution of background.
RS, terminal	Resumes foreground output spooler (S. FGSP).
AP, program, terminal	Aborts execution of specified program.
AB	Aborts execution of background.
SP, program, (Unnn,)interval, priority, parameter	Schedules foreground program for periodic execution with the specified optional parameter at the specified priority.
BP, program, (Unnn,) time-of-day, priority, parameter	Schedules the foreground program for initiation at the specified time of day at the specified priority with an optional parameter.
TP, program	Terminate timer scheduling of a foreground program.
CP, program, (Unnn,) interrupt, priority, parameter	Connects a foreground program to a priority interrupt for execution at a specific priority with an optional parameter.
DP, interrupt	Disconnects a foreground program from a specific interrupt.
EI, interrupt	Enables a foreground interrupt.
DI, interrupt	Disables a foreground interrupt.
AS, program, LFN, device or file name	Makes an assignment for the specified active program. Subsequent executions of the program will not have this assignment made.
AL, program, LFN device or file name	Alters the logical device assignment at a specified foreground program. The assignment is changed on disc such that subsequent executions of that program will utilize the new assignment.

Table 9-1. Operator Communications Commands (Cont'd.)

Command Format	Function
QP, program, terminal	Queries the status of a program.
QB	Queries the status of background.
PL	Displays on the console TTY a list of active programs, their terminal numbers, if any, and priorities.
MM,device	Prints on the specified device a map of the current memory allocations.
NOTE: The following eight spooling control commands are not available in unspooled systems.	
TO, terminal	Terminates the current spool output file on the specified device.
KO, terminal	Specifies that the current file being spooled out to the specified terminal, is to be saved and not deleted when output is complete.
AO, terminal, lines	Advances the spooled output on the specified terminal to skip outputting the specified number of records.
BO, terminal, lines	Backs up the spooled output on the specified terminal the specified number of records. If no line count is given, then the file is rewound.
MO, terminal, copies	Specifies that multiple copies of the current output file are to be printed on the specified terminal.
SO, filename, (Unnn,) terminal	Causes the specified file to be spooled out to the specified terminal.
IJ, filename, priority	Causes the contents of the specified file to be queued for background execution.
BQ	Causes the contents of the background job spool-input queue to be displayed on the operator communications device.
CD, device	Clears the specified device.
NP, pack#, disc#	Indicates that a new disc pack with the specified ID number has been mounted on the specified disc#.
TA, address	Loads and enables the Address Trap. When the specified memory address is subsequently referenced, a message is output.
MB, size	Modifies the background memory boundary to provide a background area of the specified size.

Table 9-1. Operator Communications Commands (Cont'd.)

Command Format	Function
JC	Causes the current operation in background to be terminated (effectively aborted) and assigns the job stream logical file to the console teletypewriter. If background was checkpointed when the keyin was entered, then the size of background is reduced to that available.
IR,XXXX	Initialize the Remote Job Entry System (where applicable) with parameter XXXX.
OT, terminal	Causes the specified terminal (must be TTY or CRT) to become the Operator Communications Device (Physical device 2). If no terminal is specified control is returned to the default OCD.
SD, XXXXXXXX	Sets the current system date to that specified.
PD	Prints on the console devices the current date.
ST, time-of-day	Sets the current time of day to that specified.
PT	Prints the current time of day on the console device.
DD, filename, (password,) sector 1 - sector 2, device	Causes the contents of the specified sector (sector 1) or sectors (sector 1 - sector 2) which are either absolute sector numbers (no filename specified) or relative to the start of the specified file to be printed on the standard list output device (or other device if specified.)
MS, filename, (password,) sector, word, fromdata, todata	Modifies the contents of the specified word (0-111) of the specified sector, which is either absolute (no filename specified) or relative to the start of the specified filename.
BS,pack#,sector	Creates a permanent type file over the specified sector of the specified pack.
Accounting Commands: The following accounting control commands are valid in accounting systems only.	
AU, Unnn, account, name	Add user number to system.
CU, Unnn, new account, new name	Charges the name and/or account number for the specified user to that given on this command.
RU, Unnn	Removes the specified user number from the system.
DA	Dumps the current accounting information in the in-core buffer to disc.

Table 9-1. Operator Communications Commands (Cont'd.)

The command parameters are defined as follows:

Command Parameter	Format
program	One to six characters that identify a foreground program or the name "BAKGND" for background programs.
priority	A number between 1 and 377 representing a foreground program execution priority.
parameter	Any 24-bit value which is passed as a parameter to the specified program in its A-register at initiation.
interval	A single precision integer representing 120 Hz clock counts per program initiation.
time-of-day	A time of day in the format of HH:MM or HH:MM:SS where the HH are decimal hours, MM is a decimal minutes, and SS is decimal seconds as a 24-hour clock value.
interrupt	An octal number of the form "XYY" where X identifies the hardware interrupt and YY identifies the level. The interrupt must have been defined during system configuration as being available for foreground program initiation.
address	A number within the addressing range of the specific series 6000 computer.
filename	One to six ANSCII characters, the first of which must be alphabetic.
device	A physical device number in the range 0 to 77.
LFN	A logical device code in the range 0 to 76.
word	A word number within a sector in the range of 0-111.
sector	An integer representing a sector within the addressing range of the specific disc drive.
terminal	Any physical device number which represents a device which is configured as a "terminal" in a spooled/interactive configuration of DMS.
pack#	Any integer in the range of 2-377 representing a disc pack ID value.
disc#	An integer in the range of 2-n where n is the number of disc drives in the specific configuration.
Unnn	A "U" followed by the decimal user number, used to designate a user number where required.

The operator Communications programs perform validity checking on command parameters whenever possible. When an error is detected, a message is output as shown in Table 9-2. However, some error conditions can be detected only by resident DMS routines. When such an error occurs, the Operator Communications module involved is aborted and an abort message is output to the Operator Communication Device. The abort message has the form:

OPCOMn: ABT cc (1) xxxxxx

where n designates the module involved, and cc consists of an octal number that identifies the reason for which the module was aborted (see Table 9-7, Program Abort Messages).

In some cases, it may be necessary to abort an Operator Communications Module itself. This can be done by entering an abort program command for the module involved as follows:

AP,OPCOMn

The names of the various Operator Communications modules are given in Table 9-3,along with the commands which they process. For example, if the operator wishes to abort the output of a DD command, he would enter

AP,OPCOM2

which would cause termination of the output.

Table 9-2. Operator Communications Error Messages

Error Code	Meaning
OC 1	Illegal character in constant.
OC 2	Number too large.
OC 3	Terminal number required to distinguish multiple users.
OC 4	Specified terminal is non-existent.
OC 5	Program not there.
OC 6	Input line too long.
OC 7	Undefined command.
OC 8	Invalid logical file number.
OC 9	Invalid physical device number.
OC 10	File not there.
OC 11	Program not there.

Table 9-2. Operator Communications Error Messages (Cont'd.)

Error Code	Meaning
OC 12	Program of wrong type.
OC 13	Cannot change permanent assignment.
OC 14	No room in assign table.
OC 15	Device is not spooled.
OC 16	Invalid priority value.
OC 17	Time invalid.
OC 18	Terminal not in output mode.
OC 19	Invalid pack number.
OC 20	Invalid disc number.
OC 21	Pack not there.
OC 22	Invalid size field.
OC 23	Space not available.
OC 24	Missing or invalid line count value.
OC 25	$JOB not first line of file.
OC 26	Cannot assign re-entrant foreground.
OC 27	Logical file in use.
OC 28	File permanently assigned.
OC 29	Timer schedule full.
OC 30	Program already scheduled.
OC 31	Invalid interrupt designation.
OC 32	Program already connected.
OC 33	No program is connected to interrupt.
OC 34	Word number must be 0 - 111.
OC 35	"FROMDATA" does not match disc data.
OC 36	Cannot modify "EOF" sector.
OC 37	Address invalid.
OC 38	User number not present where required or not valid.
OC 39	User number already exists.
OC 40	Program not on timer schedule.
OC 41	Date invalid.

Table 9-3. Operator Communications Modules

Commands beginning with the letters:	Are processed by:
A-B	OPCOM1
C-I	OPCOM2
J-M	OPCOM3
N-R	OPCOM4
S-Z	OPCOM5

9-2 OPERATOR MESSAGES

There are four general types of operator messages generated by DMS: Physical device error, manual I/O reqiest, foreground operational error, and user program operator requests.

9-2.1 Physical Device Errors

Physical device error messages occur when the I/O system detects a hardware problem while attempting to process a request. After taking appropriate action, the operator may release the program hold condition, permitting the request to be reinitiated by the I/O supervisor. Table 9-4 defines each error code.

For format of the error messages is as follows:

XXXXXX : DEV YY ERROR ZZ

where: XXXXXX is the program name.
YY is the physical device number
ZZ is the error code.

The operator should respond to an error message of this type with one of the following commands.

RP,XXXXXX

or

AP,XXXXXX

to resume or abort program "XXXXXX" respectively.

Error code 01 does not require an operator release other than correction of the off-line condition. Program execution is automatically continued.

Table 9-4. Physical Device Error Messages

Error Code	Definition
1	Device not "ON-LINE".
2	Read error (Checksum, Parity, etc.).
3	Write error.
4	Paper Tape reader gate open.
5	Paper tape punch paper low.
6	Card reader command buffer full.
7	Card reader stacker full.
10	Line printer trouble.
11	File protected.
12	Unrecoverable disc error.

9-2. 2 Manual I/O Requests

A manual I/O request occurs when a program has initiated a request such as reposition paper tape, which cannot be performed under program control. In such cases the I/O handler will issue an operator hold message. When the operation is performed and the program hold condition released, control will be returned to the calling program with theoperation complete status. Table 9-5 defines each message.

The format of the error message is as follows:

XXXXXX : YYY ZZ

where: XXXXXX is the name of the requesting program.

YYY is the abbreviated function code definition.

ZZ is the device identification.

The operator should perform the requested function and then resume the program with the following command:

RP,XXXXXX

Table 9-5. Manual I/O Request Messages

Function Code	Definition
RPF	Reposition file.
BSF	Backspace file.
BSR	Backspace record.
BSR+1	Backspace 2 records.

9-2.3 Foreground Operational Error

A foreground operational error occurs when a foreground program cannot be initiated when it is required. The format of the associated error message is shown below.

OC :FG ERR N - XXXXXX

where: N is the error code (see Table 9-6).
XXXXXX is the program name.

Table 9-6. Foreground Error Messages

Error Code	Definition
2	Program is not present in the disc directory.
3	Too many programs are already active; insufficient table space.
4	Insufficient core available for loading a foreground module.
5	Attempt to load an invalid file type into foreground.

9-2.4 User Program Operator Requests

A user program operator request is generally given by a program whenever a specific action of the operator is desired. These messages are of the form

program : message

where: program is a six ASCII character program name and message is one to fifteen characters of text designating the activity to be performed.

For example, a background program request to mount a magnetic tape might be output as follows:

```
BAKGND: TAPE 3 DRIVE 1
```

Once the operation has been performed, the program should be released with the following command.

```
RP, program
```

9-3 PROGRAM MESSAGES

Program messages are those designated for the programmer or the person controlling the execution. These include program abort messages (see Table 9-7), file system error conditions (see Table 6-1), job control messages and normal program output.

Messages generated by DMS, including program abort messages and file system errors are output to the program's output device. This is determined as follows.

1) If the message is an operator message, such as a HOLD message, it is output to the Operator Communication Device.

2) If it is a Background message, it is written to Background logical file 6 (List Output file).

3) If the system is a spooled DMS configuration, and the program for whom the message was intended was initiated at a terminal (TTY or CRT), the message is output to that terminal.

4) If none of the above, the message is output to the Operator Communication Device.

A summary of all messages output by DMS and the standard DMS processors is included in Appendix C.

Table 9-7. Program Abort Messages

Error Codes	Definition
1	Program called "ABORT" service.
2	Program aborted via Op. Comm. key-in.
3	Instruction trap violation.
4	Memory protect violation.
5	Stall alarm interrupt.
6	Floating point overflow.
7	Invalid "BLU" operand.
10	Unassigned device code.
11	Invalid function code.

Table 9-7. Program Abort Messages (Cont'd.)

Error Codes	Function
12	Failed to open a device before using it.
13	Program attempted to input below its lower bound.
14	Program attempted to input above its upper bound.
15	Temporary storage exceeded.
16	File extents overrun.
17	Program read a job statement record.
20	System service called with invalid parameter.
21	Unable to load "BAKGND" program due to size limitations.
22	Invalid absolute load module origin.
23	Invalid open request (disc file).
24	Invalid file type for loading.
25	Invalid foreground interrupt specification.
26	Invalid program starting address.
27	Relocatable Background over-laps 32K boundary.
32	Dynamic Memory Request for more core than is in the system.
33	Invalid request to return Dynamic Core.
34	Invalid parameters on Dynamic Assign call.
35	Assign table overflow.
36	Invalid parameters on Dynamic Core Manager call.
37	Abort foreground program by terminal user request.
40	Attempt to use duplicate or nested $ADD cards.
42	No room on disc for spool file.
43	Irrecoverable disc I/O error.
44	Spool Assignment made to device which is not spooled.
45	Accounting disc pack full. Unable to create new accounting file.
46	Read Access to file prohibited for this user.
47	Write access to file prohibited for this user.
50	Background Job time - limit exceeded.

9-4 DMS SYSTEM START-UP PROCEDURE

The following procedure is used to load DMS from disc and start-up the system.

Master clear the CPU, mount the DMS master pack on the proper disc drive, and power up the system hardware. Enter the disc bootstrap, select the desired boot-in options by setting the appropriate sense and/or control switches, and load DMS. The boot-in options are described in Table 9-8 and Table 9-9

The system will output the message:

BAKGND:*** START DMS***

on the console device. If this message is not output and the display lights are flashing in a right circular motion, the master pack was not mounted on the proper drive and the procedure should be re-started.

In accounting systems, the system will next ask for the current date by outputting the message:

ENTER DATE: MM-DD-YY

The date should now be entered starting in column 1 in the form MM-DD-YY where MM is the number of the current month (01-12), DD is the current day (01-31) and YY is the last two digits of the year. If the date is not entered in the proper form, the message:

INVALID DATE: RE-ENTER

will be output and the date must again be input.

If desired, an empty line may be entered for the date. This will cause the system to use the latest date found in the accounting files on disc as the current date. If no date has been found in an accounting file and a blank line is entered for the date, the message:

DATE MUST BE ENTERED
ENTER DATE: MM-DD-YY

will be output and a date must be entered as described in the preceeding paragraph.

After properly inputting the date, the system will request the time of day with the message:

ENTER TIME: HH:MM:SS

The time must now be entered starting in column 1 in the form HH:MM:SS where HH is the current hour (00-23), MM is the current minute (00-59), and SS is the current second (00-59). The time may be entered in the form HH:MM in which case 00 is used as a value for seconds. If the time is not entered in the proper form, the message:

INVALID TIME: RE-ENTER

will be output and the time must again be input. If the current date and time are less than a previous date and time stored in an accounting record on disc, the system will output the warning message:

DATE/TIME IS LESS THAN PREVIOUS, (MM-DD-YY HH:MM:SS)

The newly entered date and time have not yet been stored on disc. If desired, they may be changed by re-booting the system at this time.

Both accounting and non-accounting systems will output the message:

BAKGND:***DMS READY***

when the initialization process is complete.

The stall alarm, program restrict, and console lock may now be enabled, if desired.

If it is desired to change the date and/or time which were entered above, the operator communication commands /SD (set date) or /ST (set time) may be used.

Examples:

NON-ACCOUNTING SYSTEM:

BAKGND:***START DMS***
BAKGND:***DMS READY***

ACCOUNTING SYSTEM:

BAKGND:***START DMS***
ENTER DATE: MM-DD-YY

10-23-74 (October 23, 1974)
ENTER TIME: HH:MM:SS

14:27:45 (2:27:45 PM.)
BAKGND:***DMS READY***

Table 9-8. System Boot Sense Switch Options

Switch	Function
1	Set-Use physical device number specified in control switches 0-5 as console PDN. Reset-Use default console PDN (specified at sysgen time).
2	Set-Halt after loading DMS from disc. This allows patches to be made to the system before system initialization is started. Restart execution at location 0 after patches are completed. Reset-Do not halt after loading DMS from disc.
3	Set-Select options as specified in control switches 6-23. These options are described in Table 9-9. Reset-Do not select options specified in control switches 6-23.
4	Set-Zero core before loading the system from disc. This will initialize semi-conductor memory which may have been powered off. Reset-Core is not zeroed before system is loaded.

Table 9-9. System Boot Control Switch Options

Switch	Function
0-5	Alternate console PDN-If sense switch 1 is set, the physical device number of the console device is selected control switches 0-5. If an invalid console PDN is selected, the system will halt. System initialization may be continued by selecting a valid console PDN and pressing halt/run.
15*	Set-Do not automatically spool out leftover output spool files. Reset-Leftover output spool files are automatically spooled out to the proper spooled device.
16*	Set-Delete all input and output spool files (S:NNNN and S#NNNN). Leftover spool files will be deleted and will not be spooled out. Operator verification on the console device is required for this option. Reset-Spool files are not deleted.
17*	Set-Delete all acronim work files (CI:/NN-C9:/NN, ED:/NN,RC:/NN and TP:/NN). Operator verification on the console device is required for this option. Reset-Acronim work files are not deleted.
18*	Set-Compress pack specified by operator. Operator entry and verification of a pack number on the console device is required for this option. Files on the specified pack are moved down such that all available space is combined into one large contiguous area. If control switches 16 and/or 17 are also set, the spool and/or Acronim files will be deleted prior to compressing the pack. The system must not be halted while the pack is being compressed. Reset-No pack compression occurs.
19*	Set-Inhibit loading of resident foreground programs. Reset-Resident foreground programs are loaded.
20*	Set-Inhibit entry of in-core-directory (MDD) entries. Reset-In-core-directory entries are entered.
*Sense switch 3 must also be set to select this option.	

SECTION X
1108 REMOTE JOB ENTRY SYSTEM

10-1 GENERAL DESCRIPTION

The 1108 Remote Job Entry System (RJE) is a software communication package designed to handle data transfers between a UNIVAC 1108 operating under the EXEC 8 Operating System and the Datacraft 6024 operating under DMS. A Universal Synchronous Interface is utilized for communication on the 6024.

The 1108 RJE system handles all message compression/decompression which is unique to the EXEC-8 1004 Symbiont processing system. The 1004 Symbiont provides for submission of 1108 batch job streams from a remote batch terminal (i. e., the Datacraft 6024 under DMS) and for processing of that job in the normal EXEC 8 environment. The 1004 Symbiont provides for two methods of output back to the terminal. One is the completed job's list output (PRINT) file, which the 1108 RJE system automatically spools out to the Datacraft 6024 List Output device. The second form of output is the "PUNCH" file which, under normal UNIVAC procedure, is designed to be punched out as normal job punched card output. However, the 1108 RJE system treats these "PUNCH" files as data files and leaves them on the Datacraft 6024 disc as permanent DMS card image data files. This provides the ability to transfer partial results of data computations made on the UNIVAC 1108 to the Datacraft 6024 for additional processing. These permanent files may be later deleted in the normal DMS manner.

Operating under the spooled DMS, 1108 RJE will accept input for transmission from the Background Batch Stream. Since DMS accepts background input from any input terminal on the Datacraft 6024, this provides the ability to transfer 1108 job streams from any remote or batch terminal on the 6024. Similarly, output from the 1108 may be accessed and displayed at any output terminal on the Datacraft 6024.

10-2 SYSTEM COMPONENTS

1108 RJE consists of three software modules, as described below.

10-2.1 Resident Device Handler

The Resident Device Handler is a regular DMS resident handler, which communicates with the 1108 over a Synchronous Interface Unit. The handler handles all data communication operations, such as message formatting, message checking, and line communication via interrupts.

10-2.2 Foreground Communication Handler

The Foreground Communication Handler (R1108) is a regular DMS non-resident foreground program which provides an extension to the resident handler. R1108 handles message decompression and transmission, input/output queue processing, and operator control facilities for the RJE system. This foreground program is resident only when needed for actual transmission/reception, and is not resident when the 1108 is not connected or only in a "probe/ack" mode, which is used to maintain communication when no data blocks are being transmitted.

10-2.3 Background Processor

The RJE Background Processor (REMOTE) is a normal background program which runs in a DMS background batch stream. REMOTE is used for message compression and input queueing. Each job processed by REMOTE is placed on a transmission queue. Jobs wait on this queue until the 1108 requests input, at which time they are sent for processing on a first in-first out basis.

10-3 COMMUNICATIONS CONTROL

The following paragraphs discuss the methods of controlling communications to the Remote 1108.

10-3.1 Establishing Communications

There are two ways to establish 1108 communications. One is to enter a job for input transmission through the background processor REMOTE. This will automatically initiate the foreground program R1108. R1108 can also be started manually via an operator communications (OPCOM) key-in to initiate program.

IP,R1108,.200

(200 is the priority of execution in this example)

Once initiated, R1108 begins probing the 1108 once every 10 seconds. This process will continue until either the 1108 responds with a message. or the operator makes a "terminate" key-in.

10-3.2 Terminating Communications

When it is desired to stop communication with the 1108, the special "terminate" key-in (12) is used on an initiate request as follows:

IP,R1108,.200,.12

If R1108 has not yet established communications when this key-in is made, it will be told to stop attempting to do so.

If files remain on the transmission queue when input is terminated, they will remain on queue until communication is re-established in the normal manner. These files will, however, be lost if the DMS system has to be re-booted.

10-3.3 Status

The status of the 1108 system can be determined by the initiate key-in with the parameter "1" as follows:

IP,R1108,.200,1

The response to this request will be a message displayed on the operator communications device of either "ACTIVE", if normal communications is proceeding, or "PROBE", if R1108 is still attempting to connect to the 1108.

10-3.4 Special Functions

Two additional special control functions can be sent to the 1108. These are "Halt" (parameter = 7) and "Halt-Go-Voice" (parameter = 8), both of which are used to indicate to the 1108 a request to temporarily suspend communications. These parameters are passed on a standard initiation call as follows:

IP,R1108,.200 (.7 / .8)

10-3.5 Special Message from 1108

If the Datacraft computer receives a "Halt-Go-Voice" function from the 1108, the following message will be typed on the operator communications device.

R1108 : HALT-GO-VOICE

Communications is then suspended until released via the OPCOM key-in:

RP,R1108

10-3.6 1108 Down

If the 1108 does not respond for 20 seconds, a "NACK" (negative knowledge) is automatically generated. If this happens again, the message

1108 DOWN??

is typed on the operator communications device and the system waits for either an 1108 response or operator input.

10-4 TRANSMITTING A JOB

Input job transmission is accomplished through the background processor REMOTE. REMOTE reads input jobs, compresses them for transmission, and calls R1108 to place the job on the send queue.

10-4.1 Initializing RJE

Whenever DMS is loaded from disc, the RJE system must be initialized with the first transmit number via the special OPCOM key-in:

IR,XXXX

where: XXXX is the first transmit number to be used.

The RJE system will automatically increment the transmit number for each job transmitted, and output the number of each job as it is sent to the 1108 on the operator console.

If no transmit number is specified when a job is input for transmission, the operator message

INIT 1108 TRNO!

will be output to the operator communications device and background suspended. The special job number should then be entered and background released.

10-4.2 Background Job Stream

The first card of an 1108 input job should be a @RUN card and the last card must be a @FIN card. Thus, the background job stream will be as follows.

```
$JOB
  .
  .
  .
$REMOTE,options
@RUN
  .
  .    (deck to be transmitted)
  .
@FIN
  .
  .
  .
$EOJ
```

10-4.3 $REMOTE Card

Various options may be placed on the $REMOTE card. The generalized form of this card is as follows.

```
          C
$REMOTE,X ,N  ,(size)
          D
          W
```

10-4.3.1 Class of Service

Various 1108 systems allow a specification of class of service or priority on the input deck. This class of service is represented by the (C, X, D, W) field on the "REMOTE card.

The field may be left empty to indicate the standard class of service (blank). This is denoted by two consecutive commas if other options are present, or no option field if there are not other options.

10-4.3.2 Listing Option

The ",N" field is optional, and if present, indicates that no listing is to be made of the deck being processed. If the field is omitted, the deck being transmitted is listed.

10-4.3.3 Size Option

The ",(size)" field specifies the size in sectors to be used in creating the compressed input file. About 8-10 cards per sector is the standard compression factor. If this field is omitted, the default size of 40 sectors is used.

10-4.4 Source Input

REMOTE reads its input from logical file (job stream). All card images, including cards containing a dollar sign in column 1 are transmitted until a @FIN card is read.

10-4.5 Error Messages

Table 10-1 lists the error messages that may be printed by REMOTE and the meaning of each. An abort is made after each error.

Table 10-1. $REMOTE Error Messages

Message	Cause	Action to be Taken
BAD DATA ON CARD	Invalid $REMOTE card format.	Correct $REMOTE card and rerun.
NO ROOM ON DISC FOR FILE	There is insufficient disc space available for the spooled input file.	Rerun later when disc space is available or create some disc space by deleting unneeded files.
FILE NAME ALREADY USED	This @TRNO number has already been used indicating an invalid number was keyed in on the IR,XXXX key-in.	Re-enter the "IR,XXXX" command to reset the TRNO to the correct number.
INVALID CLASS OF SERVICE SPECIFIED	The class of service field on the $REMOTE card contains an invalid character.	Correct $REMOTE card and rerun.

10-5 REMOTE OUTPUT

Output received from the 1108 is divided into two classes: 1) output to be spooled to the printer, and 2) output to be saved on disc. Output from an 1108 normally comes as either punch files or print files. This provides a suitable classification system. Print files are automatically spooled out, while punch images are left on disc.

10-5.1 Print Files

Output print images are automatically uncompressed and spooled out to physical device 6 (printer) through DMS spooled files. All necessary carriage controls is supplied by the decompression process.

10-5.2 Punch Files

When a punch file is received from the 1108, a disc file with public access is automatically created to receive the data. The punch file is decompressed using the usual technique and placed in the new file in symbolic form.

10-5.2.1 File Names

Dynamic files created for punch files are named according to:

RXYYYY

where: YYYY is the transmit number of the job which created the file.

X is a single character which is normally "#". However, if the resultant file name is already in use, the X character will be incremented by one to the next ASCII character until a unique file name is found.

10-5.2.2 Operator Notification

When such a punch file is received, the operator is notified via the message:

RXYYYY RECEIVED FROM 1108

SECTION XI
SYSTEM GENERATION

11-1 SCOPE OF DOCUMENTATION

Sections XI - XIV describes the procedures for generating a Disc Monitor System (DMS).

11-2 GENERAL

System generation is the procedure used to produce a functional DMS for a specific hardware configuration.

The procedure for generating a customized DMS requires and uses a functional DMS system.

Each installation is supplied with a functional DMS which is configured to the hardware supplied with the system. In addition to an operation DMS, the basic configuration modules, the disc initialization file, and the source modules for system and background data are supplied.

Generating a customized system is accomplished in three phases: Phase I consists of modifications to the source file in order to incorporate the data pertaining to the user application. Phase II produces the input modules for disc initialization. This phase requires an operational DMS. Phase III uses the System Generation System (SGS) to initialize the disc. This phase produces a functional DMS.

Phase III is also used to regenerate the system supplied with the installation. Section XIV describes the operating procedures necessary to generate a copy of the original DMS.

SECTION XII
PHASE I - SYSTEM CONFIGURATION

12-1 GENERAL

This section describes the procedure for constructing or modifying the various tables required for DMS.

12-2 PARAMETERIZATION

Prior to generating a Disc Monitor System, several system tables and constants must be constructed to define the specific hardware configuration and operating environment under which the system will function.

12-2.1 Required Definitions

The items which must be defined in the system tables are:

- The peripheral devices contained in the configuration; their interrupt levels, channel and unit numbers; and their operational status in regard to the DMS application.
- The size and type of the disc(s).
- The interrupt linkages for all system interrupts.
- The total available memory size.
- The size of the System Common (SYSCOM) memory area.
- The size of: the Master Disc Directory; of each system disc file; of system reserved areas on disc; and of dynamic spool files.
- The system service linkages for all BLU instructions.
- The maximum number of programs that will be executed concurrently.
- The maximum number of resident foreground programs.
- The maximum number of timer-scheduled programs.
- The size of the disc I/O queue.
- The default density, characters/word and mode for magnetic tape devices.
- The usage characteristics and priorities of each terminal device in the system.

- The default options, memory size and device assignments for background jobs.

The tables and constants are contained in two DMS modules: The System Data Module (SYSDAT) and Background Data Module (BGDATA). It is not necessary to modify the BGDATA module unless other than standard default assignments are desired for background logical files. The standard BGDATA is sufficient for all hardware configurations, but may be modified if desired. To simplify the initial generation of a DMS for a new configuration, SYSDAT and BGDATA are provided in assembler source form. These modules reflect the installation configuration. As new system requirements arise, these modules can be modified as necessary.

A listing of a sample SYSDAT module is shown in Appendix E. Appendix F contains a listing of a sample BGDATA module. Each module has an index of the tables contained in the module. The index is on page 1 of the listing.

12-3 PERIPHERAL DEVICES

The proper definition and configuration of the functional parameters of a peripheral device is one of the most important aspects of generating a customized Disc Monitor System. In particular, the use of a peripheral device as a terminal is the most involved and demanding task confronting the user.

The following paragraph discusses this area of application.

12-3.1 Terminals

DMS provides the ability for any peripheral device other than disc to be configured as a terminal. Any terminal that is an output device can have program output spooled to it via disc files. Any terminal that is an input device can interact with the re-entrant editor (ACRONIM) or enter spooled background jobs, providing that the proper configuration of the appropriate DMS modules is present in the system. ACRONIM and spooling can only be used on terminal devices. A device is established as a terminal by two operations:

a. Constructing a Terminal Control Block (TCB) for the device in the SYSDAT.

b. Making sure that the required device handler is assembled with Flag bit 20 reset (off).

When terminals are to be used with DMS, the I/O Executive Module must be incorporated in the system, along with the necessary support routines for the terminal and the IOEXEC.

In the case where only certain devices are to be treated as terminals, only those modules associated with the device should be assembled with Flag bit 20 reset. This approach saves substantial memory space in the resultant system. Refer to Section XIII for details of assembly options.

12-3.2 Defining Peripheral Devices

Each peripheral device that must operate under DMS requires a set of External Equivalence statements to define the device characteristics. These definitions must be inserted in the External Equivalence Table (Table B of the SYSDAT module).

In the following paragraphs a generalized three-character name is represented by XXX, e.g., in the channel/unit definition for Line Printer A the XEQV statement is:

```
XEQV LPACU, '0300
```

and the general form is

```
XEQV XXXCU, 'n
```

The required definitions are:

Channel/Unit Number XXXCU

Teletypes, CRTs and Mag Tapes	OTHER DEVICES
Device Interrupt Level:	
Interrupt Number: XXXI (Note 1)	Unitary Mask: XXXIL
Interrupt Control Instructions:	
None	Unitary Arm UAXXXI Unitary Disarm UDXXXI Unitary Enable UEXXXI Unitary Inhibit UIXXXI

NOTE 1.

The interrupt level for teletypes, alphanumeric displays, and magnetic tape drives is specified as a number from 0-71 which represents the level. The value of the number determines the group in which the level belongs, e.g., 0-23 = Group 1, 24-47 = Group 2, 48-71 = Group 3. The necessary interrupt control instructions are built by the appropriate handlers. The console teletype may operate from separate input and output interrupts. This is done by using RT0 as the name of the device, and coding the equivalence RT 0I as the input interrupt number. The output interrupt must be the next lower priority interrupt.

The operator communication device must be defined in spooled systems by the definition

```
XEQV OPCOMD, 'XX
```

where 'XX is the Physical Device Number of the terminal which is also the console teletype. The System Initialization Module will then establish the necessary control linkages.

For examples of the various formats used in the XEQV statements, refer to Table 8 of the SYSDAT Module (Appendix E).

The devices must also be defined in the Peripheral Device Coordination Table (PDCT). This table contains an entry for each device used in the system, except for discs. Regardless of the number of drives or types of disc systems in the configuration, only one entry is made in the PDCT. If the console terminal is used as an interactive terminal in spooled systems, this device will have two entries in the PDCT, one for Physical Device Number 01, and one entry for its terminal number.

Each entry in the PDCT contains the handler address, the device type code and the device busy flag address. Refer to Table 4-3 for definition of type codes. The relative position of an entry in the PDCT determines the Physical Device Number (PDN) of a device. If a PDN is to be unused, a dummy entry (of zeros) can be placed in the table.

The last entries in the PDCT consist of a table of device busy flags. There is one flag for each device controller. These flags must be defined in the External Definition Table (Table A of SYSDAT).

Note that a card reader and line printer have separate busy flags, but two magnetic tape drives operating from one controller have only a single busy flag.

For examples of entries in the PDCT refer to Table L of SYSDAT. For examples of the required XDEF statements concerning a device, refer to Tables A and L.

The Magnetic Tape Handler requires each software transport number being utilized to be associated with a specific PDN (physical device number). This association is accomplished by identifying each software transport number being utilized with a specific service routine. The relative position of the Service routine entry in the PDCT determines the physical device number with which the transport number is associated. This identification is made by specifying the additional XEQV's: MCA0TN, MCA1TN, MCA3TN, MCB0TN, etc. The standard default retry limits/positioning and special action options are described in section 5-2.3. If it is desired to use the standard default options no additional XEQV's need be specified. If it is desired to change any of the standard defaults, any or all of the following XEQV's may be specified. If any of these are specified they will be flagged on the cataloger map as being multiply defined, this is correct however. The default number of retries are specified with the XEQV's MCA0R1, MACA1R1, etc. Bits 23-16 specify the number of read retries bits 15-8 the number of write retries, and bits 7-0 the number of erase retries. The default software EOT location is specified with the XEQV's MCA0R2, MCA1R2, etc. This is specified as a value less than 256 which is the net number of forward requests allowed past hardware EOT before reaching software EOT.

The error positioning and special close action are specified with the XEQV's MCA0PM, MCA1PM, etc. The bits of this word are defined as follows:

B23-	output read error hold message
22-18-	zero
17-	output "UNLOAD TAPE" message on deallocation
16-	rewind upon deallocation
15-	wait for off-line on deallocation
14-13-	zero
12-	output EOT hold message
11-	zero
10-	position after write fault record
9-	position between erasure and faulty record
8-	psotion before erasure
7-	output write error hold message
6-5-	zero
4-	treat all write functions (2, 4, 5 and 6) as null functions
3-	output "WP VIOLATION" hold message upon receiving a write function (2, 4, 5 and 6)

2- zero
1- position after read fault record
0- position before read fault record

12-3.3 Terminal Control Blocks

A Terminal Control Block (TCB) must be established for each device which will be used for spooled I/O or as an interactive terminal, or both.

Each TCB consists of 21 words. The format and content of each word are shown in the following table.

Table 12-1. Terminal Control Block

Word	Contents
1	Contains the PDN as defined in the Peripheral Device Coordination Table.
2-9	Zero
10	Contains the address of the first word of the next TCB. If there are no more TCB's in the table, this word is set to zero.
11	Zero
12	Contains the standard word count for ASCII I/O, e.g., the TTY and Card Reader word count is 27. This permits up to 80 column I/O.
13	Zero
14	Foreground execution priority of the terminal.

Table 12-1. Terminal Control Block

Word	Contents
15	Defines the default priority of the terminal. In addition, bits 22 and 23 are used as follows: If B23=1 then the stated priority is the maximum priority for the terminal. If B22=1 then the stated priority is a fixed value and cannot be changed by the terminal.
16	Zero.
17	Defines the device mode: -1 = input only 0 = input/output +1 = output only
18	Bits 0-7 contain the lines per page for the device. Bits 8-23 define the characters per line.
19	Contains the related TCB address: For an input device, the pointer is to an output TCB. For output devices, the word is 0. For I/O devices, the pointer is to this TCB.
20	Zero.
21	Contains ACRONIM parameters: Bits 0-7 define the number of alphanumeric registers. Bits 8-15 define the number of numeric registers -1. Bits 16-23 define the number of concurrently active files. NOTE Word 1 of the first TCB in the table must have the label "TCB" and be defined in the External Definition Table.

The label "EOTCB" must follow the last TCB. EOTCB must also be defined in the External Definition Table.

Refer to Table M of SYSDAT for examples of TCB parameters.

12-3.4 Maximum Number of Programs

The maximum number of concurrently executable programs is determined by the size of the Dispatcher Control Table. The standard SYSDAT module (Table F) has provision for five programs. The table may be modified to contain up to 128 entries. To increase the size of the table, the statements

```
DAC     *+9
RDAT    8(Ø)
```

must be inserted for each program to be added. As an example, suppose it is necessary to increase the size of the Dispatcher Control Table in Appendix E. Referring to Page 11 of the listing, the last RDAT statement appears on line 409. If it is desired to increase the number of table entries by two, then the sequence

```
DAC     *+9
RDAT    8(Ø)
DAC     *+9
RDAT    8(Ø)
```

must be inserted after line 407.

To establish the maximum number of programs that can be time-scheduled concurrently, the value of the label TPROGS (Table H) must be changed to the required value.

The maximum number of resident foreground programs is determined by the contents of the word labeled RPLIST (Table G). Each resident foreground program requires a four-word block. For example, to change from four entries to six entries, lines 423 and 424 (on page 12 of the SYSDAT listing) must be changed from

```
RPLIST  DATA /4,Ø/
        RDAT 16(0)
```

to

```
RPLIST  DATA /6,Ø/
        RDAT 24(Ø)
```

12.3.5 Disc Storage Characteristics

The Disc Definition Table (Table I) defines the physical characteristics of each drive in the system and links the drive to its controller service routine.

The first word in the table is labeled D/NUMB and it is used to define the number of disc drives in the system.

NOTE

The Model 5204 and 5208 Disc systems are each considered as two drives.

Each disc drive requires a 16-word block that defines the disc parameters. The position of the block within the table defines the disc number of a drive.

The entries for each block are as follows:

Word 1	Each disc controller has associated with it a Controller Busy Flag, labeled DCxBF, where "x" is the disc controller identification letter. Each controller must be uniquely identified and have an appropriate disc service routine associated with the controller. Necessary External Equivalences should be set up in Table B. The master disc of the system must be disc controller "A". All disc drives sharing the same controller should have word 1 pointing to the same controller Busy Flag.
Word 2	Words per sector.
Word 3	Sectors per track.
Word 4	Tracks per cylinder.
Word 5	Cylinders per drive.
Word 6	Default pack number. This word contains the number of the pack currently on the drive. The word is initialized to the default pack number.
Word 7	Absolute sector number of Space Allocation Map on the drive. For all discs, other than the main one, the sector number must be 2.
Word 8	Sectors per space allocation bit. This is the number of sectors represented by one bit in the Space Allocation Map (SAM). The SAM is restricted to a maximum of 224 x 24 or 5376 bits. The number in this word should be such that (Word3* Word 4* Word5)/word8 $\leq$ 5376.
Word 9	Retry Count-Initially zero.
Word 10	Drive number. Hardware Command word drive number positioned at correct bits for disc I/O commands.
Word 11	Channel/Unit number.
Word 12	Address of Disc controller Service Routine (should be: DAC $DCxSR).
Word 13	Output address word with correct Channel/Unit number. (OAW #DCxCU).
Word 14	Output Command Word Instruction (OCW #DCxCU)
Word 15	Address of Disc Controller Queue String Pointer (should be DAC $DCxSP).
Word 16	Reserved for future expansion.

12-3.6 Disc Storage Area Definition

The special system storage areas on disc are defined via entries in the External Equivalence Table (Table B). The sizes of the areas can be changed to meet the requirements of the application. A typical disc configuration is shown in Figure 12-1.

An entry in the Master Disc Directory (MDD) requires eight words, thus each sector can contain 14 entries. There are 60 sectors allocated for the MDD in Figure 12-1, and up to 830 entries can be made.

In order to change the size of the MDD, the entries

```
XEQV  MDDSC,n          (Table B, line 215)
```

and

```
XEQV  MDDE/D,m         (Table B, line 224)
```

must be changed. The parameter, m, must be a prime number, because a hash coding technique is used to make entries in the MDD. The number of sectors required for the MDD is determined by the relationship.

$$n = \frac{m}{14} + 0.99$$

With the parameters given on lines 224 and 225 of Table B, the required value of n is

$$\frac{839}{14} + 0.99 = 60$$

The checkpoint buffer need only be large enough to hold the background memory size for the configuration. To modify the size of the checkpoint area, the statement

```
XEQV  CKPTSC,601       (Table B, line 221)
```

must be changed.

On multi-drive systems, the checkpoint area can be placed on other than the master pack. If the user does not want the background to be checkpointed, CKPTSC can be set to zero.

The Reserved System Space Sector Count, labeled SYSSC, may be changed as necessary. This parameter is used to reserve the storage for the resident DMS system. By assigning a value larger than the current system size, the user can insure that extra space will be available so that a new, large DMS resident system will fit in the area when loaded by a REPSYS operation (see Section 6-3.11). If it is not desired to allocate any extra disc storage, the equivalence SYSSC may be set to zero.

It should be noted that the reserved areas are ordered such that the first sector of one plus its sector count must yield the first sector of the next. Therefore, if any sector count is modified, all first sector values following those areas must be altered accordingly.

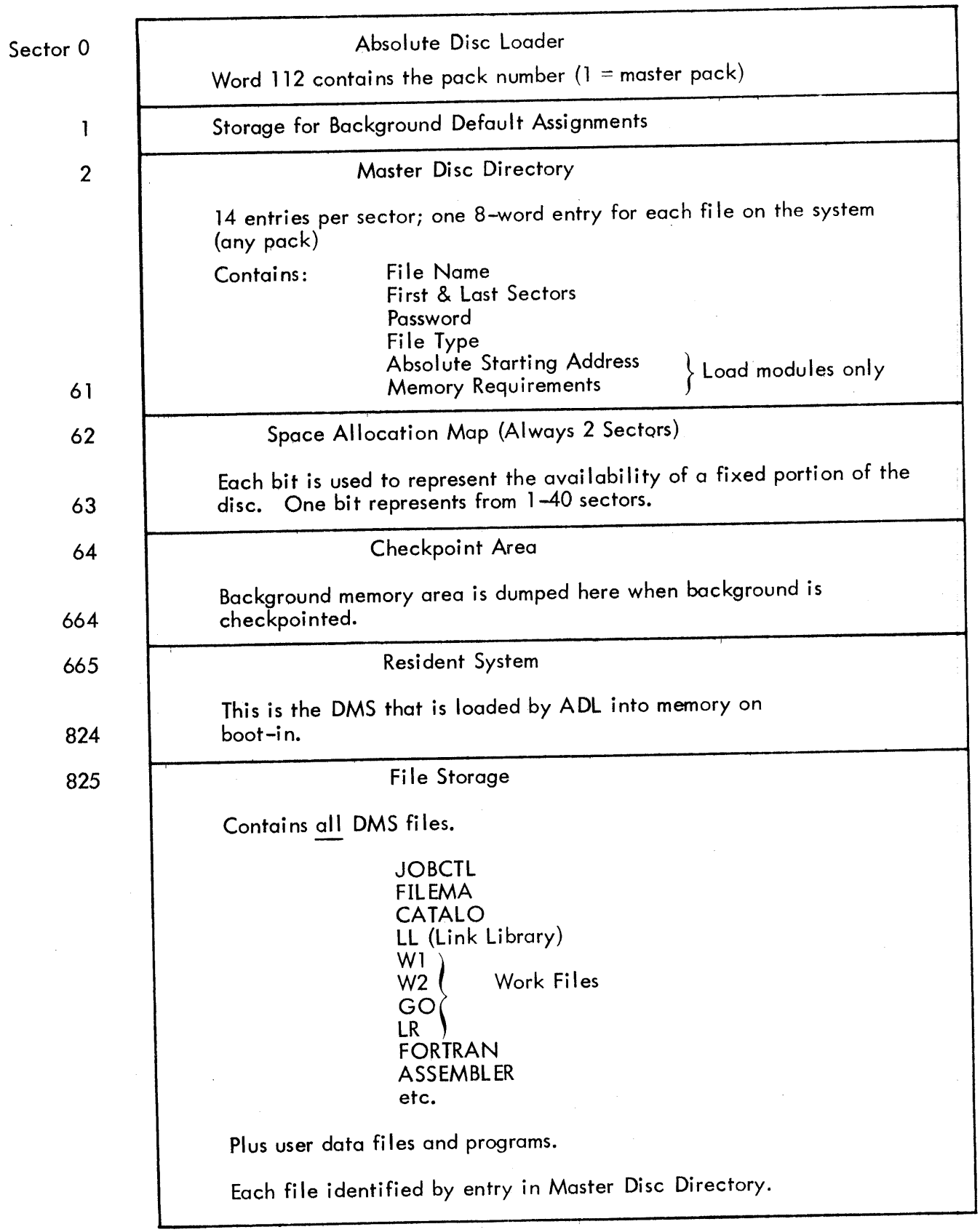


Figure 12-1. Typical Disc Layout

12-3.7 Disc I/O Queue Size

The Disc I/O Queue accommodates six entries, which should be sufficient for most applications. If a large number of programs will be concurrently executed, or several disc drives are present in the system, the size of the queue should be increased.

To increase the size of the Disc I/O Queue (Table K of SYSDAT), the statements

```
DAC     *+8
RDAT    7(0)
```

must be inserted for each additional entry required in the table.

12-3.8 Magnetic Tape Options

The Magnetic Tape Options Table (Table N) defines the density, characters/word, mode, and type of transport (7 or 9 tracks) as a function of the transport number. The format of this data is given in Appendix A.

The Magnetic Tape Density Table (Line 738) defines the command word density bits for each of the standard densities available. Each 3-bit segment of the word is used for one drive, with bits 23-21 of the word for drive Ø and bits 2-0 for drive 7. An octal digit of 4 is used to indicate an invalid density.

12-3.9 System Service Linkage

The System Service Linkages are defined by two tables in SYSDAT: Table C and Table O.

The linkage table occupies locations $0\text{-}37_8$ (Branch and Link Unrestricted calls). An unconditional branch to an appropriate system service is contained in these locations.

The identification Table (Table O) contains the label corresponding to each system service. This table is used by the Link Loader to satisfy external BLU requests, e.g., BLU $I/O.

No changes should be made in these tables unless additional services are to be included in the system.

12-3.10 Dynamic Cell Pool

The Dynamic Cell Pool contains 4-word cells used by various system routines to hold small quantities of information, e.g., core allocation threads for dynamic core blocks, and to maintain spool file queues.

The standard table size is 100 words (25 cells). The table may be decreased in size if spooling is not incorporated in the system. The size may also be reduced if the Dynamic Core Manager is not used.

To increase the size of the table the statement

```
NUMCELS EQIV 100          (Table P, line 771)
```

must be changed. The value selected must be an even multiple of 4.

12-3.11 External Interrupt Definition Blocks

Any external interrupt that will initiate a foreground program must be defined via an External Interrupt Definition Block. The format of a block is shown in Table Q of SYSDAT.

Word 1 of each additional block must define the group and level of the external interrupt. The group number must be in bits 6-7 and the level must be coded in bits 0-5.

A linkage to the block must be placed in the dedicated location associated with the group and level of the interrupt. This entry is made in Table E and provides the linkage to the block defined in Table Q. Note that the entry points to the TRM instruction -1 (Table Q, line 786).

The location EXIDBS (Table Q, line 783) must be changed to reflect the number of blocks defined in the table.

12-3.12 Spooling Parameters

There are two parameters which may need changing for a given spooled I/O configuration.

The spool pack number (SPPACK in Table B) should correspond to a pack which is always mounted on a drive, or for single drive systems, the pack which is usually mounted.

The default spool file size (SPLSIZ in Table B) defines the file size (in sectors) that will be dynamically created in the absence of explicit size parameters.

12-3.13 Blocked I/O Parameters

Two parameters (both in Table B) determine the size of the blocking buffers. The label BLOKSC determines the number of sectors to be used and must be greater than four. The label BLOKWC specifies the word count of the blocking buffer and must exactly 112 times BLOKSC.

If the parameters are changed when creating a new system or when modifying the present system, previously created block files cannot be read. Therefore, care must be exercised so that no files are lost due to parameter changes.

12-3.14 Background Default Options

The background default options should be set to establish the conditions under which most of the background processors will operate. The option word, the flag word, the list output logical file, and the lines per page parameters can be set via entries in the External Equivalence Table (Table B). Refer to lines 232 - 235 for the configuration supplied with the standard SYSDAT module.

The default values are established when the system is loaded and when a $JOB card is encountered. If a spooled system is used, the List Output assignment is taken from the $JOB card.

The parameter BGDTL is the background job default time-limit in seconds. This is used only in accounting systems.

The JOBMES parameter (Line 236) is used to control the logging of background $JOB and $EOJ statements. If this parameter is set to zero, then these statements will not be logged on the operator console. If the JOBMES value is non-zero the background $JOB and $EOJ statements will logged.

12-3.15 In-Core Directory Table

The In-Core directory table (Table R) is used to hold the Master Disc Directory entries for those files which the customer desires to have core resident directory entries. It must be large enough to hold all of the entries which could ever be typed as core-resident files at any one time. The size of the table may be changed by setting the negative number of entries (Line 800) and the table size (number of entries times eight into line 801).

12-3.16 Accounting Parameters

The Accounting Parameters are external equivalences (Table B) used to define operational parameters for accounting systems only. These include the following:

AFSC	Accounting buffer sector count
ABUF	Accounting buffer size in words (AFSC X 112)
AFSIZ	Accounting file size in sectors
ACPACK	Accounting file disc pack number
LOUSR	Lowest user-number allowed in system
HIUSR	Highest user-number allowed in system.

12-3.17 Real-Time Peripherals

For installations utilizing the RTP equipment line, and desiring to use the DMS RTP Handler, some information must be coded in SYSDAT and the RTP equipment should be set up accordingly.

The RTP Handler is configured into the System Data Module similar to most other handlers. However, a single RTP Handler is used to control up to 30 RTP "devices" and thus uses up to 30 DMS physical device numbers. Each RTP "device" on equipment chassis, as listed in Figure 12-3 is to be assigned one DMS physical device number.

Each entry in the Peripheral Device Coordinator Table (PDCT), has the following form:

```
DAC     $RTPH
'13     RTPBF
```

These two lines should be copied for each peripheral device number assigned as an RTP device. The label RTPBF should be assigned one storage location.

The RTP Handler requires the following external equivalences in Table B of SYSDAT:

RTPCU	Channel/Unit number
RTPIL	Interrupt level bit mask
RTPED#	Device Number of I/O Expander (Normally 1)
UARTPI	Unitary arm instruction
UERTPI	Unitary enable instruction
UIRTPI	Unitary inhibit instruction

Three additional tables must be provided in SYSDAT to define the RTP equipment to the handler. These are defined as follows and must be externally defined via XDEF cards in SYSDAT Table A.

Table RTPTAB is used to correlate DMS physical device numbers to RTP "device" numbers (RTP device numbers are encoded in switches on each device and are thus selectable at any time). RTPTAB is 30 words long and each entry position corresponds to the RTP device number. The table entry consists of the DMS physical device number. For example, if there are 3RTP devices in a system, numbered 2, 3, and 4 (normally device 1=I/O expander) to which DMS physical device numbers 22, 23, and 24 were assigned, RTPTAB would have the following structure:

```
RTPTAB    DATA     0        . Device 1= I/O Expander
          DATA     '22      . Device 2= DMS PDN '22
          DATA     '23      . Device 3= DMS PDN '23
          DATA     '24      . Device 4= DMS PDN '24
          RDAT     '26(0)   . Device 5-31 unassigned
```

The second table is RTPTB2, which is used to define the interrupt/non-interrupt status of each RTP card or channel. RTPTB2 is also 30 words long and each entry corresponds to the same relative position in RTPTAB (RTP Device 1 = first entry, etc.). Bits are set in this word indicating that the appropriate channel, card, on device has interrupt capabilities and is always to be programmed via interrupts. Zero positions indicate non-interrupt I/O with optional interrupt capabilities described in Section B. In the case of the 9470 (7430/20) or 9471 (7430/30 equipment chassis, a bit position in the device word is reserved for each channel or card slot. Bit 0 is used for channel 0, Bit 1 for channel 1, etc. For all other RTP devices listed in Figure 12-3, the whole word should be set to -1 if the device is to be programmed with interrupts and zero otherwise.

The third table, RTPTB3 is used to define which devices or channels contain Analog I/O Modules, which require special programming considerations. This table is also 30 words long and corresponds to RTP device numbers exactly as do RTPTAB and RTPB2. For 9410 (7480), 9430 (7471), 9440 (7460), and 9460 (7440) the entire word should be set to -1 (77777777_8) for 9484, 9485 (7435/47, 49), the bit position (Bit 0 = card slot 0, etc.) containing these cards must be set to 1.

The only other RTP Handler configuration item is to insure the dedicated interrupt location contains a BSL to $RTPIR if any interrupt devices are to be used.

12-3.18 Dispatcher Interrupt Definition

The channel/unit number of the software controlled interrupt on /1 and /3 computers must be defined by the external equivalences TTY. On /4 and /5 computers, thus must be set to zero. Thus the equivalence

```
XEQV TTY, Ø
```

is used on all DMS systems. Note that this equivalence is not in the Master System Data Module but is required for proper DMS operation.

DATACRAFT MODEL#	RTP MODEL#	DESCRIPTION
9470	7430/20	Digital Common
9471	7430/30	Universal Common
9460	7440/21	Analog Out Common
9461	7440/21	Analog Out Common
9450	7450/20	DAC Common
9440	7460/20	High Level Common
9442	7460/21	High Level Common
9441	7460/22	High Level Common
9430	7471/20	Low Level Common
9410	7480/30	Wide Range Common
9411	7480/31	Wide Range Common
9412	7480/32	Wide Range Common
9413	7480/33	Wide Range Common

Figure 12-2. RTP Devices (Equipment Chassis)

12-3.19 VERSATEC PRINTER/PLOTTER

The VERSATEC PRINTER/PLOTTER requires the configuration of several model dependent parameters. These parameters are set as External equivalencies (XEQV's) which are listed below:

PPXCU - device channel/unit ('CCUU).
PPXIL - interrupt level ('2000000 would represent level 19).
UAPPXIL - unitarily arm instruction.
UEPPXIL - unitarily enables instruction.
PPXC/L - characters per plot line (see Figure 12-3).
PPXW/L - words per print line (see Figure 12-3).

where "PPX" designates the appropriate printer/plotter device ("PPA" for systems with a single device). Figure 12-3 describes the model dependent settings-

MODEL	PPXC/L	PPXW/L
200	70	27
1100	128	45
1600	200	34
2030	232	78
2160	360	61

Figure 12-3. VERSATEC Configuration Parameters

12-4 BACKGROUND DEFAULT FILE/DEVICE ASSIGNMENTS

Default assignments must be specified in the Background File/Device Control Table within the BGDATA module (refer to Appendix F). Default assignments determine the logical file/physical device assignments established at the start of each job. The standard BGDATA contains assignments required for standard background processors. Additional assignments may be included as required for a specific system.

The total size of the table must be limited to 112 words. The final entry must be followed by a "DATA 0", which is used as a terminator by DMS.

12-5 ASSEMBLY OPTION CONFIGURATION

The remainder of the configuration process is accomplished by assembly of the various modules comprising DMS. It is essential that the correct options are set for the assembly. Normally, when a system is configured for the customer, all of the modules are assembled with the necessary options set for the configuration. However, when re-assembly of any modules is necessary, care should be taken with regard to the options.

12-5.1 SAU Option

Bit 23 of the $FLAGS word should be set for configuration having a Scientific Arithmetic Unit (SAU). This option affects the assembly of the Executive module (61607-01), and Executive Trap module (61612-01).

12-5.2 Bit Processor Option

Bit 22 of the $FLAGS word should be set for configuration having a Bit Processor. This option affects the assembly of the Executive module (61607-01) only.

12-5.3 Machine Type

Bit 21 of the $FLAGS word should be set as follows.

6024/1 and 6024/3 systems: Bit 21 = 0
6024/5 and 6024/4 systems: Bit 21 = 1

This flag affects the assembly of the SYSDAT module (61606-01).

12-5.4 Spooled/Interactive Option

Bit 20 of the $FLAGS word is used to control the spooled/interactive option in various system modules and device handlers. Normally, all relevant handlers and system modules are configured with the spooled/interactive option set. However, by setting Bit 20, the handlers can be configured to operate in the normal direct IOCS mode only.

If the spooled/interactive mode is not a system requirement, then all relevant handlers must be assembled with flag bit 20 set. Once this is done, the blocks of system space used to control the spooling and terminal interaction features may be removed since they will not be referenced. In particular, if no spooling or terminal interaction is to be done, and all relevant device handlers have been assembled with flag bit 20 on, then the following system modules should also be assembled with bit 20 set:

SYSDAT	61606-01
Executive	61607-01
Monitor Service Linkage Routines	61608-01
I/O Control Supervisor	61609-01
System Initialize	61614-01
Job Control	61619-01
Disc File Handler	61670-01

Operator Communications Modules:

61695-00
61695-01
61695-02
61695-03
61695-04
61695-05

These assemblies remove all references to the I/O Executive Module (61663-00) and the Spooling Service Routine (61669-00).

12-5.5 Remote Job Entry Option

The IR (Initialize Remote Job Entry) command at OPCOM is used to initialize the Remote Job Entry System. Thus on these systems, the Operator Communications module (61695-02) must be assembled with flag bit 19 set to include this command.

12-5.6 Accounting Option

Bit 18 of the $FLAGS word is used to control the accounting system option in the system routines and handlers. Normally all system routines are configured without accounting, but by setting bit 18, the routines can be configured into an accounting version of DMS.

All of the following modules, if present, must be assembled with flag bit 18 turned on to produce functional accounting system:

SYSDAT	61606-01
Executive	61607-01
Monitor Service Linkage Routines	61608-01
Background Data	61610-01
Executive Traps	61612-01
Program Scheduler	61613-01
System Initialize	61614-01
Job Control	61619-01

Operator Communications Modules	61695-01
	61695-02
	61695-03
	61695-05

Also, all I/O handlers present in the configuration, if accounting is to be done for them, must have flag bit 18 set when assembled. The accounting version of DMS references one additional module, the Accounting Service Routine, 61703-00.

12-5.7 50 Cycle Power Option

Bit 17 of the $FLAGS word is used to control the 50 cycle power option, in conjunction with the 120/100 Hz clock. Bit 17 should be set for 50 Hz power and reset for 60 Hz power. This effects the assembly of the program scheduler module (61613-01) only.

12-6 FINAL MODULE PREPARATION

As final input to Phase II of the SYSGEN procedure, three modules must be assembled in their configured state and left on specially named disc files. Appendix G contains a complete Job Stream list for configuring, assembling, and establishing the link modules.

12-6.1 System Linkage Module - SYSGEN

SGS is a core resident System Generation System that is configured by Datacraft to include those devices associated with a specific system. The System Generation System Linkage Module (SLMSGS) is equivalent to SYSDAT of DMS in that it contains all pertinent System Configuration Parameters. SLMSGS must be on the disc in link module form for input to Phase II.

12-6.2 SYSDAT and BGDATA

After any modifications are made, the System Data Module (61606-01) and Background Data Module (61610-01) must be assembled and left on disc in link modules files with the names SYSDAT and BGDATA respectively. The System Data Module must be assembled with the correct options.

12-6.3 DSR

DSR is the appropriate Disc Service Routine for the disc controller which controls the main disc (disc 1) on which DMS resides. This is used by Phase II to supply the Format Absolute Sector Address (FASA) routine to the Disc Initialization Routine (DISINT).

SECTION XIII
PHASE II - SYSTEM DEVELOPMENT

13-1 GENERAL

The objective of Phase II of the System Generation procedure is to create a disc initialization file on an appropriate media for the installation.

Phase II procedures require a functional DMS and the presence of the required Phase I (System Configuration) files.

The disc initialization file produced by Phase II consists of:

- o Absolute Device Loader
- o System Generation System
- o Disc Initialization Program
- o Resident DMS
- o Processors
- o Utilities

13-2 PHASE II PROCEDURES

The following paragraphs describe the steps required for file development. Appendix G contains a sample job stream. Pages G-1 through G-4 pertain to the assemblies and file creation required by Phase I procedures.

The remaining pages show the necessary steps to create the disc initialization file for a new DMS. These pages will be referenced in the following paragraphs.

13-2.1 Bootstrap Module of Device Loader

The Absolute Device Loader development consists of building a bootstrap module for one of the following devices: Absolute Magnetic Tape Loader (AML), Absolute Card Loader (ACL) or Absolute Paper Tape Loader (ATL). The appropriate module, selected according to hardware configuration, should be cataloged and dumped (in bootstrap format relative to location 20_8) as the first element of the disc initialization file. This loader is used to load SGS for Phase III of System Generation.

Page G-5 shows a job stream for creating an AML in bootstrap format.

13-2.2 SGS Absolute Load Module

The SGS absolute load module consists of the System Linkage Module (SLM) and SGS. The SLM for the system is configured to the hardware of the installation prior to delivery of the system, and unless a hardware change is made, this module should not be changed.

The job stream pages G-1 and G-2 shows the customizing performed to configure SGS to the installation equipment.

The file S54390 contains the configured source for SLMSGS. This module should be assembled and the object output should be put on a file named SLMSGS. This is part of Phase I.

During Phase II, an absolute load module of SLMSGS should be created. If magnetic tape is used for the initialization file, the module must be dumped in bootstrap format (DUMPBF). For files on cards or paper tape, the Absolute Device Loader format is required (DUMP). In either case, the module is dumped relative to location 0. Refer to page G-5 for a sample job stream.

13-2.3 SGS Disc Initialization Load Module

The Disc Initialization Program (DISINT) is built in an SGS Load Module format for execution under SGS. SYSDAT, as configured in Phase I, defines DMS parameters for DISINT. The FASA (Format Absolute Sector Address) routine from the Disc Service Routine (DSR) for the main disc must also be present to allow DISINT to build correctly formatted command words. This combination of SYSDAT, DSR, and DISINT should be cataloged, making sure that DISINT loads last. This is possible since SYSDAT references both DSR and DISINT via External Requests. The resulting module should then be dumped relative to some location greater than SGS (20000_8 should be sufficient in all cases) in DUMP format to the appropriate Disc Initialization file.

Page G-5 shows a sample job stream of this operation.

13-2.4 SGS Load Module of Resident DMS

The resident DMS module consists of the handlers and services required for operation of the system.

The configured SYSDAT module (from Phase I), the services, and the handlers are cataloged using the job stream shown on page G-3. All DMS modules except SYSDAT are placed on a file which becomes a library file for the cataloger, thus producing a DMS with only the required modules included. When setting up modules for the cataloger, the following precautions must be observed:

a) The modules must be ordered such that there is only one pass made through this "library file" by the cataloger.

b) The System Initialize routine (61614-01) must be the last module on the file, preceded immediately by the BGDATA module.

c) The service routines for the teletypes and CRT's must precede the respective handlers in order to accomplish a) above.

When the cataloging is complete, the load map produced should be checked to ensure that no modules were linked such that they would load above the System Initialize module (link map label SYINIT). If this has occurred, then the preceding precautions were not observed, and the DMS produced will not correctly function.

13-2.5 Load Modules of Operator Communications Segments

The non-resident portions of Operator Communications must be cataloged using the external definitions produced by the link cataloging of the DMS module, utilizing the RESTRT option of the Link Cataloger.

The six phases of Operator Communications should follow SYSDAT on file 15, separated by end-of-files. They are each cataloged as shown on page G-7 of the sample Job Stream, using names OPCON, OPCON1,...,OPCON5.

13-2.6 Saving Master Disc Director Entries for Multiple Discs

If the system has more than one disc, then the Master Disc Directory Entries for all packs except the master pack should be saved for reloading onto the new system. This is done by using the SAVMOD command of the File Manager as shown on page G-6 of the sample job stream. The output file must then be terminated with an EOF.

13-2.7 Creation of Dynamic Disc Modules

The SAVE operations consist of first dumping the resident DMS load module in save format, followed by the non-resident modules which are required by the new DMS.

There are nine non-resident modules necessary for a basic DMS: Job Control (JOBCTL), File Manager (FILEMA), Link Cataloger (CATALO), and the six operator communications segments: OPCOM, OPCOM1, OPCOM2, OPCOM3, OPCOM4, and OPCOM5.

Other modules can be included as required. For example, processors such as the Macro Assembler (ASSEMB) and Utility Package (UTILIT) are saved as shown on page G-8. DMS link and source module files required for System Generation are also saved.

In general, any load, link, or source module files contained on the DMS in use may be retained in this manner for inclusion in the DMS being generated.

The preceding steps, properly performed, produce a binary input file for Phase III, Disc Initialization. Phase II produces a "link map" of the configured DMS. This map should be retained for reference.

13-2.8 Deletion of Temporary Files

In the process of developing a new system, several files were established during Phase I and Phase II, e.g., SLMSGS, SYSDAT, BGDATA, DSR, and the Operator Communications segments. These files must now be deleted. A sample job stream which deletes the files is shown on page G-8.

SECTION XIV
PHASE III - DISC INITIALIZATION

14-1 GENERAL

Phase III of System Generation is used to produce an operational DMS. The system created may be a re-generation of the existing system, or a modified system incorporating new processors, services or hardware in which case Phases I and II must be executed.

If a back-up copy of the original DMS is required, or if the original DMS has been inadvertently destroyed, the procedures of Phase III, using the supplied Disc Initialization file, will produce a DMS which is identical in all respects to the original.

The Disc Initialization file (the one supplied with the system, or one created by the user) may be on cards, magnetic tape, or paper tape. The file format is shown in Figure 14-1.

14-2 PROCEDURES

This paragraph describes the Phase III operating procedures. There are four major steps in the procedure. Step 3 is iterative and continues until the initialization file is completely read.

14-2.1 Loading SGS

The first module on the Disc Initiazalition file is in bootstrap format, therefore the appropriate device bootstrap must be entered (either manually or via the FILL switch). Appendix H contains listing of the various bootstraps.

Load the Disc Initialization file on the input device and, via the bootstrap loader, read the data. When the SGS module is read, the message:

ABORT

will be typed and SGS is then in an idle loop, awaiting operator action.

14-2.2 Initial Job Control Statements

Release the system by entering the control key BELL and enter the following statements:

```
$JOB
$ASSIGN 4,X (input device assignments)
$LOADGO DISINT
```

The input device (X) is normally 7 for cards, 4 for high speed paper tape, 2 for teletype paper tape and 23 for magnetic tape.

START OF FILE

Module 1	Absolute Device Loader in Bootstrap Format: Absolute Magnetic Tape Loader (AML) Absolute Card Loader (ACL) Absolute Tape Loader (ATL) - Selects ASR reader if sense Switch 1 is set (if not, selects highspeed reader).
Module 2	SGS in Absolute Device Loader Format: AML assumes DUMPBF (single record) format for SGS. ACL assumes DUMP format for SGS. ATL assumes DUMP format for SGS.
Module 3	Disc Initialization Program in DUMP Format: DISINT in DUMP format at some location greater than SGS background low. ('20000 will always be adequate).
Modules 4-m	Master Disc Directory Entries for disc packs 2-n, 8-word entries blocked to 112-word records.
	End-of-File record.
Modules m-n	Dynamic Disc Files in DMS SAVE/RESTORE Format: DMS - Resident System JOBCTL - Job Control CATALO - Cataloger FILEMA - File Manager ASSEMB - Assembler ETC.
	End-of-File Record.

Figure 14-1. Disc Initialization File Format

These statements assign the binary input file to the appropriate physical device and load the Disc Initialization program.

14-2.3 Disc Initialization

Once loaded, DISINT types out three messages:

MOUNT SCRATCH PACK FOR DISC INITIALIZATION
SET SS2 TO DELETE OTHER PACK DICTIONARY ENTRIES
RELEASE TO INITIALIZE PACK

When the system is released, DISINT will clear the Master Disc Directory. If SS2 is not set, the entries for other packs will not be restored.

After initializing the Master Disc Directory the system will print the message:

RELEASE TO ENTER RESIDENT DMS

When the system is released, the Resident DMS module is input and transferred to the disc. The starting sector is the one specified in SYSDAT.

The system will print the message:

RELEASE TO ENTER DMS LOAD MODULES (SET SS1 FOR PAUSE)

When the system is released, DISINT transfers the next encountered dynamic load module to disc, enters the name in the Master Disc Directory and prints the name of the module on the list output device. If Sense Switch 1 is set a pause will follow each load module, allowing operator intervention for paper tape input. A system release is required to continue. During any appropriate pause in this procedure, the operator may re-assign the input file via operator communications by entering a "rub-out" ASSIGN statement to change devices.

The reading of the load modules continues until an end-of-file is detected, at which time the system prints the message:

YOUR DISC IS NOW INITIALIZED

14-2.4 Preliminary DMS Steps

Master clear the CPU, enter the disc bootstrap and load DMS.

It is necessary to create four files for the new DMS: Link Ready (LR), Work 1 (W1), Work 2 (W2), and the GO file. The following statements are required:

```
$FILEMA
CREATE,LR,0,pack no., sector count,B6,R,W,D
CREATE,W1,0,pack no., sector count,B6,R,W,D
CREATE,W2,0,pack no., sector count,6,R,W,D
CREATE,GO,GORP,pack no.,sector count,6,R,W,D
```

System generation is now complete.

SECTION XV
BACKGROUND PROCESSOR OPERATING PROCEDURES

15-1 MACRO ASSEMBLER

This subsection contains operating procedures for using the Macro Assembler in conjunction with the Series 6000 Disc Monitor System (DMS). Included are the necessary Job Control statements and examples of typical assemblies.

15-1.1 Linkage with DMS

The following items must be specified to Job Control prior to making an assembly:

Input/Output Assignments

The assembler requests Input/Output from the following logical devices which must be assigned to their respective physical devices. These can be made via the $ASSIGN Job Control statement, if the system default values are not appropriate.

Logical Device	Logical Device Number
Source Input (SI)	7
Source Output (SO)	10
Binary Output (BO)	5
List Output (LO)	6

Options

The assembler is provided with a 24-bit option word from DMS which is set by the $OPTIONS Job Control statement and indicates the following:

Bit 0 (Generally referred to as the "Scratch Option") - If set, signifies that all statements encountered on pass 1 will be written on the source output logical device; and that this device is to be rewound and used as the source input device on pass 2. This option is mainly intended for those systems having an available mass storage device.

If bit 0 is reset, then the source input device is to be used for input to both pass 1 and pass 2. Upon completion of pass 1, the source input device is to be repositioned.

NOTE: If bit 0 of the option word is not set, then the source output is not used and hence need not be assigned.

Bit 1 If set, signifies that control is to be transferred directly to pass 2 upon execution of the assembler. The underlying assumption is that the program to be assembled has been assembled previously and therefore, pass 1 need not be repeated.

If bit 1 is not set, then the assembly will proceed in the normal manner; i.e., pass 2 after pass 1.

NOTE: If bit 1 is set, then bit 0 of the option word is examined to determine the pass 2 input device.

Lines

The assembler is provided a 24-bit number from DMS which is set by the $LINES Job Control statement. This number specifies the total number of lines to be printed prior to initiating a Top-of-Form request on the list output device; also included is the heading line followed by two blank lines. Note that if continuous line output is desired (i.e., no heading lines other than the initial one), the Lines specification should be made to be a large number.

Date

The assembler is provided a three-word (nine-character) date from DMS. This date will be placed on the heading line of each list output page.

Flags

The assembler is provides a 24-bit user flag word from DMS which is set by the $FLAGS Job Control statement. This word will be examined upon encountering a SKFS or SKFZ pseudo-operation.

15-1.2 Assembly Examples

The following examples illustrate typical Job Control statements and their use in making an assembly.

```
$JOB user-job-name
$ASSIGN 5,LR,10,W1 (Note 1)
$OPTIONS. 0 (Note 2)
$LINES 55
$DATE JAN. 1,70
$REW 5 (Note 3)
$LOADGO ASSEMBLER
            IDEN MAIN PROGRAM
            TMA CAT
            .               MAIN
            .               MAIN PROGRAM
            .
            BLL $SUB1
            .
            .
            .
            END
SUB 1       XDEF SUB1,SUB1
            TJM EXIT
            .
            .
            .
            END$            SUBROUTINE
(Note 4)
```

NOTES

1. The $ASSIGN statement causes the assembler's logical devices to be assigned to the following physical devices:

Logical Device	Physical Device
Binary Output (05)	Disc (Link Ready File)
Source Output (10)	Disc (Work File #1)

2. The $OPTIONS statement indicates that the "SCRATCH OPTION" is to be set.

3. It is necessary to rewind the source output logical device (10) since this is done by the assembler upon detecting a "scratch option" bit.

4. A $WEF statement is not necessary since the assembler will write an EOF and then backspace over it upon completion of the assembly.

Note in addition that all of the parameters set by the Job Control statements above are set as defaults by the $JOB statement, and that therefore, the following example will yield similar results to the above assembly.

```
$JOB user-job-name
$ASSEMBLER
          IDEN MAIN PROGRAM
          .
          .
          .
          END$
```

15-1.3 Assembler Messages and Error Codes

The following is a list of the basic assembler error codes which appear to the immediate left of the statement on the list output device.

M Signifies that the operand contains a reference to a label that has multiple definition.

U Signifies that the operand contains a reference to a label which has not been defined.

O Indicates one of the following:

a. Operand syntax error
b. Improper index specification
c. Improper indirect specification
d. Improper literal usage
e. Improper text usage
f. Improper expression mode
g. Improper external specification

h. Operand address exceeds permissible limits
i. Improper statement ordering

C Signifies that the contents of the operation field contains an unrecognizable operation code.

L Signifies the absence of a required label in the label field.

F Signifies incorrect card format (i.e., columns 8, 13 and 14 must be blank on all statements other than comment cards).

P Signifies a MACRO parameter error:

a. Greater than 20 parameters in a single call.
b. Expansion of statement exceeded column 72 of line.
c. Improper parameter formation.

The following is a list of the basic assembler error messages which are output to the operator communications device:

ASSEMB 1 Indicates that insufficient storage is available for the assembler symbol table and that the job in progress will be aborted upon issuing a RELEASE command. (Each label encountered in the label field requires three locations of storage.)

ASSEMB 2 Indicates that the number of distinct common block names encountered has exceeded 25. (The number of common variables is unlimited.) Upon issuing a RELEASE command the job will be aborted.

ASSEMB 3 Indicates that an operand error was encountered in a SKOZ, SKOS, SKFS, or SKFZ pseudo-operation. Upon issuing a RELEASE command the job will be aborted.

ASSEMB 4 Indicates that an excess number of ESKP pseudo-operations were encountered. Upon issusing a RELEASE command the job will be aborted.

ASSEMB 5 Indicates that an insufficient number of ESKP pseudo-operations were encountered upon completion of pass 1. Upon issuing a RELEASE command the job will be aborted.

ASSEMB 6 Indicates that the expansion of a MACRO resulted in an overflow of the macro temporary storage area. (Nesting level is too deep or the total number of characters in the macro calling parameter is too great.) Upon issuing a RELEASE command the job will be aborted.

15-2 FORTRAN IV COMPILER

This section contains operating procedures for using the Series 6000 FORTRAN Compiler in conjunction with the Disc Monitor System (DMS). Included are the necessary Job Control statements and examples of typical compilation.

15-2.1 Linkage with DMS

The following items must be specified to Job Control prior to compilation.

Input/Output Assignments

The compiler requests input/output from the following logical devices which must be assigned to their respective physical devices. These may be made via the $ASSIGN Job Control statement, if the system default assignments are not appropriate.

Logical Device	Logical Device Number
SOURCE INPUT	7
BINARY OUTPUT	5
LIST OUTPUT	6

NOTE

It should be noted that either compiler output, i.e., list output or binary output, may be suppressed by assigning the logical devices to the null device (0).

Options

The compiler is provided with a 24-bit option word from DMS which is set by the $OPTIONS Job Control statement and indicates the following.

Bit 4 controls implicit declaration of DOUBLE PRECISION variables and constants. If set, all variables beginning with the letters A through H and O through Z are implicitly typed as DOUBLE PRECISION. Calls to single precision FORTRAN functions are recognized and replaced with calls to their double precision counter-parts. If bit 4 is reset, implicit typing operates normally, as explained in the FORTRAN specifications.

Bit 5 controls assembly language list output of the generated code. If bit 5 is set, an assembly language listing of the generated code is output to the list output device interspersed with normal compiler list output. If bit 5 is reset, only the normal compiler listing is output to the list output device.

Bit 8 controls the map output to the list output device at the end of each FORTRAN module. If bit 8 is set, the map will be suppressed. If bit 8 is reset, the map will be output.

Bit 9 controls the generation of code for upper memory map. If the program to be compiled is to run above 32K in memory, then bit 9 should be set to generate correct upper map code. If the program will never run in an upper memory area, then bit 9 may be reset to produce less code.

Lines

The compiler is provided a 24-bit word which specifies the total number of lines to be printed prior to initiating a Top-of-Form request. Also included is the heading

line followed by one blank line. Note that if continuous list output is desired (i.e., no heading lines or Top-of-Forms other than the initial one) the lines specification should be made a large number.

Date

The compiler is provided a three-word (nine-character) date from DMS which is set by the $DATE Job Control statement which will be placed on the heading line of each list output page.

15-2.2 Compiler Examples

The following is a typical compile and execute Job Stream under DMS.

```
$JOB ...
$ASSIGN 5,LR
$OPTIONS .
$LINES 55
$DATE 1/1/71
$REW 5
$LOADGO FORTRAN
C FORTRAN MAIN PROGRAM
                .
                .
                .
  END
C FORTRAN SUBPROGRAM
                .
                .
                .
  END$
$CATGO
$EOJ
```

Note that through use of system defaults, the same compile and execute could be done with the following job stream.

```
$JOB
$FORTRAN
C FORTRAN MAIN PROGRAM
                .
                .
                .
  END
C FORTRAN SUBPROGRAM
                .
                .
                .
  END$
$CATGO
$EOJ
```

15-2.3 FORTRAN Diagnostic Messages

The Series 6000 FORTRAN system includes an extensive set of compile-time and run-time diagnostic messages. Compile-time diagnostics include in the message a complete phrase which normally will indicate to the programmer the exact nature of the error as well as the point in the source statement where the error was discovered. The run-time diagnostics do not produce a verbose message, but attempt to pinpoint the location in the source program where the error occurred.

Diagnostic messages will appear on the list output file in-line with the source statements. In most cases, the diagnostic message will appear on the line immediately following the questionable statement. Although this is not always possible due to the one-pass nature of the compiler, the "snapshot" feature should pinpoint the location of the questionable text.

The format of a compile-time diagnostic message is:

ERROR XX YYYYYY message,

where: XX is an octal number corresponding to the type of error, and YYYYYY is the last six characters encountered within the statement when the decrepancy was discovered.

The message may be up to 30 characters in length and generally indicates the exact nature of the error. Table 15-1 is a list of all compile-time diagnostics and a description of all conditions which might cause the error.

Table 15-1. Compile-Time Diagnostic Messages

Error Number	Message	Cause
0	DATA POOL OVERFLOW	The entire data storage area available to the compiler is filled. This includes symbol table and all other table storage combined, since table storage is allocated dynamically. This is the only error condition which prevents compilation from continuing.
1	INVALID OPERATOR	a) Two or more consecutive operators. b) A special character not recognized as an operator. c) Using .NOT. as a leading operator following an arithmetic or relational operator. d) An attempt to shift or rotate by a value that is not an integer constant whose absolute value is less than 24.
2	INVALID CONSTANT	a) Too many digits in a constant. b) Magnitude of a real or double precision constant is out of range. c) An octal constant contains an "8" or "9". d) A constant appears where an identifier is expected.

Table 15-1. Compile-Time Diagnostic Messages (Cont'd.)

Error Number	Message	Cause
3	INVALID SYNTAX	The compiler has accepted the statement as a legitimate FORTRAN statement, but the construction of some particular element does not conform to the rules of the language. In particular, this message is caused by: a) Incorrectly formed exponent in a real or double precision constant. b) Unrecognizable logical operator. c) Identifier with more than six characters. d) Invalid construction of a statement function. e) Invalid use of "=" (equal sign). f) Invalid construction of a DO or ASSIGN statement. g) Two consecutive relational operators not separated by a logical operator. h) Invalid construction of an argument list.
4	MISSING OPERAND	An identifier or constant does not exist in the text at a point where expected.
5	RETURN STATEMENT	a) A RETURN statement has been recognized in a main program. b) A subprogram contains no RETURN statement.
6	INVALID STATEMENT NUMBER	a) Incorrectly constructed statement number. b) A statement number contains more than five numeric characters. c) A reference exists to an undefined statement number. These errors are discovered immediately following the output of the statement number map.
7	DATA IN COMMON	The variable list of a DATA statement contains an item which has been allocated common storage and the program is not a BLOCK DATA subprogram.
10	SUBSCRIPT USAGE	a) An array is declared using variable subscripts which are not dummies (i.e., do not appear as arguments of a subroutine or FUNCTION statement). b) An array element is accessed whose number of subscripts do not match the declared number of subscripts for that array.
11	INVALID STATEMENT	The statement is not recognized as a legitimate FORTRAN statement.

Table 15-1. Compile-Time Diagnostic Messages (Cont'd.)

Error Number	Message	Cause
12	PARENTHESIS	a) A left or right parenthesis does not occur where expected. b) The number of left and right parentheses within an expression do not agree in number.
13	MIXED MODES	Invalid mixing of item modes (data types) within a) An expression. b) An assignment statement. c) A DATA statement.
14	INVALID DELIMITER	a) An invalid special character has caused the processing of a statement to terminate. b) The "/" character does not appear where expected in a DATA or COMMON statement. c) In an expression, an invalid character follows an array or SUBPROGRAM identifier. (Only "," and "(" are acceptable.)
15	INVALID DATA	a) A DATA statement variable list does not contain the same number of items as its corresponding constant list. b) A DATA statement in a BLOCK DATA subprogram is attempting to initialize data into a non-COMMON variable. c) An H-specification is too large to fit into its corresponding variable. d) An H-specification that is preceded by a repeat-count is contained on more than one card by use of a continuation line.
16	MULTIPLE DEFINITION	More than one statement has the same statement number.
17	INVALID ITEM USAGE	a) An identifier previously defined as a variable subroutine or array issued in a context that requires a different item usage. b) A constant is used where an identifier is expected.
20	INVALID LOGICAL IF	A logical IF statement is used as the executable statement of a logical IF statement.
21	MISSING STATEMENT NO.	A FORMAT statement is unlabeled.
22	INVALID H SPECIFICATION	An H-specification is longer than the remaining characters in the statement.

Table 15-1. Compile-Time Diagnostic Messages (Cont'd.)

Error Number	Message	Cause
23	INVALID BLOCK DATA PROGRAM	An executable statement has been detected in a BLOCK DATA subprogram.
24	INVALID COMMON USAGE	Multiply defined common variables.
25	INVALID MODE	a) Logical operation attempted using other than logical or integer typed variables. b) Arithmetic operation attempted on logical variables.
26	FUNCTION NEEDS ARGUMENTS	A FUNCTION statement has been encountered that does not specify any arguments.
27	EQUIVALENCE	a) Two items are equivalenced, both of which have been allocated to a COMMON block. b) An equivalence group demands the extension of a COMMON block in a negative direction. c) Invalid construction of an EQUIVALENCE statement. d) Contradicting equivalence groups.
30	STATEMENT ORDER	a) Specification statement follows DATA, statement function or executable statement. b) DATA statement follows statement function or executable statement. c) Statement function follows an executable statement. d) FUNCTION, SUBROUTINE or BLOCK DATA statement is not the first statement of a program.
31	NO PATH TO HERE	An executable statement which is not labeled with a statement number immediately follows with an arithmetic IF or a GO TO statement.
32	INVALID DO	a) The terminal statement of a range of a DO is a GO TO, Arithmetic IF, RETURN, PAUSE, STOP or DO statement. b) Illegally nested DO statements. c) An error in an implied DO within an I/O list.

15-2.4 Run-Time Diagnostics

If an error condition occurs during execution of a FORTRAN program or any program that calls functions from the FORTRAN library, a message and/or an abort may occur. The format of the error message is:

FER xx yyyyyy or
SAU xx yyyyyy

where: xx is a two-digit error number, and

yyyyyy is an octal address which is the error address or the return address of the routine in which the error occurred.

Table 15-2 is a list of all run-time diagnostics produced by FORTRAN and the FORTRAN support libraries. A description of the result, if no abort occurs, is also given.

Table 15-2. Run-Time Diagnostics

Error	Meaning	Result	Flag
SAU/FER 01	SQUARE ROOT OF X -- X <0	0.0	OA
SAU/FER 02	OVERFLOW DURING FIX	FSP OR FSN	OA
SAU/FER 03	DIVISION BY ZERO	FSP	OA
SAU/FER 04	UNDERFLOW DURING A/S/M/D	0.0	UA
SAU/FER 05	OVERFLOW DURING A/S/M/D	FSP OR FSN	OA
FER 06	SIN/COS OF X -- X >10**7	0.0	OA
FER 07	ATAN2(0.0'0.0)	0.0	OA
FER 08	LOG(X) OR DLOG(X) -- X .LE. 0	0.0	OA
FER 09	UNDERFLOW DURING EXP(X)	0.0	UA
FER 10	OVERFLOW DURING EXP(X)	FSP	OA
FER 11	UNDERFLOW DURING X**Y	0.0	UA
FER 12	OVERFLOW DURING X**Y	FSP OR FSN	OA
FER 13	X**0.0 -- X = 0.0 OR X<0.0	1.0	OA
FER 14	0.0**Y -- Y <0.0	FSP	OA
FER 15	X**Y -- X >0.0 AND Y .NE. 0.0	0.0	OA
FER 41	FORMAT ERROR DURING OUTPUT	---	--
FER 42	FORMAT ERROR DURING INPUT	---	IO
FER 43	ILLEGAL CHARACTER DURING INPUT	---	IO
FER 44	UNDERFLOW DURING INPUT CONVERSION	0.0	UA

Table 15-2. Run-Time Diagnostics (Cont'd)

Error	Meaning	Result	Flag
FER 45	OVERFLOW DURING INPUT CONVERSION	FSP OR FSN	OA
FER 46	MORE THAN 10 BUFFER IN/OUT FILES	---	IO
FER 47	MORE THAN 10 RANDOM ACCESS FILES	---	IO
FER 48	I/O ON AN UNDEFINED RANDOM FILE	---	IO
FER 49	INVALID RANDOM RECORD NUMBER	---	IO

NOTES:

1. Error message or abort is controlled as follows:

Flag	Error Message		Error Abort	
	YES	NO	YES	NO
OA	OM	NOM	OA	NOA
UA	UM	NUM	UA	NUA
IO	ALWAYS		IOA	NIOA

Control is thru catalog "TYPE" specifications as described above. Error messages do not cause HOLDs.

IOA and NIOA may be specified as AFER and HFER respectively on older cataloger versions.

2. FSP = '37777777 for integer results

= '37777777, '37777577 for all SAU and Double precision non-SAU results

= '37777777, '00000177 for Real non-SAU results.

FSN = '40000001 for integer results

= '40000000, '00000577 for all SAU and Double precision non-SAU results.

= '40000001, '00000177 for Read non-SAU results.

0.0 = '00000000 for integer results

= '00000000, '00000201 for all non-integer results.

3. SAU/FER indicates an error output as follows:

   SAU XX for SAU systems

   FER XX for non-SAU systems

4. FER 41 always results in an error message and is never aborted.

5. Other than 43 thru 45, I/O errors result in the I/O operation in error being terminated, if the abort was suppressed.

6. FER 43, if not aborted, will ignore the invalid character.

15-3 LIBRARY FILE EDITOR

15-3.1 Use

The Library File Editor is activated in the background area via the Job Control Statement $EDITLF. Once activated, the Editor requests commands from the Job Control logical device (0). Commands must start in column one and be terminated by a blank. If a name is used, it must appear as six consecutive characters following the blank.

The library file ('12) and link ready file ('15) are rewound by the Editor for each command. Logical file '10 must be assigned to a work file for proper operation of the ORDER command. The Editor may be used with blocked or unblocked files.

NOTE

When the Add, Replace, or Order commands are executed, the end-of-file is rewritten at the end of the library file. The execution of these commands must not be interrupted by the operator.

ADD name

Each module to be added must be externally defined at assembly or compilation time. If "name" is absent, all modules on the link ready file (15) are added to the end of the library file ('12). The link ready file must be terminated by an end-of-file. If "name" is present, the link ready file is scanned and the module specified is added to the library file. Any external definitions within a module may be used as the referenced name.

DELETE name

The library module whose name is specified by the parameter is deleted from the library file ('12). Any external definition within a module containing multiple definitions may be referenced to delete the module.

REPLACE name

Replace is a combination of Delete and Add. Each module to be added must be externally defined at assembly or compilation time. The modules on the library file ('12) that have external definitions the same as those being added from the link ready file ('15) are deleted.

If "name" is absent, all modules on the link ready file ('15) are added to the end of the library file ('12). The link ready file must be terminated by an end-of-file. If "name" is present, the link ready file is scanned and the module specified is added to the end of the library file. Any external definitions within a module can be used as its name.

RENAME name1,name2

The library file (12) is scanned and the external definition specified as name1 is replaced by name2. Any external definition within a module can be used as name1.

ORDER

The modules on the library file ('12) are ordered, that is, modules are arranged such that all external requests precede the module being requested. This allows the Link Cataloger to satisfy all external requests from the library file in one pass.

Deleted modules are eliminated from the library file. File '10 is rewound by the Order command and should be assigned to a temporary work file.

LIST

This command outputs to the listout file (6) a cross reference of the modules on the library file ('12) and continues with an alphabetic sort of the external definitions with their corresponding module number. The format of the cross reference is as follows:

```
*XXX  JJJJJJ  KKKKKK  YYY  LLLLLL  UNDEF
      MMMMMM
```

where: * indicates (if present) that the module is out of order (i.e., the module requests an external definition which was previously defined on the library file).

XXX is the sequence number of the module.

JJJJJJ is the first external definition in module XXX.

KKKKKK is the first external request in the module XXX.

YYY is the module number where KKKKKK is defined.

LLLLLL is the second external request in the module XXX.

UNDEF if present, indicates that the request LLLLLL is undefined on the library file.

MMMMMM is the second external definition in module XXX.

The following example illustrates the list output format.

```
  1     AAAAAA          BBBBBB    UNDEF
  2     DELETED
  3     CCCCCC
* 4     EEEEEE          FFFFFF    5       CCCCCC  3
        DDDDDD
* 5     FFFFFF          AAAAAA    1

AAAAAA  1
CCCCCC  3
DDDDDD  4
EEEEEE  4
FFFFFF  5
```

Note that module number two has been deleted causing module number one to have an undefined request. Note that module number four is out of order because request CCCCCC is defined in module number three.

A listing of the same library file after an Order command is given below.

```
1       EEEEEE      FFFFFF      3     CCCCCC     2
        DDDDDD
2       CCCCCC
3       FFFFFF      AAAAAA      4
4       AAAAAA      BBBBBB      UNDEF

AAAAAA  4
CCCCCC  2
DDDDDD  1
EEEEEE  1
FFFFFF  3
```

15-3.2 Library File Editor Error Message Codes

During the editing process, checking is performed to ensure a valid library file. If an error condition is detected, a message will be output to the list output logical device (6) and the editing aborted. The error message codes are listed in Table 15-3.

The format of the error message is as follows.

ELF XX MOD YYY AAAAAA

where: XX is a two decimal digit error code. (Definitions of the error codes are given in the table below.

YYY is a three decimal digit module number in which the error occurred.

AAAAAA is the last encountered external definition. This will be blank if the definition cannot be determined.

15-4 UTILITY PACKAGE

The Series 6000 Utility Package consists of a Main Processor that provides access to three subprocessors: Card List, Source Update, and Binary Update. The Card List Subprocessor lists source input statements and maintains control over horizontal and vertical formatting. Other miscellaneous functions, such as sequencing and variable card code conversion may also be performed.

The Source Update Subprocessor performs insertions, deletions, and copying of source language programs (FORTRAN or assembly language). All manipulations are performed in terms of End-of-File and source program statement numbers which may be found on assembly or compiler listings.

Table 15-3. Library File Editor Error Messages

Number	Meaning
1	There is insufficient background area to build the external definition and external request tables.
2	Invalid control statement.
3	An external definition is missing from the beginning of a module being added to the library file.
4	An attempt is being made to add a module which duplicates a name on the library file.
5	The specified name is not on the link ready file.
6	The word count was not complete when a binary input record was requested.
7	The modules being added will not fit on the library file. To increase the size of the library file refer to ESTAB (6-3. 3) and DMAP (6-3. 8). Once this has been done, the current ADD or REPLACE should be repeated for all modules which did not fit previously. Note that it is possible that an ORDER command may create sufficient space on the library file for new entries.
8	A name was not specified in the control statement.
9	The specified name is not on the library file.
10	An attempt is being made to add modules which exceed the 1000 module input.
11	An invalid loader code was encountered on the input file.
12	A source program error was encountered on the input file.
13	An end-of-file was encountered at an improper position on the library file.
14	An end-of-file was encountered at an improper position on the link ready file.
15	A checksum error was encountered on an input record from the library file.
16	A checksum error was encountered on an input record from the link ready file.
17	An end-of-file is present on the start of the library file, (i. e., no program is on the library file).
18	An end-of-file is present on the start of the link ready file, (i. e., no program is on the link ready file).
19	The work file for the ORDER statement (LFN '16) is not large enough.

The Binary Update Subprocessor performs insertions, deletions, and copying of binary link modules as created by the Assembler or FORTRAN compiler. All manipulations are performed in terms of End-of-Files, END codes, END$ codes, and subprogram names.

15-4.1 Linkage with DMS

The following items must be specified to Job Control prior to using the Utility Package.

Input/Output Assignments

The Utility Package requests Input/Output from the following logical devices which must be assigned to their respective physical devices via the $ASSIGN Job Control Statement.

Processor	Logical Files Used	Logical File Number
MAIN	COMMAND INPUT	0
CARD LIST	SOURCE INPUT	7
	LIST OUTPUT	6
SOURCE UPDATE	COMMAND INPUT	0
	SOURCE INPUT	7
	SOURCE OUTPUT	10
	LIST OUTPUT	6
BINARY UPDATE	COMMAND INPUT	0
	BINARY INPUT	4
	AUXILIARY BINARY INPUT	2
	BINARY OUTPUT	5

Lines

The Card List and Source Update Subprocessors are provided a 24-bit number from DMS which is set by the $LINES Job Control statement. This number specifies the total number of lines to be printed on the list output device prior to initiating a Top-of-Form request. If a heading was specified, then the heading line plus two blank lines will be deducted from the line count specification. Note that if continuous list output is desired (i. e., no heading lines or Top-of-Forms other than the initial one), the Lines specification should be made to be a large number.

Date

The Card List and Source Update Subprocessors are provided a three-word (nine-character) date from DMS which is set by the $DATE Job Control statement. This date will be placed on the heading line of each list output page.

$UTILITY

Refer to document AA61515 (Utility Package, General Specification) for processor command definitions.

15-4.2 Error Message Definition

The Utility Package error messages conform to the following format.

UTL XX

where: XX is a two-digit error code. The message is output to the system operator communications device and a system "hold" condition ensues.

A list of the various error codes, their meaning, and corresponding action to be taken is provided below.

Error Code	Meaning	Action to be Taken
01	Invalid command identifies character	Correct statement and issue "release"
02	Invalid command	Correct statement and issue "release"
03	Invalid command parameter	Correct statement and issue "release"
04	Invalid parameter delimiter	Correct statement and issue "release"
05	Invalid sequence specification (parameter is too large, small, or specifies a conflict in sequence)	Correct statement and issue "release"
06	Invalid ANSCII equivalent of BCD specification in CDCODE command	Correct statement and issue "release"
07	Checksum error on B/I device	If "release" issued, a record will be backspaced on B/I
08	Checksum error on A-B/I	If "release" issued, a record will be backspaced
09	An END$ record was encountered on the binary input device	Prepare I/O on A-B/I device and issue "release"
10	An END$ record was encountered on the auxiliary binary input device	Prepare I/P on A-B/I device and issue "release"
11	The number of ANSCII and BCD equivalents has exceeded 64	Issue "release". No more specifications may be made

15-5 DEBUG

15-5.1 Background Debug

Background Debug provides the capability of controlled execution of a background program. This is accomplished by providing Debug as a link module file which can be cataloged along with the program to be tested. The procedure for this is:

```
$ASSIGN 5=LR,15=LR
$REW 5
$INCLUDE DEBUG
$..INCLUDE,ASSEMBLE,OR COMPILE"PROGRAM"
$CATALOG
...
```

When the cataloged program is loaded for execution, Debug provides operator control at the console typewriter. The user's program bias may be acquired through the Debug command OPB. With this address, the user may set breakpoints within his program for controlled execution.

Debug commands are accepted from the typewriter and processed until an EXIT command is received. Invalid control statements are ignored and the message "ICS" is typed. Valid commands are described in Table 15-4.

Table 15-4. Debug Commands

Command	Description
OPB	Format: OPB Causes the program bias (i.e., the first address of the user program) to be typed as PB=XXXXXX.
SRA	Format: SRA XXXXXX Permits addressing relative to listing address references. Any address reference followed by "R" adds the relative address bias to the address.
I	Format: I X,d1,d2,... or I XR,... Indicates that the octal data constants (d) are to be input starting at octal address X. If d is a memory reference instruction, dR adds the relative address to the address portion of the instruction.
O	Format: O X,Y or O XR,YR Indicates that the contents of octal memory locations X through Y are to be output in octal, one word per line. A blank may be substituted for the comma and the Y parameter is optional.
OD	Format: OD X,Y or OD XR,YR Indicates that the contents of octal memory locations X through Y are to be output to the list output logical device. The output format is eight octal words per line preceded by the address of the first word. A blank may be substituted for the comma.

Table 15-4. Debug Commands (Cont'd.)

Command	Description
OA	Format: OA X,Y or OA XR,YR Indicates that the contents of octal memory locations X through Y are to be output in ANSCII format. (Characters are assumed to be packed 3 characters per word.) The output line is limited to 24 word (72 characters).
SB	Format: SB X or SB XR The contents of octal address X are saved and then replaced by a BSL instruction to the Debug routine. When the BSL instruction is executed, the contents of all registers are saved and the address X preceded by the identifier "B" is output. A maximum of eight "breaks" are permitted.
RB	Causes all previously set "breakpoints" to be reset to their contents.
IB	Initializes the "breakpoint" routine. This command should be issued prior to using the "SB" and "RB" commands.
C	Format: C X or C XR If the optional parameter X is absent, registers saved from the last break are restored and the instruction that belongs in the break address is executed (not restored) and the background process is continued at the last encountered "break" plus one. Since the instruction associated with a break address is executed 'out-of-line', a breakpoint should never be set at any address containing a linkage instruction (i.e., BLL, BLU, BSL, etc.). If the parameter X is present, registers are restored and control is transferred directly to octal address X.
RC	Format: RC X or RC XR If the optional parameter X is absent, the last encountered breakpoint is reset, registers are restored, and control transferred to the address associated with the last break. If the parameter X is present, the above is performed except that control is transferred to octal address X.
OR	Format: OR r Causes the specified register(s) (saved from the last "break") to be output in an octal format. The r specification may be I, J, K, E, A or C or any combination thereof. If a register specification is absent, all registers will be output.
X=	Format: X=d Causes the specified octal data d to replace the current contents of register X (may be I, J, K, E, A or C). Refer to commands "C", "RC" and "SB".)
Z	Format: Z X,Y or Z XR,YR Causes the contents of octal memory locations X through Y to be set to zero. A blank may be substituted for the comma.
S	Format: S X,Y,d,m or S XR,YR,d,m Causes the contents of octal memory address X through Y to be searched for the following condition: (Memory .AND. m) d. Each time the condition is true, the associated memory address and its contents are output. If a true condition is not detected, no action will take place.
EXIT	The command is necessary only to non-resident Debug processors and causes an exit to the operating system.

Debug Messages are:

ICS	"INVALID CONTROL STATEMENT"	The statement is ignored.
BSO	"Break Stack Overflow"	More than 8 "SB" commands have been given. The statement is ignored.

15-5.2 Foreground Debug

The same DEBUG program may also be cataloged with foreground programs to provide a foreground debugging capability. This also provides a method to control debug from devices other than the operator's console. In particular, DEBUG can be cataloged with a foreground program and run from a remote TTY or CRT terminal. The procedure for cataloging DEBUG into a foreground program is as follows.

```
$ASSIGN 5,LR,15,LR
$REW 5
$INCLUDE DEBUG
$...INCLUDE,ASSEMBLE,OR COMPILE "PROGRAM"
$CATALOG
NAME=xxxxxx
TYPE=FG
ASSIGN 1,YY  (where YY is the device through which communications
              with DEBUG is to be made)
```

Once the program has been cataloged, it may be initiated as any foreground program. Once started, DEBUG provides input and output communications through the device to which logical file 01 is assigned. The same commands and messages apply as in background.

15-6 CROSS REFERENCE

This paragraph contains the operational procedures for using the Series 6000 Cross Reference program in conjunction with the Series 6000 Disc Monitor System (DMS). Included are the necessary Job Control statements and examples of typical jobs. Detailed information is contained in document number AA61514, Cross Reference, General Specification.

The following paragraphs describe the information that must be specified to Job Control before using the Cross Reference program.

15-6.1 Input/Output Assignments

The Cross Reference program requests input/output from the following logical devices which must be assigned to their respective physical devices.

Logical File	Logical Device Number
SOURCE INPUT (SI)	7
LIST OUTPUT (LO)	6

The job control statement for making input/output assignments is as follows.

$ASSIGN L_i, P_i,...,L_n, P_n

This statement assigns a logical device number L_i to physical device number P_i, etc.

15-6.2 Lines

The Cross Reference program is provided a 24-bit word that specifies the total number of lines to be printed on the list output device prior to initiating a Top-of-Form request. Included is the heading line followed by two blank lines. Note that if continuous list output is desired (i.e., no heading lines other than the initial one), the lines specification should be made a large number. The job control statement for specifying lines is as follows.

$LINES N

Where: N is a decimal number.

15-6.3 Date

The Cross Reference program is provided a three-word (nine-character) date that will be placed on the heading line of each list output page. The job control statement for specifying a date is as follows.

$DATE XXXXXXXXX

where: XXXXXXXXX is a nine-character date.

15-6.4 Cross Reference Examples

Example 1 illustrates typical Job Control statements and their use in running a Cross Reference with input from job stream file; where job stream is assigned to the card reader.

Example 1:

```
$JOB USER-JOB-NAME
$LINES 55
$LOADGO XREF  (Note 1)
                    IDEN        MAIN PROGRAM
                    TMA         CAT
                    .
                    .
                    .
                    END
                    XDEF        SUB1,SUB1
SUB1                TMA         DOG
                    .
                    .
                    .
                    END$  (Note 2)
$EOJ
```

NOTES

1. An absolute load module of the Cross Reference program will be loaded from the processor file on encountering the $LOADGO statement.

2. The Cross Reference program terminates on encountering the assembler statement END$.

Example 2 illustrates a Cross Reference following an assembly wherein the work file W1 was used for pass two input.

Example 2:

```
$ASSIGN 7,W1
$REW 7
$LOADGO XREF
$EOJ
```

NOTE

Only single module programs can be cross referenced this way because the assembler rewinds the work file for each module.

15-6.5 Error Messages

The following message will be output to the system operator communications device if insufficient storage is available.

TABLE OVERFLO, RELEASE TO PRINT TABLE

A system "hold" condition will ensue until operator action is taken.

NOTE:

Each label requires three locations of storage plus one location for each operand reference to a label.

15-7 FORGO

FORGO is a diagnostic FORTRAN compiler and library. It consists of a compiler program which compiles FORTRAN IV source code directly into memory, and an execution module which contains all the library routines for the executing program.

15-7.1 Linkage with DMS

Several items must be provided by the user to use FORGO under DMS. The system date is used by FORGO in the heading for each page. The JOB name is also used in these headings. These may be specified by the user, or the user may elect to use the system default values.

The lines per page obtained from the system is also used by FORGO to format the listing of the source program, and to format the source program output (if it is output to LFN 6). The user may wish to specify a value for lines or he may use the system default.

In addition to any logical files that the user may reference in his source program, there are two logical files that must be properly assigned before FORGO can run. These files are LFN 7, which is used for Source/Input, and LFN 6, which is used for List/Output. Note that when referencing FORTRAN unit numbers from FORGO, that the unit numbers do not always correspond to the same logical file numbers. The correspondence is given below:

FORTRAN Unit Number (Decimal)	System Logical File Number (Octal)	
	Input	Output
< 0	Illegal	Illegal
0	Used for Reread or Decode	Used for Encode
1	Illegal	23
2	2	Illegal
3	Illegal	3
4	1	1
5	7	Illegal
6	Illegal	6
7	7	23
8-12	Illegal	Illegal
13-18	15-22	15-22
> 18	Illegal	Illegal

15-7.2 Execution Job Stream

In order to execute a FORGO program, the above requirements must be met, and then the following job stream should be entered.

```
$FORGO
      . . .
      source program
      . . .
$CATGO
      . . .
      data from LFN 7
      . . .
```

Note that all appropriate assignments must be made prior to the $FORGO statement, and may not be made between the program and the $CATGO statement.

15-7.3 FORGO Error Comments

All of the FORGO error comments appear on LFN 6 along with any List/Output from the program. The error comments are self-explanatory, and contain an indication as to where the program error occurred. A complete list of the error comments is available in the FORGO General Specification and Users Manual (AA61657-00).

15-8 SNOBOL

SNOBOL is a string manipulation language. It is used to do high-level string manipulation and pattern-matching. The SNOBOL program is a combination compiler and interpreter. The source program is "compiled" directly into core and then is executed interpretively. The SNOBOL program therefore contains all routines necessary for complete program execution.

15-8.1 Linkage with DMS

Several items must be provided by the user to use SNOBOL under DMS. The system date is used by SNOBOL for the user function DATE. If this function is not used, or if the user desires to use the system default, then the user need not specify a date to the system.

The lines per page obtained from the system is also used by SNOBOL to format the listing of the source program, and to format the source program output (if it is output to LFN 6). The user may wish to specify a value for lines or he may use the system default.

In addition to any logical files that the user may reference in his source program, there are two logical files that must be properly assigned before SNOBOL can run. These files are LFN 7, which is used for Source/Input, and LFN 6, which is used for List/Output. In addition, if the SNOBOL user uses the "output-associated" name PUNCH, then he must provide an assignment for LFN 23, or he must redefine PUNCH.

15-8.2 Execution Job Stream

In order to execute a SNOBOL program, the above requirements must be met, and then the following Job Stream should be entered:

```
$SNOBOL
        . . .
        source program
        . . .
END
        . . .
        data from LFN 7
        . . .
```

15-8.3 SNOBOL Error Comments

All of the SNOBOL error comments appear on LFN 6 along with any List/Output from the program. The error comments are self-explanatory, and contain an indication as to where the program error occurred. A complete list of the error comments is available in the SNOBOL IV Programming Language Manual, described in the SNOBOL IV General Specification, (AA61659-00).

15-9 RPG II

RPG II is a high level business programming language. The RPG II system consists of a compiler that produces link ready code, and a set of library subroutines to perform some of the functions of RPG II.

15-9.1 Linkage with DMS

Several items must be provided by the user to use RPG II under DMS. The system date is used by RPG II in the heading for each page in the source listing. The date is also used in the reserved field name UDATE. The date may be specified by the user, or the user may elect to use the system default value.

The system time is also used by RPG II to set the value of the reserved field name UTIME. The user should ensure that the system time is correct or set to the desired time, if the user references any of the reserved time fields.

The lines per page obtained from the system is used by RPG II to format the listing of the source program. The user may wish to specify a value for lines or he may use the system default.

The RPG II compiler also responds to two options. Option 5 causes the RPG II compiler to produce an object code listing of the source program as it is being compiled. Option 9 is used to determine the processing of DEBUG commands. When Option 9 is reset, all DEBUG commands are treated as comments. When Option 9 is set, then all DEBUG commands are compiled for execution. Note, that Option 9 need not be on during execution for DEBUG output to occur, however, it must have been on during the compilation of that program. Refer to Paragraph 6-2.6 for further information.

During compilation, the user must supply three assignments for proper compilation. These assignments consist of LFN 7, which is used for Source/Input; LFN 6, which is used for List/Output; and LFN 5, which is used for binary output.

To execute an RPG II compiled program under DMS, the only assignments necessary are those which are specified in the source program on the File Description statements.

15-9.2 Compilation Job Stream

In order to compile an RPG II program, the above requirements must be met, and then the following Job Stream should be entered:

```
$RPGII
      . . .
      source program
      . . .
/* (or eof)
      compile time table or array data   } (this may occur zero or more times)
/*
```

Note, that the last End-of-File or /* card may be omitted since the system provides an EOF status upon reading the next Job Control card.

15-9.3 Execution Job Stream

In order to execute an RPG II program, the above requirements must be met, and then the following Job Stream should be entered:

```
$CATGO          (if the program resides in the link ready file)

        or

$program name   (if the program was previously Cataloged)
```

15-9.4 RPG II Error Comments

All of the RPG II compilation error comments appear on LFN 6 along with any List/Output from the program. The error comments are self-explanatory, and occur immediately following the line in error. A complete listing of the errors is available in the RPG II General Specification, (AA61711-00).

All execution errors from RPG are written out to LFN 1. LFN 1 is automatically assigned to physical device 1 for background programs. This assignment should be made if the program is run in foreground. Following the error message, the program initiates a system HOLD. The user should release or abort his program based on the error message and the task involved.

15-10 INDEXED SEQUENTIAL UTILITY

The Indexed Sequential Utility is a utility designed to enable users to perform basic manipulations on indexed sequential files.

15-10.1 Linkage with DMS

The Indexed Sequential Utility requires only that appropriate assignments be made prior to executing the utility. These assignments consist of LFN 0, which is used for command input; LFN 6 which is used for listing of the commands and output from the LIST command; and LFN 1, which is used for error listing. LFN 1 should be assigned to the operator communications device so that the operator may respond to the error condition. In addition, any logical files that are referenced on LFN, SAVE, etc. cards must also be assigned to their appropriate devices of files.

15-10.2 Execution Job Stream

In order to execute the Indexed Sequential Utility, the above assignments must be made, and then the following Job Stream should be entered:

```
$ISUTIL
   . . .
   utility commands
   . . .
```

15-10.3 UTILITY Error Comments

Error comments from the Indexed Sequential Utility appear on LFN 6 or LFN 1 according to whether they must be acted upon immediately or can be referenced later. The HOLD message PAUSE ERROR is an indication of some syntax or other error within the commands. An actual explanation of the error is available on LFN 6. For a further discussion of the Indexed Sequential errors, refer to the Datacraft Indexed Sequential Package General Specification, (AA61676-00).

15-11 SORT/MERGE UTILITY

The Sort/Merge Utility is a utility designed to enable users to perform Sorts and/or Merges from Job Control Statements.

15-11.1 Linkage with DMS

The Sort/Merge Utility requires only that appropriate assignments be made prior to executing the utility. These assignments consist of LFN 01, which is used for command input; LFN 6, which is used for listing of the commands; and LFN 1, which is used for error listing. LFN 1 should be assigned to the operator communications device so that the operator may respond to the error condition. In addition, any logical files that are referenced on FILE cards must also be assigned to their appropriate devices or files.

15-11.2 Execution Job Stream

In order to execute the Sort/Merge Utility, the above assignments must be made, and then the following Job Stream should be entered:

```
$MSUTIL
   . . .
   utility commands
   . . .
```

15-11.3 Utility Error Comments

Error comments from the Merge/Sort Utility appear on LFN6 or LFN1 according to whether they must be acted upon immediately or can be referenced later. For a further discussion of the Sort/Merge errors, refer to the Sort/Merge General Specification, (AA61712-00).

15-12 PRINT FILE

PRINT FILE is a utility for printing the contents of a logical file. The file may be assigned to any input device. The print format may be either octal or alphanumeric.

15-12.1 Commands

The PRINT FILE utility is executed via the $PRINTF or $LOADGO PRINTF Job Control statement. The command card input format is as follows:

```
xx,a,b,c,f
Axx,a,b,c,f
Cxx,a,b,c,f
```

where:
- xx is the logical file number (octal)
- a is the starting record number (0, 1, 2,. . . ;decimal)
- b is the ending record number (0, 1, 2,. . . ;decimal)
- c is the number of words to be printed for each record. This may be greater or less than the actual number of words in the record (decimal).
- f is the desired format (A for alpha, 0 for octal, or blank for octal)

xx implies rewind file xx and print c words from record a-b in format f.

Axx implies advance one file before starting the print cycle.

Cxx implies a continue on file xx without advancing or rewinding, the next sequential record is to be considered record number zero.

15-12.2 Adding PRINTF to the Processor File

The adding of PRINTF to DMS is the same as any background processor, make the appropriate assignments, include, and catalog.

PRINTF is shipped to the user in a link module format. The following is one procedure which may be used:

```
$JOB ...
$ASSIGN 4, binary input device
$INCLUDE
$CATALOG
NAME=PRINTF,,R
BEGIN
$EOJ
```

15-12.3 Examples

The following is an octal dump of the first 5 records on a magnetic tape.

```
$JOB
$ASSIGN  11=11
$PRINTF
11,0,4,500          (octal output)
11,0,4,500,A        (alphanumeric output)
EXIT
$EOJ
```

The following procedure will output record 0 of the first file and records 5 through 10 after the first file mark.

```
$JOB
$ASSIGN  20=FILE
$PRINTF
20,0,0,3            (Record 0)
A20,5,10,100        (Records 5-10, second file)
EXIT
$EOJ
```

15-13 VERSATEC PLOTTER PACKAGE

This section describes the software support provided by the VERSATEC Plotter Package. The plotter support library consists of two distinct sets of routines. VERSAPLOT-I is a custom plotting package which offers the user maximum flexibility and control. VERSAPLOT-II is a series of routines provided for compatability with existing plotting packages provided by the manufacturers of pen plotters. In addition to the library, the second pass processors, "VPLOT" is responsible for accepting user directives which dictate how the output of the first pass should be processed. One such directive will initiate the foreground plot copy program, freeing the background area for further processing.

15-13.1 VERSAPLOT-I Support Library

The VERSAPLOT-I library consists of a series of FORTRAN callable subroutines which determine the generation of an intermediate (VERSADATA) file. This intermediate file must be subsequently processed by the "VPLOT" program (Section 15-13.3).

The intermediate (VDATA) file is written to LFN 20. For maximum efficiency this file should be a "blocked binary" disk file. Additionally, diagnostics are printed to LFN 6.

The DMS version of the support library agrees in all respects to the description given in the Versaplot Graphics Programming Manual (Versatec document number 50001-90003) except for those differences documented within this section.

The VERSAPLOT-I library consists of the following modules:

AXES - draws either or both X and Y axes including tick marks, scaled annotations, and axis labels.

DRAW - processes input data into graphic movements for straight lines, curves or point plots.

FORM - draws grid patterns and overlay forms.

MODE - provides program control over the total mode table organization allowing the user to either request or alter variables as required.

NOTE - performs plot generation of character texts numeric values, or symbol plots.

SCAN - searches unscaled data for ranges of maximum and minimum; computes scale factors and sets mode table scaling entries.

TONE - shades (with user designed tone patterns) or clears user defined polygonal areas.

Because certain parameters within the plotter package are plotter dependent, an independent procedure has been added which will dynamically configure the plotter package to a specific environment. An initialization call must be made prior to any references to the library. If such a call is not made, results will be unpredictable. This call must also be made when using the VERSAPLOT-II package. The form of this call is:

CALL MODE (0, MODEL, 0, 0)

Where MODEL is an integer designating the Versatec model number of the device for which the plot is ultimately destined. This call must be made once prior to any other library references but only once. Note that this Mode zero call is the only MODE call requiring an integer argument.

15-13.2 VERSAPLOT-II Support Library

For computer systems where extensive pen plotter software is already in use, VERSAPLOT-II provides a method to generate raster scan plots from existing pen plotter programs. The modules included in this package are briefly described below:

PLOTS	-	initializes the PLOT routine and sets the mode table for pen plotter emulation.
PLOT	-	process straight line "pen" moves with the "pen up" or "pen down".
NEWPEN	-	for pen plotters this routine selects a different writing pen; in VERSAPLOT-II NEWPEN changes the line width.
FACTOR	-	allows the user to enlarge or reduce the size of the plot.
WHERE	-	returns the current "pen position" as well as indicating whether the pen is "up" or "down".

As indicated in Section 15-13.1, an initialization call must be made prior to any references to the package. Logical file assignments are the same as those for the VERSAPLOT-I package.

15-13.3 VPLOT - Versaplot Second Pass Processor

References to the Versaplot library during phase one of the plotting process cause the generation of the intermediate VDATA file (LFN 20). This VDATA file must be processed by the phase two routine, VPLOT. The final raster output may be plotted directly online, buffered to mass storage, or saved on a mass storage device. This saved plot file might then be copied online or it might be copied "off-line" by the foreground plot copy program.

VPLOT is executed as a background processor. It reads and processes a series of control cards which describe user requests. A discussion of these user requests follows.

The VPLOT program reads options from LFN 7 and prints diagnostics to LFN 6. A series of other logical file assignments are required:

1. LFN 20 is assigned to the VDATA file produced by the output from phase one plotter library calls.

2. LFN 21 is assigned to the VWORK file. This file is used by phase two as a scratch work file when creating the final raster output. For maximum efficiency, this file should be assigned to an unblocked binary file.

3. LFN 22 is dynamically ASSIGNED by VPLOT. The nature of this assignment is determined by user option. This file will contain the saved VPLOT raster file, which may be plotted offline at a later time. An unblocked binary file provides maximum efficiency, but any mass storage medium is suitable.

4. LFN 23 should be assigned to the online printer/plotter if the online plotting option is selected.

The following definitions are required in the discussion of the various control options supplied to VPLOT:

FILE (PW) - a file name fully qualified by password (if necessary).

PDN - a valid peripheral device no. (octal).

N1-N2 - two decimal integers separated by a dash, or a single integer. This parameter designates which files of a multi-file area should be processed. (1<=N1<=N2<=32).

C=n - where n is a decimal integer (1<=n<=32) specifying the plot copy count.

D/K - any string beginning with a "D" or "K". This option specifies whether the VPLOT file should be "deleted" or "kept" upon completion of processing.

PACK # - a decimal integer describing a pack number upon which the VPLOT file should be saved.

All VPLOT control options must begin in column one. Parameters may be separated by blanks or a comma embedded in optional blanks. If a parameter is given as null or missing then it will assume its default value. The syntax of the control options is given below. A slash ("/") designates an alternative. Command names may be abbreviated to six characters.

Command defaults follow:

PDN = 0
N1-N2 = 1-32 (all files)
C = n defaults to C=1
D/K defaults to "D"
Pack # = 1

SAVEPLOT, FILE(PW)/PDN, N1-N2. Save the raster output onto FILE(PW)/PDN. N1-N2 designates which files of the VDATA to save. FILE(PW) must have been created prior to its use.

DSAVEPLOT, FILE(PW), N1-N2, PACK# Create FILE(PW) with the appropriate size to save the raster output generated by files N1-N2 of the VDATA file. FILE(PW) will be deleted if it existed prior to creation.

PLOTD, N1-N2. Plot files N1-N2 of VDATA directly to an online device. LFN 23 is assumed to be assigned to the printer/plotter.

PLOTS, FILE(PW)/PDN, N1-N2, C=99, D/K. Plot the previously saved area given by FILE(PW)/PDN, N1-N2 may be used to select various files from the saved area. Multiple copies may be specified. FILE(PW) will be deleted after processing, unless "keep" was specified.

PLOT, FILE(PW)/PDN, N1-N2, C=99, D/K, PACK #. Initiate the foreground copy program to plot files N1-N2 of FILE(PW)/PDN. Multiple copies may be specified as well as the delete option. If no FILE(PW)/PDN is given, then a file will be dynamically created from files N1-N2 of the VDATA file, and the foreground plot copy program will be initiated to plot the saved file (spooled DMS only).

15-14 COMPLOT PLOTTER PACKAGE

The incremental plotter package consists of a FORTRAN encoded subroutine library written by Houston Instruments, and Harris written interface routines. In addition, several foreground programs exist to support "off-line" operations.

15-14.1 COMPLOT BASIC SOFTWARE Library

The COMPLOT BASIC SOFTWARE plotter package consists of 11 user callable subroutines which are normally written in FORTRAN. The user writes a FORTRAN or Assembly Language program which contains calls to the desired subroutines. These calls will move the plotter pen to a specified point, draw symbols and numbers, raise or lower the pen, draw with labels, provide the necessary scaling, and plot a series of points when an array of X- and Y-coordinates is supplied.

Output from the library routines is to LFN 5.

Detailed information on the subroutine arguments and programming examples are available in the manual "COMPLOT Software Descriptions" by Houston Instruments. Harris has implemented the following routines:

FUNCTIONS OF THE SUBROUTINES

PLOT - Drives the pen from its present position to a new position with the pen raised or lowered. It is also used to redefine the present position of the pen or to locate the present position of the pen.

SCALE - Sets and stores scaling information for AXIS and LINE routines.

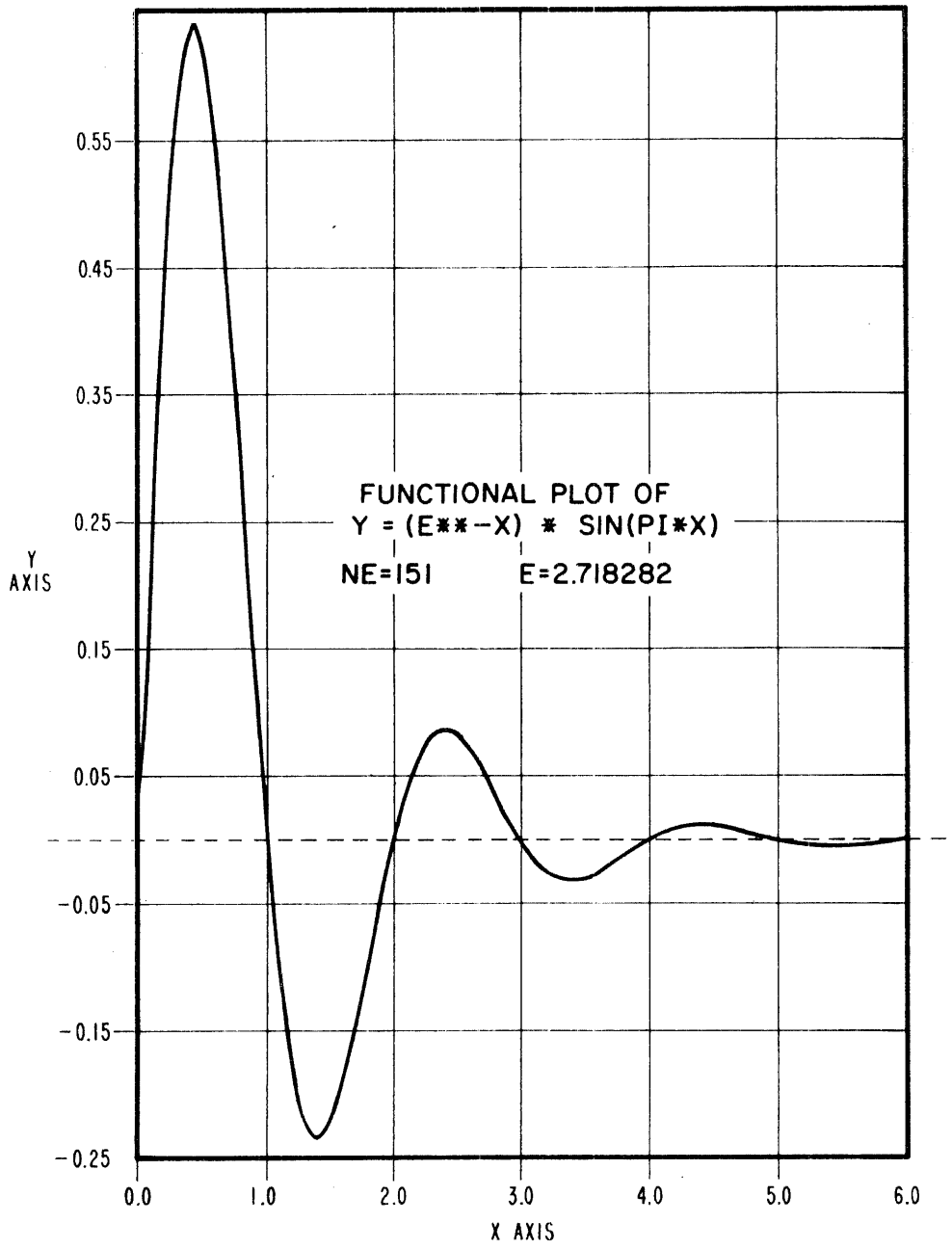


Figure 15-1. Versaplot Function Test

AXIS - Draws an axis with "Tic" marks at one-inch intervals and an identification label. Numerals representing the magnitude of the plotted data are drawn at each "Tic" mark.

LINE - Drives the pen through an array of points for point-to-point plotting.

SYMBOL - Positions the pen and forms the characters to be plotted. Available characters are 0-9, A-Z, and various special characters. See Figure 15. 3.

NUMBER - Positions the pen and forms the numberals that define the magnitude of an internal floating point number.

MARKER - Draws event marker at present pen position.

FACTOR - Scales all plotting.

PENUP - Raises the pen.

PENDN - Lowers the pen.

RSTR - Restores plotter to Z-fold position on a new page and positions the pen at the bottom of page. A call to RSTR is required for the dumping of the output buffer in time-shared and offline software.

NOTE: Hollerith strings contain 3 characters per word.

15-14. 2 Harris Library Routines

Harris also supplies several subroutines to compliment the above.

15-14. 2. 1 Subroutine PLTOUT

The subroutine PLTOUT performs the actual output to the operating system. The COMPLOT routine DRVR formulates a buffer of plot data and passes the information to PLTOUT. This enables the user to supply his own routine for special applications.

ENTRY: CALL PLTOUT (LFN,IBUFF,IWC)

WHERE:

LFN - OUTPUT LOGICAL FILE NUMBER
IBUFF - PLOT BUFFER ADDRESS
IWC - WORD COUNT
- POS - OUTPUT BUFFER
- ZERO - OPEN ONLY (PAGE RESTORE)
- NEG - CLOSE AFTER OUTPUTTING ABS (IWC) WORDS

Plot data is packed three commands per word. The commands contain pen number, pen-up or down, and direction. See Figure 15-2.

For example, a user wants to convert the plot command data to be compatible to a second computer, the following job stream may be used.

```
$JOB ....
$FORTRAN
      .
      .
      (MAIN PROGRAM)
      .
      .
      END
      SUBROUTINE PLTOUT (LFN,IBUFF,IWC)
            (user supplied routine)
      END$
$CATALOG
BEGIN
$ASSIGN 5=11      (Output from PLTOUT to mag tape)
$REW 5
$GO
      .
      .
      DATA (if any)
      .
      .
$EOJ
```

	NOT USED			PEN		DIRECTION			
			PEN UP	0	0				
	0	0	PEN #1	0	1	-X	-X	-Y	-Y
			PEN #2	1	0				
			PEN #3	1	1				
BIT	7	6		5	4	3	2	1	0

Figure 15-2. PLOT Command Format

15-14.2.2 Subroutine STPLOT

STPLOT may be called to initiate a special foreground plot program, S. PLOT. Once initiated, the plot data stored on a disc file, mag tape, or other media, will be output to the plotter. Thus an "off-line" operation is available.

ENTRY:

```
CALL STPLOT (FILE,PASS,COPIES,FIRST,LAST,IERROR)
```

WHERE:

FILE:	Six character file name or a valid physical device number.
PASS:	Four character password of disc file or zero.
COPIES:	The number of plot copies desired.
FIRST:	First plot file number (one or greater).
LAST:	Last plot file.
IERROR:	Return address upon an invalid file/device name.

For example, a user wants to output directly from his test program to the plotter, the following may be used.

```
$JOB ...
$FORTRAN
     .
     .
     (PROGRAM)
     .
     .
     END$

$CATALOG
BEGIN
$ASSIGN 5=nn    (assign output to plotter directly)
$GO
$EOJ
```

The user decides to perform a long plot and does not want to dominate the computer. The following may be used:

```
$JOB ...
$FILEMA
CREATE  PLDATA, MINE, 1,500,B6
$FORTRAN
     .
     .
     .
     (PROGRAM)
     CALL RSTR (2)
     CALL STPLOT (6HPLDATA, 4HMINE, 1,1,1)
     .
     .
     END$
```

```
$ASSIGN 5=PLDATA
$OPEN 5, MINE
$CATGO
$EOJ
```

15-14.3 Foreground Programs INTPLT and S. PLOT

S. PLOT is the main foreground program for "off-line" plots. Once initiated, S. PLOT obtains several plot parameters from either the subroutine STPLOT or the foreground program INTPLT. The parameters include device, file name, password, number of reproductions and plot file numbers. A buffer length of 110 packed words is assumed. S. PLOT may be released or terminated via use of the standard operator communication commands.

INTPLT is a foreground program input and verify the plot parameters before passing the information to S. PLOT. The following is the input format:

NAME(PASS),COPIES,FIRST-LAST

WHERE:

NAME	-	1 to 6 character file name containing the plot commands.
	-	An octal device number which the plot data is currently stored.
PASS	-	An optional password for disc files, in parenthesis
COPIES	-	Number of reproductions of the desired plot(s).
FIRST	-	First plot file number, starting with 1 (decimal, optional, default 1).
LAST	-	Last plot file number. (Decimal, optional, default, first plot file.)

The following are examples of INTPLT terminal usage.

```
$ON Unnn
$INTPLT
PLDATA          1 copy of first plot file on PLDATA
$OFF

$ON Unnn
$INTPLT
PLDATA (MINE),3     three copies of first plot file on PLDATA, password MINE
$OFF

$ON Unnn
$INTPLT
11,3,1-5        3 copies of plot file, 1 thru 5 on mag tape
$OFF
```

The following are the same examples but requested by the operator.

```
/IP,INTPLT,Unnn,100,0
PLDATA        1 copy of first plot file on PLDATA

/IP,INTPLT,Unnn,100,0
PLDATA (MINE),3      3 copies of first file on PLDATA with password MINE

/IP,INTPLT,Unnn,100,0
11,3,1-5      3 copies of plot files 1 thru 5 on mag tape
```

The syntax error messages are self-explanatory. After the error message is printed, the correct command may be input.

INTPLT must be cataloged foreground while S. PLOT must be cataloged as privileged foreground with LFN 21 assigned to the plotter physical device number (see Cataloger Assign Statement).

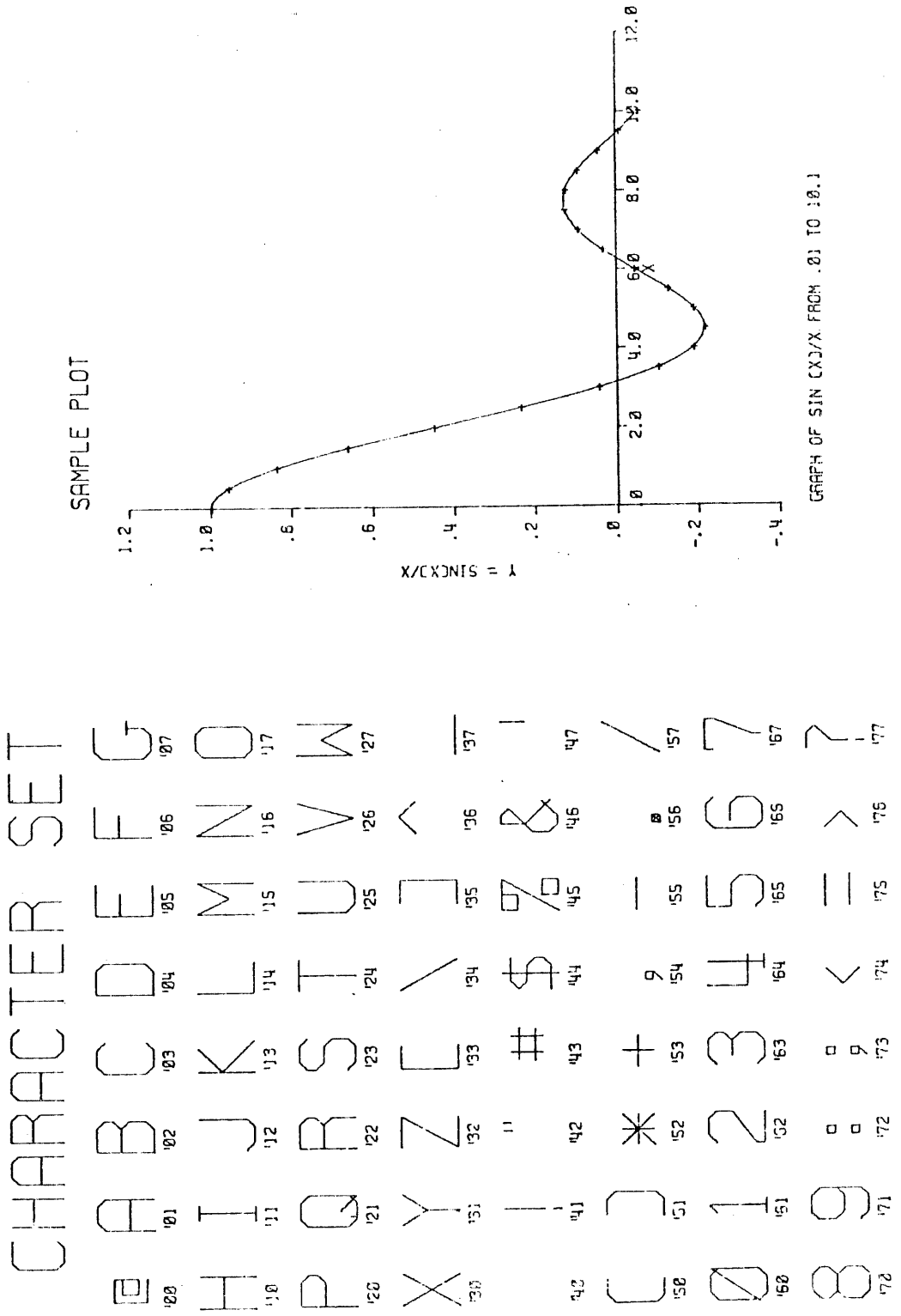


Figure 15-3. Sample Plot

APPENDIX A
DMS TABLES

DISPATCHER CONTROL TABLE

The Dispatcher Control Table consists of a fixed area of memory containing two threaded lists, the Active Program List and the Available Entry List. The table also contains the Operator Communications Block.

The Active Program List (APL) contains a nine word entry for each program which is currently being executed by DMS. The list is threaded in priority order and always contains at least one entry (for background which is always active). For spooled systems the entry for "IOEXEC" is also always active This thread is started with the pointer which is labeled $ACT and contained in the EXEC module.

Each entry in the APL contains the following information:

Word 0	Pointer to next entry in the list (Zero indicates last entry).
1	Program name -
2	(6 ANSCII characters)
3	Program status word (see Table 4-2).
4	Core allocator threads (if allocated)
5	Low memory address (start of program service area)
6	High memory address
7	Program priority value in bits 7-0 (0 to 255--- 0 is highest) Accounting user# in bits 23-8 (0-65735 --- 0 for non-accounting systems).
8	Parameter passed to program at initiation

The Available Entry List consists of similar nine word entries with the value of words 1-8 having no significance. The purpose of this list is to allow the Initiate Program subroutine to locate table space rapidly. This list is started by the Available Entry List pointer, which is labeled $AVAIL and is contained in the EXEC module. When no entries remian in this list, $AVAIL contains a zero.

The Operator Communications Block ($OCBLOK) reserves 9 memory words in the Dispatcher Control Table which are used as an active program list entry for the operator communications program. This block insures that OPCOM functions can be processed when the available entry list is empty.

DISPATCHER CONTROL VARIABLES

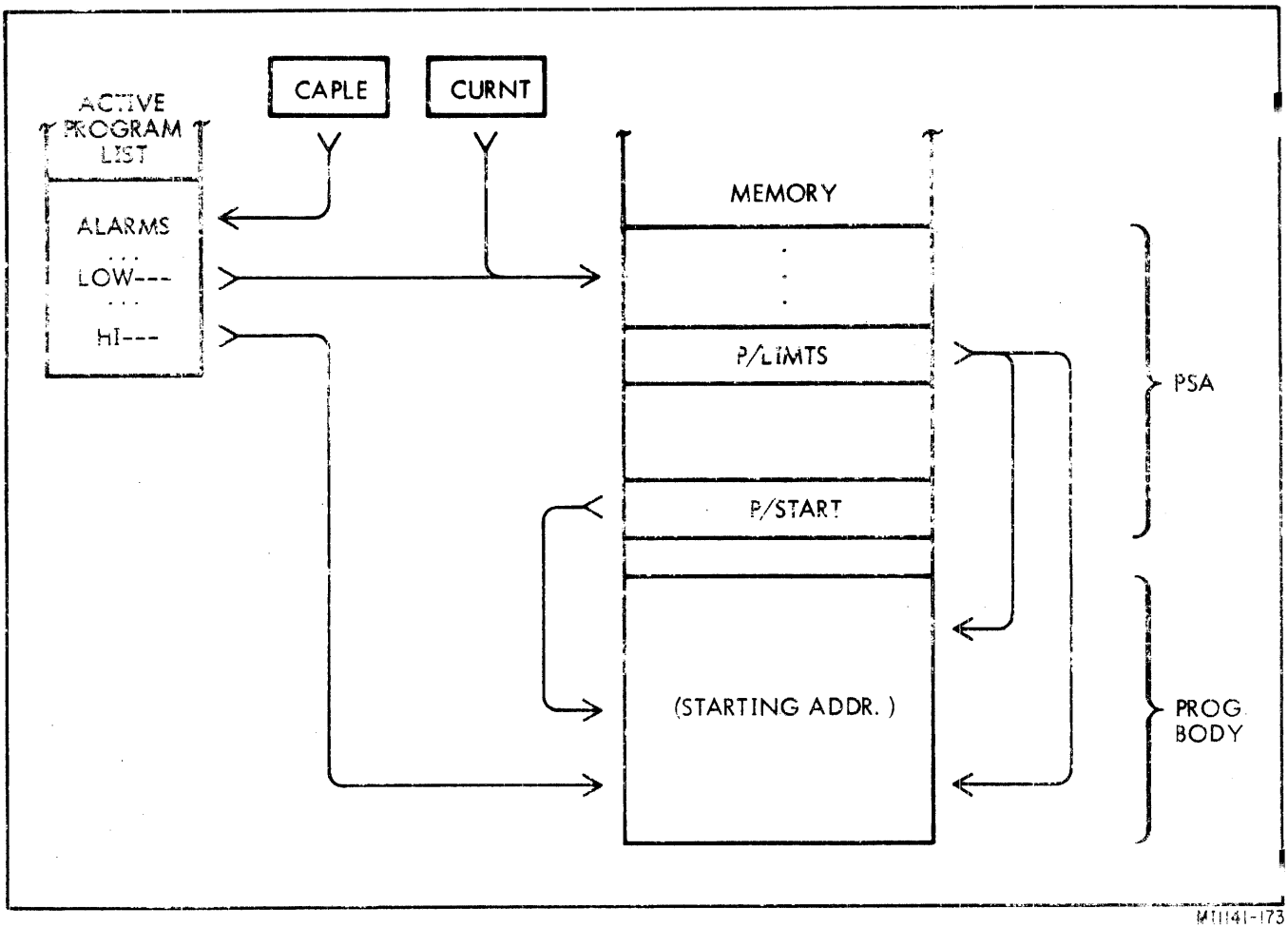
CURNT - When any program is being executed, this variable contains the address of the first word of the Program Service Area (PSA) associated with that program. When the Disc Monitor System is not executing any program, it is considered to be in the 'null state' and CURNT must be zero.

CAPLE - This variable contains the address of the Active Program List (APL) entry for the active program corresponding with the PSA pointer in CURNT.

ACT - This pointer contains the address of the first (highest priority) entry in the Active Program List. Since the background is always active, this pointer is never zero.

AVAIL - This pointer contains the address of the first available entry in the Active Program List. If the entire Dispatcher Control Table is full, this pointer will be zero.

The following figure shows a typical Active Program List. The foreground program ALARMS, the I/O Executive, and background are in the active program list, and there are two free entries available.



PROGRAM SERVICE AREA

The Program Service Area (PSA) consists of register save blocks, system variables, and device control tables which are unique to each single program. The PSA precedes each program in memory (except for re-entrant foreground programs) and is initially established by the foreground allocator before each foreground program is loaded.

The format of the PSA is shown below:

Word	Label	Function
0-4	P/PSA	General register storage (I,J,K,E,A).
5	P/PC	Program counter storage.
6	P/BIT	Bit processor storage (H,V).
7-9	P/SAU	S.A.U. register storage (X,Y).
10	P/SWIT	Program Switch Word (Used for foreground communications).
11-12	P/LIMTS	Values for loading the limit registers (Computed by the allocator).
13	P/CTEMP	Pointer to the highest 5-word block of temporary storage that is currently allocated.
14-145	P/TEMP	Re-entrant storage area (132 words including sector buffer).
34-145	P/SBUFF	Sector buffer area (112 words).
146	P/ACCTG	Pointer to Accounting Information Block.
147-154	P/FDCB	Reserved file control block (for LFN '77 used by loader).
155-266	P/CFDCB	Standard File/Device Control Blocks.
267	P/MODE	Execution mode switches.
268	P/START	Program execution starting address.

Execution Mode Switch Word format

The following table explains the significance of the various bits of the Execution Mode Switch Word, (word 267 of the PSA).

Bit 23 = 1 Privileged
0 Restricted

Bit 22 = 1 Ignore floating-point overflow
0 Test further bits on floating-point overflow

Bit 21 = 1 Do not type message on SAU underflow.
= 0 Type SAU message on SAU underflow.

Bit 20 = 1 Do not abort on SAU underflow
0 Abort on SAU underflow

Bit 19 = 1 Do not type message on SAU overflow
0 Type SAU message on SAU overflow

Bit 18 = 1 Do not abort on SAU overflow
0 Abort on SAU overflow condition

Bit 17 = 1 Do not abort on FORTRAN I/O error
0 Abort on FORTRAN I/O error

Bit 16 = 1 Program is a non-accounting foreground type

Bit 6 = 1 Program is re-entrant foreground

Bit 5-0 = Count of users of foreground re-entrant program (Only in P/MODE of actual program body)

Bit 15=1 Option 23 is cataloged with program
0 Option 23 not cataloged with program

Bit 14=1 Option 22 is cataloged with program
0 Option 22 not cataloged with program

Bit 12=1 Option 20 is cataloged with program
0 Option 20 not cataloged with program

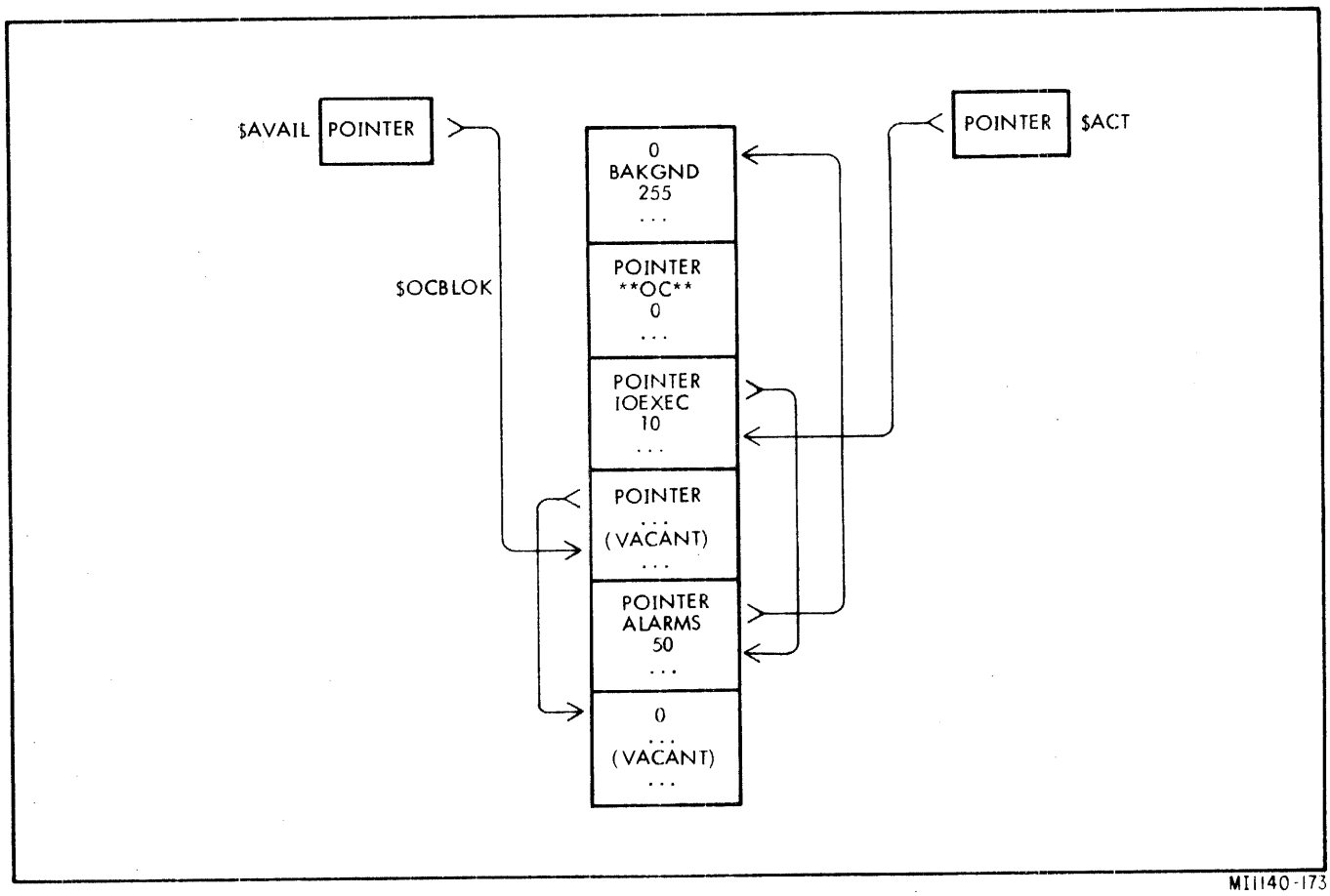
Bit 10=1 Option 18 is cataloged with program
0 Option 18 not cataloged with program

Bit 9=1 Option 17 is cataloged with program
0 Option 17 not cataloged with program

Bit 8=1 Option 16 is cataloged with program
0 Option 16 not cataloged with program

Example of APL/PSA Linkages

The following chart shows the linkages which are present when the foreground program ALARMS is loaded in memory and is active.



FILE/DEVICE CONTROL BLOCKS

File/Device Control Blocks are used to handle logical I/O through the I/O Control System and the device handlers. Each Logical File Number which is assigned to a disc file or physical device has an associated File/Device Control Block. The general layout of these blocks is shown below:

Device Control Block

Bits	23	22	21	20	19	18	17	16-12	11-6	5-0
Word 0	Busy	WCNC	EOF	Err	Open	0	PA	Err#	LFN	PDN
1	Reserved For Handler Use									

File Control Block

Bits	23	22	21	20	19	18	17	16-12	11-6	5-0
Word 0	Busy	WCNC	EOF	Err	Open	1	PA	Err#	LFN	0
1	Reserved For Handler Use									
2	First 3 ANSCII Characters of File Name									
3	Second 3 ANSCII Characters Of File Name									
4	Disc Number							Initial Sector #		
5	BL	SP	R	W	O	F	Final Sector #			
6	Current Sector Number									
7	Sector Number Of Start Of Current File									

* If file is blocked (BL = 1) then

Word 6 = undefined

Word 7 = pointer to blocked work area

The definition of the bits of the File/Device Control Blocks Are:

BUSY - Set to indicate that the user's buffer is busy.

WCNC - Set to indicate word count not complete.

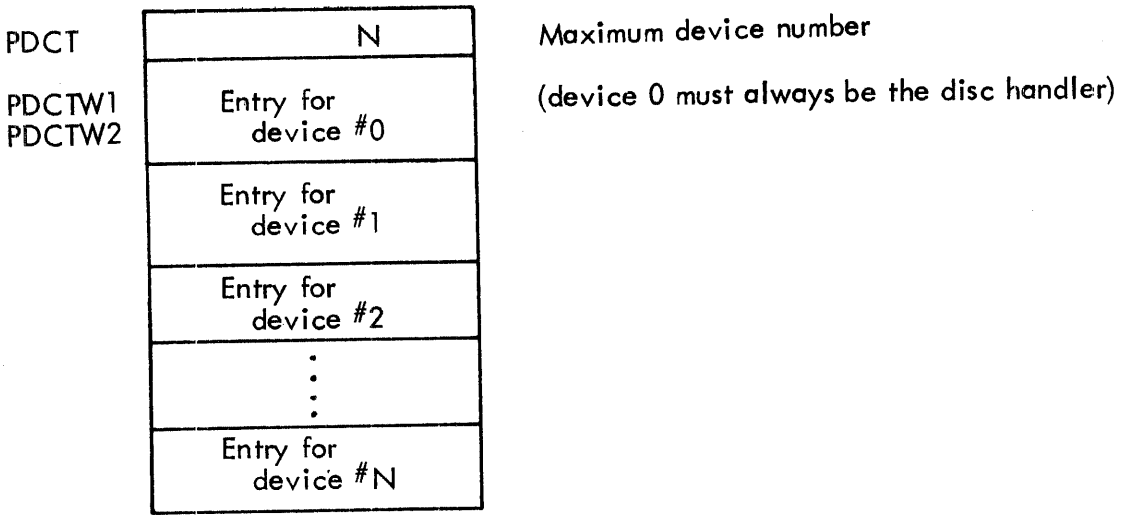
EOF - Set to indicate EOF read on last input.

ERR - Set to indicate fatal device error (Error number in 16-12).

OPEN	-	Set to indicate that the file or defice has been opened.
PA	-	Permanent assign - set to indicate that this assignment cannot be changed.
ERR #	-	Error code number of device error.
LFN	-	Logical File Number.
PDN	-	Physical Device Number to which the LFN is assigned.
BL	-	When set, indicates a blocked file.
SP	-	When set, indicates a spooled file.
R	-	Set to indicate file is read-prohibited for this user.
W	-	Set to indicate file is write-prohibited for this user.
O	-	Set to indicate file has overflowed.
F	-	Set to indicate file has been written into.

PERIPHERAL DEVICE COORDINATION TABLE

The Peripheral Device Coordination Table (PDCT) is used to associate a particular device and its handler with a Physical Device Number and to coordinate and control the use of the device among the various active programs.



Note that the physical device number associated with a device is determined by its position in this table. Unused entries may occur, but must have both words of their entry equal to zero.

Table Entry Layout:

Bit	23- 16	15- 0
Word 1	Current User	Device Handler Entry Address

	23	22-18	15- 0
Word 2	AF	Device Type	Device Busy Flag Address

Current User - Contains the address of the Active Program List entry of the program to which it is allocated. This address is masked by '3770 in order to generate a unique 8-bit value for each program.

Device Handler Entry Address - Provides the memory address of the handler entry point used to initiate an I/O request.

AF (Allocated Flag) is set to "1" (negative) when the device is allocated to a particular program and cleared upon release.

Device Type is the type of physical device. The various valid device types are:

0	Disc
1	Console TTY
2	Remote TTY
3	Remote CRT
4	Paper Tape Reader
5	Paper Tape Punch
6	Line Printer
7	Card Reader
10	Card Punch
11	Magnetic Tape
12	Synchronous Communications Devices
13	Real-time peripherals
14	Incremental Plotter
15	Printer/Plotter

Device Busy Flag Address - Contains the memory address of the busy flag that is associated with this device. Note that one busy flag might be shared among several devices, as is the case with several magnetic tape units on a single controller. Each tape unit can be allocated to a separate program but this activity must be coordinated by a single busy flag.

Example Of Peripheral Device Coordination Table

```
PDCT    DATA    '11         . Maximum Device Number
PDCTW1  DAC     $DFH        . Device 0 - Disc
PDCTW2  DAC     = 0         . Disc Is Never Busy
        DAC     $ASR        . Device 1 - ASR TTY
        '01     ASRBY
        DAC     0           . Device 2
        DAC     0
        DAC     0           . Device 3
        DAC     0
        DAC     0           . Device 4
        DAC     0
        DAC     0           . Device 5
        DAC     0
        DAC     $LPAH       . Device 6 - Line Printer
        '06     LPABF
        DAC     $CRAH       . Device 7 - Card Reader
        '07     CRABF
        DAC     0           . Device 10
        DAC     0
        DAC     $MCAH0      . Device 11 - Mag Tape
        '11     MCABF
```

RESIDENT PROGRAM LIST

The Resident Program List contains the ANSCII name and memory bounds for each resident program. The size of this table must be established when DMS is generated, and must be large enough to satisfy the worst case requirements of each particular foreground environment.

Entires are made in this list by the system initialization routine (SYSINIT), which directs the loading of all programs which have been cataloged as resident foreground types.

RPLIST

WORD 0	N	M
1	ENTRY #1	
5	ENTRY #2	
9	ENTRY #3	

N - Maximum number of entries allowed in the table

M - Number of entries in current use.

Entry Format:

Word 1 - Program name
2 - (6 char. ANSCII)

3 - Low memory address

4 - High memory address

TIMER SCHEDULE TABLE

The Timer Scheduler must reserve a six word entry for each program to be executed concurrently on a periodic basis. The size of this table must be specified at system configuration to be large enough for worst case conditions of requests for timer schedule services.

MXTPRG N	Maximum number of programs on timer schedule
NAME1	First 3 characters of program 1 name or zero * First 3 characters of program 2 name or zero*
	First 3 characters of program N name or zero *
NAME2	Second 3 characters of program 1 name Second 3 characters of program 2 name
	Second 3 characters of program N name
LEVEL	Program 1 priority level Program 2 priority level
	Program N priority Level
INTRV	Program 1 negative of interval Program 2 negative of interval
	Program N negative of interval
INCTR	Program 1 interval counter (incremented each cycle) Program 2 interval counter
	Program N interval counter
PARAM	Program 1 initiation parameter Program 2 initiation parameter or Program N initiation parameter — APL pointer for suspend calling program for interval mode.

* Note that a vacant entry is denoted by having the NAME1 value equal to zero.

If the INTRV value is zero, and if the LEVEL value is zero, the program is suspended until the clock period is completed, at which time it will be released by the Timer Scheduler. If the LEVEL value is not zero, then the specified program will be initiated when the interval has been completed, and the entry will be removed from the timer schedule list.

MAGNETIC TAPE OPTIONS TABLE

This table is used to define the characteristics of and the default tape options of the various tape drives in the system. There is a one word entry for each drive. Thus, there can be up to 8 words in this table. The position of an entry in this table is determined by the software transport number (T0-T7) associated with the entry. E.g., Word 1 is the default tape-op for transport T0 and word 7 is the default tape-op for transport T6. An 8 word block is also set aside here which contains the background tape options word actually being used.

The entry format is shown below:

Bits	23-17	16	15-14	13-12	11-10	9-6	5-3	2-0
Entry	0	Type	Mode	Density	CPW	0	TR.	0

Type (0) Specifies 9-track tape
(1) Specifies 7-track tape

Mode(0) Specifies binary-ASCII (no conversion) for symbolic I/O
(1) Specifies BCD or EBCDIC conversion for symbolic I/O

Dens. (0) Specifies 200 BPI - or - low density PEC
(1) Specifies 556 BPI - or - Hi density PEC
(2) Specifies 800 BPI
(3) Specifies 1600 BPI

CPW (0) Specifies 1 character per word
(1) Specifies 2 character per word
(2) Specifies 3 character per word
(3) Specifies 4 character per word

TR is hardware transport number on controller

TERMINAL CONTROL BLOCKS

Terminal Control Blocks are established for each physical device which is either to be spooled or to interact with foreground through the remote terminal handler.

Terminal Control Blocks define the physical characteristics of these physical devices, along with holding certain control information to allow the spooling and foreground I/O tasks to be performed.

Each Terminal Control Block (TCB) consists of 21 words as defined below:

Word	Label	Contents
0	T/PDN T/FCB	Bits 23-12 Normal File Control Block Bits for word 0 of FCB (status). (bit 23 set indicates TCB busy) bits 5-0 Physical Device Number
1		Reserved for handler use
2-3	T/NAME	File name for spooling or processor name for foreground communications.
4-7		Reserved for handler or processor use
8	T/C	Control word for TCB (see below)
9	T/LINK	Link to next TCB in chain or 0 (for end of chain)

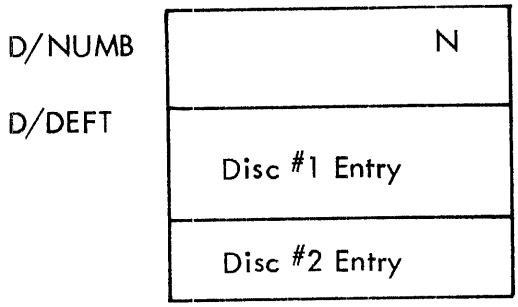
Word	Label	Contents
10	T/PL	Parameter list storage for I/O call ('XXYY)
11	T/WC	Word count of I/O buffer for device
12	T/BA	Address of I/O buffer (0 at SYSGEN -- filled in via dynamic core allocation request) (storage beyond I/O buffer in this core block is used as permanent storage by ACRONIM.)
13	T/PRI	Foreground execution priority of terminal
14	T/FPRI	Priority limit for device. (default priority set in bits 7-0; if bit 23 set, then this is a maximum limit for this terminal; if both bits 23 and 22 set, then this is a fixed priority for this device -- none other will ever be used.)
15	T/SQP	Spool queue pointer
16	T/MODE	Mode of terminal : -1 = Input only 0 = Both input and output +1 = Output only
17	T/LP	Bits 23-8 Characters per line 7-0 Lines per page TTY = 1 L/P; 72 C/L. Printer = 1 L/P; 132 C/L.
18	T/ALT	Address of related TCB. Gives output TCB for any input only or input/output device. Should be address of first word of TCB. For output devices, gives address of other output TCB to take output from it if that device is busy, or zero if this is not desired.
19	T/PROG	Gives negative of Active Program List entry address for current foreground program connected to this terminal, or last initiated by this terminal. If input spooling, gives input job stream queue priority.
20	T/REGS	Gives storage definitions for ACRONIM function. Bits 23-16 = Maximum # of files available at any one time under ACRONIM 15-8 = Number of "number" registers available to the terminal minus 1. 7-0 = Number of "alphanumeric" registers available to the terminal.

The TCB control word is used by the I/O executive module to manipulate the TCB and keep track of its current status and operations. The various bits of the control word are defined below:

TCB CONTROL WORD

Bit 22 -	Terminate Output	(Abort) - set by handler to indicate that user wishes to abort his current foreground or spooling process
Bit 21 -	Input Ready -	Indicates that the handler has input ready for I/O executive to pass on either to spooling or foreground processor.
Bit 20 -	Special Action Request -	used to return $JOB card from foreground interpreter to initiate input spooling process.
Bit 18 -	Input Request -	signals I/O executive to request input from device.
Bit 16 -	Operator Open Request -	Indicates that the terminal user wishes to begin communications through his terminal. I/O executive opens input from device and reads one line.
Bit 15 -	Close Request	I/O executive closes the device and releases the I/O buffer and permanent storage area.
Bit 14 -	Delayed Open Request	An operator open request is made when the current program is disconnected from the terminal.
Bit 8 -	Waiting for Cell	The I/O Executive was unable to obtain a 4-word cell for this TCB.
Bits 2-0 -	Mode	000 = Closed 001 = Input spool 010 = Processor 011 = Control 100 = Output spool

DISC DEFINITION TABLE



Entry format:

Word 0	Address of disc controller busy flag
1	Number of words per sector
2	Number of sectors per track
3	Number of tracks per cylinder
4	Number of cylinders per pack
5	Pack number currently mounted (default set at SYSGEN time)
6	Sector number of space allocation map on the disc
7	Sectors per space allocation bit
8	Retry count
9	Drive number at appropriate bit position for command word

Word 10	Channel/Unit number
11	Address of disc controller service routine
12	Output Address Word (OAW #CU) instruction
13	Output Command Word (OCW #CU) instruction
14	Address of disc controller string pointer
15	Reserved

DISC I/O QUEUE

The Disc I/O queue is a queue in which all disc I/O requests are stacked for processing according to priority and disc controller availability. Queue entries are prepared and entered into the stack by the resident disc handler, as each request is made, and processed according to priority.

Each entry consists of 8 words, defined as follows:

Word 0	String address pointing to NEXT AVAILABLE SPACE or the NEXT PRIORITY ENTRY FOR THAT CONTROLLER.
1	Disc definition block pointer
2	Word count
3	Buffer address
4	Disc command word
5	File Control Block pointer
6	Priority of request
7	Bit 23 = 1 For adv. file; 0 otherwise. Bit 22 = 1 For I/O executive call; 0 otherwise. Bit 21 = 1 For absolute R/W call; 0 otherwise.

Each controller maintains its own string through the queue, only those entries which are for that disc controller are kept on the string. An Available Space List is also maintained to link together the unused entries for later use.

The variable DAVAIL maintains the Available Space List through the vacant cells in the queue. The variable(s) DC-SP maintain the individual disc controller string for disc controller "n".

All strings are terminated with a 0 pointer value.

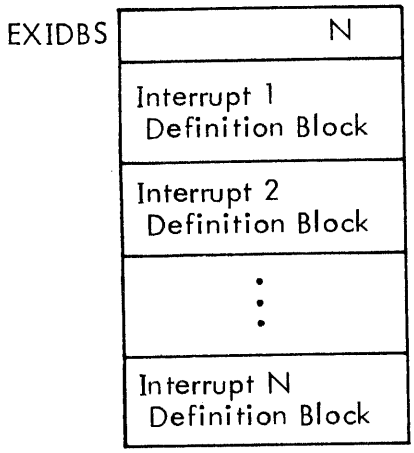
When an entry is taken from the string for processing, its address is negated and stored as the disc controller busy flag DC-BF. After successful completion of a request, word 0 is removed from the disc controller string, and added to the available space string.

MASTER DISC DIRECTORY (MDD)

WORD																								
0	FIRST THREE ASCII CHAR OF FILE NAME																							
1	SECOND THREE ASCII CHAR OF FILE NAME																							
2	INITIAL SECTOR NUMBER																							
3	23 PRIVT	22 PRO	21 PRO	20 PRO													10-0 FINAL SECTOR NUMBER							
4	23 BLOCK	22 SPOOL	21 CORE	20 PERM	19-16 TYPE												15-0 MEMORY REQUIREMENTS							
5	23-16 PACK NO																15-0 ABSOLUTE PROGRAM ORIGIN							
6	FIRST THREE ASCII CHAR OF PASSWORD																							
7	23-16 4TH CHAR OF PASSWORD																15-0 USER NUMBER							

EXTERNAL INTERRUPT DEFINITION BLOCKS

An External Interrupt Definition Block is required for each interrupt which might be used for foreground program activation in a particular configuration. The format of this table is:



Each entry takes the form:

```
IDBXYY   DATA  'XYY      . Interrupt group (X) and level (YY)
         DATA  0,0       . Name of connected program
         DATA  0         . Priority of connected program
         DATA  0         . Initiation parameter of connected program
         RDAT  5(0)      . Register save block
         DATA  0         . Interrupt address
         TRM   *-6       . Save registers upon entry
         TMD   *-11      . (D) = program name
         TMI   *-10      . (I) = execution priority
         TMK   *-10      . (K) = parameter
         BSL   $INIT#    . Call initiate subroutine
         TMR   *-11      . Restore registers
         BRL*  *-7       . Return to interrupted program
```

The corresponding dedicated interrupt location must be filled with a BSL instruction to the location IDBXYY + 10.

APPENDIX B
ACCOUNTING RECORD STRUCTURE

This appendix defines the contents of DMS accounting records, produced by accounting configurations of DMS as discussed in Section VII.

All accounting records are eight words long and follow a standard format as discussed below, and shown in Table B-1.

Word 0 contains the identification of the type of accounting record. The bits are defined as follows.

Bits 23-18 - Device type (see Table 4-3)

Bit 8 - Set to indicate remote terminal accounting record.

Bit 7 - Set to indicate re-entrant foreground program accounting record.

Bit 6 - Set to indicate background accounting record.

Bits 5-0 - Physical device number.

Some records use special combination of these fields for special records. These are as follows.

Device type 0 - Disc Record

36_8 - System Idle Time Record

37_8 - CPU Usage Record

Word 0 = -1 (77777777_8) - Date Record

The exact format of each of these records is discussed later.

Word 1 of all records contains the user number, or zero if none.

Word 2 defines the starting time-of-day for accumulation of information in the record. This is expressed in tenths of seconds since midnight.

Word 3 defines the ending time-of-day for accumulation of information within the record. This is also expressed as tenths of seconds since midnight.

Words 4-7 are variable depending on record type. See Table B-2 for these words.

Table B-1. Standard Accounting Record

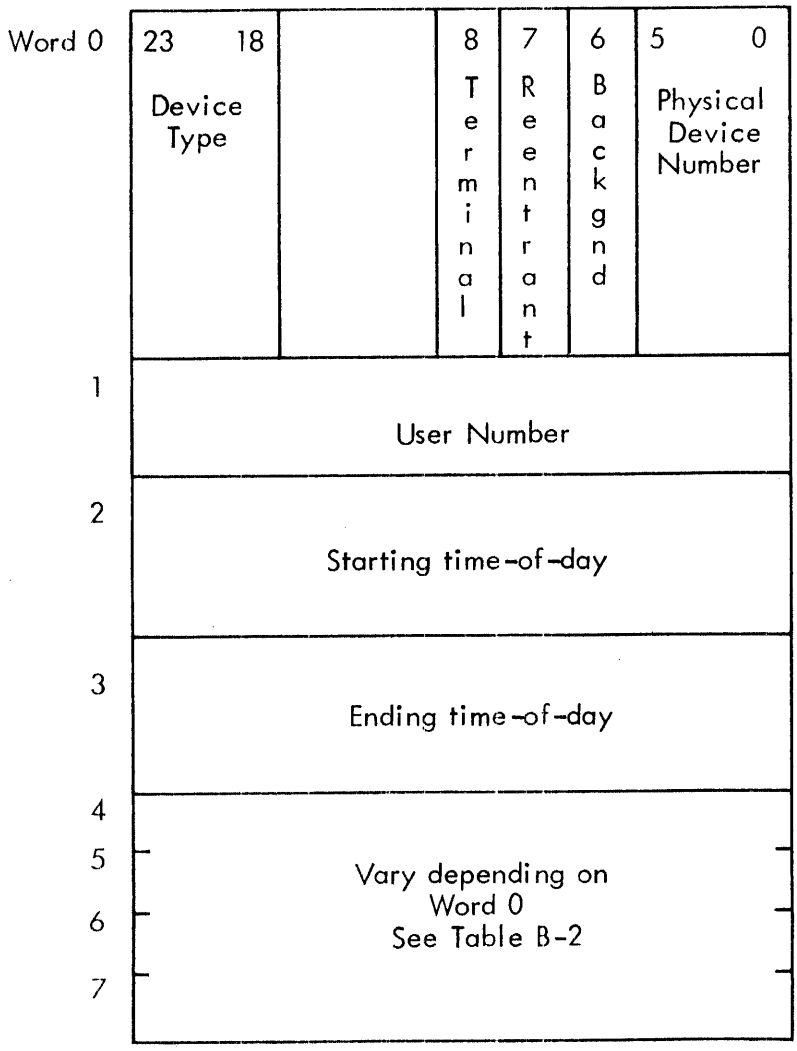


Table B-2. Accounting Record Contents

Record Type	Device Type in Word 0	Word 4	Word 5	Word 6	Word 7
Date	Word 0 = '77777777	First 3 ASCII characters of date	Second 3 ASCII characters of date	Last 2 ASCII characters of date & blank	N/A
CPU Usage	'37	CPU time in milliseconds	Core usage in words	N/A	N/A
System Idle time Accumulator	'36	Time spent in Idle loop in millisec.	N/A	N/A	N/A
Disc	'00	Disc Accesses	Hundreds of words transferred	N/A	Disc errors
Console Device	'01	Lines input or output	N/A	N/A	N/A
Remote TTY	'02	Lines input or output	N/A	N/A	N/A
Remote CRT	'03	Lines input or output	N/A	N/A	N/A
Paper Tape Reader	'04	Characters read	N/A	N/A	N/A
Paper Tape Punch	'05	Characters punched	N/A	N/A	N/A
Printer	'06	Lines printed	N/A	N/A	Error stops
Card Reader	'07	Cards read	N/A	N/A	Error stops
Magnetic Tape	'11	I/O requests	Hundreds of words transferred	N/A	Mag. Tape
Synchronous Communications	'12	Characters transferred	N/A	N/A	Errors
Real-Time Peripherals	'13	Words transferred	N/A	N/A	N/A
Printer/Plotter	'15	Lines printed	Lines Plotted	N/A	Error Stops
Incremental Plotter	'14	Hundred of commands transferred	Number of physical pages used	N/A	N/A

APPENDIX C
SYSTEM ERROR MESSAGES

System Error Messages

Message	Generated by	Meaning	Hold Message ?	Action taken when released
$ADD FILNAM NON-EXISTENT	DMS	Attempted to ADD a non-existent file to job stream.		
ABS BIAS ERROR	File Manager	Invalid program BIAS specified on DUMP or DUMPBF command.	No	
ABS START ERROR	File Manager	Invalid program starting address specified.	No	
ABT XX cccccc	DMS	Program aborted at location cccccc for reason XX - see below:	No	
ABT 01		Program called abort.		
ABT 02		Operator aborted program.		
ABT 03		Instruction trap violation.		
ABT 04		Memory protect violation.		
ABT 05		Stall Alarm violation.		
ABT 06		Floating point overflow.		
ABT 07		Invalid "BLU" operand.		
ABT 10		Unassigned device code.		
ABT 11		Invalid function code.		
ABT 12		Failed to open a device before using it.		
ABT 13		Program attempted to input below its lower bound.		
ABT 14		Program attempted to input above its upper bound.		
ABT 15		Temporary storage exceeded.		
ABT 16		File extents overrun.		
ABT 17		Program attempted to read job statement record.		
ABT 20		System service called with invalid parameter.		
ABT 21		Unable to load "BAKGND" program due to size limitations.		
ABT 22		Invalid absolute load module origin.		
ABT 23		Invalid open request (disc file).		
ABT 24		Invalid file type for loading.		
ABT 25		Invalid external interrupt specification.		
ABT 26		Invalid program starting address.		
ABT 27		Attempt to load non-absolute background program across 32K map.		
ABT 33		Invalid request to return dynamic core.		
ABT 34		Invalid parameters on dynamic assign call.		
ABT 35		Assign table overflow.		
ABT 36		Invalid parameters on Dynamic Core Manager call.		
ABT 37		Program aborted by terminal user request.		
ABT 40		Attempt to use duplicate or nested $ADD cards.		
ABT 42		No room on disc for spool file.		
ABT 43		Irrecoverable disc I/O error.		
ABT 44		Spool assignment made to device which cannot be spooled out to.		
ABT 46		Read access to disc file prohibited for this user.		
ABT 47		Write access to disc file prohibited for this user.		
ABT 50		Background Job time-limit exceeded.		
ABT 51		Too many currently open RTP devices.		
ABT .	DMS	Operator of TTY aborted current program with X-OFF key.	No	
ACCTG DISC FULL	DMS	Accounting disc pack full. Unable to create new accounting file. Accounting records will be lost and message will repeat at 20 second intervals until space becomes available.		
ACCTG INOPERTIV	DMS	Accounting disc pack full. Unable to create ACCUSR files. The /JC command may be used to run background jobs without user number in correcting this situation. When room has been created on the pack, the system should be re-booted.		
ASSEMB 1	Macro Assembler	Symbol table overflow.	Yes	Abort
ASSEMB 2	Macro Assembler	Greater than 25 common block names	Yes	Abort
ASSEMB 3	Macro Assembler	Error on skip pseudo operation.	Yes	Abort

System Error Messages (Cont'd)

Message	Generated by	Meaning	Hold Message ?	Action taken when released
ASSEMB 4	Macro Assembler	Excess number of ESKP commands.	Yes	Abort
ASSEMB 5	Macro Assembler	Insufficient number of ESKP operations.	Yes	Abort
ASSEMB 6	Macro Assembler	Macro expansion overflow.	Yes	Abort
BSF CR	DMS	Operator should backspace file on card reader.	Yes	Program Continues
BSF TR	DMS	Operator should backspace file on paper tape reader.	Yes	Program Continues
BSO	DEBUG	Break stack overflow.	No	
BSR+1 CR	DMS	Operator should backspace one card plus one record on card reader.	Yes	Program Continues
BSR CR	DMS	Operator should backspace one record on card reader.	Yes	Program Continues
BSR TR	DMS	Operator should backspace one record on tape reader.	Yes	Program Continues
CD nn	ACRONIM	Error codes below.	No	
CD 1		Syntax error.		
CD 2		Dynamic core unavailable.		
CD 3		Too many concurrently active disc files.		
CD 4		Cannot copy to CRT terminal.		
CD 5		First output device or file name invalid.		
CD 6		Second output device or file name invalid.		
CD 7		Third output device or file name invalid.		
CD 8		Invalid command code.		
CD 9		BASIC statement number error.		
CD 10		No area is in edit mode.		
CD 11		Not in BASIC mode.		
CD 12		Cannot do I/O to another interactive terminal.		
CD 13		Cannot copy an "area" from non-disc device.		
CD 14		Initialization request must be on first card.		
CD 15		Unexpected end-of-area.		
CD 16		Disc area overflow on destination file.		
CD 17		Invalid standard scratch area name.		
CD 18		More than 8 numbers on TV statement.		
CD 19		Cannot change processor modes while editing.		
CD 20		Invalid variable on "mode value" statement.		
CD 21		BASIC work area "C1" not present.		
CD 22		Disc area name already exists, or not enough space on disc.		
CD 23		BASIC statement number not in 1-9999 range.		
CD 24		Edit area full: update needed.		
CD 25		RC area full: update needed.		
CD 26		Cannot specify RC size as zero.		
CD 27		No edit area currently defined.		
CD 28		Edit area too small - system error.		
CD 29		Edit area not on disc.		
CD 30		Insufficient disc space to create scratch area.		
CD 31		Output disc area overflow on update command.		
CD 32		Core allocation problems: Delete edit area name then rename TP area to old edit area name.		
CD 33		Run name not same as edit area name.		
CD 34		Function not allowed in BASIC mode.		
CD 35		Requested disc area(s) not on disc.		
CD 36		New disc name already in use.		
CD 37		Cannot delete or rename standard terminal scratch areas.		
CD 38		Disc area is nonexistent.		
CD 39		No disc area name supplied where required.		
CD 40		No list or view area defined.		
CD 41		Non-BASIC record found in disc area.		

System Error Messages (Cont'd)

Message	Generated by	Meaning	Hold Message ?	Action taken when released
CD 42		Request to list or view beyond end of disc area.		
CD 43		Cannot delete "output area" disc area.		
CD 44		Key (password) must be alphabetic.		
CD 45		Edit requests for records past end of disc area ignored.		
CD 46		Cannot delete or change line number 0.		
CD 47		Too many (more than 7) arguments on CR command.		
CD 48		Specified text not found on search request.		
CD 49		Column number not in 1-80 range.		
CD 50		Specified text cannot be found: not enough room in search area.		
CD 51		Cannot edit on end-of-file record.		
CD 52		Requested text not found in edit record.		
CD 53		No "output area" defined.		
CD 54		Edit line not defined.		
CD 55		Cannot insert end-of-file line in edit line mode.		
CD 56		Requested word in line does not exist.		
CD 57		Cannot delete across tab field.		
CD 58		Non-existent register(s).		
CD 59		Register(s) not large enough for given text.		
CD 60		Command only valid when Acronim control is from disc area.		
CD 61		Non-numeric values in number register.		
CD 62		Requested deconcatenated text not found.		
CD 63		More than 80 characters building text.		
CD 64		List or view values out of range.		
CD 65		Name not in disc area table.		
CD 66		Register(s) requested are beyond maximum.		
CD 67		Disc area does not contain valid $JOB card as first record.		
CD 68		Invalid $JOB card parameter.		
CD 69		Invalid on non-existent user-number.		
CD 70		Invalid output device on $JOB card.		
CD 71		Terminal already on. $ON treated as $OFF. Reenter $ON.		
CD 72		Invalid user-number.		
CD 73		Edit record greater than 32767.		
CD 74		User number on $ON statement has not been entered.		
CD 75		Invalid password.		
CD 76		Old area gone or new area already exists.		
CD 77		File being updated does not have proper access bits. Rename TP to new name.		
CD 78		Cannot delete edit area file.		
CD 79		Register setting command is invalid.		
CD 80		Buffer size too large (>999 characters), too small (<10 characters) or from a non-interactive device.		
CD 81		Illegal PDN as first parameter on "SM" card		
CD 82		Illegal optional parameter on "SM" card		
CD 83		Transport was not specified on "SM" card		
CD 84		Non-existant disc area to be spooled		
CD 85		Invalid PDN as spool out device.		
CHECKSUM ERROR	Job Control INCLUDE	Checksum error on binary input record.	Yes	Reread
CHECKSUM ERROR	$FILEMA-Restore	Checksum error on binary input record.	Yes	Reread
CORE DICT FULL	DMS	The in-core-directory of MDD entries is full.		
CORE NOT AVAIL	$FILEMA-DMAP	Insufficient core available for map.	No	
CSE	SGS	Checksum Error in Load		
DEV XX ERROR YY	DMS	Physical I/O error on device XX. See below.	Yes	
DEV XX ERROR 01		Device XX not "ON-LINE".	Yes	Retry
DEV XX ERROR 02		Read error (checksum, parity, etc.) on device XX.	Yes	Retry

System Error Messages (Cont'd)

Message	Generated By	Meaning	Hold Message ?	Action when when released
DEV XX ERROR 03		Write error on device XX.	Yes	Retry
DEV XX ERROR 04		Paper tape reader gate open.	Yes	Retry
DEV XX ERROR 05		Paper tape punch paper low.	Yes	Retry
DEV XX ERROR 06		Card reader command buffer full.	Yes	Retry
DEV XX ERROR 07		Card reader stacker full.	Yes	Retry
DEV XX ERROR 10		Output device trouble.	Yes	Retry
DEV XX ERROR 11		File protected.	Yes	Retry
DEV XX ERROR 12		Irrecoverable disc error.		
DISC ERR XX	DMS	Disc I/O error on disc XX.	Yes	Abort
DISC READ ERROR	DMS	Unrecoverable disc read error.	No	
DISC WRITE ERROR	DMS	Unrecoverable disc write error.	No	
DMS NOT ON FILE	FILEMA-REPSYS	The module on LFN 4 was not DMS.	No	
DMS SA ERROR	FILEMA-REPSYS	The starting address was not zero.	No	
DMS TOO LARGE	FILEMA-REPSYS	Now DMS does not fit in the current resident DMS area.	No	
ELF	EDITLF	Insufficient background area to build external definition tables.	No	
ELF 02	EDITLF	Invalid control statement.	No	
ELF 03	EDITLF	An external definition is missing from the beginning of a module being added to library file.	No	
ELF 04	EDITLF	Attempt to add module which duplicates already existing name to library file.	No	
ELF 05	EDITLF	Specified name not on link ready file.	No	
ELF 06	EDITLF	Word count not complete on binary read request	No	
ELF 07	EDITLF	Modules being added will not fit on current library file.	No	
ELF 08	EDITLF	Name not specified where required.	No	
ELF 09	EDITLF	Specified name is not on library file.	No	
ELF 10	EDITLF	An attempt is being made to add modules which exceeds 1000 module limit.	No	
ELF 11	EDITLF	An invalid loader code was encountered on input file.	No	
ELF 12	EDITLF	A source program error was encountered on input file.	No	
ELF 13	EDITLF	An end-of-file was encountered at an improper position on library file.	No	
ELF 14	EDITLF	An end-of-file was encountered at an improper position on link ready file.	No	
ELF 15	EDITLF	A checksum error was encountered on a library file record.	No	
ELF 16	EDITLF	A checksum error was encountered on a link ready file record.	No	
ELF 17	EDITLF	An end-of-file is present at start of library file.	No	
ELF 18	EDITLF	An end-of-file is present at start of link ready file.	No	
ELF 19	EDITLF	Work file (LFN'16) for ORDER statement is too small.	No	
EOF..	DMS	An end-of-file was written to the output device.	No	
EOT Tx	DMS	End-of-tape detected on Transport X.	Yes	Program Continues
ERR File-xxxxxx	DMS	Irrecoverable disc error on file xxxxxx.	No	
FER 01	Fortran Library	Square root of a negative number.		
FER 02	Fortran Library	Overflow in conversion from real to integer.		
FER 03	Fortran Library	Division by zero.		
FER 04	Fortran Library	Arithmetic underflow.		
FER 05	Fortran Library	Arithmetic overflow.		
FER 06	Fortran Library	SIN or COS of a number so large that all significance is lost.		
FER 07	Fortran Library	ATAN2 or DATAN2 called with both arguments zero.		
FER 08	Fortran Library	Logarithm of zero or a negative number.		
FER 09	Fortran Library	Underflow during EXP or DEXP.		
FER 10	Fortran Library	Overflow during EXP or DEXP.		
FER 11	Fortran Library	Underflow during exponentiation (X**Y).		
FER 12	Fortran Library	Overflow during exponentiation (X**Y).		
FER 13	Fortran Library	Exponentiation error (0**0 or neg**0).		
FER 14	Fortran Library	Exponentiation error (0**neg).		

System Error Messages (Cont'd)

Message	Generated by	Meaning	Hold Message ?	Action taken when released
FER 15	Fortran Library	Exponentiation error (neg**neg or neg**pos).		
FER 41	Fortran Library	Invalid output FORMAT specification.		
FER 42	Fortran Library	Invalid input FORMAT specification.		
FER 43	Fortran Library	Illegal character during input.		
FER 44	Fortran Library	Underflow during numeric input conversion.		
FER 45	Fortran Library	Overflow during numeric input conversion.		
FER 46	Fortran Library	More than 10 BUFFER IN or OUT files.		
FER 47	Fortran Library	More than 10 files were defined via the DEFINE FILE statement.		
FER 48	Fortran Library	Attempted READ, WRITE, or FIND a record on an undefined random access file.		
FER 49	Fortran Library	Attempted READ, WRITE, or FIND a record number larger than the defined number of records in the random access file.		
FG ERR 2xxxxxx	DMS	Attempts to initiate nonexistent foreground program named xxxxxx.	No	
FG ERR 3xxxxxx	DMS	Too many programs already active. Table full.	No	
FG ERR 4xxxxxx	DMS	Foreground program too large.		
FG ERR 5xxxxxx	DMS	Attempt to load invalid file type into foreground.	No	
FILE ISN'T I.S.	Index Sequential Package	The disc file has never been initialized with an index sequential initialization request.	Yes	Request Ignored
FILE NOT THERE	INCLUDE	Attempt to include nonexistent file.	No	
FILE NOT THERE	FILEMA	Attempt to SAVE nonexistent file.	No	
FILE NOT THERE	DMS	File name does not start with alphabetic character.	No	
FILE NON-EXIST	DMS	Attempt to delete nonexistent disc file.	No	
FILE SIZE ERROR	FILEMA	Invalid file size parameter.	No	
FILE TYPE ERROR	FILEMA	Invalid file type parameter.	No	
FILE IN USE	DMS	Attempt to delete a file which is open to another program.	No	
FORMAT ERROR	FILEMA	Improper parameter sequence on statement.	No	
ICS	DEBUG	Invalid Control Statement.	No	
ICS	SGS	Invalid Control Statement.		
ILR	SGS	Invalid Load Request.		
INPUT SPOOL	DMS	Input spool file of insufficient size to hold input job. Or, user "X-OFF" ed the spooled input.	No	
INSUF FG SPACE	DMS	Resident foreground area is not large enough to load all programs typed as resident foreground.		
INV FILE TYPE	INCLUDE	Attempt to "include" load module file.	No	
INVAL FUNCTION	FILEMA	Unrecognized command.	No	
INVALID PRIORITY FIELD	DMS	Invalid priority value on SJOB statement.	No	
INVALID SJOB CARD SYNTAX	DMS	Invalid format of SJOB statement.	No	
INVALID OUTPUT DEVICE	DMS	Invalid spool output device specification on SJOB statement.	No	
INVALID USER NUMBER	DMS	Invalid user number on SJOB statement.	No	
JCL ERROR	Job Control	Unrecognized or invalid parameter or command on Job Control statement.	Yes Console input only	Reread
KEY TOO LARGE	Index Sequential Package	Key size of greater than 111 words requested.	Yes	Request Ignored
LCT xx [...]	Cataloger	Link Cataloger error message. See below:	No	
LCT 01		Invalid control statement.		
LCT 02		A "TYPE" control statement was encountered after an ASSIGN control statement.		
LCT 03		An invalid parameter was encountered in a "TYPE" control statement.		
LCT 04		Invalid delimiter following a parameter on a control statement.		
LCT 05		An end-jump-relative code was encountered within an overlay segment.		
LCT 06		File access specification on a "NAME" control statement is not "R", "W", or "D".		

System Error Messages (Cont'd)

Message	Generated by	Meaning	Hold Message ?	Action taken when released
OC 10		File not there.		
OC 11		Program not there.		
OC 12		Program of wrong type.		
OC 13		Cannot change permanent assignment.		
OC 14		No room in Assign table.		
OC 15		Device is not a spooled terminal device.		
OC 16		Invalid priority value.		
OC 17		Invalid time.		
OC 18		Terminal not in output mode.		
OC 19		Invalid pack number.		
OC 20		Invalid disc number.		
OC 21		Pack no there.		
OC 22		Invalid size field.		
OC 23		Space not available.		
OC 24		Missing or invalid record count value.		
OC 25		SJOB not first line of file.		
OC 26		Cannot assign re-entrant foreground.		
OC 27		Logical file in use.		
OC 28		File permanently assigned.		
OC 29		Timer schedule full.		
OC 30		Program already scheduled.		
OC 31		Invalid interrupt designation.		
OC 32		Program already connected.		
OC 33		No program is connected to interrupt.		
OC 34		Word number must be less than 112.		
OC 35		"FROMDATA" does not match disc data.		
OC 36		Cannot modify "EOF" sector.		
OC 37		Address Invalid.		
OC 38		User number not present where required or not valid.		
OC 39		User number already exists.		
OC 40		Program not on Timer schedule.		
OC 41		Invalid date.		
PASSWORD ERROR	DMS	Password does not have first character alphabetic or is incorrect on file operation.	No	
PAUSE ERROR	Index Sequential Utility	Error - See logical file 6 for message.	Yes	Command ignored
POWER FAIL	DMS	Power fail/restore activated - reboot is necessary.	No	
RAR	SGS	Restricted Address Reference		
RE-ENTER SPOOL JOB	DMS	The spool input job is cancelled due to a previously stated error on the SJOB statement.	No	
REL ORG ERROR	FILEMA	Invalid relative origin specified on DUMP statement.	No	
RPF CR	DMS	Operator should reposition file on card reader.	Yes	Processing Continues
RPF TR	DMS	Operator should reposition file on paper tape reader.	Yes	Processing Continues
RPLIST OVERFLOW	DMS	Resident foreground program list is full.		
SAU 00	DMS	Unrecognized SAU trap.	No	
SAU 01	DMS	Square Root of a Negative Number.	No	
SAU 02	DMS	Overflow during FIX.	No	
SAU 03	DMS	Division by Zero.	No	
SAU 04	DMS	Arithmetic Underflow.	No	
SAU 05	DMS	Arithmetic Overflow.	No	
SORT INHIBITED	Sort/Merge	Error - See logical file 06 for message.	Yes	Program Execution
TABLE OVERFLOW, RELEASE TO PRINT TABLE	Cross Reference	The internal table in XREF is full.	Yes	Table printed and execution continues
TOO MANY FILES	Index Sequential Package	More than 10 unique files are active at one time.	Yes	Request ignored
UNDEFINED LFN	Index Sequential Package	The specified logical file number was never specified on an ISINIT call.	Yes	Request ignored

System Error Messages (Cont'd)

Message	Generated by	Meaning	Hold Message ?	Action taken when released
UTL XX	Utility Package	See below for meanings.	Yes	
UTL 01	Utility	Invalid command identifier character.	Yes	Command Ignored
UTL 02	Utility	Invalid command.	Yes	Command Ignored
UTL 03	Utility	Invalid command parameter.	Yes	Command Ignored
UTL 04	Utility	Invalid command delimiter.	Yes	Command Ignored
UTL 05	Utility	Sequence error.	Yes	Command Ignored
UTL 06	Utility	Invalid ASCII equivalent of BCD character.	Yes	Command Ignored
UTL 07	Utility	Checksum error on Binary Input Device.	Yes	Record will be backspaced.
UTL 08	Utility	Checksum error on Alternate Binary Input Device.	Yes	Record will be backspaced.
UTL 09	Utility	END$ record encountered on Binary Input Device.	Yes	Prepare Input on Binary Input Device & Release
UTL 10	Utility	END$ record encountered on Alternate Binary Input Device.	Yes	Prepare input on alternate Binary Input Device & Release.
UTL 11	Utility	More than 64 ANSCII equivalence of BCD code have been input.	Yes	Command Ignored
WORK TCO SMALL	Index Sequen-	Insufficient working buffer space.	Yes	Request Ignored
XXXXXX NOT ENTERED	DMS	XXXXXX was not entered in the in-core-directory of MDD entries.		
XXXXXX NOT LOADED	DMS	Resident foreground program XXXXXX was not loaded.		
DISC# NN NOT READABLE	SYSINT	Pack on disc NN cannot be read		
ERROR : PK NNN	DMS	A disc error has occurred on pack NNN, the sector number follows on the next line.		
S#NNNN(OVERFLW)	DMS	Spool file S#NNNN overflowed when it was written.		

APPENDIX D
SYSTEM FLAGS AND OPTIONS

SYSTEM FLAGS AND OPTIONS

This appendix lists the standard bits of the $FLAGS and $OPTIONS (see Paragraphs 6-2.6 and 6-2.7) used by Datacraft processors and by DMS. Table D-1 lists the $OPTION bits. The $FLAGS bits are normally used in the assembly of programs to produce different versions of what is basically the same program. Table D-2 lists these assembly $FLAGS values.

Table D-1. $OPTIONS Bits

Bit	Used In	Meaning
0	Macro Assembler	Set to indicate write pass one to disc and read for pass two.
1	Macro Assembler	Set for pass two only.
3	Optimizer	Set to produce expanded (object code) listing of optimized code.
4	FORTRAN Compiler	Set forces all type REAL variables to type DOUBLE PRECISION.
5	FORTRAN Compiler	Set to produce expanded (object code) listing
5	RPG II Compiler	Set to produce expanded (object code) listing
6	Link Cataloger	Set to output hold message upon encountering END$
8	FORTRAN Compiler	Set to suppress map listing
9	FORTRAN Compiler	Set to produce code to operate in upper memory map
9	RPG II Compiler	Set to produce DEBUG command code. Reset to treat DEBUG commands as comments.
16*	Not used	
17*	Not used	
18*	Not used	
19	FORTRAN Compiler	Set to optimize FORTRAN code
20*	Not used	
21	FORTRAN Compiler	Set to move sequence number to left side of source listing
22*	FORTRAN Compiler and I/O Library	Set to suppress internal translation of 026 to 029 source code.
23*	DMS	Set to cause Card Punch Handler to convert symbolic data to 026 code instead of 029.

NOTE: OPTION 23 is checked by the Card Punch Handler only at the time the punch is opened. If it is desired to change the option, the Punch must be opened after the change is made for it to have any effect.

* Options 16, 17, 18, 20, 22 and 23 may be cataloged with a program through the use of the Link Cataloger OPTION statement.

Table D-2. $FLAGS Bits

Bit	Used in Assembly of	Meaning
0	Fortran Compiler	Set - Always execute DO loops at least once. Reset - Skip execution of DO loop if limits satisfied on initial entry.
1	Fortran Compiler	Set - Produce extended compiler. Reset - Produce basic compiler.
5	Fortran Compiler	Set - Produce SAU Compiler. Reset - Produce non-SAU Compiler.
5	Macro Assembler	Set - Produce Macro Assembler. Reset - Produce basic assembler.
17	DMS-61613-01	Set - 50 Hz power. Reset - 60 Hz power.
18	DMS - Modules: 61601-01 61607-01 61608-01 61610-01 61612-01 61613-01 61614-01 61616-01 61619-01 61622-01 61624-01 61625-01 61626-01 61627-01 61629-01 61640-02 61663-00 61670-00 61677-01 61685-00 61690-01 61695-00 61695-01 61695-02 61695-03 61695-04 61695-05	Set - Produce accounting version of system. Reset - Produce non-accounting version.
19	DMS-OPCOM 2 61695-02	Set - Produce Remote job entry version. Reset - Produce Non-RJE version.

Table D-2. $FLAGS Bits (Cont'd.)

Bit	Used in Assembly of	Meaning
20	DMS - 61606-01 61607-01 61608-01 61609-01 61614-01 61619-01 61670-01 61695-00 61695-01 61695-02 61695-03 61695-04 61695-05	Set - Produce Spooled/Interactive version. Reset - Produce Non-Spooled/Non-Interactive version.
21	DMS - SYSDAT 61606-01	Set - Configure for 6024/4 or 6024/5. Reset - Configure for 6024/1 or 6024/3.
22	DMS - Executive 61607-01	Set - Produce code for bit processor. Reset - Do not produce code for bit processor.
23	DMS - Executive 61607-01 Executive Traps 61612-01	Set - Produce code for Scientific Arithmetic Unit (SAU). Reset - Do not produce code for SAU.

APPENDIX E
SYSTEM DATA MODULE

```
07-13-73          SYSTEM DATA MODULE (DMS)          61606-01  01.070173      PAGE     1

                * THE SYSTEM DATA MODULE (SYSDAT) IS THE "MAIN" MODULE OF A DMS SYSTEM.
                * PROPER CONFIGURATION OF THIS MODULE IS NECESSARY FOR A CONFIGURED
                * SYSTEM GENERATION.
                *
                *
                *
                ****************  T A B L E   I N D E X  ***************************
                * EXTERNAL DEFINITION TABLE                                  TABLE A
                * EXTERNAL EQUIVALENCE TABLES                                TABLE B
                * MONITOR SERVICE LINKAGE TABLE                              TABLE C
                * SYSTEM ABSOLUTE LINKAGE TABLE                              TABLE D
                * INTERRUPT LINKAGE TABLE                                    TABLE E
                * DISPATCHER CONTROL TABLE                                   TABLE F
                * RESIDENT PROGRAM LIST                                      TABLE G
                * TIMER SCHEDULER TABLE                                      TABLE H
                * DISC DEFINITION TABLE                                      TABLE I
                * MISCELLANEOUS PARAMETERS TABLE                             TABLE J
                * DISC I/O QUEUE                                             TABLE K
                * PERIPHERAL DEVICE COORDINATION TABLE                       TABLE L
                * TERMINAL CONTROL BLOCK TABLE                               TABLE M
                * MAGNETIC TAPE OPTIONS TABLE                                TABLE N
                * SYSTEM SERVICE IDENTIFICATION TABLE                        TABLE O
                * DYNAMIC CELL POOL DEFINITION TABLE                         TABLE P
                * EXTERNAL INTERRUPT DEFINITION BLOCKS                       TABLE Q
                * DISC DICTIONARY IN CORE TABLE                              TABLE R
                ********************************************************************
                *
                *
                *
                * NOTE: ASSEMBLE WITH FLAGS 21 SET  FOR 6024/5 SYSTEMS
                *                AND FLAGS 21 ZERO FOR 6024/1,3 SYSTEMS
                *       ASSEMBLE WITH FLAGS 20 SET FOR UNSPOOLED/NON-INTERACTIVE DMS
                *       ASSEMBLE WITH FLAGS 20 ZERO FOR SPOOLED/INTERACTIVE DMS
                *       ASSEMBLE WITH FLAGS 18 SET   FOR ACCOUNTING SYSTEM
                *       ASSEMBLE WITH FLAGS 18 RESET FOR NON-ACCOUNTING SYSTEM
```

** TABLE A

```
38                      * EXTERNAL DEFINITION TABLE
39                      *
40   00000000 2                 XDEF  SYSDAT,ABORT    .MODULE NAME & INITIAL ENTRY ADDRESS
41   00000046 2                 XDEF  ALLOLO,ALLOLO   .ALLOCATOR LOW MEMORY DEFINITION
42   00000037 2                 XDEF  ALLOHI,ALLOHI   .ALLOCATOR HIGH MEMORY DEFINITION BLOCK
43   00000054 2                 XDEF  BAKLO,BAKLO     .BAKGND LOW MEMORY ADDRESS (LOWEST EXECUTABLE INST)
44   00000211 2                 XDEF  BAKHI,BAKHI     .BAKGND HIGH MEMORY ADDRESS
45   00000211 2                 XDEF  BG,HI,BAKHI
46   00000047 2                 XDEF  NRFLO,NRFLO     .NON-RESIDENT FOREGROUND LOW MEMORY ADDRESS
47   00000044 2                 XDEF  NRFHI,NRFHI     .NON-RESIDENT FOREGROUND HIGH MEMORY ADDRESS
48   00000046 2                 XDEF  COMLO,COMLO     .SYSTEM COMMON LOW MEMORY ADDRESS
49   00000201 2                 XDEF  CURNT,CURNT     .ADDRESS OF CURRENTLY ACTIVE PSA
50   00000202 2                 XDEF  CAPLE,CAPLE     .ADDRESS OF CURRENT ACTIVE APL ENTRY
51   00000203 2                 XDEF  APL,DCT         .START OF ACTIVE PROGRAM LIST
52   00000214 2                 XDEF  OCBLOK,OCBLOK   .APL ENTRY FOR OPERATOR COMMUNICATIONS PROCESSOR
53   00000206 2                 XDEF  CKPTFL,CKPTFL   .CHECKPOINT FLAG
54   00000236 2                 XDEF  DCTFR,DCTFREE   .DISPATCHER CONTROL TABLE FREE ENTRY LIST
55   00000313 2                 XDEF  RPLIST,RPLIST   .RESIDENT FOREGROUND PROGRAM LIST
56   00000200 2                 XDEF  DISINT,DISINT   .DISPATCHER INTERRUPT UNITARY CONTROL MASK
57   00000520 2                 XDEF  PDCT,PDCT       .PERIPHERAL DEVICE COORDINATION TABLE-INITIAL LOC.
58   00000521 2                 XDEF  PDCTW1,PDCT+1   .PERIPHERAL DEVICE COORDINATION TABLE-WORD 1
59   00000522 2                 XDEF  PDCTW2,PDCT+2   .PERIPHERAL DEVICE COORDINATION TABLE-WORD 2
60   00000576 2                 XDEF  S,CHA,CHA       .CONVERT HOLLERITH TO ASCII
61   00000577 2                 XDEF  S,CAH,CAH       .CONVERT ASCII TO HOLLERITH
62   00000600 2                 XDEF  S,CBA,CBA       .CONVERT BCD TO ASCII - SEVEN TRACK TAPE
63   00000601 2                 XDEF  S,CAB,CAB       .CONVERT ASCII TO BCD - SEVEN TRACK TAPE
64   00000602 2                 XDEF  S,CEA,CEA       .CONVERT EBCD TO ASCII - NINE TRACK TAPE
65   00000603 2                 XDEF  S,CAE,CAE       .CONVERT ASCII TO EBCD - NINE TRACK TAPE
66   00000374 2                 XDEF  D/NUMB,D/NUMB   .NUMBER OF DISC DRIVES
67   00000375 2                 XDEF  D/DEFT,D/DEFT   .DISC DEFINITION TABLE
68   00000436 2                 XDEF  DQFULL,D/QFULL  .DISC QUEUE FULL FLAG
69   00000437 2                 XDEF  DAVAIL,D/AVAIL  .DISC QUEUE AVAILABLE ENTRY POINTER
70   00000571 2                 XDEF  CRABF,CRABF     .CARD READER "A" BUSY FLAG
71   00000574 2                 XDEF  LPABF,LPABF     .LINE PRINTER "A" BUSY FLAG
72   00000572 2                 XDEF  CPABF,CPABF     .CARD PUNCH "A" BUSY FLAG
73   00000573 2                 XDEF  CPBBF,CPBBF     .CARD PUNCH B BUSY FLAG
74   00000334 2                 XDEF  MAXPRO,MAXPRO   .MAXIMUM PROGRAM SIZE (FOREGROUND)
75   00000575 2                 XDEF  MCABF,MCABF     .MAGTAPE BUSY FLAG
76   00001005 2                 XDEF  MCATOT,TOT      .MAGTAPE OPTIONS TABLE
77   00000045 2                 XDEF  MEMSIZ,MEMSIZ   .TOTAL MEMORY SIZE (MINUS ONE)
78   00000604 2                 XDEF  TTYIN,TTYIN     .CONSOLE ASR INPUT INTERRUPT MASK
79   00000605 2                 XDEF  TTYOUT,TTYOUT   .CONSOLE ASR OUTPUT INTERRUPT MASK
80   00001032 2                 XDEF  SSTAR,SSTAR     .SYSTEM SERVICES TABLE
81   00000570 2                 XDEF  TPABF,TPABF     .PAPER TAPE PUNCH "A" BUSY FLAG
82   00000567 2                 XDEF  TRABF,TRABF     .PAPER TAPE READER "A" BUSY FLAG
```

```
                    * EXTERNAL DEFINITION TABLE (CONTINUED)                    * TABLE A
                    *
                          XDEF  NAME1,NAME1     .1ST 3 CHAR. PROG NAME-TIMER SCHEDULE
00000336 2                XDEF  NAME2,NAME2     .LAST 3 CHAR. PROG NAME-TIMER SCHEDULE
00000343 2                XDEF  LEVEL,LEVEL     .PROG. EXECUTION PRIORITY LEVEL-TIMER SCHEDULE
00000350 2                XDEF  INTRV,INTRV     .INTERVAL COUNTER-TIMER SCHEDULE
00000355 2                XDEF  INCTR,INCTR     .INTERVAL DEFINITION-TIMER SCHEDULE
00000362 2                XDEF  PARAM,PARAM     .PARAMETERS FOR TIMER SCHEDULED PROGRAMS
00000367 2                XDEF  MXTPRG,MXTPRG   .MAXIMUM NUMBER OF TIMER SCHEDULED PROGRAMS
00000335 2                XDEF  EXIOBS,EXIOBS   .EXTERNAL INTERRUPT DEFINITION BLOCKS
00001220 2                XDEF  DELTIM,DELTIM   .DELTA TIME=ONE SECOND
00000606 2                XDEF  CELLS,CELLS     .POOL OF AVAILABLE 4-WORD CELLS
00001053 2                XDEF  BGSPQP,BGSPQP   .BACKGROUND SPOOL QUEUE POINTER
00000424 2                XDEF  BGIDLE,BGIDLE   .BACKGROUND IDLE FLAG
00000423 2                SKFS  B20             .ONLY IN SPOOLED VERSION
                          XDEF  IOXAPL,IOXAPL   .I/O EXEC APL ENTRY
00000225 2                XDEF  TCB,TCB         .TERMINAL CONTROL BLOCKS
00000607 2                XDEF  EOTCB,EOTCB     .ADDR OF END OF TCB AREA
00001005 2                ESKP
                          XDEF  DICT,DICT       .IN-CORE DICTIONARY ADDRESS
00001243 2                XDEF  DUMPBG,DUMPBG   .BACKGROUND CORE DUMP ACTIVE FLAG
00000431 2                XDEF  ADUMP,ADUMP     .BACKGROUND ABORT DUMP REQUEST FLAG
00000432 2
```

```
107                     * EXTERNAL EQUIVALENCE TABLES                                   ** TABLE B
108                     *
109                     * REAL TIME CLOCK
110         03200000 6          XEQV  TMERIL,!20000000 ,INTERRUPT MASK WORD
111         03200000 6          XEQV  UATMER,!600100 ,UNITARY ARM INSTRUCTION
112         03200000 6          XEQV  UETMER,!620100 ,UNITARY ENABLE INSTRUCTION
113         03200000 6          XEQV  UITMER,!630100 ,UNITARILY INHIBIT INSTRUCTION
114                     * DISPATCHER INTERRUPT CONTROL
115         03200000 6          XEQV  ALLOW,!00620100 ,UNITARY ENABLE
116         03200000 6          XEQV  PREVNT,!00630100 ,UNITARY INHIBIT
117                             SKFS  B21
118                     *                             ,TRIGGER /1 OR /3
119         03200000 6          XEQV  TINST1,!62000000 ,NOP
120         03200000 6          XEQV  TINST2,!62500400 ,TOA B8
121         03200000 6          XEQV  TINST3,!00714000 ,OAW #TTY
122                             FSKP
123                             SKFZ  B21
124                     *                             ,TRIGGER /5
125                             XEQV  TINST1,!04000200 ,TME DISINT
126                             XEQV  TINST2,!00300020 ,TZA
127                             XEQV  TINST3,!00644100 ,TD4
128                             FSKP
129                     * DISC CONTROLLER PARAMETERS
130         03200000 6          XEQV  DCACU,!0500     ,MOVING HEAD DISC CONTROLLER C/U=5/0
131         03200000 6          XEQV  DCAIL,!00040000  ,DISC CONTROLLER "A" INTERRUPT LEVEL = 14
132         03200000 6          XEQV  UADCAI,!00600100 ,UNITARY ARM
133         03200000 6          XEQV  UEDCAI,!00620100 ,UNITARY ENABLE
134         03200000 6          XEQV  UDDCAI,!00610100 ,UNITARY DISARM
135         03200000 6          XEQV  UIDCAI,!00630100 ,UNITARY INHIBIT
136                     * CONSOLE ASR TELETYPE
137         03200000 6          XEQV  OPCOMD,!20      ,OPERATOR COMMUNICATIONS PDN
138         03200000 6          XEQV  RTOI,12         ,CONSOLE TTY INPUT INTERRUPT LEVEL
139         03200000 6          XEQV  RTOCU,!0000     ,CHANNEL/UNIT NUMBER
140                     * CARD READER "A"
141         03200000 6          XEQV  CRACU,!0400     ,CARD READER-A C/U=4/0
142         03200000 6          XEQV  CRAIL,!400      ,INTERRUPT LEVEL = 8
143         03200000 6          XEQV  UACRAI,!00600100   ,UNITARY ARM CRA
144         03200000 6          XEQV  UDCRAI,!00610100 ,UNITARY DISABLE
145         03200000 6          XEQV  UECRAI,!00620100   ,UNITARY ENABLE CRA
146         03200000 6          XEQV  UICRAI,!00630100 ,UNITARY INHIBIT
147                     * CARD PUNCH "A"
148         03200000 6          XEQV  CPACU,!0600     ,CARD PUNCH-A C/U=6/0
149         03200000 6          XEQV  CPAIL,!1000     ,INTERRUPT LEVEL 9
150         03200000 6          XEQV  UACPAI,!00600100 ,UNITARY ARM
151         03200000 6          XEQV  UDCPAI,!00610100 ,UNITARY DISARM
152         03200000 6          XEQV  UECPAI,!00620100 ,UNITARY ENABLE
153         03200000 6          XEQV  UICPAI,!00630100 ,UNITARY INHIBIT
```

```
                        * EXTERNAL EQUIVALENCE TABLES (CONTINUED)                    * TABLE B
                        *
                        * CARD PUNCH "B"
03200000 6                      XEQV  CPBCU,'0401     .CARD PUNCH-B C/U=4/1
03200000 6                      XEQV  CPBIL,'4000000  .INTERRUPT LEVEL 20
03200000 6                      XEQV  UACPBI,'600100  .UNITARILY ARM
03200000 6                      XEQV  UDCPBI,'610100  .UNITARILY DISARM
03200000 6                      XEQV  UECPBI,'620100  .UNITARILY ENABLE
03200000 6                      XEQV  UICPBI,'630100  .UNITARILY INHIBIT
                        * LINE PRINTER "A"
03200000 6                      XEQV  LPACU,'0300     .LINE PRINTER-A C/U=3/0
03200000 6                      XEQV  LPAIL,'1000000  INTERRUPT LEVEL = 18
03200000 6                      XEQV  UALPAI,'00600100   .UNITARY ARM LPA
03200000 6                      XEQV  UDLPAI,'00610100 .UNITARY DISARM
03200000 6                      XEQV  UELPAI,'00620100   .UNITARY ENABLE LPA
03200000 6                      XEQV  UILPAI,'00630100 .UNITARY INHIBIT
                        * PAPER TAPE READER "A"
03200000 6                      XEQV  TRACU,'0100     .TAPE READER-A C/U=1/0
03200000 6                      XEQV  TRAIL,'200000   .INTERRUPT LEVEL = 16
03200000 6                      XEQV  UATRAI,'00600100 .UNITARY ARM
03200000 6                      XEQV  UDTRAI,'00610100 .UNITARY DISARM
03200000 6                      XEQV  UETRAI,'00620100 .UNITARY ENABLE
03200000 6                      XEQV  UITRAI,'00630100 .UNITARY INHIBIT
                        * PAPER TAPE PUNCH "A"
03200000 6                      XEQV  TPACU,'0101     .TAPE PUNCH-A C/U=1/1
03200000 6                      XEQV  TPAIL,'400000   .INTERRUPT LEVEL = 17
03200000 6                      XEQV  UATPAI,'00600100 .UNITARY ARM
03200000 6                      XEQV  UDTPAI,'00610100 .UNITARY DISARM
03200000 6                      XEQV  UETPAI,'00620100 .UNITARY ENABLE
03200000 6                      XEQV  UITPAI,'00630100 .UNITARY INHIBIT
                        * MAGNETIC TAPE CONTROLLER "A"
03200000 6                      XEQV  MCACU,'0700     .MAGTAPE CONTROLLER-A C/U=7/0
03200000 6                      XEQV  MCAIL,'100000   .INTERRUPT LEVEL = 15
03200000 6                      XEQV  UAMCAI,'00600100 .UNITARY ARM
03200000 6                      XEQV  UDMCAI,'00610100 .UNITARY DISARM
03200000 6                      XEQV  UEMCAI,'00620100 .UNITARY ENABLE
03200000 6                      XEQV  UIMCAI,'00630100 .UNITARY INHIBIT
                        * REMOTE TELETYPE-1
03200000 6                      XEQV  RT1CU,'0200     .CHANNEL/UNIT
03200000 6                      XEQV  RT1PIN,10       .REMOTE TTY PRIORITY INTERRUPT NUMBER
                        * REMOTE TELETYPE-2
03200000 6                      XEQV  RT2CU,'0201     .CHANNEL/UNIT
03200000 6                      XEQV  RT2PIN,11       .REMOTE TTY PRIORITY INTERRUPT NUMBER
                        * SYNCHRONOUS INTERFACE-A (1108)
03200000 6                      XEQV  SIACU,'0201     .CHANNEL/UNIT
03200000 6                      XEQV  SIAIL,'40       .INTERRUPT LEVEL = 5
03200000 6                      XEQV  UASIAI,'600100 .UNITARILY ARM
03200000 6                      XEQV  UESIAI,'620100 .UNITARILY ENABLE
```

```
                    * EXTERNAL EQUIVALENCE TABLES (CONTINUED)                    * TABLE B
                    *
                    * RESERVED DISC STORAGE PARAMETERS
                    *
                    *          FS SUFFIX MEANS FIRST SECTOR
                    *          SC SUFFIX MEANS SECTOR COUNT
                    *
03200000 6                XEQV  ADLFS,0          .ABSOLUTE DISC LOADER
03200000 6                XEQV  ADLSC,1
03200000 6                XEQV  BASFS,1          .BAKGND DEFAULT ASSIGNMENTS
03200000 6                XEQV  MDDFS,2          .MASTER DISC DIRECTORY
03200000 6                XEQV  MDDSC,60         .MASTER DISC DIRECTORY SECTOR COUNT
03200000 6                XEQV  SAM1FS,62        .SPACE ALLOCATION MAP FIRST SECTOR
03200000 6                XEQV  SAM1SC,2         .SAM SECTOR COUNT
03200000 6                XEQV  SAMNFS,2         .ADDRESS OF "SAM" ON NON-MASTER PACKS
03200000 6                XEQV  CPDISC,1         .CHECKPOINT DISC NUMBER
03200000 6                XEQV  CKPTFS,64        .CHECKPOINT FIRST SECTOR
03200000 6                XEQV  CKPTSC,601       .CKPT SECTOR COUNT
03200000 6                XEQV  SYSFS,665        .SYSTEM FIRST SECTOR
03200000 6                XEQV  SYSSC,160        .RESERVED SYSTEM SPACE
03200000 6                XEQV  MDDE/D,839       .MASTER DISC DIRECTORY-ENTRIES/DIRECTORY
03200000 6                XEQV  MDDE/S,14        .MASTER DISC DIRECTORY-ENTRIES/SECTOR
03200000 6                XEQV  MDDW/S,112       .MASTER DISC DIRECTORY-WORDS/SECTOR
                    * TIMER RELATED CONSTANTS
03200000 6                XEQV  MPC*10,10        .MICROSECONDS PER CYCLE * 10 FOR 6024/4 & 5
03200000 6                XEQV  CPS/10,100000    .CYCLES PER SECOND/10 FOR 6024/4 & 5
03200000 6                XEQV  CYC/MS,100       .CYCLES PER MILLISECOND/10 FOR 6024/4 & 5
                    * BACKGROUND DEFAULT OPTIONS
03200000 6                XEQV  BGDLDA,6         .DEFAULT LIST OUTPUT ASSIGNMENT
03200000 6                XEQV  BGDOPS,1201      .DEFAULT OPTIONS B0B7
03200000 6                XEQV  BGDLNS,55        .DEFAULT $LINES VALUE
03200000 6                XEQV  BGDFGS,0         .DEFAULT $FLAGS VALUE
03200000 6                XEQV  JOBMES,1         .BG JOB MESSAGE FLAG; SET TO TYPE MESSAGES
                    * BLOCKED I/O PARAMETERS
03200000 6                XEQV  BLOKSC,2         .BLOCK SECTOR COUNT
03200000 6                XEQV  BLOKWC,224       .BLOCK WORD COUNT = 2.* SECTOR SIZE
                    * SPOOL SYSTEM CONSTANTS
03200000 6                XEQV  SPPACK,1         .PACK NUMBER FOR SPOOL FILES
03200000 6                XEQV  SPLSIZ,200       .DEFAULT SPOOL FILE SIZE IN SECTORS
                          SKFZ  B18              .ACCOUNTING SYSTEM PARAMETERS
03200000 6                XEQV  AFSC,2           .ACCOUNTING FILE BUFFER SECTOR COUNT
03200000 6                XEQV  ABUF,224         .ACCOUNTING BUFFER SIZE (112*AFSC)
03200000 6                XEQV  LOUSR,1          .LOWEST VALID USER NUMBER IN SYSTEM
03200000 6                XEQV  HIUSR,3000       .HIGHEST VALID USER NUMBER IN SYSTEM
03200000 6                XEQV  AFSIZ,100        .SIZE OF ACCOUNTING FILES IN SECTORS
03200000 6                XEQV  ACPACK,1         .ACCOUNTING FILES PACK NUMBER
                          ESKP
```

```
252                     * MONITOR SERVICE LINKAGE TABLE                                  ** TABLE C
253                     *
254                     *                                ,FOR EACH LOCATION WHICH CAN BE ACCESSED
255                     *                                 BY A "BLUM", THIS TABLE CONTAINS A "BUC"
256                     *                                 TO THE APPROPRIATE SUBROUTINE.
257 000000 00200000 6           RORG  0
258 000000 21000000 3  ABORT   BUC   $SYINIT            ,REPLACED WITH "ABORT" ADDRESS BY SYINIT
259 000001 21000000 3  I/O     BUC   $IOC               ,REQUEST I/O FUNCTION
260 000002 21000000 3  EXIT    BUC   $EXIT*             ,TERMINATE EXECUTION
261 000003 21000000 3  HOLD    BUC   $IDLE              ,PRINT MESSAGE & SUSPEND EXECUTION
262 000004 21000000 3  CHAIN   BUC   $CLOR              ,LOAD CHAIN SEGMENT
263 000005 21000000 3  INFO    BUC   $INFOS             ,GET BACKGROUN INFORMATION
264 000006 21000000 3  WAIT    BUC   $WAITS             ,WAIT UNTILL FLAG IS SET (OR RESET)
265 000007 21000000 3  SFUNC   BUC   $SFUNCT            ,PERFORM SPECIAL SYSTEM FUNCTION
266 000010 21000000 3  CONV    BUC   $CONVRT            ,PERFORM DATA CONVERSION
267 000011 21000000 3  FROGS   BUC   $FROGS             ,PERFORM SPECIAL FOREGROUND SERVICE
268 000012 21000000 3  SYSTER  BUC   $STEMS             ,PERFORM RESTRICTED SYSTEM SERVICE
269 000013 21000000 3  TERMIN  BUC   $TERMIN            ,TERMINATE (EXIT WITHOUT DE-ALLOCATION)
270 000014 21000000 3  ASSIGN  BUC   $ASSIGN            ,DYNAMIC ASSIGN SERVICE
271 000015 21000000 3  DCM     BUC   $DCM               ,DYNAMIC CORE MANAGER
272 000016 21000000 3  UNTRAP  BUC   $UNTRAP            ,BACKGROUND $ CARD CONTROL
273 000017 21000000 3  FPACK   BUC   $FPACK             ,FIND PACK
274 000020 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
275 000021 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
276 000022 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
277 000023 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
278 000024 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
279 000025 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
280 000026 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
281 000027 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
282 000030 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
283 000031 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
284 000032 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
285 000033 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
286 000034 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
287 000035 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
288 000036 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
289 000037 21000000 3          BUC   $SERR              ,**PROGRAM ERROR**
```

```
291                     * SYSTEM ABSOLUTE LINKAGE TABLE                          ** TABLE D
292                     *
293                     *                              ,IF THE FOLLOWING EQUIVALENCES ARE SET
294                     *                               PROPERLY, THE REMAINDER OF THE TABLE
295                     *                               WILL BE AUTOMATICALLY COMPUTED.
296           00117777  MSIZE   EQIV  '117777          ,MEMORY SIZE = 40K
297           00000144  SCSIZE  EQIV  100              ,SYSTEM COMMON SIZE
298           00025370  NRFSIZE EQIV  11000            ,NON-RESIDENT FOREGROUND SIZE
299                     *
300 000040 00200040 6             RORG  '40
301 000040 00000001 3             DAC   $U$FLAG       '40,ADDRESS OF BACKGROUND CONTROL AREA
302 000041 00000001 3             DAC   $D/DEFT       '41,ADDRESS OF DISC DEFINITION TABLE
303 000042 00000001 3             DAC   $SSTAB        '42,ADDRESS OF SYSTEM SERVICE TABLE
304           00000037  ALLOHI   EQIV  ABORT+'37        ,IDENTIFIES THE START OF DUMMY ALLOCATOR LIST
305 000043 00000001 3             DAC   $ALLOLO       '43 ENTRY THAT RESERVES SYSCOM & RFS FG AREA
306 000044 00117633 0   NRFHI     DAC   MSIZE-SCSIZE  '44,NON-RESIDENT FOREGROUND HIGH
307 000045 00117777 0   MEMSIZ    DAC   MSIZE         '45,TOTAL MEMORY SIZE
308 000046 00117634 0   COMLO     DAC   MSIZE-SCSIZE+1 '46,SYSTEM COMMON LOW ADDRESS
309 000047 00072244 0   NRFLO     DAC   MSIZE-SCSIZE-NRFSIZE+1 '47,NON-RESIDENT FG LOW ADDRESS
310 000050 00000415 4             DAC   D/MDD           '50,ADDRESS OF MISC. PARAMETER LIST
311 000051 00000211 4             DAC   BAKHI         '51,ADDRESS OF CURRENT BAKGND HI
312           00000046  ALLOLO   EQIV  ABORT+'46        ,IDENTIFIES START OF DUMMY ALLOCATOR LIST ENTRY
313 000052 00370000 0             DATA  '370000       '52, WHICH RESERVES THE RESIDENT SYSTEM AREA.
314 000053 00000000 0             DAC   0             '53,RESIDENT SYSTEM LOW MEMORY ADDRESS
315 000054 00000001 3   BAKLO     DAC   $BSTAR        '54,START OF BAKGND EXECUTION AREA
316 000055 00000001 3   PDCTA     DAC   $PDCT         '55,ADDRESS OF PERIPH DEVICE COORD TABLE
317 000056 00001005 4   MTSUPA    DAC   TOT           '56,ADDRESS OF MAG TAPE UNIT SETUP TABLE
318 000057 00000001 3   LOADRG    DAC   $BGLOAD       '57,ADDRESS OF BAKGND PROGRAM LOAD ROUTINE
```

```
320                     * INTERRUPT LINKAGE TABLE                                   ** TABLE F
321 000060 00200060 6           RORG  '60
322                     *
323                     *          THIS TABLE CONTAINS A "BSL" TO THE INTERRUPT SERVICE
324                     *       ROUTINE FOR EACH INTERRUPT WHICH IS USED IN THIS
325                     *       SPECIFIC DC-6024 CONFIGURATION.  THE TABLE IS ORDERED
326                     *       SEQUENTIALLY WITH GROUP 0, LEVEL 0 FIRST AND GROUP 3,
327                     *       LEVEL 23 LAST.  THE SLOT FOR EACH UNUSED INTERRUPT
328                     *       SHOULD CONTAIN A "BSL $INVINT" SUCH THAT THE INVALID
329                     *       INTERRUPT ERROR ROUTINE IS ENTERED.
330                     *
331 000060 25000001 3           BSL   $PFAIL          .-000-POWER FAIL
332 000061 25000001 3           BSL   $PRESTR         .-001-POWER RESTORE
333 000062 25000001 3           BSL   $MEMPRO         .-002-MEMORY PROTECT VIOLATION
334 000063 25000001 3           BSL   $INSTRP         .-003-INSTRUCTION TRAP VIOLATION
335 000064 25000001 3           BSL   $STALL          .-004-STALL ALARM
336 000065 25000001 3           BSL   $TIMER          .005-INTERVAL TIMER
337 000066 25000001 3           BSL   $SAUOUT         .-006-SAU TRAP
338 000067 25000001 3           BSL   $ADTRAP         .-007-ADDRESS TRAP
339 000070 25000001 3           BSL   $INVINT         .-100-INVALID INTERRUPT
340 000071 25000001 3           BSL   $INVINT         .-101-INVALID INTERRUPT
341 000072 25000001 3           BSL   $INVINT         .-102-INVALID INTERRUPT
342 000073 25000001 3           BSL   $INVINT         .-103-INVALID INTERRUPT
343 000074 25000001 3           BSL   $INVINT         .-104-INVALID INTERRUPT
344 000075 25000001 3           BSL   $INVINT         .-105-INVALID INTERRUPT       LEVEL 5
345 000076 25000001 3           BSL   $INVINT         .-106-INVALID INTERRUPT       LEVEL 6
346 000077 25000001 3           BSL   $INVINT         .-107-INVALID INTERRUPT       LEVEL 7
347 000100 25000001 3           BSL   $CRAIR          .-110-CARD READER INPUT       LEVEL 8
348 000101 25000001 3           BSL   $INVINT         .-111-INVALID INTERRUPT       LEVEL 9
349 000102 25000001 3           BSL   $RT1IR          .-112-REMOTE TELETYPE I/O     LEVEL 10
350 000103 25000001 3           BSL   $RT2IR          .-113-REMOTE TTY #2           LEVEL 11
351 000104 25000001 3           BSL   $RTOIR          .-114-CONSOLE TELETYPE INPUT  LEVEL 12
352 000105 25000001 3           BSL   $RTOOR          .-115-CONSOLE TELETYPE OUTPUT LEVEL 13
353 000106 25000001 3           BSL   $DCAIR          .-116-DISC CONTROLLER         LEVEL 14
354 000107 25000001 3           BSL   $MCAIR          .-117-MAGTAPE CONTROLLER READ LEVEL 15
355 000110 25000001 3           BSL   $TRAIR          .-120-TAPE READER INPUT       LEVEL 16
356 000111 25000001 3           BSL   $TPAIR          .-121-TAPE PUNCH OUTPUT       LEVEL 17
357 000112 25000001 3           BSL   $LPAIR          .-122-LINE PRINTER            LEVEL 18
358 000113 25000001 3           BSL   $INVINT         .-123-INVALID INTERRUPT       LEVEL 19
359 000114 25000001 3           BSL   $CPBIR          .-124-CARD PUNCH B            LEVEL 20
360 000115 25000001 3           BSL   $INVINT         .-125-INVALID INTERRUPT       LEVEL 21
361 000116 25000001 3           BSL   $ITIR           .-126-120 HZ CLOCK            LEVEL 22
362 000117 25000001 3           BSL   $DINT           .-127-DISPATCHER INTERRUPT    LEVEL 23
```

```
364                     * DISPATCHER CONTROL TABLE                                  ** TABLE F
365                     *
366 000200 00200200 6          RORG  '200
367 000200 40000000 0  DISINT  DATA  B23          .MUST REMAIN AT '200
368 000201 00000000 0  CURNT   DATA  0            .MUST REMAIN AT '201.  CONTAINS ADDRESS OF
369                     *                          PROGRAM SERVICE AREA OF CURRENT ACTIVE
370 000202 00000000 0  CAPLE   DATA  0            .MUST REMAIN AT '202.  CONTAINS THE RELATIVE
371                     *                          ADDRESS OF THE APL ENTRY OF THE PROGRAM
372                     *                          CURRENTLY BEING EXECUTED.
373                     .                          PROGRAM OR ZERO IF NONE ACTIVE.
374                     .
375                     *
376                     *                          THE DISPATCHER REQUIRES ONE 8-WORD ENTRY FOR
377                     *                          EACH ACTIVE PROGRAM. THE TABLE SHOULD
378                     *                          THEN BE ESTABLISHED LARGE ENOUGH TO SATISFY
379                     *                          THE WORST-CASE CONDITIONS WITH REGARD TO
380                     *                          THE MAXIMUM NUMBER OF PROGRAMS WHICH
381                     *                          COULD BE CONCURRENTLY ACTIVE.
382                     *
383                     *
384 000203 00000000 0  DCT      DATA  0,"BAKGND"    .BAKGND ENTRY - ALWAYS ACTIVE
384 000204 20440513 0
384 000205 21647104 0
385 000206 10000000 0  CKPTFL  DATA  B21,0
385 000207 00000000 0
386 000210 00000001 3          DAC   $BPSA
387 000211 00000000 0  BAKHI   DAC   0             SET BY SYSINIT EQUAL NRFLO-BUFSZ
388 000212 00000377 0          DATA  255           NON-CHANGABLE PRIORITY
389 000213 00000001 3          DAC   $JSBUFR       .JOB STREAM BUFFER IS PARAMETER FOR BKGND
390 000214 00000000 0  DCBLOK  DATA  0,"**DC**",0,0
390 000215 12425117 0
390 000216 20625052 0
390 000217 00000000 0
390 000220 00000000 0
391 000221 00000001 3          DAC   $DCPSA        .DP COMM ENTRY - REMAINS IN DCT
392 000222 00000000 0          DATA  0,0,0         EVEN WHEN DP-COMM IS INACTIVE.
392 000223 00000000 0
392 000224 00000000 0
393                            $KFS  B20           .ONLY IN SPOOLED VERSION
394 000225 00000203 4  IOXAPL  DAC   DCT           .I/O EXEC ENTRY
395 000226 22247505 0          DATA  "IOEXEC"
395 000227 26042503 0
396 000230 10000000 0          DATA  '10000000,0   .STATUS = SUSPENDED INITIALLY
396 000231 00000000 0
397 000232 00000001 3          DAC   $IOXPSA
398 000233 00000000 0          DATA  0,10,0        .PRIORITY = 10
398 000234 00000012 0
398 000235 00000000 0
399                            ESKP
400 000236 00000247 4  DCTFREE DAC   *+9           .INITIAL AVAILABLE LIST - ENTRY 1
401 000237 00000000 0          RDAT  8(0)
402 000247 00000260 4          DAC   *+9           .                       - ENTRY 2
```

```
403 000250 00000000 0          RDAT   8(0)
404 000260 00000271 4          DAC    *+9              .                    - ENTRY 3
405 000261 00000000 0          RDAT   8(0)
406 000271 00000302 4          DAC    *+9              .                    - ENTRY 4
407 000272 00000000 0          RDAT   8(0)
408 000302 00000000 0          DAC    0                .                    - ENTRY 5
409 000303 00000000 0          RDAT   8(0)
```



```
429                            * TIMER SCHEDULER TABLE                                    ** TABLE H
430                            *
431                            *       THE FOLLOWING LABEL TPROGS MUST BE SET EQUAL TO THE
432                            *       TO THE MAXIMUM NO. OF PROGRAMS THAT WILL BE ON THE
433                            *       TIMER SCHEDULE
434          00000005          TPROGS  EQIV   5                . NO. OF PROGS. ON TIMER SCHEDULE
435 000335 00000005 0          MXTPRG  DAC    TPROGS           .MAXIMUM TIMER SCHEDULER PROGRAMS
436 000336 00200343 6          NAME1   BLOK   TPROGS           . 1ST 3 CHAR. OF PROG. NAME
437 000343 00200350 6          NAME2   BLOK   TPROGS           . 2ND 3 CHAR. OF PROG. NAME
438 000350 00200355 6          LEVEL   BLOK   TPROGS           . PROG. LEVEL
439 000355 00200362 6          INTRV   BLOK   TPROGS           . PERMANENT INTERVAL COUNTER
440 000362 00200367 6          INCTR   BLOK   TPROGS           . TEMPORARY INTERVAL COUNTER
441 000367 00200374 6          PARAM   BLOK   TPROGS           . PARAMETER PASSED FROM CALLING PROGRAM
```

```
411                     * RESIDENT PROGRAM LIST                 ** TABLE G
412                     *
413                     *                       ,CONTAINS ONE 4 WORD ENTRY FOR EACH RESIDENT
414                     *                        PROGRAM WHICH CONSISTS OF THE PROGRAM NAME
415                     *                        AND MEMORY ADDRESSES.
416                     *
417                     *                       ,THIS TABLE CONTAINS ONLY ZEROES WHEN THE
418                     *                        RESIDENT SYSTEM IS INPUT FROM DISC, BUT
419                     *                        THE ENTRIES ARE MADE BY THE BOOTSTRAP
420                     *                        INITIALIZATION PROGRAM AS IT BRINGS IN
421                     *                        THE RESIDENT FOREGROUND PROGRAMS.
422                             FORM  12,12
423 000313 00040000 0  RPLIST  DATA  /4,0/      ,MAY # OF ENTRIES, CURRENT # OF ENTRIES
424 000314 00000000 0        RDAT  16(0)
425                     *
426 000334 00000000 0  MAXPRO  DAC   0          ,MAX FG PROGRAM SIZE, COMPUTED BY SYINT
427                     *
```

```
429                     * TIMER SCHEDULER TABLE                                  ** TABLE H
430                     *
431                     *       THE FOLLOWING LABEL TPROGS MUST BE SET EQUAL TO THE
432                     *       TO THE MAXIMUM NO. OF PROGRAMS THAT WILL BE ON THE
433                     *       TIMER SCHEDULE
434           00000005  TPROGS  EQIV  5              . NO. OF PROGS. ON TIMER SCHEDULE
435 000335 00000005 0   MXTPRG  DAC   TPROGS        ,MAXIMUM TIMER SCHEDULER PROGRAMS
436 000336 00200343 6   NAME1   BLOK  TPROGS        . 1ST 3 CHAR. OF PROG. NAME
437 000343 00200350 6   NAME2   BLOK  TPROGS        . 2ND 3 CHAR. OF PROG. NAME
438 000350 00200355 6   LEVEL   BLOK  TPROGS        . PROG. LEVEL
439 000355 00200362 6   INTRV   BLOK  TPROGS        . PERMANENT INTERVAL COUNTER
440 000362 00200367 6   INCTR   BLOK  TPROGS        . TEMPORARY INTERVAL COUNTER
441 000367 00200374 6   PARAM   BLOK  TPROGS        . PARAMETER PASSED FROM CALLING PROGRAM
```

```
443                     * DISC DEFINITION TABLE                                 ** TABLE I
444                     *
445 000374 00000001 0   D/NUMB  DATA  1           ,NUMBER OF DISCS
446        00000375     D/DEFT  EQIV  *
447 000375 00000001 3   DAC1    DAC   $DCABF      0,DISC CONTROLLER BUSY FLAG ADDRESS
448 000376 00000160 0           DATA  112         1,# OF WORDS PER SECTOR
449 000377 00000024 0           DATA  20          2,# OF SECTORS PER TRACK
450 000400 00000024 0           DATA  20          3,# OF TRACKS PER CYLINDER
451 000401 00000313 0           DATA  203         4,# OF CYLINDERS
452 000402 00000001 0           DATA  1           5,PACK NUMBER
453 000403 03400000 6           DAC   #SAM1FS     6,ADDRESS OF SPACE ALLOCATION MAP
454 000404 00000024 0           DATA  20          7, SECTORS PER SPACE ALLOCATION BIT
455 000405 00000000 0           DATA  0           8,RETRY COUNT
456 000406 00000000 0           DATA  0           9,DRIVE NUMBER @ 819
457 000407 03400000 6           DAC   #DCACU      10,CHANNEL/UNIT NUMBER
458 000410 00000001 3           DAC   $DCASR      11,ADDRESS OF DISC CONTROLLER SERVICE ROUTINE
459 000411 03400000 6           DAW   #DCACU      12, DAW INSTRUCTION
460 000412 03400000 6           DCW   #DCACU      13, DCW INSTRUCTION
461 000413 00000001 3           DAC   $DCASP      14, DISC CONTROLLER QUEUE STRING POINTER
462 000414 00000000 0           DAC   0           15, RESERVED
```

```
464                    * MISCELLANEOUS PARAMETERS TABLE                    ** TABLE J
465                    *
466                    *              MISC. PARAMETERS TABLE
467                    *
468                    *              THESE PARAMETERS MAY BE REFERENCED AS ABSOLUTE LINKAGES
469                    *               BY PICKING UP THE BASE ADDRESS WHICH IS IN LOCATION 150
470                    *               AND THEN REFERING TO THESE PARAMETERS INDIVIDUALLY.
471                    *
472 000415 03400000 6  D/MDD   DAC   #MDDE/D        0,MDD ENTRIES PER DISC
473 000416 03400000 6         DAC   #MDDE/S        1,MDD ENTRIES PER SECTOR
474 000417 03400000 6         DAC   #MDDFS         2,MDD FIRST SECTOR
475 000420 03400000 6         DAC   #MDDSC         3,MDD SECTOR COUNT
476 000421 00000000 0         DATA  0              4,ACRONIM CONTROL FLAG
477 000422 03400000 6         ZZZ   #BGDLDA        5,BACKGROUND DEFAULT LIST OUTPUT ASSGNMENT
478 000423 77777777 0  BGIDLE DATA  -1             6,BACKGROUND IDLE FLAG (SET INITIALLY)
479 000424 77777777 0  BGSPQP DATA  -1             7,BACKGROUND SPOOL QUEUE POINTER (EMPTY)
480                           SKPZ  B18            ,ACCOUNTING SYSTEM ONLY
481 000425 00000001 3         DAC   $BGACBK        8,ADDRESS OF BACKGROUND ACCOUNTING BLOCK
482 000426 03400000 6         ZZZ   #LOUSR         9,LOW SYSTEM USER NUMBER
483 000427 03400000 6         ZZZ   #HIUSR         10,HIGH SYSTEM USER NUMBER
484                           ESKP
485                           SKFS  B18            ,FOR NON-ACCOUNTING SYSTEMS
486                           DATA  0,0,0          8,9,10, NULL
487                           ESKP
488 000430 03400000 6         ZZZ   #CKPTFS        11,ABS. SECTOR NUMBER OF CHECKPOINT AREA
489 000431 00000000 0  DUMPBG DATA  0              12,BACKGROUND MEMORY DUMP ACTIVE FLAG
490 000432 00000000 0  ADUMP  DATA  0              13,BACKGROUND REQUESTS ABORT DUMP FLAG
491 000433 00000000 0         DATA  0              14, LOW DUMP LIMIT
492 000434 00000000 0         DATA  0              15, HIGH DUMP LIMIT
493 000435 03400000 6         ZZZ   #JOBMES        16,BG JOB MESS FLAG: 0 FOR NO MESS
```

```
                                                                                ** TABLE K
                    * DISC I/O QUEUE
                    *
                    *
000436 00000000 0   D/QFULL DATA  0          ,QUEUE FULL FLAG
000437 00000440 4   D/AVAIL DAC   *+1        ,AVAILABLE ENTRY STRING ADDRESS
000440 00000450 4           DAC   *+8        ,ENTRY #1
000441 00000000 0           RDAT  7(0)
000450 00000460 4           DAC   *+8        ,ENTRY #2
000451 00000000 0           RDAT  7(0)
000460 00000470 4           DAC   *+8        ,ENTRY #3
000461 00000000 0           RDAT  7(0)
000470 00000500 4           DAC   *+8        ,ENTRY #4
000471 00000000 0           RDAT  7(0)
000500 00000510 4           DAC   *+8        ,ENTRY #5
000501 00000000 0           RDAT  7(0)
000510 00000000 0           DAC   0          ,ENTRY #6
000511 00000000 0           RDAT  7(0)
```

```
513                     * PERIPHERAL DEVICE COORDINATION TABLE                    ** TABLE L
514                     *
515                     *
516                     *
517                     *    THIS TABLE CONTAINS AN ENTRY FOR EACH ALLOCATABLE DEVICE PRESENT IN A
518                     * CONFIGURATION PLUS A SINGLE ENTRY FOR ALL DISCS. EACH ENTRY CONTAINS THE
519                     * HANDLER ADDRESS, THE DEVICE BUSY FLAG ADDRESS AND STORAGE FOR THE CURRENT USER
520                     * ID WHEN THE DEVICE IS ALLOCATED. THE RELATIVE POSITION OF AN ENTRY IN THIS
521                     * TABLE DETERMINES THE "PERIPHERAL DEVICE NUMBER" OF THE DEVICE.
522                     *    THE ENTRIES WHICH DEFINE DC-6024 DEVICES ARE FOLLOWED BY THE DEVICE BUSY
523                     * FLAG - ONE FOR EACH DEVICE CONTROLLER. FOR EXAMPLE, THE CARD READER AND LINE
524                     * PRINTER HAVE SEPARATE DEVICE FLAGS; TWO MAGNETIC TAPE DRIVES ON A SINGLE
525                     * CONTROLLER HAVE ONLY ONE DEVICE BUSY FLAG.
526                     *
527 000520 00000022 0  PDCT    DATA  '22           ,MAXIMUM DEVICE NUMBER
528 000521 00000001 3        DAC   $DFH          ,DEVICE 0 - ALWAYS DISC
529 000522 00001314 4        DAC   =0             DISC NEVER BUSY.
530 000523 00000001 3        DAC   $RTOH         ,DEVICE 1 - CONSOLE TTY - ALWAYS
531 000524 01000000 3        '01   $RTOBF
532 000525 00000000 0        DAC   0             ,DEVICE 2 - UNASSIGNED
533 000526 00000000 0        DAC   0
534 000527 00000000 0        DAC   0             ,DEVICE 3 - UNASSIGNED
535 000530 00000000 0        DAC   0
536 000531 00000001 3        DAC   $TRAH         ,DEVICE 4 - TAPE READER
537 000532 04000567 1        '04   TRABF
538 000533 00000001 3        DAC   $TPAH         ,DEVICE 5 - TAPE PUNCH
539 000534 05000570 1        '05   TPABF
540 000535 00000001 3        DAC   $LPAH         ,DEVICE 6 - LINE PRINTER
541 000536 06000574 1        '06   LPABF
542 000537 00000001 3        DAC   $CRAH         ,DEVICE 7 - CARD READER
543 000540 07000571 1        '07   CRABF
544 000541 00000001 3        DAC   $CPBH         ,DEVICE 10 - CARD PUNCH
545 000542 10000573 1        '10   CPBBF
546 000543 00000001 3        DAC   $MCAH0        ,DEVICE 11 - MAGTAPE TRANSPORT 0
547 000544 11000575 1        '11   MCABF
548 000545 00000001 3        DAC   $MCAH1        ,DEVICE 12 - MAGTAPE TRANSPORT 1
549 000546 11000575 1        '11   MCABF
550 000547 00000000 0        DAC   0
551 000550 00000000 0        DAC   0
552 000551 00000000 0        DAC   0
553 000552 00000000 0        DAC   0
554 000553 00000000 0        DAC   0
555 000554 00000000 0        DAC   0
556 000555 00000000 0        DAC   0
557 000556 00000000 0        DAC   0
558 000557 00000000 0        DAC   0             ,DEVICE 17
559 000560 00000000 0        DAC   0
560 000561 00000001 3        DAC   $RTOH         ,DEVICE 20 - CONSOLE TELETYPE
561 000562 02000000 3        '02   $RTOBF
562 000563 00000001 3        DAC   $RT1H         ,DEVICE 21 - REMOTE TELETYPE 1
563 000564 02000000 3        '02   $RT1BF
564 000565 00000001 3        DAC   $RT2H         ,DEVICE 22 - REMOTE TELETYPE 2
```

```
565 000566 02000000 3          '02    $RT2BF
566                    *
567 000567 00000000 0  TRABF    DATA   0          .DEVICE BUSY FLAGS
568 000570 00000000 0  TPABF    DATA   0          .PAPER TAPE READER "A" BUSY FLAG
569 000571 00000000 0  CRABF    DATA   0          .PAPER TAPE PUNCH "A" BUSY FLAG
570 000572 00000000 0  CPABF    DATA   0          .CARD READER "A" BUSY FLAG
571 000573 00000000 0  CPBBF    DATA   0          .CARD PUNCH "A" BUSY FLAG
572 000574 00000000 0  LPABF    DATA   0          .CARD PUNCH B BUSY FLAG
573 000575 00000000 0  MCABF    DATA   0          .LINE PRINTER "A" BUSY FLAG
574 000576 21000000 3  CHA      BUC    $C02629    .MAGTAPE CONTROLLER "A" BUSY FLAG
575 000577 21000000 3  CAH      BUC    $A12629    .CONVERT HOLLERITH(026/029) TO ASCII
576 000600 21000000 3  CBA      BUC    $B1A       .CONVERT ASCII TO 026/029 HOLLERITH
577 000601 21000000 3  CAB      BUC    $A1B       .CONVERT BCD TO ASCII (7-TRACK TAPE)
578 000602 21000000 3  CEA      BUC    $E1A       .CONVERT ASCII TO BCD
579 000603 21000000 3  CAE      BUC    $A1E       .CONVERT EBCD TO ASCII (9-TRACK TAPE)
580 000604 03400000 6  TTYIN    DAC    #TTYII     .CONVERT ASCII TO EBCD
581 000605 03400000 6  TTYOUT   DAC    #TTYOI     .TELETYPE INPUT INTERRUPT MASK
582 000606 03400000 6  DELTIM   ZZZ    #CPS/10    .TELETYPE OUTPUT INTERRUPT MASK
                                                  .DELTA TIME=ONE SECOND IN T-COUNTS
```

```
584                          SKFS  B20            ,ONLY IN SPOOLED SYSTEM
585                    * TERMINAL CONTROL BLOCK TABLE                          ** TABLE M
586                    *
587                    *
588                    *              TERMINAL CONTROL BLOCKS
589                    *
590                    *              A TERMINAL CONTROL BLOCK (TCB) MUST BE ESTABLISHED FOR
591                    *              EACH DEVICE WHICH IS TO BE SPOOLED.
592                    *
593                           FORM  8,8,8
594 000607 00000007 0  TCB    DATA  !07             ,PDN=07 FOR CARD READER
595 000610 00000000 0         RDAT  7(0)
596 000617 00000000 0         DATA  0               ,CONTROL WORD = 0 FOR NOW
597 000620 00000634 4         DAC   TCB2            ,LINK
598 000621 00000000 0         DATA  0               ,PL
599 000622 00000033 0         DATA  27              ,WC
600 000623 00000000 0         DATA  0               ,BA
601 000624 00000062 0         DATA  50              ,PRIORITY
602 000625 00000062 0         DATA  50              ,FIXED PRIORITY (DEFAULT = 50)
603 000626 00000000 0         DATA  0               ,SPOOL QUE?E POINTER
604 000627 77777777 0         DATA  -1              ,MODE=INPUT ONLY
605 000630 00000000 0         DATA  0               ,LINES PER PAGE = NULL
606 000631 00000634 4         DAC   TCB2            ,RELATED TCB ADDR = PRINTER
607 000632 00000000 0         DATA  0               ,PROGRAM
608 000633 01200000 0         DATA  /5,0,0/         ,5 FILES, BYT NO REGISTERS
609                    *
610 000634 00000006 0  TCB2   DATA  !06             ,PDN 06 = PRINTER
611 000635 00000000 0         RDAT  7(0)
612 000644 00000000 0         DATA  0               ,CONTROL WORD
613 000645 00000661 4         DAC   TCB3            ,LINK
614 000646 00000000 0         DATA  0               ,PL
615 000647 00000055 0         DATA  45              WC
616 000650 00000000 0         DATA  0               ,BA
617 000651 00000074 0         DATA  60              ,PRIORITY
618 000652 00000074 0         DATA  60              ,FIXED PRIORITY
619 000653 00000000 0         DATA  0               ,SPOOL QUEUE POINTER
620 000654 00000001 0         DATA  1               ,MODE=OUTPUT ONLY
621 000655 00102067 0         DATA  /0,132,55/      ,132 CHAR/LINE; 55 LINES/PAGE,
622 000656 00000607 4         DAC   TCB             ,RELATED TCB ADDR = CARD READER
623 000657 00000000 0         DATA  0               ,T/PROG = NULL
624 000660 00000000 0         DATA  /0,0,0/         ,REGS DEFINITIONS NULL
625                    *
```

```
627                     * TERMINAL CONTROL BLOCK TABLE (CONTINUED)                    * TABLE M
628                     *
629 000661 00000005 0   TCB3    DATA  '05           ,PDN = '05 FOR PAPER TAPE PUNCH
630 000662 00000000 0           RDAT  7(0)
631 000671 00000000 0           DATA  0             ,CONTROL WORD
632 000672 00000706 4           DAC   TCB4          ,LINK TO NEXT TCB
633 000673 00000000 0           DATA  0             ,PL
634 000674 00000033 0           DATA  27            ,WC
635 000675 00000000 0           DATA  0             ,BA
636 000676 00000106 0           DATA  70            ,PRIORITY
637 000677 00000106 0           DATA  70            ,FIXED PRIORITY
638 000700 00000000 0           DATA  0             ,SPOOL QUEUE POINTER
639 000701 00000001 0           DATA  1             ,MODE = OUTPUT ONLY
640 000702 00000000 0           DATA  0             ,LINES PER PAGE = NULL
641 000703 00000000 0           DATA  0             ,RELATED TCB = NONE
642 000704 00000000 0           DATA  0             ,T/PROG = NULL
643 000705 00000000 0           DATA  /0,0,0/       ,REGS DEFINITIONS NULL
644                     *
645 000706 00000020 0   TCB4    DATA  '20           ,PDN = '20 FOR CONSOLE TELETYPE
646 000707 00000000 0           RDAT  7(0)
647 000716 00000000 0           DATA  0             ,CONTROL WORD
648 000717 00000733 4           DAC   TCB5          ,LINK TO NEXT TCB
649 000720 00000000 0           DATA  0             ,PL
650 000721 00000033 0           DATA  27            ,WC
651 000722 00000000 0           DATA  0             ,BA
652 000723 00000106 0           DATA  70            ,PRIORITY
653 000724 00000106 0           DATA  70            ,FIXED PRIORITY
654 000725 00000000 0           DATA  0             ,SPOOL QUEUE POINTER
655 000726 00000000 0           DATA  0             ,MODE = BOTH INPUT AND OUTPUT
656 000727 00044001 0           DATA  /0,72,1/      ,72 CHAR/LINE; 1 LINE/PAGE.
657 000730 00000000 0           DATA  0             ,RELATED TCB = NONE
658 000731 00000706 4           DAC   TCB4          ,RELATED TCB = SELF
659 000732 03001403 0           DATA  /12,3,3/      ,REGS DEFINITIONS FOR ACRONIM
660                     *
```

```
662                    * TERMINAL CONTROL BLOCK TABLE (CONTINUED)                    * TABLE M
663                    *
664 000733 00000021 0  TCB5    DATA  '21             ,PDN = '21 FOR REMOTE TTY
665 000734 00000000 0          RDAT  7(0)
666 000743 00000000 0          DATA  0               ,CONTROL WORD
667 000744 00000760 4          DAC   TCB6            ,LINK
668 000745 00000000 0          DATA  0               ,PL
669 000746 00000033 0          DATA  27              ,WC
670 000747 00000000 0          DATA  0               ,BA
671 000750 00000055 0          DATA  45              ,PRIORITY
672 000751 40000055 0          DATA  '40000055       ,PRIORITY MAX LIMIT FOR TERMINAL = 45
673 000752 00000000 0          DATA  0               ,SPOOL QUEUE POINTER
674 000753 00000000 0          DATA  0               ,MODE = BOTH I/O
675 000754 00044001 0          DATA  /0,72,1/        ,72 CHAR/LINE, 1 LINE/PAGE,
676 000755 00000733 4          DAC   TCB5            ,RELATED TCB = SELF
677 000756 00000000 0          DATA  0               ,T/PROG = NULL
678 000757 03001403 0          DATA  /12,3,3/        ,12 FILES, 3 OF EACH TYPE OF REGISTERS
679                    *
680 000760 00000022 0  TCB6    DATA  '22             ,PDN = '22 FOR REMOTE TTY #2
681 000761 00000000 0          RDAT  7(0)
682 000770 00000000 0          DATA  0               ,CONTROL WORD
683 000771 00000000 0          DAC   0               ,LINK
684 000772 00000000 0          DATA  0               ,PL
685 000773 00000033 0          DATA  27              ,WC
686 000774 00000000 0          DATA  0               ,BA
687 000775 00000050 0          DATA  40              ,PRIORITY
688 000776 60000050 0          DATA  '60000050       ,PRIORITY FIXED AT 40
689 000777 00000000 0          DATA  0               ,SPOOL QUEUE POINTER
690 001000 00000000 0          DATA  0               ,MODE = BOTH I/O
691 001001 00044001 0          DATA  /0,72,1/        ,72 CHAR/LINE, 1 LINE/PAGE,
692 001002 00000760 4          DAC   TCB6            ,RELATED TCB = SELF
693 001003 00000000 0          DATA  0               ,T/PROG = NULL
694 001004 03001403 0          DATA  /12,3,3/        ,12 FILES, 3 OF EACH TYPE OF REGISTERS
695        00001005    EOTCB   EQIV  *               ,END OF TCB AREA
696                    ESKP
```

```
698                     * MAGNETIC TAPE OPTIONS TABLE                                   ** TABLE N
699                     *
700                     *           TYPE (0) SPECIFIES 9-TRACK TAPE
701                     *                (1) SPECIFIES 7-TRACK TAPE
702                     *           MODE (0) SPECIFIES BINARY-ASCII(NO CONV) FOR SYMBOLIC I/O
703                     *                (1) SPECIFIES BCD OR EBCD CONVERSION FOR SYMBOLIC I/O
704                     *                   (ODD PARITY IS FORCED FOR BINARY I/O REQUESTS)
705                     *           DENSITY:     (0) IS 200 BPI -OR- LOW DEN PEC
706                     *                        (1) IS 556 BPI -OR- HI DEN PEC
707                     *                        (2) IS 800 BPI
708                     *           CWPI         (0) IS 1 CH/WD
709                     *                        (1) IS 2 CH/WD
710                     *                        (2) IS 3 CH/WD
711                     *                        (3) IS 4 CH/WD
712                     *
713                     *           (OC) AND (PI) IN THE FOLLOWING MUST BE ZERO
714                     *           TYP@16, MOD@14, DEN@12, CPW@10, OC@6 T#@3, PI@0
715                           FORM  8,2,2,2,4,3,3
716                     *     ...                   ,BG TAPE OPTION TABLE; SET BY JOB CONTROL
717                     *     !!!                   ,TOT MUST BE 8 WORDS
718 001005 00000000 0   TOT   RDAT  8(0)
719                     *     ...                   ,BG/FG DEFAULT TAPE OPTIONS TABLE
720                     *     !!!                   ,DTOT MUST BE 8 WORDS
721 001015 00024000 0   DTOT  DATA  /0,0,2,2,0,0,0/     TRANSPORT 0 - 9T,BIN,3CPW,800BPI
722 001016 00024010 0         DATA  /0,0,2,2,0,1,0/     TRANSPORT 1 - 9T,BIN,3CPW,800BPI
723 001017 00014010 0         DATA  /0,0,1,2,0,1,0/     TRANSPORT 1
724 001020 00014020 0         DATA  /0,0,1,2,0,2,0/     TRANSPORT 2
725 001021 00014030 0         DATA  /0,0,1,2,0,3,0/     TRANSPORT 3
726 001022 00014040 0         DATA  /0,0,1,2,0,4,0/     TRANSPORT 4
727 001023 00014050 0         DATA  /0,0,1,2,0,5,0/     TRANSPORT 5
728 001024 00014060 0         DATA  /0,0,1,2,0,6,0/     TRANSPORT 6
729 001025 00014070 0         DATA  /0,0,1,2,0,7,0/     TRANSPORT 7
730                     *     ...                   ,DENSITY CONTROL TABLE
731                     *                           ,ONE OCTAL DIGIT (3 BITS) IS USED FOR EACH
732                     *                            DRIVE, LEFTMOST (BITS 23-21) FOR TRANSPORT
733                     *                            0; VALUES OF 0-3 INDICATE DENSITY BITS IN
734                     *                            COMMAND WORD AS APPROPRIATE FOR PHYSICAL
735                     *                            DEVICE; VALUE OF 4 INDICATES DENSITY IS
736                     *                            INVALID.
737                     *
738 001026 44444444 0   MTDENS DATA  !44444444       ,200 BPI
739 001027 44444444 0         DATA  !44444444       ,556 BPI
740 001030 22444444 0         DATA  !22444444       ,800 BPI
741 001031 44444444 0         DATA  !44444444       ,1600 BPI
```

```
07-13-73              SYSTEM DATA MODULE (DMS)              61606-01  01.070173      PAGE   23

743                    * SYSTEM SERVICE IDENTIFICATION TABLE                              ** TABLE D
744                    *
745 001032 00000020 0  SSTAB    DAC    SSTABE-SSTAB-1 ,NO OF ENTRIES
746 001033 20241117 0           DATA   "ABO"           ,0  $ABORT
747 001034 22227517 0           DATA   "I/O"           ,1  $I/O
748 001035 21254111 0           DATA   "EXI"           ,2  $EXIT
749 001036 22047514 0           DATA   "HOL"           ,3  $HOLD
750 001037 20644101 0           DATA   "CHA"           ,4  $CHAIN
751 001040 22247106 0           DATA   "INF"           ,5  $INFO
752 001041 25640311 0           DATA   "WAI"           ,6  $WAIT
753 001042 24643125 0           DATA   "SFU"           ,7  $SFUNC
754 001043 20647516 0           DATA   "CON"           ,10 $CONVERT
755 001044 21451117 0           DATA   "FRO"           ,11 $FROGS
756 001045 24654523 0           DATA   "SYS"           ,12 $SYSTER
757 001046 25042522 0           DATA   "TER"           ,13 $TERMIN
758 001047 20251523 0           DATA   "ASS"           ,14 $ASSIGN
759 001050 21041515 0           DATA   "DCM"           ,15 $DCM
760 001051 25247124 0           DATA   "UNT"           ,16 $UNTRAP
761 001052 21450101 0           DATA   "FPA"           ,17 $FPACK
762        00001053     SSTABE  EQIV   *               ,END OF SYSTEM SERVICES ID TABLE
```

```
07.13.73              SYSTEM DATA MODULE (DMS)              61606.01  01.070173      PAGE    24

764                     * DYNAMIC CELL POOL DEFINITION TABLE                          ** TABLE P
765                     *
766                     *                                 .POOL OF AVAILABLE 4.WORD CELLS
767                     *
768                     *              THIS POOL IS REFERENCED BY THE "PUTCEL" AND "GETCEL" ROUTINES
769                     *               TO ACCESS 4.WORD CELLS AS NEEDED BY THE SYSTEM.
770                     *
771           00000144  NUMCELS EQIV  100             .SIZE OF AVAILABLE SPACE
772 001053 00000144 0  CELLS   DAC   NUMCELS
773 001054 00201220 6          BLOK  NUMCELS         .RESERVE SPACE
```

```
                    * EXTERNAL INTERRUPT DEFINITION BLOCKS                          ** TABLE Q
                    *
                    *ENTRY FORMAT (ONE PER EXTERNAL INTERRUPT):
                    * WORD 0      INTERRUPT GROUP/LEVEL
                    * WORD 1-2    PROGRAM NAME (6 ANSCII CHARACTERS) - INITIALLY 0
                    * WORD 3      EXECUTION PRIORITY                 - INITIALLY 0
                    * WORD 4      EXECUTION PARAMETER
                    *
001220 00000001 0   EXIDBS  DATA  1             ,NUMBER OF BLOCKS
001221 00000113 0   IOB113  DATA  '113          ,GROUP 1, LEVEL '11
001222 00000000 0           RDAT  10(0)
001234 20001226 1           TRM   *-6
001235 06001222 1           TMD   *-11
001236 01001224 1           TMI   *-10
001237 03001225 1           TMK   *-10
001240 25000001 3           BSL   $INIT#        ,INITIATE CONNECTED PROGRAM
001241 10001226 1           TMR   *-11          ,RESTORE REGS
001242 25601233 4           BRL*  *-7
```

```
07-13-73            SYSTEM DATA MODULE (DMS)          61606-01  01,070173      PAGE    26


794                      * DISC DICTIONARY IN CORE TABLE                          ** TABLE R
795                      *
796                      *       ...                 ,ALL DISC FILES TYPED AS "CORE RESIDENT
797                      *                            DIRECTORY ENTRY" (C) ARE ENTERED IN THIS
798                      *                            TABLE BY SYSINIT.
799                      *
800 001243 77777773 0   DICT    DATA  -5              ,NEG # OF ENTRIES AVAILABLE
801 001244 00000000 0           RDAT  40(0)           ,TABLE AREA (8 WORDS PER ENTRY)
802                      *
803 001315 00400000 6           END$                  ,END OF SYSDAT
```

APPENDIX F
BACKGROUND DATA MODULE

```
61773-00 MACRO ASSEMBLER - REVISION LEVEL 04.092275
12-01-75              BACKGROUND DATA MODULE (DMS)          61610-01  03.122275       PAGE     1


 2        00000003     REV#    EQIV   3              .CURRENT REVISION LEVEL
 3                     *
 4                     *             ASSEMBLE WITH FLAGS 18 SET   FOR ACCOUNTING SYSTEMS
 5                     *                                    RESET FOR NON-ACCOUNTING SYSTEMS
 6                     *
 7                     ******************************  I N D E X  ********************************
 8                     * EXTERNAL DEFINITION TABLE                                      - LINE 20
 9                     * FLAGS                                                         - LINE 36
10                     * BACKGROUND INFORMATION TABLE                                  - LINE 46
11                     * BACKGROUND PROGRAM SERVICE AREA                               - LINE 58
12                     * BACKGROUND FILE/DEVICE CONTROL BLOCK                         - LINE 77
13                     * BACKGROUND EXECUTION AREA                                     - LINE 108
14                     ********************************************************************************
```

```
12-01-75              BACKGROUND DATA MODULE (DMS)          61610-01  03.122275      PAGE

16                      * NOTE: THIS MUST BE THE LAST RESIDENT MODULE, PRECEDING SYSINT!!
17                      *
18                      * EXTERNAL DEFINITION TABLE
19                      *
20          00000000 2          XDEF    B61001,*          .BEGINNING OF 61610-01
21          00000000 2          XDEF    BGDATA,U$FLAG     .MODULE NAME
22          00000000 2          XDEF    U$FLAG,U$FLAG     .UNSOLICITED $ FLAG
23          00000001 2          XDEF    ABORTF,ABORTF     .JOB ABORT FLAG
24          00000004 2          XDEF    REDUCE,REDUCE     .REDUCE BACKGROUND SIZE FLAG
25          00000317 2          XDEF    JSFDCB,JSFDCB     .JOB STREAM'S F/D CONTROL BUFFER
26          00000006 2          XDEF    JSBUFR,JCBUFR     .JOB STREAM'S INPUT BUFFER
27          00000003 2          XDEF    JSFLAG,JSFLAG     .JOB STREAM FLAG
28          00000041 2          XDEF    INFORM,INFORM     .BACKGROUND INFORMATION TABLE
29          00000042 2          XDEF    DATE,INFORM+1     .SYSTEM DATE
30          00000041 2          XDEF    OPTION,INFORM     .SYSTEM OPTION WORD
31          00000064 2          XDEF    BPSA,BPSA         .BACKGROUND PSA
32          00000306 2          XDEF    BACCTG,BACCTG     .BAKGND ACCTG AREA PARAMETERS
33          00000307 2          XDEF    BFDCB,BFDCB       .BACKGROUND F/D CONTROL BUFFER
34          00000501 2          XDEF    BSTAR,BSTAR       .BACKGROUND STARTING ADDRESS
35                      *
36                      * FLAGS
37 000000 00000000 0    U$FLAG  DATA    0                 .UNSOLICITED CONTROL STATEMENT FLAG
38 000001 00000000 0    ABORTF  DATA    0                 .BACKGROUND JOB ABORTED FLAG
39 000002 00000000 0    GOFLAG  DATA    0                 ."GO" FLAG
40 000003 00000000 0    JSFLAG  DATA    0                 .JOB STREAM FLAG
41 000004 00000000 0    REDUCE  DATA    0                 .REDUCE BAKGND SIZE FLAG - SET BY
42                      *                                 OP. COMM. WITH SPECIFIED SIZE
43 000005 10020040 0            DATA    "   "             .BUFFER PRINT CONTROL CHACTER(JCB - 1)
44 000006 10020040 0    JCBUFR  BDAT    27("   ")         .CONTROL STATEMENT BUFFER
45                      * BACKGROUND INFORMATION TABLE
46 000041 00000000 0    INFORM  DATA    0                 .$OPTIONS
47 000042 10020040 0            BDAT    3("   ")          .$DATE
48 000045 00000067 0            DATA    55                .$LINES
49 000046 00000000 0            DATA    0                 .$FLAGS
50 000047 10020040 0            DATA    "      "          .$JOB
50 000050 10020040 0
51 000051 00000000 #    DOPTS   ZZZ     #BGDOPS           .DEFAULT OPTIONS
52 000052 00000000 #    DLINES  ZZZ     #BGDLNS           .DEFAULT LINES/PAGE
53 000053 00000000 #    DFLAGS  ZZZ     #BGDFGS           .DEFAULT FLAGS
54 000054 00000000 0    SPQPNT  DATA    0                 .BACKGROUND'S SPOOL QUEUE ENTRY POINTER
55 000055 00000000 0    BAKHI#1 DATA    0                 .ORIGINAL 'BAKHI' VALUE
56 000056 00000000 0    BAKHI#2 DATA    0                 .MODIFIED 'BAKHI' VALUE
57 000057 00000000 0            BDAT    5(0)              .RESERVED FOR FUTURE SYSTEM EXPANSION
58                      * BACKGROUND PROGRAM SERVICE AREA
59 000064 00000000 0    BPSA    BDAT    5(0)              .GENERAL REGISTER STORAGE
60 000071 00000000 0    BPC     DATA    0                 .PROGRAM COUNTER STORAGE
61 000072 00000000 0    BBIT    DATA    0                 .H AND V REGISTERS
62 000073 00000000 0    BSAU    DATA    0,0,0             .SAU REGISTERS
62 000074 00000000 0
62 000075 00000000 0
63 000076 00000000 0    BSWIT   DATA    0                 .PROGRAM SWITCH WORD
64 000077 00000000 0    BLIMTS  DATA    0,0               .LIMIT REGISTER VALUES
```

```
64 000100 00000000 0
65 000101 00000000 0  BCTEMP   DAC     0              .CURRENT ALLOCATION POINTER
66 000102 00200306 6  BTEMP    BLOK    132            .TEMPORARY STORAGE AREA
67                             SKF7    B18            .ACCOUNTING VERSION
68 000306 00000001 3  BACCTG   DAC     $BGACBK        .POINT TO BACKGROUND ACCOUNTING BLOCK
69                             ESKP
70                             SKFS    B18            .NON-ACCOUNTING
71                     BACCTG  DATA    0              .NO ACCTG PARAMETERS
72                             ESKP
73                             FORM    7,11,6
74 000307 01007700 0  BFDCB    DATA    /2,'77,0/
75 000310 00000000 0           BDAT    7(0)
```

```
12-01-75              BACKGROUND DATA MODULE (DMS)          61610-01  03.122275      PAGE     4

                        * BACKGROUND FILE/DEVICE CONTROL BLOCK
                        * FORMAT /X,Y,Z/
                        *              X=1 IF NOT REASSIGNABLE
                        *              X=2 IF DISC FILE
                        *              X=3 IF NOT REASSIGNABLE DISC FILE
                        *              X=0 OTHERWISE
                        *              Y=LOGICAL FILE #
                        *              Z=PHYSICAL DEVICE # (0 FOR DISC FILES)
                        *
                        *..... THE FOLLOWING ARE DEFAULT ASSIGNMENTS, INITIALIZED BY A $JOB
                        *
000317 00000001 0   JSFDCB  DATA    /0,0,1/,0       .ASSIGN  0(JS) TO  1(TTY)
000320 00000000 0
000321 00400101 0           DATA    /1,1,1/,0       .ASSIGN  1(OC) TO  1(TTY) NOT REASSIGNABLE
000322 00000000 0
000323 00000201 0           DATA    /0,2,1/,0       .ASSIGN  2     TO  1
000324 00000000 0
000325 00000301 0           DATA    /0,3,1/,0       .ASSIGN  3     TO  1
000326 00000000 0
000327 00000404 0           DATA    /0,4,4/,0       .ASSIGN  4(BI) TO  4(PTR)
000330 00000000 0
000331 01000500 0           DATA    /2,5,0/,0       .ASSIGN 5(BO) TO "LR"(LINK READY)
000332 00000000 0
000333 23051040 0           DATA    "LR     ",0,0,0,0
000334 10020040 0
000335 00000000 0
000336 00000000 0
000337 00000000 0
000340 00000000 0
000341 00000606 0           DATA    /0,6,6/,0       .ASSIGN  6(LO) TO  6(LP)
000342 00000000 0
000343 00000777 0           DATA    /0,7,'77/,0     .ASSIGN 7(SI) TO 77(JOB STREAM)
000344 00000000 0
000345 01001000 0           DATA    /2,'10,0/,0     .ASSIGN 10(SO) TO "W1"(WORK FILE ONE)
000346 00000000 0
000347 25630440 0           DATA    "W1     ",0,0,0,0
000350 10020040 0
000351 00000000 0
000352 00000000 0
000353 00000000 0
000354 00000000 0
000355 01001200 0           DATA    /2,'12,0/,0     .ASSIGN 12(LL) TO "LL"(LINK LIBRARY)
000356 00000000 0
000357 23046040 0           DATA    "LL     ",0,0,0,0
000360 10020040 0
000361 00000000 0
000362 00000000 0
000363 00000000 0
000364 00000000 0
000365 01001500 0           DATA    /2,'15,0/,0     .ASSIGN 15(LR) TO "LR"(LINK READY)
000366 00000000 0
000367 23051040 0           DATA    "LR     ",0,0,0,0
```

```
102 000370 10020040 0
102 000371 00000000 0
102 000372 00000000 0
102 000373 00000000 0
102 000374 00000000 0
103 000375 01001600 0          DATA    /2,,16,0/,0     .ASSIGN 16(GO) TO "GO" (CATALOGER WORK FILE)
103 000376 00000000 0
104 000377 21647440 0          DATA    "GO     ",0,0,0,0
104 000400 10020040 0
104 000401 00000000 0
104 000402 00000000 0
104 000403 00000000 0
104 000404 00000000 0
105 000405 00000000 0          DATA    0               .THIS VALUE TERMINATES THE SEARCH
106 000477 00200477 6          RORG    JSFDCH+112      .PRESERVE EXACTLY 112 LOCATIONS
```

```
108                     * BACKGROUND EXECUTION AREA
109 000477 00000000 0   BMODE   DATA    0           .BAKGND EXECUTION MODE WORD
110 000500 00000000 0   BSTART  DATA    0           .BAKGND PROGRAM STARTING ADDRESS POINTER
111        00000501     BSTAR   EQIV    *           .FIRST LOCATION OF BACKGROUND PROGRAM
112 000501 21000501 1           BUC     *
```

```
12-01-75          BACKGROUND DATA MODULE (DMS)          61610-01  03.122275      PAGE      7


114                       DASH    MACRO
115                               SKOB    :01
116                       :00     DASH1   (:02),:01
117                               ESKP
118                               SKNB    :01
119                       :00     DASH2   :02
120                               ESKP
121                               MEND
122                       DASH1   MACRO
123                               SKFZ    B:02
124                               SKOB    :03
125                       :00     DASH1   (:01),:03,:04,:05
126                               ESKP
127                               SKNB    :03
128                               SKOB    :01
129                       :00     DASH2   :01
130                               ESKP
131                               SKNB    :01
132                               XEQV    :00,REV#
133                               ESKP
134                               ESKP
135                               ESKP
136                               MEND
137                       DASH2   MACRO
138                               SKFS    B:01
139                               SKOB    :02
140                       :00     DASH2   :02,:03,:04
141                               ESKP
142                               SKNB    :02
143                               XEQV    :00,REV#
144                               ESKP
145                               ESKP
146                               MEND
147                       L61001  DASH    (),(18)
148                       L61002  DASH    (18),()
149            00000003           XEQV    L61002,REV#
150 000502 00400000 6             ENDB                      .END OF BGDATA
```

APPENDIX G
SYSTEM GENERATION JOB STREAM

```
$JOB SYSGEN-PHASE-11 U1
$DATE 27 JUNE 73
$ ... CUSTOMER: DATACRAFT
$ ... OUTPUT FILE TYPE: 9-TR MAG TAPE
$ ... DISC TYPE: 5204 CARTRIDGE
$ ... STEP 0. ESTABLISH CONFIGURED MODULES OF THE FOLLOWING:
$ ...            - SYSGEN LINK FILES -
$ ...         SLMSGS - SYSTEM LINKAGE MODULE/SGS
$ ...         SYSDAT - SYSTEM DATA MODULE/DMS
$ ...         BGDATA - BACKGROUND DATA MODULE/DMS
$ ...         DSR    - LINK MODULE OF DISC SERVICE ROUTINE
$ ...            - CUSTOMER SOURCE FILES -
$ ...         S54390 - SYSTEM LINKAGE MODULE/SGS
$ ...         S60690 - SYSTEM DATA MODULE/DMS
$ ...         S61090 - BACKGROUND DATA MODULE/DMS
$ASSIGN 7=S54302,10=W1
$ ... CONFIGURE & ESTABLISH SLMSGS FOR CUSTOMER SYSTEM
$UTILITY
.SRCFUD
.REPL 1
        IDEN  SYSTEM LINKAGE MODULE/SGS#2 DATACRAFT 61543-90  6/27/73
.CHNG 63,21"6"
.CHNG 126,16"7"
.CHNG 127,16"7"
.CHNG 131,16"12"
.REPL 138
.CHNG 139,1-3"CBA"
.REPL 140
.CHNG 141,1-3"CAR"
.REPL 142
.CHNG 143,1-3"CEA"
.REPL 144
.CHNG 145,1-3"CAE"
```

SYSTEM LINKAGE MODULE CONFIGURATION

```
.REPL 167,176
G1.L8    BSL    $S.CIP          .CARD READER INPUT
         BLOK   3
G1.L12   BSL    $S.TIP          .CONSOLE TELETYPE INPUT
G1.L13   BSL    $S.TOP          .CONSOLE TELETYPE OUTPUT
G1.L14   BSL    $S.DIP4         .DISC
G1.L15   BSL    $S.MTIP         .MAG TAPE
G1.L16   BSL    $S.TRIP         .TAPE READER INPUT
G1.L17   BSL    $S.TROP         .TAPE READER OUTPUT
         BLOK   6
.CHNG 196,21"4",49"4"
.CHNG 197,21"4",49"4"
.CHNG 198,21"4",49"4"
.CHNG 199,21"4",49"4"
.CHNG 200,21"4",49"4"
.CHNG 201,21"4",49"4"
.CHNG 202,21"4",49"4"
.CHNG 203,21"4",49"4"
.CHNG 204,21"4",49"4"
.CHNG 205,21"4",49"4"
.CHNG 306,16"1",18"1"
.INSE EOF
.EXIT
```

SYSTEM LINKAGE MODULE CONFIGURATION

```
$ASSIGN 5=LR,7=W1
$OPTIONS .
$ASSEMBLE
$ADF 7
$FILEMA
ESTAB 7,S54390,0,1,0,5,R,W,D  .CONFIGURED SOURCE - SLMSGS
ESTAB 5,SLMSGS,0,1,0,4,R,W,D  .SLMSGS FOR SYSGEN
EXIT
```

CREATING A FILE FOR SLM

```
$ ... CONFIGURE & ESTABLISH SYSDAT FOR CUSTOMER SYSTEM
$ASSIGN 7=S60601,10=W1
$UTILITY
.SRCEUD
.REPL 1
        IDEN  SYSTEM DATA MODULE (DMS) DATACRAFT  61606-90 R01.6/27/73
.CHNG 130,23"6",63"6"
.CHNG 131,23-25"200",72"9"
.CHNG 228,22-23"6"
.CHNG 229,23-29"66666"
.CHNG 230,23-24"66"
.REPL 353
          BSL   $INVINT          .-116-INVALID INTERRUPT        LEVEL 14
.REPL 358,359
          BSL   $DCAIR           .-123-DISC CONTROLLER          LEVEL 19
          BSL   $INVINT          .-124-INVALID INTERRUPT        LEVEL 20
.CHNG 446,15"2"
.CHNG 451,16" "
.CHNG 452,17"4"
.CHNG 455,16" "
.INSE 463
DAC2      DAC   $DCABF           0,BUSY FLAG ADDRESS
          DATA  112              1,# OF W/S
          DATA  20               2,# OF S/T
          DATA  2                3,# OF T/C
          DATA  204              4,# OF C/D
          DATA  2                5,PACK #
          DAC   #SAMNFS          6,SAM FIRST SECTOR
          DATA  2                7,SECTORS PER SAB
          DATA  0                8,RETRY COUNT
          DATA  B1^              9,PLATTER ADDRESS
          DAC   #DCACU           10,CHANNEL/UNIT
          DAC   $DCASR           11,SERVICE ROUTINE
          DAW   #DCACU           12,DAW INSTRUCTION
          DCW   #DCACU           13,DCW INSTRUCTION
          DAC   $DCASP           14,QUEUE STRING POINTER
          DAC   0                15,RESERVED
.REPL 544,545
          DAC   0                .DEVICE 10
          DAC   0
.CHNG 722,16"1",18"1",49"7"
.CHNG 739,16-17"00"
.CHNG 740,16-17"11"
.INSE EOF
.EXIT
```

SYSDAT CONFIGURATION

```
$ASSIGN 7=W1,5=LR
$OPTION .
$ASSEMBLE
$ADF 7
$FILEMA
ESTAB 7,S60690,0,1,0,5,R,W,D  .CONFIGURED SOURCE - SYSDAT
ESTAB 5,SYSDAT,0,1,0,4,R,W,D  .SYSDAT FOR SYSGEN
EXIT
```

CREATING A FILE FOR SYSDAT

```
$ ... CONFIGURE & ESTABLISH BGDATA FOR CUSTOMER SYSTEM
$ASSIGN 7=S61001,10=W1
$UTILITY
.SRCEUD
.REPL 1
        IDEN  BACKGROUND DATA MODULE (DMS)        61610-90 R01.6/27/73
.INSE EOF
.EXIT
$ASSIGN 7=W1,5=LR
$OPTION .
$ASSEMBLE
$ADF 7
$FILEMA
ESTAB 7,S61090,0,1,0,5,R,W,D  .CONFIGURED SOURCE - BGDATA
ESTAB 5,BGDATA,0,1,0,4,R,W,D  .BGDATA FOR SYSGEN
EXIT
$REW 5
$INCLUDE L67200               CARTRIDGE DISC SERVICE ROUTINE
$FILEMA
ESTAB 5,DSR,0,1,0,4,R,W,D     .DSR FOR SYSGEN
```

BGDATA CONFIGURATION

```
$ ... STEP 1. CREATE BOOTSTRAP OF ABSOLUTE DEVICE LOADER
$ASSIGN 5=11  (OUTFILE FILE = 9-TR MAG TAPE)
$REW 5
$ASSIGN 15=L55700,16=GO
$CATALOG
NAME=AML
BEGIN
$FILEMA
DUMPBF AML,0,'20              (BOOTSTRAP MODULE @ 20)
DELETE AML,GORP               (REMOVE TEMPORARY FILE)
$ ... STEP 2. CREATE ABSOLUTE MODULE OF SGS
$ASSIGN 15=SLMSGS,12=LR,5=LR
$REW 5
$INCLUDE L53000               (FOREGROUND EXECUTIVE MODULES)
$INCLUDE L58100               (I/O CONTROL SUPERVISOR)
$INCLUDE L58200               (CONSOLE TELETYPE HANDLER)
$INCLUDE L58400               (PAPER TAPE READER HANDLER)
$INCLUDE L58500               (PAPER TAPE PUNCH HANDLER)
$INCLUDE L58600               (CARD READER HANDLER)
$INCLUDE L58700               (026/029 : ASCII)
$INCLUDE L58800                 "
$INCLUDE L58900                 "
$INCLUDE L59000               (ANALEX LINE PRINTER HANDLER)
$INCLUDE L59900               (MAG TAPE HANDLER)
$INCLUDE L69100               (ASCII:EBCD CONVERSION)
$INCLUDE L69200               (ASCII:BCD CONVERSION)
$INCLUDE L65600               (CARTRIDGE DISC HANDLER)
$INCLUDE L50300               (SYSTEM SERVICE MODULE/ROS)
$INCLUDE L50400               (LINK LOADER/ROS)
$CATALOG
NAME=SGS
BEGIN
$ASSIGN 5=11  (OUTPUT FILE = 9-TR MAG TAPE)
$FILEMA
DUMPBF SGS,0,0                (AML REQUIRES BOOTSTRAP FORMAT)
DELETE SGS,GORP
EXIT
$ ... STEP 3. CREATE SGS LOAD MODULE OF DISC INITIALIZATION PROGRAM
$ASSIGN 5=LR
$INCLUDE SYSDAT               (CONFIGURED SYSDAT)
$INCLUDE DSR                  (DISC SERVICE ROUTINE)
$ASSIGN 15=LR,12=L61801       (DISINT = L61801)
$CATALOG
NAME=DISINT
BEGIN
$ASSIGN 5=11  (OUTPUT FILE = 9-TR MAG TAPE)
$FILEMA
DUMP DISINT,0,'20000          (ABSOLUTE LOAD MODULE @ 20000)
DELETE DISINT,GORP
EXIT
```

CREATE AML MODULE IN BOOTSTRAP FORMAT

CREATE SGS MODULE IN ABSOLUTE LOAD MODULE FORMAT

CREATE SGS/DISC INITIALIZATION MODULE

```
$ ... STEP 4. OUTPUT NON-DISC 1 MOD ENTRIES
$ASSIGN 5=11  (OUTPUT FILE = 9-TR MAG TAPE)
$ ...    $FILEMA                (NO MOD ENTREIS FOR CUSTOMER SYSGEN)
$ ...    SAVMOD
$ ...    EXIT
$WEF 5          (TERMINATE MOD ENTRIES WITH END-OF-FILE)
$ ... STEP 5. CREATE DMS SAVE MODULES OF RESIDENT DMS AND OPCOM
$ASSIGN 5=W1
$REW 5
$INCLUDE SYSDAT
$WEF 5
$INCLUDE L69500                (MAIN MODULE OF OPERATOR COMMUNICATIONS)
$WEF 5
$INCLUDE L69510
$WEF 5
$INCLUDE L69520
$WEF 5
$INCLUDE L69530
$WEF 5
$INCLUDE L69540
$WEF 5
$INCLUDE L69550
$ASSIGN 15=W1=12=LR,5=LR
$REW 5
$INCLUDE L60701                (EXECUTIVE)
$INCLUDE L60801                (MONITOR SERVICE LINKAGE ROUTINES)
$INCLUDE L61101                (LOAD MODULE LOADER)
$INCLUDE L61201                (EXECUTIVE TRAPS)
$INCLUDE L61301                (PROGRAM SCHEDULER)
$INCLUDE L61601                (FILE HANDLER)
$INCLUDE L66300                (I/O EXECUTIVE)
$INCLUDE L60901                (I/O CONTROL SUPERVISOR)
$INCLUDE L67300                (DISC HANDLER)
$INCLUDE DSR                   (DISC SERVICE ROUTINE)
$INCLUDE L66900                (SPOOLING SERVICE ROUTINE)
$INCLUDE L62201                (CARD READER HANDLER)
$INCLUDE L62301                (026-029 : ASCII CONVERSION)
$INCLUDE L62401                (ANALEX LINE PRINTER HANDLER)
$INCLUDE L62501                (PAPER TAPE READER HANDLER)
$INCLUDE L62601                (PAPER TAPE PUNCH HANDLER)
$INCLUDE L62701                (MAG TAPE HANDLER)
$INCLUDE L60201                (ASCII:BCD CONVERSION)
$INCLUDE L60100                (ASCII:EBCDIC CONVERSION)
$INCLUDE L71300                (TELETYPE #0 - CONSOLE - SERVICE ROUTINE)
$INCLUDE L71301                (REMOTE TELETYPE #1 SERVICE ROUTINE)
$INCLUDE L71302                (REMOTE TELETYPE #2 SERVICE ROUTINE)
$INCLUDE L67702                (RE-ENTRANT TELETYPE HANDLER)
$INCLUDE L70900                (OPERATOR COMMUNICATIONS INTERFACE)
$INCLUDE L70300                (ACCOUNTING SERVICE ROUTINE)
$INCLUDE BGDATA                (BACKGROUND DATA MODULE)
$INCLUDE L61401                (SYSTEM INITIALIZATION PROGRAM)
$FILEMA
ESTAB 5,L60001,0,1,0,4,R,W,D  .LINK MODULES OF CURRENT SYSGEN
EXIT
```

BUILD INPUT FILE FOR DMS CONSTRUCTION

```
$CATALOG
NAME=DMS
TYPE=SYSGEN
RESTRT
BEGIN
NAME=OPCON,,R
TYPE=FG,PRIV
NOMAP
RESTRT
BEGIN
NAME=OPCON1,,R
TYPE=FG,PRIV
NOMAP
RESTRT
BEGIN
NAME=OPCON2,,R
TYPE=FG,PRIV
NOMAP
RESTRT
BEGIN
NAME=OPCON3,,R
TYPE=FG,PRIV
RESTRT
NOMAP
BEGIN
NAME=OPCON4,,R
TYPE=FG,PRIV
RESTRT
NOMAP
BEGIN
NAME=OPCON5,,R
TYPE=FG,PRIV
BEGIN
$ASSIGN 5=11  (OUTPUT FILE = MAG TAPE)
$FILEMA
SAVE DMS,GORP
DELETE DMS,GORP
EXIT
```

PRODUCE RESIDENT DMS AND NON-RESIDENT OPERATOR COMMUNICATIONS LOAD MODULES

```
$ ... STEP 6. CREATE DMS DYNAMIC DISC MODULES OF PERTINENT FILES
$FILEMA
SAVEP JOBCTL,GORP
SAVEP FILEMA,GORP
SAVEP CATALO,GORP
SAVEP OPCON,GORP,,OPCOM
SAVEP OPCON1,GORP,,OPCOM1
SAVEP OPCON2,GORP,,OPCOM2
SAVEP OPCON3,GORP,,OPCOM3
SAVEP OPCON4,GORP,,OPCOM4
SAVEP OPCON5,GORP,,OPCOM5
SAVEP A.BAED,GORP
SAVEP A.COPY,GORP
SAVEP A.DISP,GORP
SAVEP A.INIT,GORP
SAVEP A.LINE,GORP
SAVEP A.MANP,GORP
SAVEP A.MIS1,GORP
SAVEP A.POSN,GORP
SAVEP A.MISC,GORP
SAVEP A.REGS,GORP
SAVEP A.UDAT,GORP
SAVEP S.ABSJ,GORP              (ABORT SPOOL INPUT JOB)
SAVEP S.FGSP,GORP              (FOREGROUND OUTPUT SPOOLER)
SAVEP S.FGCP,GORP              (ACRONIM COMMAND PROCESSOR)
SAVEP BASIC,GORP               (BASIC COMPILER)
SAVEP DEBUG                    (LINK MODULE OF DEBUG)
SAVEP SYSDAT                   (LINK MODULE OF CURRENT SYSTEM DATA MODULE)
SAVEP L60001                   (LINK MODULE OF SYSTEM)
SAVEP S54390                   (CONFIGURED SOURCE - SLMSGS)
SAVEP S60690                   (CONFIGURED SOURCE - SYSDAT)
SAVEP S61090                   (CONFIGURED SOURCE - BGDATA)
EXIT
$ ... STEP 7. DELETE TEMPORARY FILES
$FILEMA
DELETE OPCON,GORP
DELETE OPCON1,GORP
DELETE OPCON2,GORP
DELETE OPCON3,GORP
DELETE OPCON4,GORP
DELETE OPCON5,GORP
DELETE SLMSGS
DELETE SYSDAT
DELETE BGDATA
DELETE DSR
DELETE L60001
EXIT
$WEF 5
$RFW 5
```

APPENDIX H
DEVICE BOOTSTRAPS

DISC BOOTSTRAP

		CU	EQIV	'500
00	00000000		HLT	
01	62500013		TOA	WC
02	00714500		OAW	CU
03	05000012		TMA	CW
04	00700500		OCW	CU
05	00730500		ISW	CU
06	22600005		BNZ	*-1
07	00110200		QBB	B7
10	22600005		BNZ	*-3
11	21000020		BUC	'20
12	40000000	CW	DATA	B23
13	00000100	WC	DAC	'100
14	00000020		DAC	'20

ASR TAPE READER BOOTSTRAP

		CU	EQIV	'000
00	00000000		HLT	
01	00030110		TOB	'110
02	00700000		OCW	CU
03	00720000		IDW	CU
04	00140000		COB	0
05	22200003		BOZ	*-2
06	63200004		TNJ	4
07	00420006		LLA	6
10	00724000		IDW*	CU
11	22600010		BNZ	*-1
12	23200007		BWJ	*-3
13	00240020		CZA	
14	22200020		BOZ	'20
15	15100020		TAM	'20,I
16	23100006		BWI	*-8

CARD BOOTSTRAP

		CU	EQIV	'400
00	00000000		HLT	
01	00030010		TOB	'10
02	00700400		OCW	CU
03	00720400		IDW	CU
04	22600003		BNZ	*-1
05	00420014		LLA	12
06	00724400		IDW*	CU
07	22600006		BNZ	*-1
10	00240020		CZA	
11	22200020		BOZ	'20
12	15100020		TAM	'20,I
13	23100003		BWI	*-8

HIGH SPEED PAPER TAPE BOOTSTRAP

		CU	EQIV	'100
00	00000000		HLT	
01	00030110		TOB	'110
02	00700100		OCW	CU
03	00720100		IDW	CU
04	00140000		COB	0
05	22200003		BOZ	*-2
06	63200004		TNJ	4
07	00420006		LLA	6
10	00724100		IDW*	CU
11	22600010		BNZ	*-1
12	23200007		BWJ	*-3
13	00240020		CZA	
14	22200020		BOZ	'20
15	15100020		TAM	'20,I
16	23100006		BWI	*-8

7-TR. MAGNETIC TAPE BOOTSTRAP

		CU	EQIV	'700
00	00000000		HLT	
01	62500406		TOA	'406
02	00700700		OCW	CU
03	62500012		TOA	PL
04	00714700		OAW	CU
05	05000016		TMA	CW
06	00700700		OCW	CU
07	22600006		BNZ	*-1
10	00730700		ISW	CU
11	22600010		BNZ	*-1
12	00110204	PL	QBB	B7B2
13	00600020	NOP	DATA	'00600020
14	22600010		BNZ	*-4
15	21000020		BUC	'20
16	40036506	CW	DATA	'40036506

9-TR. MAGNETIC TAPE BOOTSTRAP

		CU	EQIV	'700
00	00000000		HLT	
01	62500406		TOA	'406
02	00700700		OCW	CU
03	62500012		TOA	PL
04	00714700		OAW	CU
05	05000016		TMA	CW
06	00700700		OCW	CU
07	22600006		BNZ	*-1
10	00730700		ISW	CU
11	22600010		BNZ	*-1
12	00110204	PL	QBB	B7B2
13	00600020	NOP	DATA	'00600020
14	22600010		BNZ	*-4
15	21000020		BUC	'20
16	40034506	CW	DATA	'40034506