AA61770-00
Original 1/75

# VULCAN

# GENERAL SPECIFICATION

January, 1975

HARRIS

COMMUNICATIONS AND INFORMATION HANDLING

**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: 416
CONSISTING OF THE FOLLOWING:

| Page No. | Change No. | Page No. | Change No. | Page No. | Change No. |
|---|---|---|---|---|---|
| Title | Original | | | | |
| A | Original | | | | |
| i thru xxii | Original | | | | |
| A-1 thru A-27 | Original | | | | |
| B-1 thru B-74 | Original | | | | |
| C-1 thru C-36 | Original | | | | |
| D-1 thru D-78 | Original | | | | |
| E-1 thru E-60 | Original | | | | |
| F-1 thru F-70 | Original | | | | |
| G-1 thru G-47 | Original | | | | |

This manual corresponds to Version 00 of VULCAN.

Insert Latest Revision Pages. Destroy Superceded Pages.

# CONTENTS

## CHAPTER A. GENERAL DESCRIPTION

CONTENTS (Continued)



CHAPTER B. JOB CONTROL

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

## CHAPTER D. SYSTEM SERVICES

CONTENTS (Continued)

Section | Page

CONTENTS (Continued)



CHAPTER E. INPUT/OUTPUT

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

## CHAPTER G. SYSTEM GENERATION

CONTENTS (Continued)

CONTENTS (Continued)

# TABLES

# ILLUSTRATIONS

# CHAPTER A

# GENERAL DESCRIPTION

# SECTION I
# INTRODUCTION

## 1-1 SCOPE

This manual contains the basic operational and programming considerations pertinent to the Slash 4 VMS Virtual Core Manager (VULCAN). The manual is divided into seven chapters, each of which can be considered as a manual in itself. This allows certain classes of users to skip those areas which are not of value to them.

- Chapter A is an introduction to the philosophies and structures of VULCAN. This should be read by anyone who will use VULCAN, and may also be of interest to any casual reader.
- Chapter B contains the operation procedures for batch and interactive programming, and is of value to anyone who will use this mode of operation.
- Chapter C is a discussion of system processors and program construction. This should be read by anyone who will be constructing a program to run under VULCAN.
- Chapter D contains descriptions of system services. This is of value to assembly language programmers interested in interacting with the VULCAN system.
- Chapter E is devoted to input/output. It is of value mostly to assembly language programmers interested in I/O programming, but also contains sections on operating I/O devices.
- Chapter F is the operators guide. This is a reference manual for system managers and operators.
- Chapter G is the system generation manual. This need only be read by persons who will be generating VULCAN systems.

## 1-2 SYSTEM FEATURES

VULCAN is a three level priority driven operating system. At the highest priority range are time-critical real-time programs. Concurrently, at the middle priority ranges, are interactive terminal users, who may edit, compile, and execute programs. Each interactive operation is connected with a specific teletype or CRT terminal. The lowest priorities are reserved for batch processing. Batch jobs execute via a software structure called a "control point". Multiple concurrently executing batch jobs each are controlled via and identified by a control point.

VULCAN priorities are normally of the range 0 to 63, with zero being the lowest priority. This range may, however, be changed via system generation parameters, as discussed in Chapter G.

VULCAN features include:

- Full use of Slash 4 VMS virtual hardware.
- Demand paging for control point and interactive programs.
- Full memory residency of high priority real-time programs to ensure rapid response to external stimuli.
- Up to 256K of program space regardless of physical memory size.
- Dynamically configurable operating system, through use of non-resident I/O handlers and system services, to minimize core requirements based on current program mix.
- Extensive disc file system, providing full protection along with efficient access.
- Re-entrant code generation and re-entrant system processors.
- Complete memory protection for operating system, other user programs, and re-entrant pages.
- Spooled I/O for batch devices.
- Time slicing between programs of equal priority.
- Concise job control language with flexible editing package.
- Operator control via console CRT.
- Timer scheduling of real-time programs.
- External interrupts to trigger real-time programs.
- Automatic system generation via parameters entered by the user at his site.

# SECTION II
# VMS OPERATION

## 2-1 GENERAL

The Slash 4 Virtual Memory System (VMS) hardware is an integral part of VULCAN operation. VMS was designed in conjunction with VULCAN to provide an efficient hardware/software interface.

## 2-2 ADDRESSING SCHEMES

There are two addressing modes in VMS. In the monitor mode, CPU Operation is identical to that of a non-VMS Slash 4. No address translation occurs and all memory references are absolute. No memory protection is provided in the monitor mode. The resident VULCAN system, non-resident handlers and services, and monitor programs execute in the monitor mode, in the lowest 32K of real memory.

In the user mode, all memory addresses are mapped. This includes the Program Counter (P-register) and all indirect and indexing operations. An 18-bit logical address is mapped into an 18-bit physical address by the VMS.

Logical addresses are divided into a series of 1024-word segments, called pages. Any user program can consist of up to 256 pages, of which only the first 64 can be executable code, and the remainder must be data only. Using VMS the program size is unaffected by the actual memory size.

Address translation is accomplished through use of Virtual Address Registers, or VARs. Each executing program under VULCAN requires a contiguous string of VARs equal in length to the number of pages the program requires. Associated with each program is a Virtual Base Register (VBR) to which a page displacement (upper 8-bits of an 18-bit address) is added to address a VAR. This VAR then gives the resultant physical memory page in which the logical page resides. This is combined with the low 10 bits of the virtual address to provide a physical memory location.

There are 1024 VAR's in a Slash 4 VMS system. VULCAN allocates contiguous strings of them to executing programs. Thus, if four programs each require 256 registers (maximum) then no other program can be started. To prevent this condition from occurring, program sizes should be kept to the minimum actually required for successful operation.

VULCAN may only use a portion of the VAR's depending on the size of the page swapping disc area. To ensure that a slot is always available on disc for any given core memory page, VULCAN will only use as many VAR's as there are pages in the swapping area. 10 sectors are required for each page on disc. Thus, if a swapping disc area of 2000 sectors is used, only 200 VAR's will be used. The size of the page swapping disc area is controlled by a system generation parameter (refer to Chapter G).

## 2-3 DEMAND PAGING

Programs larger than the amount of memory physically available can execute under VULCAN because VMS provides a feature called demand paging. This allows some of the program's pages to be stored on disc and loaded into core only when required. This is accomplished by setting a unique value into the VAR for pages on disc, and when such a VAR is referenced, an executive trap is triggered allowing VULCAN to perform a page swap. The requested page will be loaded into an empty page, if one exists, otherwise, a page is swapped out to the page swapping area on disc to make room for the required page. The determination as to which page is to be swapped out is made by a user tunable combination of priority and length of time since last reference.

# SECTION III
# PROGRAM TYPES

## 3-1 GENERAL

VULCAN differentiates program types based on the mode in which it executes and the memory in which it resides.

## 3-2 INTERACTIVE/CONTROL POINT PROGRAMS

Programs which execute at control points or interactive terminals are referred to as Interactive/Control Point Programs. Any program in this class may run at either structure. These programs are sometimes referred to as "background" programs.

All Interactive/Control Point programs execute in the user mode. They are all divided into 1024-word pages on disc and loaded as required by use of demand paging. These programs may not execute as real-time or monitor programs.

System diagnostic error messages for interactive programs are written to the respective terminal. Diagnostic messages for control points are written to the diagnostic PDN ( LFN 3) which is normally assigned to the list output device or disc area.

An Interactive program is always identified by its terminal physical device number. Control points are identified by a unique single letter (A-Z). This letter is assigned by the control point initiation mechanisms.

Control points and Interactive terminals are controlled by the re-entrant processor Job Control. Job Control reads the commands input and may turn control over to user programs if specified in processor call commands. When a user program at a control point or terminal exists, control is restored to Job Control to continue reading Job Control language commands. These commands are described in Chapter B.

## 3-3 CONTROL POINT ALLOCATIONS

The number of control points active at any one time is determined by an automatic control point algorithm which looks at the number of memory pages available and the number currently in use. If manual control is desired, an Operator command is available to select the number of control points.

Each control point is assigned a priority range of jobs which it can run. Any control point can run a high priority job, but as the job' s priority goes down, so does the number of control points that can run it. This works as follows: the priority range for batch jobs (normally 0-15) is divided by the number of control points (either selected by the automatic algorithm or explicitly defined by the operator). This value is then used to determine the minimum priority each control point can run. This value starts at zero and the above result is added for each additional control point. For example, if 4 control points were to be run, then the priority ranges would be assigned as follows:

| Control Point | Priority Range |
|---|---|
| A | 0-15 |
| B | 4-15 |
| C | 8-15 |
| D | 12-15 |

A priority 8 job could be run by control points A, B, or C. A priority 3 job could only be run at control point A. If A were busy with another job, the no more priority 0-3 jobs could be initiated, even if control points B, C, and D were free.

This method ensures rapid response to high priority jobs.

## 3-4 INTERACTIVE/CONTROL POINT PRIORITIES

Interactive Terminals normally execute at priorities between 16 and 31. However, any priority below 32 may be specified when the system is generated.

Control points execute at priorities from 0 to 15. The priority may be specified on a control card, but if not, a priority is calculated by VULCAN based on time limit, and spool output line limit.

## 3-5 REAL-TIME PROGRAMS

Real-time programs are the "foreground" programs of VULCAN. Most real-time programs execute in the user mode as virtually addressed programs. However, a special class of real-time program, the Monitor Program, which is discussed in Section 3-6, is available if it is necessary to run in the monitor mode.

Real-time programs are not directly attached to any terminal. That is, they are not normally executable as a standard terminal or control point processor. However, real-time programs initiated from an interactive terminal, will have their system diagnostic messages (system errors and abort conditions) output to that terminal. All other real-time programs will use the operator CRT as the diagnostic message device.

There are two classes of real-time programs. Those at priority 32 and above are designed to be immediate response type programs. They are loaded in their entirety into memory when initiated, and never have any pages swapped to disc. Thus, there is no demand paging associated with such a real-time program.

The other class of real-time program, which executes below priority 32, is designed for non-time critical operations. These programs are paged and swapped just as Interactive/Control Point programs.

A special class of real-time program is the resident real-time program. These programs are initiated automatically whenever VULCAN is loaded from disc. Any priority range real-time program may be a resident real-time program. They are executed at the priority specified when they are Vulcanized.

## 3-6 NON-RESIDENT MONITOR PROGRAMS

This class of real-time program is designed to execute in the monitor mode in real memory under VULCAN. These programs are used if it is necessary to address real memory such as when doing CBC I/O to user devices, or generating user interrupt routines.

Monitor programs are also used when it is necessary to dump a bootstrap or absolute load module as discussed under $DLOAD and $DBOOT in Chapter B.

Monitor programs are loaded into absolute memory when initiated and relocated as appropriate. When they are no longer being used, the memory is released for other uses. There is no memory protection for monitor programs under VULCAN.

## 3-7 TIMER SCHEDULED PROGRAMS

Any type of real-time or monitor program may be placed on a timer schedule. Two types of operations are available, initiate and wakeup.

Any real-time or monitor program may be placed on the timer schedule to be initiated at any specific time in the future, and re-initiated at any frequency greater than once every 2 seconds. In such cases, the program is loaded from disc each time it is initiated (unless it is re-entrant, in which case some pages may be core resident).

Any real-time or monitor program may go to "sleep" (Chapter D) and be triggered back into execution by a wakeup call. A program which goes to sleep remains core resident and ready to execute. Wakeup operations may be placed on the timer schedule for any time in the future and to be optionally repeated at any frequency. The use of sleep and wakeup is designed for rapid response programs, where high frequencies of execution are necessary.

The VULCAN timer schedule list is maintained on a disc area. Schedule entries specified as permanent remain active when VULCAN is booted from disc and are re-calibrated at that time as necessary. Temporary entries are removed when VULCAN is loaded into memory.

## 3-8 NON-RESIDENT HANDLERS

Non-Resident Handlers are Monitor Programs that are designed to be used as system services and I/O device handlers. Non-Resident Handlers function exactly as Monitor Programs except that they are not an entire program in themselves. They are designed to be a subroutine or service called by another program. All VULCAN I/O handlers (except the disc) and most System services are contained in Non-resident handlers.

User written Non-Resident Handlers may be called only by internal linkages established by Monitor Programs. The ability to access a new user written service via BLU linkages requires customer modification of the resident VULCAN.

Non-Resident Handlers may also be used for dumping load modules under the $DLOAD and $DBOOT commands as discussed in Chapter B.

## 3-9 RE-ENTRANT PROGRAMS

Any virtual program (Interactive/Control Point or Real-time) can be written with one or more re-entrant pages. In this case it becomes a re-entrant program. All VULCAN system processors (Job control, FORTRAN, Assembler, Vulcanizer, etc.) are re-entrant. In addition the FORTRAN Compiler outputs re-entrant code suitable for use as a re-entrant program.

Pages containing only executable code and fixed data are classed as re-entrant. These must consist of non-modified code only. For example, the use of the BSL (Branch and Save Long) instruction is prohibited into re-entrant pages. VMS hardware protects re-entrant pages from modification by any program using them.

Re-entrant pages are shared when two or more copies of a program are executing concurrently.

## 3-10 RE-ENTRANT LIBRARY PROGRAMS

Re-entrant library programs are effectively one or more re-entrant subroutines Vulcanized together to form a re-entrant module. This re-entrant library consists of a series of re-entrant pages followed by non-reentrant data pages, and is thus a form of a re-entrant program.

Any virtual program ( Interactive/Control Point or Real-time) can call upon at most one or more re-entrant library. When in execution, a re-entrant library may be shared by one or more concurrently executing programs, regardless of whether the programs themselves are re-entrant.

Re-entrant libraries must be Vulcanized prior to being referenced as a library by a regular program. The Vulcanizer reserves logical address space in the program for the library pages. External linkages are satisifed through a vector table of external definitions sorted alphabetically at the front of the re-entrant library.

When changing a re-entrant library that is catalogued into an existing program, a special Vulcanizer option should be set to ensure that the library remains the same size. If the external linkages or the number of pages in a re-entrant library changes, all programs referencing the library must be re-vulcanized.

The BASIC run-time library is an example of a re-entrant library.

## 3-11 MONITOR COMMON BLOCKS

Monitor Common Blocks are a type of program disc area which contains data shared by one or more programs.

Any number of Monitor Common areas may exist on a VULCAN system. A program may use one or more such blocks. They are referenced in the user program as regular COMMON blocks with a special statement to identify them as Monitor Common.

Monitor Common areas must be generated just as any other disc area. When generated, they are zeroed. Monitor Common pages are loaded into memory as needed by executing programs and returned to the disc when swapped out. Values set into Monitor Common remain there until modified or the area is eliminated, even if VULCAN is reloaded from disc.

# SECTION IV
# DISC STRUCTURE

## 4-1 GENERAL

A disc area is one or more strings of contiguous sectors of disc space which are contained on a single disc pack and having a unique disc area name. One or more disc areas are contained on a disc pack. VULCAN supports up to 256 disc packs in a system. Removable disc packs are mounted by the operator as disc areas contained on them are referenced.

Disc area names consist of an 8-character qualifier, and an 8-character areaname. These names are normally stored internally in truncated ASCII (6 bits per character). Externally they are represented as QUALIFIER*AREANAME. The components of the name are discussed below.

## 4-2 QUALIFIERS

Qualifiers consist of a 1 through 4 digit account number, followed by a 1 through 4 character identifier, the first character of which must be alphabetic. The following are examples of valid qualifiers:

1234ABCD

1A

ØØØ1A

1234U123

Accounts having less than 4 digits are assumed to be zero filled. Thus ØØØ1A and 1A are identical qualifiers.

Use of account ØØØØ is reserved for system qualifiers. The qualifier ØØØØSYST is used as the default system qualifier. ØØØØSPOL is used for input and output spool disc areas. ØØØØACNT and ØØØØACSM are used for accounting areas.

When referencing disc areas under VULCAN, the system qualifier may be merely placing an asterisk ( * ) in front of the areaname. Hence, *FORTRAN is actually ØØØØSYST*FORTRAN.

Each VULCAN user has one or more sign-on qualifiers. Once signed-on with one of these, if a disc area is referenced without a qualifier, then the sign-on qualifier is implied. For example, if a user has signed-on with qualifier 1234ABCD, then a disc area reference of the form XYZ actually references 1234ABCD*XYZ.

## 4-3 AREANAMES

All areanames must be at least two characters long with a maximum of eight. The first character must be alphabetic. The use of 2-character areanames is restricted to Control Point and Interactive Work areas (Refer to Paragraph 4-4).

All Non-Resident Handler and Monitor programs are named as V:nnnn:V, where "nnnn" is the 4-character program or service name. For example, the system magnetic tape handler is ØØØØSYST*V:MAGT:V. For a Non-Resident Handler or Monitor Program to be used as such, it must have the qualifier ØØØØSYST. Non-Resident Handlers and Monitor Programs not under the system qualifier may be used only for $DBOOT and $DLOAD dump modules (Chapter B).

Other names of the form V:nnnnnn under the system qualifier are used for system disc areas. The user should use care in selecting names starting with "V:".

## 4-4 CONTROL POINT AND INTERACTIVE WORK AREAS

Certain 2-character areanames are reserved as work disc areas for Control Point and Interactive Terminal programs. These areas are listed below.

| | |
|---|---|
| ED | "Edit Area" - for editing |
| ER | "Edit Record" - for character editing |
| LO | "List Output" - for program listings |
| LR | "Link Ready" - for binary output from compilers or the assembler |
| OA | "Output Area" - used for certain display functions while editing |
| RC | "Record" - used to hold records while editing |
| W1 | "Work One" - general work area |
| XE | "Execution" - Vulcanizer temporary program area |

Additional work area names may be specified when the system is generated (refer to Chapter G).

Whenever Control Points or Interactive Terminals reference these areas, whether being assigned or generated, the control point identification letter or terminal PDN is appended to the areaname to generate a unique name. If the resultant disc area does not exist, it is generated automatically by the system, under the user's sign-on qualifier, and made private to the current user. Work areas are eliminated when a terminal signs-off or a control point job terminates. Thus each terminal and control point has only those disc areas it is using generated on disc.

## 4-5 DISC AREA TYPES

There are three basic classifications of disc areas. These are blocked, unblocked, and random. Unblocked areas are always accessed on sector boundaries. For example, two 27 word records are written to an unblocked disc area, each will occupy a full 112 word sector, with the unused portions being zeroed. All record addressing is done in terms of sectors.

Blocked areas provide a more efficient packing scheme, in that multiple records are packed in each sector with a one word gap indicator between each record. In addition multiple blanks are compressed into a single blank count character in symbolic records.

Both of these types of areas provide read/write protection. That is, any number of programs may read the disc area simultaneously. However, no writing may be done if anyone is reading the area. If someone is writing the area, no one else may open it. The random disc area type inhibits this checking. Random areas are identical to unblocked, except that any number of programs may be simultaneously reading and writing the area.

## 4-6 DISC AREA ACCESS LEVELS

Each VULCAN user has an access level assigned to him when added to the system. Access levels range from a low of 0 to high access level of 15. Each disc area also has an access level.

When a user generates a disc area, he may request an access level for it of any value at his level or below it. In order to reference a disc area, a user must have an access level at least as great as that of the disc area, or the disc area will not exist to him.

For example, if user A generates disc area*XYZ at access level 3, then user B at access level 0 will not be able to reference the disc area, but user C at access level 12 will be able to, subject to access bit restrictions discussed in the following paragraph.

## 4-7 DISC AREA ACCESS PRIVILEGES

Associated with each disc area is a series of access bits. These bits control the read, write, delete, and execute functions for classes of users.

The owner (user who generates the disc area) is always allowed to read or execute the disc area. Access bits are available to inhibit him from writing or eliminating the disc area. A disc area is always generated with owner delete and owner write, but these may be removed by the owner at a later time.

Additionally, each disc area can have some public (any user) or account (those who are signed on in the same account as the disc area) accesses, but not a combination of these. Within public or account, read, write, delete, and execute privileges can be granted. For example, if the disc area has Account Write, then it can only be given other account accesses, but no public accesses.

Only the owner of a disc area can change the access bits of a disc area. To do Job Control editing on a disc area requires write access. Renaming or squeezing of disc areas requires delete access.

## 4-8 DISC AREA ALLOCATION

Disc Areas are divided into one or more granules. A granule is a contiguous allocation of disc space. Multiple granules of a disc area need not, however, be contiguous. The granule size for a disc area is specified when the area is generated. If not specified, the system default for the disc pack may be used. This is a system generation parameter.

As a disc area is written in a sequential mode, additional granules are automatically allocated by the system as each previous granule is filled. Granules are added until the disc pack is filled or the specified maximum size for the disc area is reached. This division of a disc area into granules is transparent to the user. The system disc handler will access the appropriate granules as necessary and even break an operation across a granule boundary into separate functions. The granule size requested by the user of a blocked disc area may be modified by the disc area generation logic to be a multiple of the blocking factor in order to minimize disc accesses.

A pointer to each granule of a disc area is maintained in the Master Area Index (MAI) sector for each disc area. This sector is appended to the front of the first granule. The MAI sector will hold pointers for up to 86 granules. If additional granules are allocated, another index sector (Extended Area Index or EAI) is allocated on the front of the next granule. This extension process continues until the disc area can no longer expand as discussed above.

Referencing data pointed to by Extended Area Index's requires additional disc accesses for each transfer. To minimize this action, granule sizes should be chosen so that in most cases the entire disc area can be contained in less than 86 granules. However, the granule size should be as small as possible to minimize disc space fragmentation.

Program disc areas always consist of only a single granule of the required size.

## 4-9 DISC DIRECTORY STRUCTURE

Each VULCAN disc pack has a disc directory structure locating those disc areas contained on it. The master pack (Pack 1) directory has entries for all disc packs. When it is desired to have a VULCAN system reference disc areas on a pack not previously used by the particular system, an operator command is available to merge the Satellite Pack directory entries into the master pack directory (Chapter F).

Each disc pack as a Master Disc Directory (MDD). This directory has four word entries for each qualifier under which disc areas exist on the pack. The MDD entry for a qualifier points to a Qualifier Disc Directory (QDD) which contains an 8-word entry for each disc area having that qualifier. This structure is demonstrated in Figure 4-1.

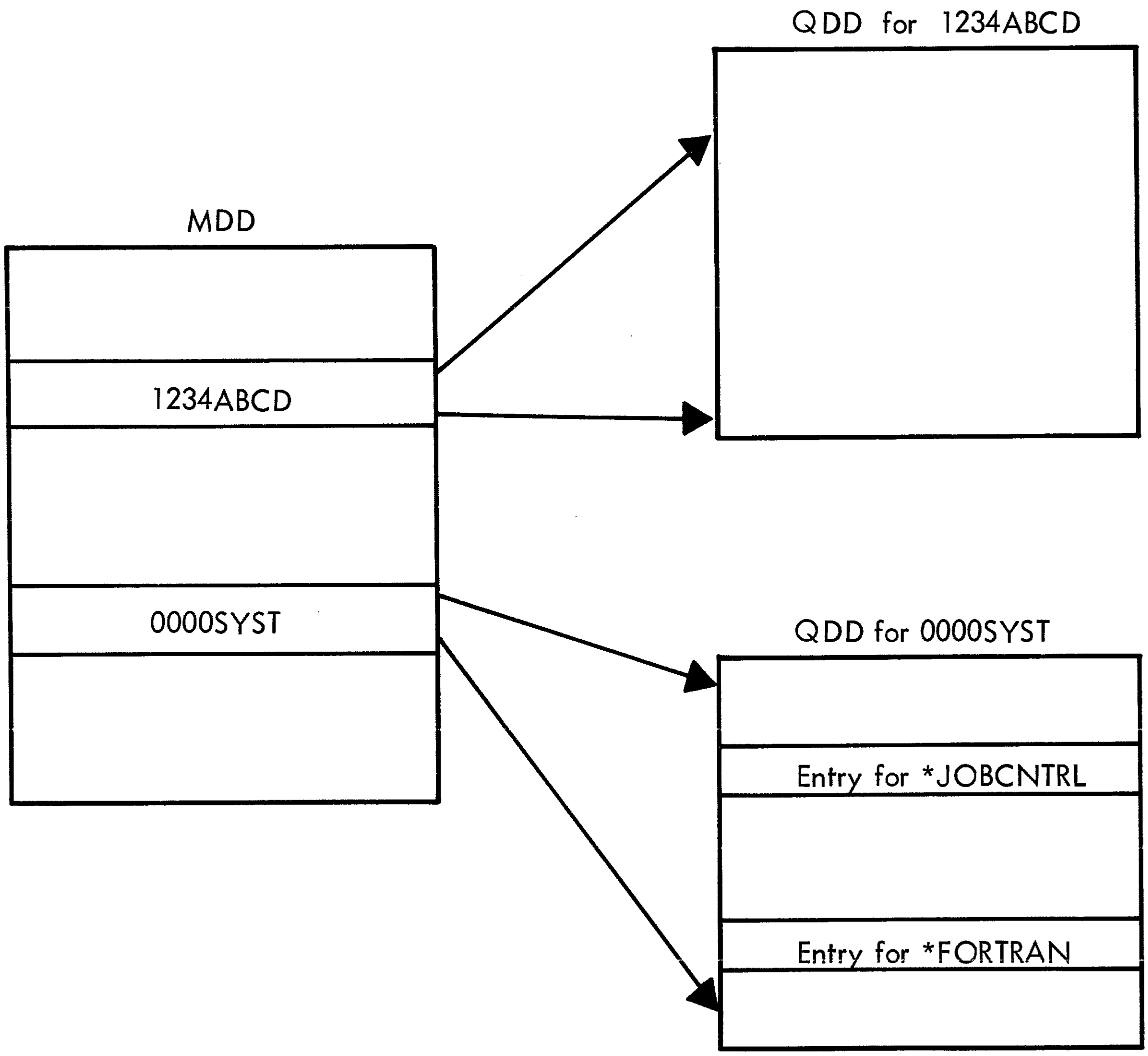



Figure 4-1. Disc Directory Structure

Both the MDD and QDD are hashed. That is, an entry is passed through a randomizing function to produce a random sector within the directory based upon the name being entered. When a particular sector of an MDD or QDD becomes full, the system will rehash the directory, producing one of approximately double the original size. MDD's start out at 7 sectors (enough for at most 196 qualifiers). QDD's start out at one sector less than the default granule size for the pack, with a minimum of 3 sectors.

A core-resident disc directory option is available for any disc area. This option causes the entire MDD and QDD entry for the disc area to be kept in memory. This core resident list is searched first when a disc area must be accessed.

## 4-10 DISC AREA LOOKUP PROCEDURE

The following procedure is executed when a disc area is assigned:

a) Scan core directory for entry.

b) Check program qualifier list for the QDD location or

c) Hash qualifier into MDD to locate QDD.

d) Hash areaname into QDD to locate entry for disc area. (MAI Sector is defined in QDD entry.)

Once a given program has located a qualifier entry in the MDD, the MDD entry is copied into a resident qualifier list for that program to eliminate the first disc access (step c above) on subsequent assignments to disc areas with that qualifier.

When the disc area is initially opened, the following procedure is performed:

a) Read in MAI sector of disc area to locate first granule.

c) Re-write MAI sector with updated time and date of last reference.

Upon detecting the first write by a program to a disc area, the MAI is again read and re-written to update the time and date of last write to the disc area.

## 4-11 DISC SPACE ALLOCATION

Disc space is always allocated in fixed units called SAB's (Space Allocation Bits). Each SAB can represent one or more sectors of disc space. When disc space is allocated, a multiple of SAB's is always allocated, possibly resulting in disc space overhead. For example, if the SAB value for a particular pack is 40, then generating a 30 sector disc area granule will allocate one SAB worth of space, or 40 sectors, wasting 10 sectors.

The SAB value for each disc pack is a sysgen or operator parameter. Smaller SAB values save disc space, but increase system overhead when allocating or deallocating disc space. The VULCAN default SAB value is 2 sectors.

The granule size for a disc pack should be at least as large as the SAB value to prevent wasting large amounts of space. Preferably, the granule size should be an integer multiple of SAB units.

# SECTION V
# USERS AND ACCESS RESTRICTIONS

## 5-1 USER NUMBERS

Under VULCAN, each user has a unique user-number. This identifies the user uniquely. Associated with each user is a user-name, which is a string of 12 ASCII characters. The user name is output as appropriate in place of the private user-number.

User numbers can be of 2 forms. If a strictly numeric user number is desired, any string of from 1 to 12 decimal digits may be used. The other form is to use a string of from one to six characters, excluding blanks or commas, the first of which must be alphabetic. Each particular VULCAN site should choose one of these schemes, as intermixing of text and numeric user number formats may produce conflicts.

## 5-2 SIGN-ON QUALIFIERS

Associated with each user is one or more sign-on qualifiers. The format of a qualifier was discussed in paragraph 4-2.

When a user signs on to VULCAN, either at a terminal, or with a control point job, he supplies in addition to his private user-number, one of his sign-on qualifiers. This qualifier becomes the default for all disc areas referenced during that session, and is also used as the account to which computer usage is charged. The only way to change sign-on qualifiers is to log-off and then sign-on again. There is no limit to the number of sign-on qualifiers a user may have, nor is there a limit to how many users may share a common qualifier.

## 5-3 ACCESS BITS

Associated with many system services, I/O devices, operator commands, and other features is an access bit mask. This is one 24-bit word with one or more bits set.

Each user also has an access bit mask, with zero to 24 bits set. In order for a particular user to access the specified function, one of his access bits must match one of those associated with the function. In other words, the logical "and" of the two access words must produce a non-zero result.

Access bits for operator commands are predefined and listed in Chapter F. Access bits for I/O devices and system services may be set via System Generation parameters.

# SECTION VI
# RESOURCE ALLOCATION

## 6-1 GENERAL

Certain types of physical devices and high-speed (semi-conductor) memory blocks are allocatable resources under VULCAN. That is, the system maintains control and allocates them as required to waiting programs. These system resources can be broken down into the four classes discussed below.

## 6-2 MAGNETIC TAPE RESOURCING

Each magnetic tape drive is an allocatable resource device. VULCAN keeps track of the type and speed of each drive, and allocates it to each program as required based on priority. Once a program has successfully allocated a tape drive, it retains control until it is freed, or the program finishes.

Tape resourcing is accomplished by specifying a tape identifier to the system along with tape drive parameters such as speed, density, frame width, etc. When an appropriate tape drive becomes available, the operator is automatically told to mount the tape. The system monitors tape drive status and when the new tape has been mounted, it becomes available for access by that program.

## 6-3 DISC PACK RESOURCING

Removable disc packs require allocatable disc drives. When a program resources a disc pack, a check is made to locate any free drives of the appropriate type. If one is found, the operator is requested to mount the indicated pack. When it becomes ready, the pack is available for use by that program and any other legitimate user of it.

If no drive is available, a search is made to locate the highest priority program that is using each appropriate drive. If the requesting program is of higher priority than the lowest priority found in this search, a clamp is placed on the disc pack on the drive and no subsequent assignments may be made to it. If not higher, the requesting program waits until all higher priority activity has ended. When the drive is no longer being used by any program, the operator is requested to mount the required pack as discussed above.

## 6-4 ALLOCATABLE PHYSICAL DEVICES

Certain other physical devices are allocatable and therefore must be resourced. All high-speed paper tape readers must be resourced. It is also necessary to resource an interactive terminal device (teletype or CRT) when a program was not initiated from it. For example, an interactive program running from terminal A must allocate terminal B via a resource command before referencing it.

No operator interaction is required with physical device allocations. When the requesting program becomes the highest priority program waiting for the device, and the device is free, the allocation is made. Other programs are prevented from accessing the device until freed by the original program. The operator CRT does not require resourcing as it is not allocatable.

## 6-5 HIGH-SPEED MEMORY ALLOCATION

In configurations utilizing high-speed (semi-conductor) memory, the high-speed memory is an allocatable resource. Any program may request any number of pages (1024-words per page) of high-speed memory, and whether it should be contiguous or which pages it must use.

No operator interaction is required for high-speed memory. The highest priority program is given its request as soon as it can be satisfied. Within a program, high-speed memory must be referenced through use of Dynamic Core Manager (DCM) allocations.

## 6-6 CONTROL POINT RESOURCING

All control point resource requests must come at the beginning of the job. That is, all resource commands must immediately follow the job command. As each job is started, a check is made to see if the required resources are available. If so, they are allocated and job processing continues. If the resources are not currently available, then the job is rejected and placed back on the jobs to-be-run queue, with a 20 second delay prior to the next try, to allow the resources to become free.

Once allocated to a control point, a resource remains allocated until the end of job is reached unless it is returned via the Free command.

## 6-7 REAL-TIME PROGRAM RESOURCING

All resource requests for real-time programs must be made before the program is placed in execution. For this reason, all resource requests for real-time programs must be catalogued into the program via Vulcanizer commands.

Real-time programs attempt to allocate resources when they are initiated. They will remain suspended in the initiation sequence until all of the required resources can be allocated.

## 6-8 INTERACTIVE RESOURCING

Interactive Terminal users may request resources at any time. Optionally they may continue with other activities while the resource operation is being processed.

Interactive terminals initially request one or more resources as necessary. They may choose to wait for it, in which case the terminal is suspended until the resource can be satisfied. Alternately, the terminal user may enter the resource request in the queue, and then continue with other processing. At a later time he may interrogate the status of the resource request via a special command. When it becomes necessary to use the resource, he may then wait for it. After placing a resource request in the queue, it is necessary to have a successful status query or a wait request before the resource can be accessed.

# SECTION VII
# SPOOLED I/O

## 7-1 GENERAL

All card reader input and output to output-only devices (printers, plotters, card punches, and tape punches) is transmitted via spooled disc areas under VULCAN, Spool areas are generated on the work pack and have the qualifier ØØØØSPOL.

## 7-2 CARD READER INPUT

Card reader input is divided into two types. These are control point jobs, and program data files.

Control point jobs are preceded by a $JOB command. The entire deck is copied to a system spool disc area and then placed on a queue for processing by the control point interpreter. When the job has been executed, the spool disc area is eliminated.

Program data files are preceded by a $DATA command, denoting the program and user which will read the data. The entire data deck is copied to a spool disc area and an entry is placed in a resident queue. When the indicated program executed by the indicated user makes an assignment to the card reader device, the assignment is actually made to the dynamic spool area. The records may then be read just as if the input were being done from the card reader. When the assignment is removed, the dynamic disc area is eliminated.

Provisions have been made to input any form of binary deck to the spool area by use of special control cards. Thus any need for direct use of the card reader device is eliminated.

If an input spool area containing a program data file or control point job is found when VULCAN is booted from disc, it is eliminated This prevents accumulation of those files left over when the system is powered down.

## 7-3 SPOOLED OUTPUT

All printers, plotters, card punches and tape punches have output transferred via spool disc areas. No direct I/O is possible.

Printers accept symbolic records only for output, except the Versatec printer/plotters, which accept binary records for plot data. Plotters accept binary records addressing plotter functions. Card and Tape punches accept symbolic, binary, and special action records to provide the appropriate output formats.

When an assignment is made to one of these devices, a spool disc area is dynamically generated, and the assignment is made to it. When the assignment is freed, the disc area is placed on the output spool queue for the appropriate device. As the appropriate device becomes free, the top priority spool area for that device is output to the device.

Output spool disc areas are eliminated after being output. If any are found when VULCAN is loaded from disc, the spool area is output to the device. Thus any spool areas being output when VULCAN halts are re-output when next booted.

## 7-4 MULTIPLE OUTPUT SPOOL DEVICES

In order to support sites having multiple printers or other output devices, a special feature is available which allows a spool output area to be destined to more than one device, and output at whichever one becomes available first. This is implemented by having each output device respond to zero, one, two or three alternate physical device numbers (ALTPDN's). These ALTPDN's may be shared between any assortment of output devices. The first device, as entered via system generation, having a specified PDN is the default if all of the devices responding to that PDN are free.

For example, if printer 2 has ALTPDN 6, and then printer 4 also has ALTPDN 6, output to physical device 2 goes to printer 2, output to physical device 4 goes to printer 4, and output to physical device 6 goes to whichever printer is first available, with preference to printer 2.

# SECTION VIII
# SYSTEM ERROR MESSAGES

All VULCAN abort and error messages are produced by a standard routine which formats the appropriate message and optionally outputs a text line. This section describes these capabilities.

VULCAN error conditions are all assigned a unique integer, in the range of 1 to 10000. System generated abort messages are assigned 1-99, with processor generated errors using the remaining values. Processors can cause the system to output the standard error message by passing the appropriate error number and optional error address through the system service SYSERR (Chapter D).

The size and content of the error message is determined by the program and type of error. For Control Point and Interactive programs, if the Job Control mode EM (Expanded message) is off, then error messages are output as:

ER xxx

where "xxx" is the error number. Also, when the EM mode is off, abort messages for these programs and all messages for Real-Time and Monitor programs are output as:

AB xxx @ yyyyyy X=zzzzz or

ER xxx @ yyyyyy

where "xxx" is the error number, "yyyyyy" is the memory location where the error occurred. The "zzzzzz" field is an optional value providing additional information about the abort. For example, on I/O aborts, the "X=" field provides the logical file number involved.

The third type of error message is the expanded message, which is available to Control Point and Interactive programs when the Job Control mode EM is on. In this form of the error message, a line of explanatory text is appended to the message formats described above, to provide a basic description of the error. These explanatory texts are contained on the system error message disc area, ØØØØSYST*VULCMESS, which contains 10,000 symbolic records, each of which corresponds to an error number, i.e. record 1 corresponds to error 1, record 9000 to error 9000, etc. If the customer site wishes to add additional error messages, he may change the appropriate lines of *VULCMESS so as to correspond to the error codes passed on the SYSERR call.

# SECTION IX
# ACCOUNTING

Accounting is an integral part of all VULCAN operation. This includes disc, CPU, and I/O device usage for all programs.

CPU execution time and charge time is kept through use of the Interval Timer option. If the individual customer site wishes to use this timer for other purposes, a special system generation option is available to inhibit use of the T register (Interval Timer) by VULCAN. Of course, this prevents calculation of program execution times, and the corresponding CPU time charges.

Accounting records for I/O usage are written to accounting disc areas by all I/O device handlers, and the program exit logic writes the accounting records for CPU and disc usage. These records may then be read by custom user programs or by the general purpose VULCAN accounting utility program ACUTIL. ACUTIL and the overall accounting structure is described in Chapter F.

# SECTION X
# VULCAN VERSION NUMBERS

Associated with each release of VULCAN is a revision level. This takes the form nnx where "nn" is the major revision level and "x" is the sub-revision level. For example, the initial VULCAN release is 00A with subsequent minor revisions making 00B, 00C, etc. The next major release would be 01A.

As any modification is made to any part of VULCAN, the sub-revision level is incremented. At this time, a new version is available to any customer. When major changes or additions are made, the major revision level designation is changed.

# CHAPTER B

# JOB CONTROL

# SECTION I
# INTRODUCTION

## 1-1 GENERAL

Job Control is the processor which interprets all the control cards or commands issued by a user.

Job Control is a standard control point or interactive terminal program. With the exception of some services which, when called, check specifically for the job control processor, Job Control obeys all the rules associated with control point or interactive programs. There is only one version of Job Control and the few differences between control point input and interactive input are specifically checked for.

Job Control is set up such that all commands requiring one input card appear to be executed by Job Control. Sometimes the commands call in another processor which in turn calls Job Control back in upon its completion. The standard compilers and assembler, when used in their short command format, have this appearance.

Example: The following is a sequence of control statements to Job Control

```
.
.
.
$ASSIGN,6=FILE1
$FORTRAN,FILE2
$LIST,FILE1
.
.
.
```

The assignment is to specify the place to output the FORTRAN listing. The command is handled by Job Control. The FORTRAN command calls in the Fortran compiler and specifies FILE2 as the disc area containing the statements to be compiled. The Fortran compiler reloads Job Control which reads the LIST command to display the output of the compilation. Thus, ostensibly all three commands are handled by Job Control.

## 1-2 CONTROL CARD FORMAT

A Job Control card consists of a command optionally followed by a string of arguments depending on the command. The command need not start in column one of the card.

All HARRIS supported processors and support packages use standard services to obtain commands and arguments from input control cards. These services are available to user written programs and are described in Chapter D. The use of the common decoding (SCANNER) services provides uniformity of input control card format. Some of the fundamental rules are listed below:

1) All blanks and commas are regarded as argument delimiters unless enclosed in double quotes ( " ).

2) Double quotes ( " ) may be enclosed in an argument. If the argument is started with double quotes two adjacent quote marks must be input in order that one be passed to the calling program.

Example:

Below are nine examples of the input card format (in the left column) and what is passed to the program as the argument (in the center column). The right column indicates whether the argument the program receives is ASCII or binary.

| | | |
|---|---|---|
| a) ...,ABCD,... | ABCD | ASCII |
| b) ...,A"BCD,... | A"BCD | ASCII |
| c) ...,"ABCD ",... | ABCDƀ | ASCII |
| d) ...,"A""BCD ",... | A"BCDƀ | ASCII |
| e) ...,"A,B,C",... | A,B,C | ASCII |
| f) ...,"3",... | 3 | ASCII |
| g) ...,3",... | 3" | ASCII |
| h) ...,3,... | 3 | Binary |
| i) ...,"A""B""C D,E",... | A"B"C D,E | ASCII |

3) All numeric values given to any processor are taken as decimal numbers. To enter octal constants a single quote must preceed the value.

Example:

a) ....,77,... translates to a decimal representation of 77

b) ....,'77,... translates to a decimal representation of 63

4) Ranges of numeric values are specifiable via a dash ( - ). A single quote mark in front of the argument converts both numeric values to octal.

Example:

a) ....,3-10,... translates as a binary 3, binary range of 8

b) ....,4-5,... translates as a binary 4, binary range of 2

c) ....,'3-10,... translated as a binary 3, binary range of 6

d) ....,' 14-17,... translated as a binary 12, binary range of 4

There are standard ways of entering certain types of arguments to Job Control and VULCAN processors. Special (non alphabetic or numeric) characters are used to denote certain functions.

1) Area names must start with an alphabetic character and be at least two characters in size.

Disc area names are written in one of three forms:

a) ...,0123ABCD*AREANAME,...

b) ....,*AREANAME,...

c) ....,AREANAME,...

In case (a) the qualifier (0123ABCD), which must always precede the name of the area, is given and is delimited from the area name (AREANAME) by the asterisk (*). Since both qualifier and area name may consist of up to eight characters, an argument that is a disc area name can be up to 17 characters in length.

In case (b) no qualifier is specified. However, since the delimiting asterisk is present, the Scanner will set the default qualifier to be the system qualifier (0000SYST). Thus for case (b) the entry "*AREANAME" is equivalent to "0000SYST*AREANAME".

In case (c) no delimiting asterisk is present and the Scanner will supply the sign-on qualifier (see section 2) as the default qualifier. Thus if the user intends to access only his "own" disc areas (created with his sign on qualifier) and work disc areas, he need never enter a qualifier associated with an area name explicitly.

2) It is necessary to distinguish between logical file numbers (LFN) and physical device numbers (PDN) as arguments where both are applicable. An example of this would be the copy command where the input or output of the copy request might be either a LFN or a PDN. The distinction is made by requiring a PDN to be preceded by a colon (:).

Example:

a) $COPY,7,6 — requests LFN 7 to be copied to LFN 6 (assuming that LFN's 6 and 7 have previously been defined)

b) $COPY,:'15,AREANAME — requests PDN 13 (decimal) to be copied to disc area AREANAME

In cases where only a PDN is applicable the colon ( : ) becomes optional.

Example:

a) $SPOOL,NAME,6 — requests disc area name NAME to be spooled to PDN 6

b) $SPOOL,NAME,:6 — is an identical request

3) Disc work areas are deleted for each user upon termination of the job. When a reference to a work area is made the area will be created if it does not already exist. The only reason for a user to explicitly create a work area would be if he wished it to have a larger granule size than it would defaultly be given. The names of work areas are two characters in length and recognized by VULCAN as work areas. The names recognized are either VULCAN required areas or work area names provided at GENASYS time.

4) An asterisk (*) appearing as the total argument indicates "self". Thus, to copy statements from the terminal to a disc area the following command might be issued:

$COPY,*,AREANAME

this is equivalent to:

$COPY,3,AREANAME

where LFN 3 is the diagnostic output device and is assigned to the input terminal.

5) A number sign (#) is a special character when used at the beginning of an argument string. It will abort the user currently since it is reserved for future VULCAN implementation.

6) The dollar sign ($) has special significance. All control point program commands are required to start each command to Job Control with a dollar sign. Interactive users may optionally start all commands with $. When two adjacent dollar signs

are encountered as the first two significant characters on a control card, Job Control treats the remaining sequence of characters in the command as the area name of a program to load.

Example:

a) LIST,... is an interactive user request to list an area

b) $LIST,.. is a request to list an area (interactive or control point)

c) $$LIST,.. is a request to load program "LIST"

The dollar sign is also used for other purposes when it stands alone on a control card. When it is the first character on the card, the control statement is taken to be a comment. When it is not the first character on a control card it acts as an "end of card" indicator.

Example:

a) ƀ$ƀ.... is treated as a blank card

b) $ƀ.... is a comment card

c) $LIST,NAME, $ ....COMMENT "COMMENT" is never read by requests to the Scanner. The "last" argument on the card is "NAME".

7) The exclamation point ( ! ), when the first character of the first argument on a control card, is recognized by Job Control as a label. Job Control will obtain the second argument and use it as the command.

Example:

a) $LIST,NAME lists area name NAME

b) ! LABEL, $LIST,NAME also lists area name NAME

## 1-3 ERRORS

When an invalid command, or a valid command with invalid arguments, is entered, Job Control will give the user an error code. This error code will, optionally, print an expanded message explaining the error. The "expanded message" ability is controlled via the $MODE command (section 3). It is switchable on or off, for example the following command will turn on the expanded message ability:

$MODE,EM=ON

The default option for all CRT terminals and control point programs is "EM=ON". It is "EM=OFF" for all teletype terminals. A $EM will give the error message command (Section 3).

## 1-4 USING THE SYSTEM

Most of the Job Control commands are equally applicable to control points and interactive terminal users. The method of getting into the system, however, is somewhat different.

Before an individual is able to use the VULCAN operating system he must be "added" to the system. This is done via the operator communication AU (add user) command. This procedure is explained in Chapter F.

### 1-4.1 Control Point Usage

To run a job at a control point the job must either be on cards and entered through the card reader or be on a disc area and be initiated via the interactive IJ (Insert Job) command. A control point job must begin with a valid job card. All control cards in the job must start with a dollar sign ( $ ). Most commands available to interactive terminal users are available to control point users. The commands which do not fall into this category are so noted in their definition (later in this chapter). The control point job is terminated by a $EOJ card.

When entering a job at the card reader an "EOT" card (6/9 punch in column one, columns two and three are blank) is required to terminate the job stream and close out the input spool area. (All card reader input is spooled.) The spool area is entered into the control point queue and the job will be run according to its priority, the work load in the control point queue and the number of available control points.

When the job is entered via the "IJ" command the same procedure is followed. No "EOT" card is required in this case since there is no input spool area to terminate.

### 1-4.2 Interactive Terminal Usage

There are two basic types of terminals supported by VULCAN. One is a teletype, the other a CRT. The CRT has to be a TEC 425 (model numbers 9050 or 9051), identical to the operator console. Other model CRT's are treated as teletypes. The starting up procedure differs for the two terminals. For a teletype, depress the bell command (control G) or "BEL". If the teletype is awaiting initialization, Job Control will be initiated, the message:

USER?

will be printed and a line feed given awaiting response. If the teletype is already on line the bell character will produce a bell in response. If nothing happens the teletype is either off line or has not been included in the system as an operational device. The user number should be entered in response to the "USER ?" request.

A CRT is initiated differently. The first command given to the CRT is both an initiation and the user number. When the transmit key is depressed for the first time Job Control is initiated, requests input from the CRT and gets, as a response, the line entered. Thus entering the user number on the CRT both starts up the CRT and supplies the user number.

Commands entered via interactive terminals only optionally need to be prefaced by a dollar sign ( $ ). The few Job Control commands that are not available to interactive users are specifically indicated as such in this chapter. A $OFF command is required to terminate use of the terminal by the user.

## 1-5 SIGNING ON

When signing on to interactive terminals two arguments are required: a sign-on qualifier and a user number. The sign-on qualifier must have been supplied as a "sign-on qualifier" when the user was added to the system. It is the default qualifier that will be used to access disc areas when an areaname is supplied without an explicit qualifier. The user number can be either six ASCII characters or, if all digits, a 12 digit decimal number and must be identical to that with which the user was added to the system.

Example:

| | |
|---|---|
| User: | "Bell" |
| Computer: | USER? |
| User: | 1234ABCD,SIGNON |
| | . |
| | . |
| | . |

where this particular user has a valid sign-on qualifier of "1234ABCD". The user's user number is the six ASCII characters "SIGNON". The printing of the user number characters can be suppressed by entering a "control Z" key after "D," on a teletype terminal. The ability to sign on is only applicable to interactive terminals.

## 1-6 $OFF

The sign-off command must be explicitly written and include the dollar ( $ ) sign. The terminal will output a message indicating who has signed off and at what time of day. Connect time (the time the user has been using the terminal) and CPU usage time are also optionally printed out. To suppress the timing message output, a mode of:

$MO,ST=ON

should be entered before the $OFF (see $MODE, section 3).

The $OFF command is only appropriate from interactive terminals. It will result in an error if entered from a control point.

## 1-7 $JOB

A $JOB card is required to precede any job stream that is to be run at a control point. The job card format is:

$JOB,JOBNAME,QUALIFIER,USERNO,PARAMETERS

where JOBNAME is the user defined name of the job (not more than 12 ASCII characters in length). QUALIFIER is a valid sign-on qualifier for this user and USERNO is the user's user number. The PARAMETERS that can be supplied are listed below:

| | |
|---|---|
| OUT= | output device for LFN's 3 and 6 |
| TIME= | maximum number of seconds for which the job will run. |
| LINES= | maximum number of lines of output the job will require |
| SIZE= | maximum number of 1024 word pages the job will require |
| PRI= | priority at which the job should run (0-15) |

The order in which these parameter can be supplied is not fixed. When a parameter is not supplied a default will be used. The defaults for TIME, LINES and SIZE may be set for an individual user when he is added to the system (operator communications command "AU"). If no default options for the particular user have been given the system default options, set at GENASYS time, will be used. The default for the output device (OUT=) is also set at GENASYS time (see Chapter G). The output device may be specified as either a physical device number (PDN), which is automatically spooled, or as a disc area. The priority may only be supplied by users who have the "priority access". This access is optionally given when the user is added to the system. The default priority is calculated from the values of TIME, LINES and SIZE supplied by the user (or the defaults if some or all of these parameters are missing).

Examples:

The following are typical job cards for control point programs for the user who signed on to an interactive terminal in section 1-5.

a) $JOB,JOB1,1234ABCD,SIGNON,LINES=2000,TIME=3

b) $JOB,JOB2,1234ABCD,SIGNON,OUT=QUAL*AREA,T=2,LINES=2500

c) $JOB,JOB3,1234ABCD,SIGNON,OUT=6,PRI=12,LINES=10000,SIZE=25

d) $JOB,JOB4,1234ABCD,SIGNON,0=6,P=15,L=10000,S=64,TIME=4

This command will produce an error if issued from an interactive terminal.

## 1-8 $EOJ

The $EOJ card is required to terminate each job stream run at a control point. This command will produce an error when issued from an interactive terminal.

# SECTION II
# PROCESSOR CALLS

## 2-1 INTRODUCTION

This section explains the control cards needed to use specific VULCAN processors. All processors may be loaded via the general processor description (Section 2-2) but there is a shortened form, for convenience, of the more commonly used processors (Sections 2-3 through 2-6).

## 2-2 GENERAL

In general a user program, or a VULCAN processor, may be loaded via:

$$PROGRAM

or $$*PROGRAM

where PROGRAM is the name of the processor or program. The double dollar sign ensures that the standard Job Control commands are not checked for a possible Job Control request. That is, the first two significant characters (non-dollar sign) of the command are not compared with the list of standard two character Job Control commands. The double dollar sign requests a load of the program name supplied. The general form of a program load is:

$$QUAL*PROG.ADSJ,PARAMETERS

where QUAL*PROG is the qualifier/name of the program to be loaded and A,D,S and J are local options. These options are set by Job Control and may be obtained in the program via a "BLU" request to "$OPTION" (see Chapter E). The parameters on the control card are not investigated by Job Control and are left for the individual processor to interpret. The user program may obtain these parameters by requesting a "backspace record" on LFN 0 (the control command input device) and follow this with a "read" request to LFN 0. This will read the preceding control card into the user program and, via the Scanner (Chapter E), the parameters may be deciphered.

## 2-3 $FORTRAN

The $FORTRAN command is a recognizable Job Control command and loads the Fortran compiler. The form of the command is:

$FO.ABC,AREANAME

where A,B and C are local options for the Fortran compiler (described in Chapter C) and the optional AREANAME is the disc area from which the source cards are to be read.

## 2-4 $VASSEM

The Vassembler command, (a Job Control command), loads the VULCAN assembler (VASSEMBLER). The form of the command is similar to the $FORTRAN command:

$VA.ABC,AREANAME

where A,B and C are local options for the Vassembler (see Chapter C) and the optional AREANAME is the disc area from which source cards are to be read.

## 2-5 $VULCANIZ

The VULCAN cataloger or linkage editor (VULCANIZER) may be called via the Job Control Vulcanizer Command. If the local option "S" (Short form) is set, the form of the command is similar to that for the $FORTRAN command:

$VU.SABC,AREANAME

where S,A,B,C are local options and AREANAME is the optional name of the disc area where the Vulcanized program is written. The "XE" (Xecute) area is used as a default.

If the Short form (local option "S") is not supplied further commands are read by the VULCANIZER from the job control input stream (logical file number zero) until a "BEGIN" card is encountered. The form of these commands is described in the Chapter C.

## 2-6 $VXECUTE

The $VXECUTE command catalogues the program (the short command form (local option "S") is automatic with a $VXECUTE command) and sets the newly cataloged program into execution. The form of the command and its argument correspond to the $VULCANIZ call:

$VX.ABC,AREANAME

The "XE" disc area is used by default if no program AREANAME is supplied.

## 2-7 SUMMARY

The above commands are incorporated in Job Control for ease of calling the common processors. All of these processors may be called via the regular job control program load requests. This form is:

$$*FORTRAN.ABC,AREANAME

$$*VASSEM.ABC,AREANAME

$$*VULCANIZ.SABC,AREANAME

$$*VXECUTE.ABC,AREANAME

The absence of a qualifier preceding the "*NAME" implies the system qualifier 0000SYST.

# SECTION III
# GENERAL COMMANDS

## 3-1 INTRODUCTION

This section describes commands within Job Control which predominantly set options or control parameters used by Job Control and subsequent processors. The copy command (Section 3-9) is the only command in this section which does more than set options or controlling parameters.

## 3-2 $MODE

The $MODE command enables different options to be enabled or disabled within Job Control. These options are all accessible by other interactive processors. Up to 10 different modes may be supplied on one mode command. The first mode request that is invalid will be indicated but all valid modes that are requested will still be set.

There are three forms of modes suppliable on a mode command:

### 3-2.1 Alphabetic

The Alphabetic mode allows arguments to be supplied to Job Control in text format. As many characters as desired may be entered for each argument but only the first two will be used to determine the request mode. The following are all the possible Alphabetic modes:

#### 3-2.1.1 Single

The SIngle mode requests one line of output for each record read. This applies to LIST, DISPLAY and COPY commands. Thus if the output is to a terminal, which has 72 characters per line, and the input is 200 characters per line only the first 72 characters will be output in the SIngle mode. When initiated the terminal has the SIngle mode by default.

#### 3-2.1.2 Double

The DOuble mode will output a maximum of two lines for each input record for LIST, DISPLAY and COPY commands. Using the example of section 3-2.1.1, in the DOuble mode two 72 character output lines would be output for each 200 character input record. Note that this still would not be sufficient to output the entire input record.

#### 3-2.1.3 Triple

The TRiple mode will output up to three lines for each input record for LIST, DISPLAY and COPY commands. With the TRiple mode set, using the example in Section 3-2.1.1, three

72 character lines will be written out. The last of these lines will output only 56 characters since this empties the input buffer.

3-2.1.4 Quadruple

The QUadruple mode will output up to four lines for each input record for LIST, DISPLAY and COPY commands. Using the example of Section 3-2.1.1, in the QUadruple mode only three of the four possible output lines are needed to write out a 200 character input buffer. Thus, in this example, the QUadruple mode would be indistinguishable from the triple mode.

3-2.1.5 Fortran

The FOrtran mode allows the user to conveniently edit Fortran source language programs. It sets a tab field (see Section 3-3) in column 7 but checks for a comment record. The comment record checking is particularly useful when character editing since it does not prevent editing across column 7. If different field Tabs are required the FOrtran mode may be set and then a $TAB request (Section 3-3) issued to set the new tab values.

3-2.1.6 Vassembler

The VAssembler mode is similar to the FOrtran mode in that tab fields are set and act as boundaries except on comment cards. The tabs set are for columns 9, 15 and 30. As for FOrtran, if different tabs are required, the VAssembler mode may be set and then the new tabs input via the $TAB command (Section 3-3).

3-2.1.7 Null

The NUll mode removes any previous VAssembler or FOrtran requests. It also removes all tab values that may have been set via other $MODE requests or via a $TAB request.

3-2.1.8 Absolute

When record numbers are supplied as input to, for example, the LIST processor or as an Editing argument the value may be ABsolute or RElative to the current position in the disc area. The position in the disc area may be moved via Logical file positioning commands (Section 4) or via positioning editing commands (Section 6). Note that the "current position" in a disc area is unchanged by LIST, DISPLAY and EDITING record number references. The ABsolute mode sets Job Control so that all record numbers input will be relative to the beginning of the disc area and independent of the "current position" in the disc area.

3-2.1.9 Relative

The RElative mode is the reverse of the ABsolute mode (See Section 3-2.1.8). Record numbers supplied to Job Control are added to the "current position" in the disc area when the

REIative mode is set. The current position in the disc area is most often controlled by the edit positioning commands (Section 6).

### 3-2.2 Numeric

The numeric mode enables options requiring numeric values to be set. The format of these modes is an alphabetic identifier followed by a numeric value all as one argument. For example, LP17 sets the number of lines per page (LP) to seventeen. This can be written in any of the following types of format:

LP17

LP=17

LPAGE=17

The first two characters must be LP and no numeric digits may appear in the argument until the value itself is to be represented. The available numeric modes are given below:

#### 3-2.2.1 Lines Per Record

This mode enables a user to request any number of lines of output for a given input buffer size. The form of the numeric mode argument is:

LP=N

where LR identifies the lines per record request and N is a numeric value. When N is one through four, the mode becomes exactly equivalent to the alphabetic modes SIngle through QUadruple (Section 3-2.1). In the event of N being greater than the number of lines required to output the input buffer, N will effectively be reduced to the number of lines actually required. On initiation VULCAN has LR=1.

#### 3-2.2.2 Lines Per Page

The lines per page option sets the number of lines per page for output to line-printers. Initially the default for VULCAN is 55 lines per page. The format of the numeric mode argument is:

LP=N

where LP identifies the lines per page request and N is the new numeric value that is required.

#### 3-2.2.3 Program Size

The program size numeric mode argument enables a user to modify the size of his "working area". The format of the argument is:

PS=N

where PS identifies the program size request and N is the number of 1024-word pages that it is required to have available when executing subsequent processors and Job Control. This command is most frequently needed to assemble large programs. The working area available to the Vassembler is fixed at Vulcanizing time. If this is not sufficient, then the working area available may be increased via the program size mode. Note that the new program size will remain in effect until changed by another user program size request. When a program is loaded the Vulcanized program size is used unless the user-set program size is larger in which case the latter is used. Thus a "PS=0" is an allowable mode argument to reset a previously entered program size numeric mode.

3-2.2.4 Buffer Size

The buffer size numeric mode allows the user to set the size of the input buffer required. This buffer is used for all reading and writing purposes. This numeric mode allows a user working with non-standard record sizes to define this buffer size. When VULCAN is initialized, the default buffer size is set to 81 characters.

3-2.3 On-Off

The On-Off type of modes set options which are either "ON" or "OFF". They all take the form:

XX=ON

or

XX=OFF

where XX defines the option that is to be enabled or disabled and the equal sign ( = ) is required to delimit the ON, OFF part from the request option.

3-2.3.1 Extended Message

The extended message facility enables a user,when he gets an abort or error code, from any of the VULCAN processors, to get an extended message explaining what the error was. The facility is enable with the ON, disabled via the OFF. On initiation all control point programs and interactive CRT terminals have this mode enabled. Teletype and teletype compatible devices have the mode disabled. The form of the mode is:

EM=ON

or

EM=OFF

When EM=OFF is in effect, the $EM command (Section 3-10) may be used to find the error code explanation.

3-2.3.2 Exit on Abort

When a program aborts, the job, optionally, may or may not terminate exit. The exit on abort mode enables or disables this option. The format of the mode argument is:

EA=ON

or

EA=OFF

When initialized control point programs have the exit on abort bit enabled whilst interactive terminals have this option disabled.

3-2.3.3 $ADD

The $ADD option allows a $ADD card to be read by the VULCAN I/O processor as a standard job control card or to be treated as a very special control card which introduces into the job stream the disc area named on the $ADD card as a replacement for the $ADD card. The usefulness of this becomes apparent when dealing with large programs. If the whole source of the program is contained in one disc area, any editing modification would take a long time due to copying the whole area in the Update Processor. By breaking the program into logically discrete disc areas, each editing is accomplished more quickly. To compile this program, it is necessary to treat these discrete areas as one and this is accomplished by a compilation job stream that is mainly composed of $ADD cards. In order to do the compilation, the $ADD mode must be enabled, however, to modify (edit) the job stream, it must be possible to access the $ADD cards themselves hence the $ADD mode must be disabled.

Thus,

$A=ON

enables the $ADD mode and no $ADD cards are ever passed to a program (the disc area that they are "adding" is passed instead) and,

$A=OFF

disables the $ADD mode and $ADD cards are passed to a program as regular control or data cards. Upon initiation, both control points and interactive terminals have the $A=OFF mode set. It is usually desirable to terminate the $A=ON mode as soon as possible after enabling it.

3-2.3.4 Suppress Timing Messages

On completion of the job ($EOJ for control points or $OFF for interactive terminals) timing information is output to the diagnostic output (LFN 3) device. To prevent these lines of output the suppress timing message option must be enabled. The default case for both interactive terminal and control points is that the suppress timing message option is disabled, which means the messages are output. The format of the mode argument is:

ST=ON

which prevents the timing messages being output, or,

ST=OFF

which allows the timing messages to be written to LFN 3.

The three types of modes described above may be intermixed in any way on the $MODE card. In the case of mutually contradicting modes, the last valid mode entered will override the previous ones.

Example:

To set the VAssembler mode, DOuble lines mode, the RElative mode, the lines per page option to 48, get extended messages output and ensure that the timing messages will be output any of the following commands will produce the desired results:

$MO,VA,DO,RE,LP=48,EM=ON,$ADD=OFF,ST=OFF

$MO,$ADD=OFF,DO,VA,ST=OFF,LP=40,EM=ON,RE,LP=48

$MO,LP=48,ST=OFF,EM=OFF,$ADD=OFF,DO,VA,RE,EM=ON

## 3-3 $TABS

The $TAB command enables the user to set up to ten tab field values for input lines from an interactive terminal. The command may be issued from control point programs though in this case its usefulness will be restricted to character editing. The tab values must be supplied in ascending numerical order and the form of the command is:

$TA,N1,N2,N3,....

where N1, N2, N3 etc., are ascending positive integers. If a processor mode is currently in effect (Section 3-2.1), the standard processor tab values will be overwritten but the other features associated with the processor mode (such as ignoring tab fields for comment cards) will remain intact. A blank argument field will remove all tab values.

## 3-4 $JSTREAM

The $JSTREAM command is only available to interactive terminals. Its functions are achievable, via the $ADD mode option, for control points. It is one of the more powerful Job Control commands available to an interactive user. It allows a string of Job Control statements, contained on a disc area, to be executed via this single statement. Any valid Job Control statement may be contained on the disc area and this particularly includes processor ($FORTRAN, etc.) and Vulcanizer calls. The $JSTREAM command reassigns logical file number (LFN) zero to the disc area. Error messages are still output to LFN 3, which is the terminal by default. Note that executing Job Control statements from a disc area does not imply or require a $JOB card. If this is supplied, an error will be output indicating an invalid interactive terminal command. This command should be carefully distinguished from the $IJOB command (Section 3-5) which is available for a totally different purpose.

Any repetitive string of Job Control statements that are frequently used may be written to a disc area and executed via a $JSTREAM command. This is particularly appropriate for compilation and cataloging of programs. The format of the command is:

$JS,AREANAME,N

where AREANAME is the disc area to which control is to be transferred and N is the record number within the disc area at which the first control command will be read. Both arguments are optional: the default for N is unity and when AREANAME is not supplied, control is returned to the interactive terminal. Thus, a string of Job Control commands on a disc area, which are to be executed via a $JSTREAM command, should logically be terminated by a $JS command with no arguments. This is required if control is to be returned at any point other than at the end of the area. Job Control commands which are end of file marks are ignored and, upon detecting an EOT, Job Control will return control to the interactive terminal.

## 3-5 $IJOB

The $IJOB allows regular control point jobs to be inserted into the control point queue from an interactive terminal. The format of the command is:

$IJ,AREANAME

where the disc area AREANAME must begin with a $JOB card and be terminated by a $EOJ card. This command should be used cautiously since execution sequences normally required by an interactive user may best be accomplished via the $JSTREAM command. (Section 3-4) The main purpose of the $IJOB command is to off load a job and achieve parallel processing.

After entering the job into the control point queue the terminal awaits further input commands from the user. When using this command, care must be exercised with respect to disc areas opened to the terminal user. If these areas have been written to by the terminal user and are required by the control point program or if they have been read by the terminal user and the control point program wishes to write on the area, then the control point program is unable to execute. Similarly if the output from the control point program is to be written to a specified disc area and the terminal user looks at this area before any output has been written, the control point output will be lost. (The control point program cannot write on a disc area that is being read by another program (unless it is a "random" area - unblocked by definition) See Chapter A.)

## 3-6 $LVALUES

The list values command enables the user to define output parameters associated with his terminal that are applicable to "LIST" commands. This command is only applicable to interactive users.

There are three forms of the command:

a) $LV,N1-N2

b) $LV,N3,N4

c) $LV,FIRST=N5,LINES=N6,CHARS=N7

For (a) this form of the command enables the user to define the first line to be written upon (only applicable to a CRT interactive terminal) and the number of lines per page for his terminal. Thus, for the above case N1 would be the first line written upon and the number of lines per page written to the device would be (N2-N1+1).

The second command form, (b), is similar to the first: N3 is the first line written upon and N4 is the number of lines per page written to the terminal.

The third command form, (c), allows the first line to be written to be defined (FIRST=), the number of lines per page to be set (LINES=) and the size of the characters per line to be set (CHARS=). The argument form may be abbreviated to: F=N5, L=N6, C=N7.

## 3-7 $DVALUES

The display values command has an identical form to the $LVALUES command. The single difference being that only "DISPLAY" commands are affected by the display parameter values supplied.

## 3-8 $ASSIGN

The assign command is a means of equivalencing user output numbers - logical file numbers (LFN) to physical device numbers (PDN). If the PDN is a resourceable device, the assignment is achieved through the $RSOURCE command (Section 10). All devices in a system have a unique number and, with device independence, it is possible, with a $ASSIGN card to output to two different physical devices on two executions of the same program. Discs are not regarded as physical devices. Instead, the disc areas within the disc, which are identified by an eight character qualifier and an eight character name, are regarded as physical devices. Thus, PDN's may be an integer in the range 1 through 255 or a disc area name. The $ASSIGN command has six different forms of assignments. Multiple assignments may be entered on an assign card and any mixture of the assignment forms is allowed.

### 3-8.1 LFN to PDN

The direct assignment of a logical file number to a physical device number takes the form:

$AS,N=PDN

or

$AS,N=:PDN

where N is the logical file number (1 through 255) and PDN is the physical device number (1 through 255). The equal sign ( = ) is required and the optional colon ( : ) indicates a physical device number.

### 3-8.2 LFN to Cassette Tape

When the direct assignment is to a model 2200 teletype (TI 733) terminal the physical device number refers to the keyboard and to two magnetic cassette tapes. The direct assignment (Section 3-8.1) makes the assignment to the keyboard. The assignments for the cassette tapes, assuming the terminal to have a physical device number of 25, are:

$AS,N=25T1 or $AS,N=:25T1

or

$AS,N=25T2 or $AS,N=:25T2

where T1 and T2 refer to the left hand and right hand magnetic cassette tapes respectively and N is the logical file number (1 through 255).

3-8.3 LFN to Disc Area

The direct assignment of a logical file number to a disc area takes the form:

$AS,N=AREANAME

where N is the logical file number (1 through 255) and AREANAME is a disc area name: QUALIFIER*AREANAME. The equal sign ( = ) is required.

3-8.4 LFN to LFN Indirect (Case 1)

It is frequently desirable to assign a LFN to a physical device which is known to have been assigned to another LFN. An example of this is when both the Fortran input and the job control command stream are on the same disc area (as often happens with control point programs). Job Control reads from LFN 0, Fortran reads from LFN 7, yet both must pick up the next record from the area when they request a read.

An indirect assignment allows a LFN to be assigned to a physical device via another LFN. The form of the command is:

$AS,N=*LFN1

where LFN1 is an existing logical file number and N is the logical file number to be assigned. The asterisk ( * ) indicates that LFN1 is a logical file number rather than a physical device number. For case one of the indirect assign (the asterisk case) N follows LFN1. That is, if LFN1 is re-assigned or de-assigned then N will "follow" LFN1 by being similarly re-assigned or de-assigned.

3-8.5 LFN to LFN Indirect (Case 2)

The second type of indirect assignment is similar to case one. The format of the command is:

$AS,N=%LFN1

where N is the logical file number to be assigned and LFN1 is an already existing logical file number. The percent ( % ) sign indicates the second type of indirect assignment. For this case, when LFN1 is re-assigned or de-assigned, N does not "follow" LFN1 and remains assigned to the physical device or disc area to which LFN1 was originally assigned.

3-8.6 LFN to Spooled Device

The assignment of a LFN to a spooled file for a device which may, optionally, be written to directly or be written to via spooling takes the following form:

$AS,N=@PDN

where N is the logical file number to be assigned and the @ sign is required to specify spooled output to physical device PDN. PDN must be a teletype or CRT terminal.

## 3-9 $COPY

The $COPY command enables data to be copied from one"device" to another "device". When the output is to a device that requires a carriage control character (e.g., lineprinters and terminals) the copy processor supplies a redundant blank as the first character to output. The general format of the command is:

$CO,IN,OUT,PARAMETERS

where IN is the input device and OUT is the output device. PARAMETERS are several, optional, parameters that describe the type of copy required. The ordering of the parameters is unimportant. The input, output devices (IN,OUT) may be disc area names, physical device numbers or logical file numbers. Physical device numbers are distinguished from logical device numbers by being preceded by a colon ( : ). When physical device number or disc area names are supplied these devices (either for input or output) will be automatically rewound since the copy processor has to assign a temporary logical file number to the device. Thus, the rewind parameter (see below) is only of functional use when used with logical file numbers as input or output devices.

The parameters available with the copy command are listed below:

ALL: The ALL parameter requests a copy from the current position for both input and output devices to the end of tape (EOT) of the input device.

FILE: The FILE parameter requests a copy from the current position for input and output devices to the next file mark on the input device.

RECORD: The RECORD parameter requests a copy from the current position for input and output devices for the next record on the input device.

N: where N is an integer number. This number specifies the number of records (if the RECORD parameter is set) or files (if the FILE parameter is set) to be copied. If the ALL parameter is used, N, if supplied, is redundant.

REWIND: The REWIND parameter requests both input and output devices be rewound before the copying begins. Note that the REWIND parameter only affects input or output devices which are logical file numbers since physical device numbers and disc area names are automatically rewound.

LIST: The LIST parameter prefixes each record copied with its record number. The LIST parameter works independently of the specified output device.

Examples:

1) Copy a disc area (QUAL*NAME) to the line printer (PDN 6) and copy the first two files and sequence number the records:

$CO,QUAL*NAME,:6,LIST,FILE,2

2) Copy LFN 19 to the disc area (QUAL*NAME), rewind the input device and copy through the EOT:

$CO,19,QUAL*NAME,ALL,REWIND

3) Copy 27 records from LFN 12 to the list output (LFN 6) numbering the records at the same time:

$CO,12,6,27,RECORD,LIST

The PARAMETERS may be abbreviated to a uniquely identifying character set (that is: A,F,REC,N,REW,L). The default parameters used by the copy processor are: unity (for N) and FILES. That is by default the copy processor will copy one file from the current position of the input device to the current position of the output device.

## 3-10 $EMESSAGE

The extended message command is used to obtain a description of a Job Control error. If the EM=ON mode is not set, an error message produces the short error message (an error number). If this is not recognized, the $EMESSAGE command should be used to get more information. The format of the command is:

$EM,N

where N is the error code number. The extended message is a record (of displacement N) in the system disc area *VULCMESS.

# SECTION IV
# LOGICAL FILE POSITIONING

## 4-1 INTRODUCTION

This section details the basic logical file positioning commands that are available in Job Control. The commands are functional only on logical file numbers (and some physical device numbers where noted). Disc area names cannot be used directly with these commands since a LFN would have to be assigned to the disc area and then de-assigned after the appropriate manipulation. Disc areas can, of course, be manipulated by assigning a logical file number to the area and then manipulating the LFN.

## 4-2 $WEOF

The write end of file command enables one or more file marks to be written on a designated logical file number. The form of the command is:

$WE,N,M

where N is the LFN and the optional parameter M is the number of end of files to write on the device. The default for M is unity.

## 4-3 $WIND

The wind command positions the designated device at its end of tape (EOT). It is only of use with a LFN that is assigned to a disc area. The form of the command is:

$WI,N

where N is the logical file number.

## 4-4 $RWIND

The rewind command positions the designated device at its beginning of tape (BOT). A PDN or a LFN can be specified as the device. PDN's are distinguished from LFN's by a colon ( : ) preceding the number:

$RW,N

or

$RW,:N

4-5 $AB

The advance file, backspace record command sets the current record position ready to write over or read in the next file mark. The form of the command is:

$AB,N

where N is a LFN assigned to a disc area or a magnetic tape.

4-6 $RF

The rewind file command sets the current record position as though it had just read in the previous file mark. The command form is:

$RF,N

where N is a LFN assigned to a disc area or a magnetic tape.

4-7 $AF

The advance file command skips records on the designated LFN until an EOF is encountered. The LFN is left positioned immediately after the EOF. The form of the command is:

$AF,N,M

where N is a LFN assigned to a disc area or magnetic tape and M is the optional number of file marks to advance. M is unity by default.

4-8 $AR

The advance record command skips the designated number of records on the requested LFN. End of files are treated as a regular record. The command form is:

$AR,N,M

where N is a LFN assigned to a disc area or magnetic tape and M is the optional number of records to advance. M is unity by default.

4-9 $BF

The backspace file command is the reverse of the $AF command. The device is positioned, upon completion, such that the next record read in will be the $M^{th}$ end of file backed over. The command form is:

$BF,N,M

where N,M are defined for the $AF command.

4-10 $BR

The backspace record command is the reverse of the $AR commands. End of file records are treated as regular records. The command form is:

$BR,N,M

where N,M are defined for the $AR command.

4-11 $FREE

The $FREE command has two forms. The first is used to close a specified logical file. This will potentially free the physical device to which the LFN was assigned or, if the physical device was a spooled device, allow the output to be spooled out. This form of the command is frequently used before spooling out a disc area to which LFN 6 (list output) has been assigned. If spooling is attempted without FREEing the disc area nothing will be output if the terminal user still has the area open for writing since the spooling program cannot open the disc area. The form of the command is:

$FR,N1,N2,....

where N1,N2, etc. are logical file numbers to be released.

The second form of the $FREE command has the form:

$FR,WORK

where the parameter WORK indicates that all the work areas associated with the users terminal or control point are to be de-assigned and, hence, eliminated. This form of the command is rarely used since already existing work areas are rare unless created by the current user in which case a regular eliminate (Section 9) will remove them.

# SECTION V
# EDITING

## 5-1 INTRODUCTION

Record editing facilities are provided within Job Control. The editing facilities apply only to disc areas and, in order to edit a particular disc area, the area has to be put into the editing mode via a $EDIT command. It is then possible to add records ($INSERT), change records ($CHANGE) and remove records ($DELETE) associated with the named disc area. Note that all these requests are pre-stored and none are implemented until a $UPDATE command is given. This means that if, in an editing sequence, it is realized a particular record was changed incorrectly, then it is possible to re-enter the record. Since the record numbers are the same, the editing processor will always use the last entered record to perform the requested function. The $EDIT command removes all previously entered editing commands and so may be used to "re-start" an editing procedure. Record numbers to the editing processor may be entered in any sequence.

## 5-2 $EDIT

The $EDIT command defines the disc area which is to be edited. The form of the command is:

$ED,AREANAME,PARAMETERS

where AREANAME is the disc area to be edited, which must be write accessible to the user, and PARAMETERS are optional parameters which can be "ABsolute", "RElative" or "COntinue". ABsolute and RElative indicate whether the record numbers subsequently to be supplied should be treated as absolute or relative to the current record position within the disc area. (See Section 3: $MODE definition) "COntinue" saves the edit commands that so far have been given as though they have already been entered for this disc area. The prime use of the continue mode is to recover editing after a system failure or to "transfer" the editing commands inadvertently given to the wrong disc area before the error was discovered. Note that all editing commands are stored and the source is never modified until a $UPDATE request is given.

The disc area on which the editing is to be performed is also made the LIST area (Section 8).

## 5-3 $INSERT

The $INSERT command enables records to be added to the area in the edit mode after the record number given on the command. The forms of the command are:

a) $IN,N

b) $IN,N,AREANAME

c) $IN,N,AREANAME,N1-N2,N3-N4

d) $IN,N,AREANAME,AF,BF,CF

For (a), N is the record after which the insertion is to occur. All subsequent records input to Job Control will be inserted into the disc area until a three dollars sign record is encountered ($$$). This is the only way of terminating a type (a) request ($IN,N). If the input to Job Control is currently from a disc area then the records to be inserted are also taken from this disc area.

For (b), the disc area AREANAME is entirely inserted into the disc area in the edit mode after record number N.

For (c), records numbers N1 through N2 and N3 through N4 are extracted from disc area AREANAME and inserted after record number N in the area in the edit mode. Note that while N may be either a relative or an absolute record number (depending on the mode value set) N1,N2,N3 and N4 are the absolute record numbers within the area AREANAME.

For (d), when area AREANAME consists of multiple files it is possible to insert only the file required by "manipulating" the area. Starting from an initially rewound state, files can be skipped (AF), backspaced (BF) and copied (CF). The files as picked out are inserted after record N in the edit area.

When inserting,it is frequently desirable to concatenate records from different sources. This may be done by multiple inserts for the same record number, provided no other editing commands (changes, deletes) have been input in between. The exception to this is the two adjacent command sequence:

```
$IN,N
$$$
```

which is an insert of nothing and is assumed to override the previous $IN,N command. There is no limit to the number of insert commands that can be concatenated.

All other similar editing commands which specify the same record number on more than one command will override all but the last command entered.

## 5-4 $CHANGE

The change command specifies the record(s) within the edit area that are to be replaced. The form of the command is:

$CH,N1,N2,N3-N4

where N1, N2 etc., are record numbers within the edit area. The next N records input to Job Control are used to replace the records given on the $CHANGE command where N is the summation of all the records on the $CHANGE command card. If it is necessary to terminate the input of the change records before N have been entered, a three dollar record may be input ($$$). This means that no record can be changed to $$$.

## 5-5 $DELETE

The delete command enables specified records or files to be deleted from the edit area. The form of the command is:

$DE,N1,N2-N3...

or

$DE,F2-3

where N1, N2, N3 are record numbers and may be relative or absolute depending on the mode. F indicates files which are absolute within the area.

## 5-6 $UPDATE

All editing requests are pre-stored and the original source is never modified until an update request is given. The form of this command is:

$UP,AREANAME

where the optional parameter AREANAME is a disc area where the updated source is to be stored. The area must exist and be accessible to the updating user. The default case has the new version of the source overwrite the old thus using the edit area as the update area.

## 5-7 EXAMPLE

Consider a disc area (*DOG) which consists of the following records:

```
AA
BBB
CCCC
DDDDD
EEEEEE
EOF      (end of file)
```

Then the following editing sequence:

```
$ED,*DOG              set edit area
$IN,0                 add 2 records at beginning
ZZZ
YYY
$$$
$IN,0,*DOG,2-5        concatenate 4 more records to the 2 already added
$DE,1-4               remove the first four records
$CH,4,1,3,6           override the delete on 3 of the records, remove EOF
LINE4
LINE1
$$$                   only want to change first two records specified
$UP                   all editing done, modify the area
```

produces the following in area *DOG

```
ZZZ
YYY
BBB
CCCC
DDDDD
EEEEEE
LINE1
LINE4
EEEEEE
EOF      (end of file)
```

The following editing sequence will restore the original area contents:

| | |
|---|---|
| $ED,*DOG | set edit area |
| $DE,1 | remove first record |
| $CH,2 | change second record to desired value |
| AA | |
| $DE,7-9 | remove the unwanted 3 records before EOF |
| $UP | restore area to original contents. |

# SECTION VI
# POSITION EDITING COMMANDS

## 6-1 INTRODUCTION

It is frequently very desirable to be able to scan a disc area looking for a given sequence of text and this is usually required for editing purposes. Job Control has six edit area positioning commands which are described in this section. These commands will only operate on the disc area in the Edit mode and they "move" the current position associated with this area. Thus, using the edit positioning commands in conjunction with the relative editing mode it is possible to edit a disc area without an updated hard-copy listing of the area's contents.

## 6-2 $AEDIT

The Advance Edit area command enables the relative position within the editing area to be moved. There are three forms of the command:

a) $AE,N

b) $AE,N1-N2,TEXT

c) $AE,TEXT

where N (default unity) is the number of records to advance the edit area position. TEXT is a group of ASCII characters for which a searching match is required. N1-N2 is a group of columns within which the search is restricted for each record.

When a match (on a TEXT search) is found the edit area is positioned on the record containing the matching TEXT. Thus, to edit or list this record a

$E,0

or

$LI,0

command is required (the relative mode having previously been set).

## 6-3 $BEDIT

The Backspace Edit area command is identical to the $AE command except the search or positioning is backward in the edit area. The same forms of the $BE command exist as for the $AE command.

6-4 $PEDIT

The Position Edit Area command is identical to the $AE command except that the edit area is rewound before the searching or positioning begins. The same forms of the $PE command exist as for the $AE command.

6-5 $AN

The Advance No match in Edit area command exist in all the forms of the $AE command. Termination for a TEXT search occurs when the given text is not found within the record. Non-TEXT searching produces identical results as for the $AE case.

6-6 $BN

The Backspace No match in Edit area command functions identically to the $AN command but searching and positioning is in the backward direction in the edit area.

6-7 $PN

The Position No match in Edit area command functions identically to the $AN command except that the edit area is rewound before positioning or text searching.

# SECTION VII
# CHARACTER EDITING AND MANIPULATION COMMANDS

## 7-1 INTRODUCTION

Character editing commands differ from all other Job Control commands in format in that they are only one character in length (there is no expanded form of these commands). A preceding dollar sign ( $ ) is required for control point commands and optional for interactive terminal commands. Character editing and manipulation requires a disc area to be in the editing mode. A $E (set edit record) command is required before any character editing or manipulation can be achieved. When a record is entered into the character editing mode, it is copied from the edit area to a work area. All subsequent character commands apply to this record. When the next record is entered into the character editing mode (or a $UPDATE command is issued) the old record is entered as a regular $CHANGE command for the editing area if the record has been modified. This can be confusing, for example:

```
 .
 .
 .
$ED,AREA
$E,14
$C,XXX,YYY
$CH,14
WRITE A NEW LINE
$E,15
 .
 .
 .
```

In the above case on encountering the $E,15 command the previous $E,14 request is entered as a $CHANGE command and thus overrides the $CH,14 command. To avoid this, a $E,14 command should be entered immediately before the $CH,14 command.

## 7-2 $E

The Edit command specifies which record is to be entered into the character editing mode. The form of the command is:

$E,N

where N is the line number of the record to be character edited. N will be either an absolute or relative line number, relative to the current record position in the area, depending on which of the two modes is currently in operation (Section 5, $EDIT command). This command also rewinds the column pointer so subsequent editing, unless manipulation commands are used, is for the whole record.

The $E command is unique in that a subsequent character editing command may be given within the same "command". Examples would be:

$E,N,L

where record N is put into the character editing mode and listed (L). Note that although L (list) is a recognized command it must not be preceded by a dollar sign when attached to a $E command. A second example would be:

$E,N,C,AAA,BBB

where, after putting record N into the character editing mode, the characters AAA within record N are searched for and replaced by BBB.

Character editing is undertaken command by command, as requested. The new, modified, record is held in a disc work area and entered as a regular $CHANGE command only upon receipt of the next $E command. If the edit record has not been modified, it will not be entered as a $CHANGE command.

7-3 $A

The Advance column pointer command enables the current displacement within the edit record to be moved. It has two forms:

$A,N
$A,TEXT

in the first case the column pointer is advanced N characters while in the second case the characters "TEXT" are searched for in the forward direction and the pointer is set pointing at the last character of the located TEXT. To subsequently modify this character a delete or change will have to reference character zero.

7-4 $B

The Backspace column pointer command is identical to the $A command except that the column pointer is moved backwards. The pointer position after a TEXT search corresponds to the $A case.

## 7-5 $P

The Position column pointer command is identical to the $A command except the column pointer is set to zero before searching for TEXT or advancing the specified number of columns.

## 7-6 $L

The List edit record command enables the user to check that his specified editing is correct.

The form of the command is:

$L,N1-N2

where N1 is the first column to list and N2-N1+1 is the number of characters to output. The columns specified are always relative to the current column pointer position. If N1-N2 is not specified the whole record is output from the current column position to the end of the record. If only N1 is specified, the output is from N1 plus the current column position to the end of the record.

For teletypes the output is direct to the terminal but for CRT terminals the output is written to the "OA" work area and a $DISPLAY,OA command is simulated. The output may be displayed again at a later time by entering this command.

## 7-7 $C

The change character command is used to change text within the record. It has three forms each of which has two arguments. Each command may contain several pairs of arguments and the three different forms may be intermixed. The three different forms are:

i) $C,TEXT1,TEXT2

ii) $C,N1,TEXT2

iii) $C,N1-N2,TEXT2

for case (i) the record is searched for a set of characters TEXT1. These are replaced by the set TEXT2. The search is forward from the current column pointer position. If TEXT1 straddles a tab field boundary no change will take place and the user will be informed.

In the second case, a specified column (relative to the current column pointer) is replaced by TEXT2. In the third case a group of columns, relative to the current column pointer, and which do not straddle a tab field boundary are replaced by TEXT2.

TEXT2, which may be a different number of characters than the characters it replaces, will be truncated if it attempts to cross a tab field. In this case, the user will not be notified. Blanks are inserted or characters dropped at the next highest tab field boundary if the replaced and TEXT2 characters are not the same size. Only one tab field is affected by one pair of $C arguments.

7-8 $D

The character Delete command is used to remove characters from the edit record. It has three forms and, as for $C, more than one form may be on a single command. The forms are:

i) $D,TEXT

ii) $D,N1

iii) $D,N2-N3

When the argument is TEXT a search is made in the forward direction from the current column position pointer. When column numbers are specified, the numbers are relative to the current column pointer. The columns or TEXT to be deleted may not cross a tab field boundary and each deletion only affects one tab field. When characters are deleted the remaining characters move down in the tab field and blanks are inserted at the next tab field position.

7-9 $I

The Insert character command is used to add characters to the edit record. It has six different forms. The characters are inserted after the requested TEXT or column and characters at the end of the tab field are dropped off the end of the tab field. Only one insert per input command is allowed and each insert affects only one tab field. The six forms are:

i) $I,N1,TEXT

where the group of characters, TEXT, are inserted after column number N1 plus the relative column pointer. If the number of characters in TEXT is greater than the space remaining in the tab field beyond the insertion column, TEXT will be truncated.

ii) $I,TEXT1,TEXT2

The same conditions apply as in case (i) but a forward search in the edit record from the current column position is made for the character string TEXT1. TEXT2 is inserted immediately after the last character of the TEXT1 string.

iii) $I,N1,AREANAME,N2,N3-N4

where N1 is the relative column position after which the insertion will take place. AREANAME is a disc area from which the character string to be inserted is obtained. N2 is the absolute record number in the disc area AREANAME and N3-N4 the columns within the record N2 that are to be inserted in the edit record. If a single character is to be inserted from the record N2 of area name AREANAME, N3 will specify it.

iv) $I,TEXT1,AREANAME,N2,N3-N4

This command is similar to case (iii) except that the character string is inserted after the character string TEXT1 in the edit record.

v) $I,N1,AREANAME,N2,WN3

This command form is the same as case (iii) except that instead of inserting columns N3-N4 from the record N2 in the disc area AREANAME, the N3 word is inserted. "W" indicates "WORD". N3 cannot be zero since word zero is not defined and, in this context, a word is defined as a contiguous string of characters not containing delimiters (blank or commas). Blanks and commas contained within double quote marks ( " ) are not regarded as delimiters. The "W" prefix must be present to distinguish between column and word insertion.

vi) $I,TEXT1,AREANAME,N2,WN3

This command form is identical to case (v) except that the insertion occurs after the character string TEXT1 in the edit record. TEXT1 is searched for, in the forward direction, from the current column position.

## 7-10 EXAMPLES

Consider the following record to be record 3 in the disc area *DOG:

ABCDEFGHIJKLMN

then the following commands will modify and restore the record:

| | |
|---|---|
| $ED,*DOG | set edit area |
| $TAB,4,8,12 | set tab field valves |
| $E,3,L | set edit record and list it |
| $I,2,XYZ | record now: ABXDEFGHIJKLMN |
| $D,DEFGH | error: deletion across a tab field |
| $D,DEFG | record now: ABXƀƀƀƀHIJKLMN |
| $D,H | record now: ABXƀƀƀƀIJKƀLMN |

| | |
|---|---|
| $A,9 | move column pointer into third tab field |
| $C,"ƀ",T | record now: ABXƀƀƀƀIJKTLMN |
| $I,3,*DOG,3,5-6 | original record read to get required characters for this command record now: ABXƀƀƀƀIJKTLEFMN |
| $P,0 | rewind current column pointer to restore record |
| $C,"ƀƀƀƀ",DEFG,3,C | record now: ABCDEFGIJKTLEFMN |
| $C,LEFM,LM | record now: ABCDEFGIJKTLMN |
| $D,11 | record now: ABCDEFGIJKƀLMN |
| $I,G,H | record now: ABCDEFGHIJKLMN |

# SECTION VIII
# DISPLAY COMMANDS

## 8-1 INTRODUCTION

Job Control display commands allow a user to conveniently look at parts of a specified disc area. Two disc areas may be in "display" modes at the same time. The LIST processor displays records in a disc area and prefixes each record with its record number. The DISPLAY processor displays the records as they appear in the disc area. By using the $LVALVES and $DVALVES commands (Section 3) it is possible, with a CRT terminal, to split the screen and view two disc areas simultaneously (one in the DISPLAY mode, the other in the LIST mode). The LIST and DISPLAY commands whilst usable by control point programs, are oriented to interactive terminal users.

## 8-2 $LIST

The $LIST command enables the user to copy to his terminal specified records of a disc area. The list command has the following forms:

a) $LI,AREANAME

b) $LI,%N

c) $LI,AREANAME,N1-N2

d) $LI,%N,N1-N2

e) $LI,N1-N2

f) $LI,AREANAME,FN1-N2

g) $LI,%N,FN1-N2

h) $LI,FN1-N2

i) $LI,N1-ALL

j) $LI,N1-EOT

k) $LI,N1-EOF

where AREANAME is a disc area name and %N indicates a logical file number, N being the LFN value.

The first argument on the $LIST command is optionally the name or the LFN of the area to be listed. If a LFN is specified, it must be assigned to a disc area and a percent sign ( % ) must preceed the LFN value to distinguish it from other LIST command formats. If a disc area name or LFN is not supplied, there must be a disc area already in the list mode. This occurs if either there is an EDIT area defined (the same area is automatically put into the LIST mode) or if a previous list request has been made which supplied the list area. If no arguments are supplied after the area name, the whole area is listed except on a CRT where one page is listed. A range of records can be supplied (N1-N2), a range of files may be requested (FN1-N2) or a group of records from a specified one to the end of the area (N1-ALL or N1-EOT) or the end of the file (N1-EOF) may be listed.

## 8-3 $DISPLAY

The $DISPLAY command is identical in form to the $LIST command. The display area name can only be set via a display command. The only difference between an area that is displayed rather than listed is that no record numbers are output in the display case. A typical form of the display command is:

$DI,AREANAME,N1-N-2

## 8-4 $LUP

The List Up command will output a "page" of records relative to the current position in the LIST area. The default page size is one line for a teletype and 23 lines for a CRT. These may be modified via the $LVALUES command. The command form is:

$LU,N1,N2

where N1 is the number of pages to skip and N2 is the number of pages to list. N2 is always set to unity for CRT's and both N1 and N2 are optional and by default unity.

## 8-5 $DUP

The Display Up command is identical to the $LUP command in format and definition but its operation applies to the area in the display mode.

## 8-6 $LDOWN

The command format and definition is the same as $LUP. For $LD, N1 refers to the number of pages to backspace past.

8-7 $DDOWN

The Display Down command is identical to the $LDOWN command in format and definition but applies to the display area rather than the list area.

8-8 ƀƀƀƀ

An input line that is blank is a "repeated request" and will repeat the last "repeatable" command that was input. All forms of the $LIST and $DISPLAY commands are repeater commands. Unless a $LD or $DD was the last repeater command input,a blank line input will cause a $LU or $DU (with the default arguments) to be processed. $LD and $DD as last repeater commands will cause a $LD or $DD to be processed for a blank line input.

When a $LD or $DD command (or a blank command which is equivalent) is entered and the beginning of the disc area is encountered, the repeater command is set to a $LU or $DU command. Similarly when a $LU or $DU encounters an end of disc area (EOT) the repeater mode is set to $LD or $DD. This enables the user to scan up and down a disc area without entering any specific commands after the initial command that sets the $LIST or $DISPLAY area. Note that the repeater applies only to the last viewing command entered. Paging through a disc area with the repeater command is, therefore, restricted to either the LIST or DISPLAY modes without a specific command being entered to force a change of the repeater mode.

# SECTION IX
# DISC AREA MAINTAINANCE COMMANDS

## 9-1 INTRODUCTION

The VULCAN disc structure is described in Chapter A. This section details the Job Control commands that are available to manipulate existing disc areas and create new ones. For these commands Job Control acts as an interface between the user and the disc area services described in Chapter D.

## 9-2 $GENERATE

The generate command is used to create disc areas. There are three types of areas which the $GENERATE command will create. These are data areas, libraries and monitor common blocks. Program areas cannot be created by this command, the Vulcanizer being the only processor capable of generating program areas.

The general form of the generate command is:

$GE,AREANAME,PARAMETERS

where AREANAME is the disc area name to be created and PARAMETERS are optional parameters, specifiable in any order, which define characteristics of the disc area.

For data areas, AREANAME is a regular 8 character qualifier, 8 character name. For library and monitor common block disc areas, the qualifier is 8 characters but the name is limited to 6 characters. This is because, for libraries, two disc areas are constructed from the name supplied- a directory and an element area. These are distinguished by making the first two characters of the name "Dƀ" and "Eƀ" and the six characters supplied become the third through eighth characters of the name. Similarly, for monitor common blocks, the size character name given is preceded by "Mƀ".

A request to generate a library or monitor common block is identified via the PARAMETERS on the $GENERATE card. These parameter groups are discussed below. The order of the PARAMETERS is unimportant.

### 9-2.1 Area Type

The area type parameter is only entered to generate library areas or monitor common blocks. If a regular disc area is required this parameter is absent. Only one of these two parameters may be entered on a generate command:

| | |
|---|---|
| MONITOR or MO | generates a monitor common block. |
| LIBRARY or LI | generates a library area. |

Note that when either of these parameters are supplied the name part of the AREANAME is restricted to six characters in length.

The $GENERATE command performs initialization on the areas created as library or monitor common blocks. A library area is initialized to "empty" by indicating the directory area is a null area. Appropriate assignments are made so that, for library editing commands the newly generated area becomes the default library area. Monitor common blocks are initialized to zero so all values can initially be predicted.

### 9-2.2 Common Block Types

If the monitor area type parameter has been entered it is possible to specify whether the common block should be resident or non-resident:

| | |
|---|---|
| RESIDENT or RE | common block will be made resident at system boot time. |
| NRESIDENT or NR | common block will be in memory only when being used by a program. |

These parameters are only meaningful when a MONITOR area type has been specified. The default is NRESIDENT.

### 9-2.3 Disc Area Structure

The disc area structure parameter defines the structure of the created area. There are three possibilities:

| | |
|---|---|
| B | Blocked |
| U | Unblocked |
| R | Random |

The parameter must be specified as one character in size. The disc area structure parameter is only appropriate to regular disc areas (libraries areas, both Element and Directory, are blocked and monitor common areas are unblocked).

Blocked and unblocked areas limit reference across programs. If program A has an area X opened (blocked or unblocked) then program B will be unable to open and write on area X. Program B may, however, open and read area X providing program A has not written to X. If B succeeds in opening area X then A, which has always been able to read X, will be unable to write on X until B has closed X. With random areas (which by definition are unblocked), these restrictions do not apply.

### 9-2.4 Directory Types

In real time situations or for programs that are continually loading it may be desirable to have a copy of the directory of a disc area in core. This saves the disc access required to locate the position of the area on the disc. The parameters specifying directory type are:

| | |
|---|---|
| CD | core directory |
| DD | disc directory |

If neither parameter is specified then DD is used as the default. Note that CD does not imply that the area directory is not on the disc, only that a copy of the disc directory for this area is held in memory at all times.

### 9-2.5 Access Parameters

Every disc area has protection associated with it. An area may be available to everyone (public) or only to people with the same account number (account) or limited to an individual user (owner). There are four protection parts: reading, writing, executing and deleting. The Account and Public protection levels are mutually exclusive. For example, an area cannot be public read and account write. The following ten parameters specify all the accessing possibilities:

| | |
|---|---|
| PR | public read |
| PW | public write |
| PX | public execute |
| PD | public delete |
| AR | account read |
| AW | account write |
| AX | account execute |
| AD | account delete |
| OW | owner write |
| OD | owner delete |

The owner of an area is always able to read (and execute) his own area. The $GENERATE command always sets the OW and OD access bits. Any account or public access required must be specified via the above parameters.

## 9-2.6 Disc Area Definition Parameters

There are nine parameters which may be specified to further define the disc area. The use of these parameters is described in Chapter A. The form of the parameter is a text identification (minimum of one character) followed by a numeric value with no delimiters interposed.

### 9-2.6.1 Access Level

The access level specifiable is less than or equal to the access level of the user generating the area. The form of the parameter is identified via a leading "A":

ACCESS=N

or A=N

or AN

where N is the access level number and has a range of 0 through 15. The default access level is zero.

### 9-2.6.2 Blocking Factor

The blocking factor parameter is only specificable if the "B" (blocked) parameter is supplied. It specifies the number of sectors in one block of a blocked area. The range of blocking factor values is 1 through 7 and the parameter is specified by:

BLOCK=N

or B=N

or BN

where N is the numeric value of the blocking factor. If no blocking factor is supplied the default for the disc pack on which the area is to be created is used.

### 9-2.6.3 Eliminate Date

The Eliminate date specifies when the file may be purged from the system. The number supplied is the number of days before the area may be purged. The general form of the parameter is:

E=N

where N is the numeric value of the number of days before purging. The default for this parameter is the default purge date supplied at GENASYS time (Chapter G).

9-2.6.4 Granule Size

A granule is the size of the smallest set of contiguous sectors within a disc area. It is specified via the general form of:

G=N

where N is the numeric value of the size of the granule. The absence of a granule size specification will result in the default for the particular disc pack being used. A granule size is not specifiable with a monitor common block.

9-2.6.5 Location

The location parameter is only allowed when generating monitor common blocks. It specifies the absolute page location at which a resident monitor common block must load. The general form of the parameter is:

L=N

where N is the numeric value of the location and is specified either in pages or words of memory. That is:

L=32

L=' 100000

are equivalent.

9-2.6.6 Maximum Size

There is a limit to the size to which a program may expand. This parameter specifies that limit. The general form of the parameter is:

M=N

where N is the numeric value of the maximum size of the disc area. If not specified the value supplied at GENASYS time is used as a default. For monitor common blocks the value supplied is rounded up to the nearest page (an integral number of 10 sector blocks) and if no value is supplied the default is one page ( 10 sectors).

9-2.6.7 Pack Number

The pack number parameter enables the user to specify the pack on which the area is to be allocated. It is specifiable via the general parameter form of:

P=N

where N is the numeric value of the pack number. If no pack is specified the default pack is taken to be the work area pack (specified at GENASYS time). This is the pack on which the work and spool areas are generated. For the generate command to be successful the pack on which the area is to be created must have previously been Resourced (section 10).

9-2.6.8 Starting Sector Number

If it sometimes desirable, particular for real time program disc area accessing, to specify that an area start at a particular sector. The general form of this parameter is:

S=N

where N is the numeric value of the starting sector. The starting sector value is not specifiable with library areas or monitor common blocks. When no value of this parameter is supplied the area is generated in the first available slot on the specified disc.

9-2.6.9 Type Number

Any non monitor common block area may specify a type number. This number, which ranges from 0 through 7, is not used by any VULCAN service and is provided for user classification of his own area types. The general form of this parameter is:

T=N

where N is the numeric value of the type and is in the range 0 through 7. The default type number is zero.

9-3 $RTYPE

The $RTYPE command may only be used by the owner (generator) of a disc area. No one else is permitted to change an area type. This command allows the parameters listed below to be changed. The contents of the disc area are unaffected. The general form of the command is:

$RT,AREANAME,PARAMETERS

where AREANAME is the disc area name of an existing area. For libraries or monitor common blocks the six character maximum applies (see $GENERATE, Section 9-2) and the LIBRARY or MONITOR parameters must be present.

PARAMETERS are a limited set of those enumerated for the $GENERATE command (Section 9-2). The ordering of these parameters is unspecified. The following parameters are specifiable with a $RTYPE command.

9-3.1 Area Type

MONITOR or LIBRARY (See Section 9-2) are required if the area to be retyped is a monitor common block or a library. Areas are not retypeable to these types via the $RTYPE command.

9-3.2 Common Block Types

If the area is a monitor common block it may be changed from a resident monitor common block to a non-resident monitor common block (or visa-versa) via the parameters:

RESIDENT

or NRESIDENT

if nothing is specified the $GENERATE parameter is left intact.

9-3.3 Disc Area Structure

None of these parameters are specifiable on a $RTYPE command. Once an area is created as blocked, unblocked or random it is not changeable.

9-3.4 Directory Types

An area may be retyped to remove a CD (core directory) parameter or add it. The allowed parameters are:

| | | |
|---|---|---|
| | CD | core directory |
| or | DD | disc directory |

if neither are specified the $GENERATE parameter is left intact.

9-3.5 Access Parameters

The access parameters will be unchanged if none of the access parameters (listed below) are supplied. However, if any one is supplied it is necessary to supply all access states required including the "free" states on the $GENERATE:OW and OD. To remove all access bits an eleventh access parameter is allowed: OO owner only. To be effective this parameter must appear with no other access specifications. The access parameters available on a $RTYPE command are:

| | |
|---|---|
| PR | public read |
| PW | public write |

| | |
|---|---|
| PX | public execute |
| PD | public delete |
| AR | account read |
| AW | account write |
| AX | account execute |
| AD | account delete |
| OW | owner write |
| OD | owner delete |
| OO | owner only |

Account and public specifications are mutually exclusive.

### 9-3.6 Disc Area Definition Parameters

Some of the area definition parameters are modifiable via the $RTYPE command. These are:

| | |
|---|---|
| A=N | Access level of the area. |
| E=N | Eliminate date of the area. |
| L=N | Location for loading of resident monitor common block. |
| M=N | Maximum size of an area. This is not a variable parameter for monitor common blocks. |
| T=N | Type number of the area. |

The format of these parameters is explained in Section 9-2. It is not possible to modify the blocking factor, granule size, pack number or starting sector number.

Unspecified area definition parameters leaves the existing value intact.

## 9-4 $RNAME

The $RNAME command is used to rename a disc (qualifier and/or name). The area is unchanged and only directory entries are modified. The format of the command is:

$RN,OLDNAME,NEWNAME,PARAMETER

where OLDNAME is the existing disc area name and NEWNAME is the new name. The PARAMETER is either LIBRARY (if OLDNAME is a library area), MONITOR (if OLDNAME is a monitor common block) or absent if the area is a "regular" disc area. An area may only be renamed if the requestor of the rename is allowed delete access to the disc area.

## 9-5 $ELIMINATE

The $ELIMINATE command removes the requested disc area from the VULCAN system. The physical space is returned and the directory entry removed from the master pack and satellite pack directories. The format of the command is:

$EL, AREANAME, PARAMETER

where AREANAME is the disc area to be eliminated and PARAMETER is, as for the $RNAME command, LIBRARY if a library area is to be eliminated, MONITOR if a monitor common block is to be eliminated and absent if a "regular" disc area is to be eliminated. The user must have delete access if this request is to be successful.

## 9-6 $SQUEEZE

The $SQUEEZE command allows a user to "garbage collect" within a specified disc area. The form of the command is:

$SQ, AREANAME, N

where AREANAME is the disc area that is to be squeezed and N, only specifiable for unblocked areas, is the new size of the unblocked area in sectors. This command should be used for areas which once contained large amounts of data and now have smaller amounts. As an area expands granules are added. These are released only by editing (Section 5), copying to a new area and eliminating the old area or by utilizing the $SQUEEZE command.

## 9-7 $MAP

The $MAP command enables a user to obtain the name and parameters associated with selected disc areas. A request for a map of a single area may be made or a group of areas may be mapped for example, a disc pack. The general form of the command is:

$MA. OPTIONS, PARAMETERS

where OPTIONS are specified as for any processor and are listed below. The PARAMETERS define the extent of the mapping required.

The following information is always output for every area mapped: Areaname, type, pack and current sector size. The disc area name is the first entry on the first line of the output. The type of disc area takes the following form for data areas:

| | |
|---|---|
| B, U or R | for blocked, unblocked or random |
| T=N | where N is the type number |
| BF=N | where N is the blocking factor for blocked areas |

For program areas the type of disc area is identified as one of the following:

| | |
|---|---|
| NRH | non resident handler |
| RLIB | re-entrant library |
| INT | interactive or control point program |
| MON | monitor program |
| RT | real time program |
| RRT | resident real time program |
| COM | monitor common (non-resident) |
| RCOM | resident monitor common |

The identifier "R" is also output if the program is re-entrant.

The pack is specified via:

| | |
|---|---|
| P=N | where N is the pack number |

and the current size in sectors is specified by:

| | |
|---|---|
| C=N | where N is the size in sectors. |

## 9-7.1 Mapping Options

There are nine options recognized by the map processor:

### 9-7.1.1 G-option

The group option (G) specifies that a map for a group of disc areas is required. When this parameter is specified at least one PARAMETER defining the "group" is required. The disc area map output will, without any other options modifying the output, be alphabetized by name and then qualifier.

9-7.1.2 Q-option

The qualifier option (Q) is only applicable if the G-option has been specified. When used it causes the mapped output to be alphabetized by qualifier first and then by name.

9-7.1.3 N-option

The numeric option (N) is only applicable if the G-option has been specified. When used it causes the mapped output to be numerically ordered, firstly on pack number and secondly on starting sector number. When options N and Q are supplied together the ordering is numeric.

9-7.1.4 A-option

If the access option (A) is set further details of the disc areas may be obtained. In particular the access levels, access bits, user name and directory residency are specified. The format of the output is:

| | |
|---|---|
| A=N | where N is the Access level number |
| CD or DD | for Core directory or Disc directory |
| PR,PW,PX,PD | access levels for Public areas (Read, Write, Xecute, Delete) |
| AR,AW,AX,AD | access levels for Account areas (Read, Write, Xecute, Delete) |
| OW,OD | access levels for Owner areas (Write, Delete) for totally protected areas none of the above are output. |
| User Name | the name of the owner of the area is output. |

9-7.1.5 S-option

The size option (S) provides information about the size of the disc area. When given it provides the following information:

| | |
|---|---|
| S=N | where N is the starting sector number |
| G=N | where N is the granule size of the area in sectors |
| M=N | where N is the maximum size to which the area may expand. |

9-7.1.6 D-option

The date option (D) provides information about the dates and times associated with the disc area. These take the form:

Day Month Year hour: minute: second

for example: 20 Jan 75 10:30:00

The date and time is output for:

| | |
|---|---|
| GE | the GEneration date of the area |
| EL | the date the area will be ELiminated |
| LW | the date something was Last Written to the area |
| LA | the date the Last Access was made to the area |

9-7.1.7 P-option

The program option (P) outputs the following information for program areas:

| | |
|---|---|
| PT=N | where N is the octal Program Temp area starting address for program areas. |
| PP=N | where N (octal) is the number of virtual address registers required by a program (bits 15-8) and the number of pages to be read in from the disc area (bits 7-0) for program areas. |
| PA=N | where N (octal) is the number of library pages the program uses (bits 11-8) and the access level (bits 3-0) for the area. |

This option is not applicable for data areas.

9-7.1.8 E-option

The everything option (E) is an amalgamation of the A,S and D options. It allows the user to obtain the maximum amount of information associated with a disc area.

9-7.1.9 L-option

The list option (L) requests that the output be sent to the list output logical file (LFN 6). The normal, default output is to the diagnostic output logical file (LFN 3). When the L-option is used the line length of the output is increased from its normal value of 72 characters (for output to LFN 3) to 120 characters. Thus, the parameters associated with the map of an area will appear in different positions in the output depending upon whether the list option is set or not.

9-7.2 Mapping Parameters

The PARAMETERS on the $MAP command take two different forms. They may be one or a string of disc area names in which case the group option (G) must not be given. When the group option is supplied at least one parameter must be included.

9-7.2.1 Qualifier Parameter

The qualifier parameter is specified via:

Q=QUALIFIER

where QUALIFIER is a disc area qualifier. The inclusion of this parameter produces a disc map for all disc areas that have this qualifier and an access level equal to or lower than the mapping user's access level. Account and Qualifier parameters are not specifiable together on the same $MAP command.

9-7.2.2 Account Parameter

The account parameter is specified via:

A=ACCOUNT

where ACCOUNT is an account number - the first four characters (digits) of a qualifier. This parameter gives a disc map for all disc areas that have this account number and have an access level less than or equal to the mapping user's access level. Account and Qualifier parameters are not specifiable on the same $MAP request (Qualifier is a subset of Account).

9-7.2.3 Pack Parameter

The Pack parameter is specified via:

P=PACK

where PACK is a particular disc pack number (1 to 255). This parameter limits the mapped output to disc areas on the specified disc pack.

9-7.2.4 User Parameter

The user parameter is specified via:

U or USER

This parameter indicates that the mapped output is to be restricted to disc areas which were generated by the mapping user.

9-7.2.5 Type Parameter

The type parameter is specified via:

T=N

where N is a digit (0 through 7) or the character 'P'. The mapped areas will all be of the type number specified. 'P' indicates all program areas. Note that the type parameter may not be the only parameter given, at least a Q=, A=, P= or U parameter is also required.

9-7.3 Examples:

a) To map the disc areas of the current user, getting full information:

$MA. GE,U

b) To map a specified qualifier (1234ABCD) for the current user:

$MA. GE,U,Q=1234ABCD

c) To map a specified qualifier (1234ABCD) for the current user on disc pack 3. Limited output is required except for program areas and numeric ordering is required:

$MA. GPN,U,Q=1234ABCD,P=3

d) To map the whole system when it has three discs:

$MA. GE,P=1

$MA. GE,P=2

$MA. GE,P=3

e) To map all Type = 0 disc areas for current user:

$MA. GE,T=0,U

f) To map two specific disc areas:

$MA. E,AREA1,AREA2

# SECTION X
# MISCELLANEOUS COMMANDS

## 10-1 INTRODUCTION

This section details Job Control commands which, by the nature of their function, do not logically fit into any other section. These commands are by no means arbitrary in nature. The KEEP, FETCH and RSOURCE commands are fundamental to system execution and stability.

## 10-2 $KEEP

The $KEEP command saves specified disc areas. The output is to logical file 4 and thus requires an assignment before being operable. The disc areas are dumped on LFN 4 via a direct copy from the disc. The dumped format is:

a) Record 1

Record 1 is 15 words long and consists of:

| | | |
|---|---|---|
| WORD | 0-7 | disc area QDD entry |
| | 8-9 | the area qualifier |
| | 10 | current area size |
| | 11 | maximum area size |
| | 12-13 | purge date |
| | 14 | checksum of words 0-13 |

b) Record 2

Record 2 is 898 words long and consists of:

| | | | | |
|---|---|---|---|---|
| WORD | 0-895 | eight sectors of the disc area | | |
| | 896 | End of File flags: | | |
| | | B23 set if Sector 1 is an EOF | | |
| | | B22 | 2 | " |
| | | B21 | 3 | " |
| | | B20 | 4 | " |
| | | B19 | 5 | " |
| | | B18 | 6 | " |
| | | B17 | 7 | " |
| | | B16 | 8 | " |
| | 897 | Checksum of Words 0-896 | | |

c) Records 3-N

Records 3 through N are identical in form to record 2.

d) Record N+1

Record N+1 is an end of file (EOF).

The $KEEP processor writes an additional end of file record on LFN 4 and then backspaces over it. Thus, further $KEEP requests will only have one EOF between kept areas but the sequence will be terminated by a double end of file. The restoring process ($FETCH) terminates its searching upon encountering two adjacent end of files.

Disc areas may only be saved by their creator or by a user having the system save access. There are two forms of the $KEEP command:

10-2.1 Specified Area Keeps

The form of the specified area keep is:

$KE,AREA1,AREA2,....

where AREA1, AREA2 are disc area names. The areas specified are written to LFN 4 in the order specified. It is possible to change the name of the disc area when keeping it, for example:

$KE,OLDAREA=NEWAREA,....

where OLDAREA and NEWAREA are disc area names, the former being the name by which the area is found on the disc, the latter the name by which it is saved on LFN 4. The equal sign ( = ) is required to delimit old and new names. The change of name request must be contained in one argument.

10-2.2 Group Keeps

To save a group of disc areas that have one or more common parameters it is necessary to do a group keep. The form of the command is:

$KE.G,PARAMETERS

where the group save is specified by the G-option. PARAMETERS have the same format as for the $MAP command (Section 9). At least one parameter listed below must be present.

10-2.2.1 Qualifier Parameter

The qualifier parameter is specified via:

Q=QUALIFIER

where QUALIFIER is a disc area qualifier. This parameter will keep on LFN 4 all disc areas that have this qualifier. Account and qualifier parameters are not specifiable on the same $KEEP command.

10-2.2.2 Account Parameter

The account parameter is specified via:

A=ACCOUNT

where ACCOUNT is an account number: the first four characters (digits) of a qualifier. This parameter will keep on LFN 4 all disc areas that are under this account. Note that Account and Qualifier parameters are not specifiable on the same $KEEP command.

10-2.2.3 Pack Parameter

The pack parameter is specified via:

P=PACK

where PACK is a particular disc pack number (1 through 255). The areas kept are all saved from the specified disc pack.

10-2.2.4 User Parameter

The user parameter is specified via:

U

or

USER

This parameter indicates that the areas to be kept are only those created by the current user.

10-2.2.5 Type Parameter

The type parameter is specified via:

T=N

where N is a digit (0 through 7) or the character 'P'. The areas kept are all of the type number specified. 'P' indicates all program areas. Note that the type parameter may not be the only parameter given, at least a Q=, A=, P=, or U parameter is also required.

## 10-3 $FETCH

The $FETCH command is used to restore disc areas that have been saved via the $KEEP processor. The fetch command reads from logical file number (LFN) 4. To fetch a disc area LFN 4 is searched until the requested area is found or two consecutive end of files are found. This indicates the termination of the $KEEP sequence.

Disc areas may only be restored by the area owner or by a user having the system save access. There are two forms of the $FETCH command.

### 10-3.1 Specified Area Fetch

It is possible to request a fetch of a specified disc area or more than one area. The form of this command is:

$FE,AREA1,AREA2,....

where AREA1, AREA2 are the disc area names of the areas to fetch. It is possible to rename an area at the same time as fetching it from LFN 4. This is done by supplying the name it was kept with:

$FE,OLDAREA=NEWAREA,....

where OLDAREA is the area's name on LFN 4 and NEWAREA is the area name it will have when restored on the disc. The equal sign ( = ) is required as a delimiter to separate old and new names.

It is also possible to restore a disc area to a different pack from the pack from which it was kept. This may be done for an area that is to be renamed as well as a regular fetch:

$FE,AREA1,N,AREA2,OLDAREA=NEWAREA,N1,...

where N and N1 are integers in the range 1 through 255 and specify the disc packs on which AREA1 and NEWAREA are to be restored. AREA2 is restored to the same disc pack as it was saved from.

When specifying more than one area to be fetched, the order should correspond to the order of the areas on LFN 4 because only one pass over LFN 4 is made.

10-3.2 Group Fetch

When it is desired to restore a group of areas from LFN 4, the group fetch may be used:

$FE.G,PARAMETERS

where the option G specifies a group fetch and PARAMETERS determine the common parameter(s) of the group to be restored. There must be at least one PARAMETER as an argument.

10-3.2.1 Qualifier Parameter

The Qualifier parameter is specified via:

Q=QUALIFIER

where QUALIFIER is a disc area qualifier. This parameter will cause all the disc areas with the requested qualifier to be fetched from LFN 4. The Account and Qualifier parameters are not both specifiable on the same $FETCH command.

10-3.2.2 Account Parameter

The account parameter is specified via:

A=ACCOUNT

where ACCOUNT is an account number: the first four characters (digits) of a qualifier. This parameter will fetch, from LFN 4, all the disc areas with the requested account number. Note that the specifying of both Account and Qualifier parameters on the same $FETCH command is not allowed.

10-3.2.3 Pack Parameters

The pack parameter has two forms:

O=PACK

N=PACK

where O represents OLD-PACK and N represents NEW-PACK. PACK is a particular disc pack number (1 through 255). The OLD-PACK specification limits the fetch to those areas that were originally on the old pack. The NEW-PACK specification determines that all the areas restored for this $FETCH command are restored to the new pack regardless of original pack.

10-3.2.4 User Parameter

The user parameter is specified via:

U

or

USER

This parameter indicates that only the areas created by the current user are to be fetched from LFN 4.

10-3.2.5 Type Parameter

The type parameter is specified via:

T=N

where N is a digit (0 through 7) or the character 'P'. The areas fetched are all of the type number specified. 'P' indicates all program areas. Note that the type parameter may not be the only parameter given, at least a Q=, A=, O=, or U parameter is also required.

10-4 $RSOURCE

The $RSOURCE command is used to allocate a resource to an interactive terminal or a control point. For a control point program to successfully use the $RSOURCE command all resource requests must follow consecutively after the $JOB card. The appearance of a $RSOURCE command at any other point in the job control command sequence will result in an error. There is no such restriction for interactive programs.

Three different functions may be performed with the $RSOURCE command: a resource may be requested, a check may be made to see if the resource is available yet or a wait may be requested until the resource becomes available. There are four types of resources which may be requested by a $RSOURCE command.

10-4.1 Disc Pack Resourcing

To request the mounting of a non-mounted disc pack from an interactive terminal, the following format of the $RSOURCE command is used:

$RS,P=N

where P denotes pack and N is the numeric value (1 through 255) of the pack required. To determine if the pack is mounted the same command is repeated. An error message is output if it is not yet mounted. When the user cannot progress further until the pack is mounted, the wait request can be used:

$RS,P=N,WAIT

where WAIT denotes not to return to the user until the pack N is mounted. This command may be used as the initial resource request if it is desired to wait until the pack is mounted.

When control point programs require a disc pack, the command may take either form (with or without the WAIT parameter) since there will be no return to the job stream until all resources have been satisfied.

10-4.2 Physical Device Resourcing

Other than discs and magnetic tapes, all resourceable physical devices may be resourced from interactive terminals, or control points with a command of the form:

$RS,N=PDN

where N is the logical file number that is to be assigned to the physical device (PDN). The status may be checked with a command of the following form:

$RS,N

where N is the logical file number to be assigned. An error message results when the resourcing is not satisfied.

As in the case of disc pack resourcing it is possible to wait until the resourcing becomes available. The command form is:

$RS,N= PDN,WAIT

or

$RS,N,WAIT

where WAIT specifies no return until the device is available for use and the form of the command depends upon whether this is the original resource and wait (first example) or check status and wait (second example). Note that the assignment is made by the resource processor before the device is made available to the user.

When control point programs require a physical device, the command resource request may or may not contain the WAIT parameter. Return to the job stream will not occur until the device has been secured regardless of the WAIT parameter specification.

### 10-4.3 Magnetic Tape Resourcing

A magnetic tape may be resourced by using the following command form:

$RS,N=TAPENAME,PARAMETERS

where N is the logical file number that will reference the magnetic tape TAPENAME. TAPENAME is the magnetic tape identifier. It may be any number of characters in length but only the first six are passed as the tape identifier to the operator. If the tape is scratch, the identifier SCRATCH (a reserved identifier) should be used. Note that TAPENAME is an identifier, not a physical tape label on the tape. TAPENAME is passed to the operator via the system which requests that he load TAPENAME on a specified physical device number.

The PARAMETERS that are specifiable are listed below. If no parameter representing a particular option is present, the system default (obtained by the $TAPEOP service - Chapter D) is used.

#### 10-4.3.1 Character Mode

The type of data on the tape may be specified as one of:

EBCDIC
BCD
ASCII
BINARY

the first two characters uniquely specify the parameter.

#### 10-4.3.2 Density

The density of data on the magnetic tape may be specified as one of:

200BPI
556BPI
800BPI
1600BPI

where BPI may be any alphabetic string beginning with a B.

#### 10-4.3.3 Characters per Word

The packing density of characters is specified as one of:

1CPW
2CPW

3CPW
4CPW

where the CPW may be written as any alphabetic string beginning with a C.

10-4.3.4 Drive Type

The particular type of magnetic tape drive that the tape should be mounted on is specified as one of:

7TRK
9TRK
LOW-SPEED
HIGH-SPEED

where the TRK may be written as any character string beginning with a T and LOW-SPEED, HIGH-SPEED may be abbreviated to LO, HI. The LO, HI specification allows a user to specify a high speed or a low speed drive when there is such a choice. The drive type specification may not be used when a drive device (see below) parameter specification is issued.

10-4.3.5 Drive Device

If a particular physical drive is required it may be specified via the drive device parameter. Up to two physical device numbers may be requested as alternative drive device numbers.

The parameter is specified as:

:N

where N is the physical device number and the colon ( : ) indicates a physical device number. Two such parameters may appear on one $RSOURCE command. The drive device parameter may not be used if a drive type parameter has been specified.

10-4.3.6 Write Enable

The write enable parameter, specified via:

WRITE

allows the user to write on the tape (it is mounted with a write ring). The parameter may be condensed to the first two characters. If this parameter is not specified, the operator will not be asked to include the write ring and the tape is only readable.

10-4.3.7 Wait

The wait parameter is specified when it is desired to wait until the resource has been completed. It is specified by:

WAIT

and may be abbreviated to the first two characters, WA. If WAIT is not specified, return will be to the user after initiation of the resource request for interactive terminal users and will automatically wait until the device has been resourced for control point programs.

After initiation of the resource request, it is possible, from interactive terminals, to check if the device is available. The command form is:

$RS,N

where N is the logical file number that was specified on the original resource request. An error will be output if the resource has not been completed. A "wait till resourced" request may also be issued. It has the form:

$RS,N,WAIT

where N is the logical file number and WAIT specifies waiting until resourced. The assignment of the LFN to the physical device number of the tape drive is done by the resource processor before making the device available for use.

When it is desirable to have more than one logical file number assigned to a magnetic tape, the second assignment should be made by an indirect assign to the first, already existing, logical file number.

10-4.4 High Speed Memory Resourcing

High speed memory, if specified in GENASYS, is a system resource and must be acquired by a program via a resource statement. The form of the resource command is:

$RS,HS,PARAMETERS

where HS specifies High Speed memory as the "device" to be resourced. The PARAMETERS are:

| | |
|---|---|
| N=N1 | where N1 is the Number (N=) of pages (1024 word blocks) that are to be resourced. This is a required parameter. |
| CO | where CO specifies COntiguous pages. This requires that the N1 pages are contiguous. The parameter is optional. |

| | |
|---|---|
| S=N2 | where N2 is the Starting page number (S=) in physical memory. Since this implies contiguity the CO parameter is not allowed with this parameter. The Starting page number parameter is optional. |
| WA | the WAit parameter functions similarly for high speed memory as for all other resourceable devices. |

Since the WAIT parameter is optional, a command exists to allow an inquiry as to the state of the resource request:

$RS,HS

an error is output if the resource has not been completed. Note that it is not possible to have more than one high speed memory request pending at a time. To wait for the resource after initiating the request the following command is used:

$RS,HS,WAIT

Control point program resource requests wait for completion before returning to the job stream regardless of whether the WAIT parameter is included.

## 10-5 $DLOAD

The $DLOAD command is used to output a load module to the dump output logical file (LFN 9). The format of this command is:

$DL,AREANAME,M,N

where AREANAME is the name of the program to be dumped. The program must be a non-resident handler or a monitor program. This restricts AREANAME to four characters (XXXX) since the monitor program name conforms to V:XXXX:V. M is the relative program origin and N is the absolute load bias address. Both M and N may be omitted, the default values being zero. The program is relocated by N-M before being dumped.

The dumped output consists of two records: a header record and a load module record. The header record consists of six words:

| | | |
|---|---|---|
| Word | 0: | "V:" followed by first character of area name |
| | 1: | Last 3 characters of area name |
| | 2: | Program low address |

Word 3: Program high address

4: Program start address

5: Checksum of Module record.

The second record consists of the load module relocated as specified (N-M).

## 10-6 $DBOOTSTRAP

The $DBOOTSTRAP command dumps a load module in bootstrap format. The command format is:

$DB,AREANAME,M,N

where AREANAME,M,N are defined as for the $DLOAD command. The output, which goes to logical file number 9 consists of one record namely the program load module relocated as specified (N-M). The output is identical to the $DLOAD command except the first record is not output.

## 10-7 $SPOOL-OUT

The $SPOOL-OUT command enables a user to spool a particular disc area to a specified physical device. The command format is:

$SP,AREANAME,PDN

or

$SP,AREANAME,:PDN

where AREANAME is the disc area to be spooled out to physical device PDN. The optional colon ( : ) denotes that the output is to a physical device rather than a logical file number. Note that all this command does is enter the requested area into the spool queue for the particular device. The disc area is output by a spooling program and so it is necessary for the user to ensure that the disc area is either not open to the terminal or control point program or that if it is open, it is only for the purpose of reading. If the disc area is open for writing, it is not possible for the spooling program to open the area for reading purposes. To avoid this, a $FREE command (Section 4) should be used to close all LFN's assigned to the disc area before the $SPOOL-OUT command is issued.

## 10-8 OPCOM COMMAND

Selected operator commands are available to users at interactive terminals. The list of the available commands, and their function, is given in Chapter F. The operator commands are in the same format as is required by the operator console except they are prefixed by a slash ( / ) character. An example of such a command from an interactive terminal is:

/QP,AREANAME

which requests the status of program AREANAME via the operator command QP.

# SECTION XI
# LIBRARY EDITING COMMANDS

## 11-1 INTRODUCTION

Library areas have their own special editing facilities. These are outlined in this section. A library area is created by the $GENERATE command (Section 9) and is two separate areas. Library area names must only be six characters in length since the two discrete areas are distinguished from each other by prefixing a "Dƀ" and a "Eƀ" in front of the names. The D prefixed area contains all the external definitions of the modules. The "D area" is re-ordered following every library edit request. Co-routines are flagged as such and only if a program has loaded a co-routine library module during Vulcanizing should it be necessary to make a second pass through the library. The "E" prefixed area contains all the library elements (modules) and this area is re-ordered only when a squeeze ($LSQUEEZE) request is encountered. The "E area" consists of relocatable binary modules.

The last library editing command encountered that contained a reference to a library area sets the named library as the default library for subsequent editing commands. The generation request for a library also sets the library created as the default for subsequent editing commands. On all the editing commands the first argument, if present, is the library area to be edited. When absent the default library is used.

## 11-2 $LADD

The $LADD command allows new modules to be added to the specified library. The form of the command is:

$LA,LAREA,LIST

where LAREA, if present, is the library area to be edited and LIST, if present, is a list of external definitions to be added to the library. The input is from logical file number 10 and this is searched for the external definitions in the LIST. When a match is found, the module containing the external reference is added to the library. When LIST is absent all modules on LFN 10 are added to the library. If LFN 10 is unassigned, a default assignment of LFN 10 to the LR (link ready) work area will be made. When wishing to reference the default library area, the command form is:

$LA,,LIST

where LIST again is an optional string of external references. Note that an error will result if an external reference requested to be added is already present in the library.

## 11-3 $LCHANGE

The $LCHANGE command allows new modules to replace out-dated old modules and add original new modules. The $LCHANGE command differs only from the $LADD command in that the external definitions encountered that are already in the library have their modules removed from the library and the new version is added. The form of the command is:

$LC,LAREA,LIST

where LAREA, if present, is the library to be changed. The default library is used if LAREA is not specified. LIST, if present, is a list of external references to be changed in the specified library. The input is from logical file number 10. If LFN 10 is not assigned, then a default assignment to the LR (link ready) work area is made. When the element containing the requested external definition is found, it is added to the library and the old version removed if the external definition existed previously. When LIST is not present, all the modules on LFN 10 are added to the library.

Example:

LFN 10 contains a module with entry points A and B. The library element area contains an element with entry points B and C. The command:

$LC,,A

will result in the module with entry points A and B being added and this will cause the module in the library with entry points B and C to be removed. Thus C will no longer exist, although only A was specified to be "changed". If $LADD was the command used, an error would have been returned to the user.

## 11-4 $LREMOVE

The $LREMOVE command removes all specified external definitions from the requested library area. The command format is:

$LR,LAREA,LIST

where LAREA, if present, is the library area to be searched. When LAREA is absent, the default library is used. For the $LREMOVE command LIST is a required parameter and is a list of external definitions which are to be removed from the library. If an external definition is found, its module is removed from the library so that to remove a particular module it is necessary to specify only one external definition in the module.

## 11-5 $LSQUEEZE

Periodically, after several editing commands have been issued ($LADD, $LCHANGE and $LREMOVE commands), the library should be squeezed. Due to the expandable disc area structure (Chapter A) the area does not become filled (depending on the M (maximum size) value specified at the generation time of the library - Section 9) but large areas of disc space may be wasted after many successive library editing requests. The $LSQUEEZE command recovers this space. The removal of old modules does not occur when the $LCHANGE or $LREMOVE commands are encountered. The disc space is marked as unaccessible. The squeeze command regenerates the element area of the library by copying the element (E) area to a work area in the order of the directory (D) area. When the copy is completed, the work area is renamed to the element area and the old element area is eliminated.

The form of the command is:

$LS,LAREA

where LAREA, if present, is the library area to be squeezed. When LAREA is absent, the default library is squeezed.

## 11-6 $LLIST

The $LLIST command outputs a map of the external definitions and modules contained in a library. The format of the command is:

$LL,LAREA

where LAREA, if present, is the library area to be mapped. When no LAREA is specified, the default library is assumed. The mapped output is written to logical file number 6. If LFN 6 is not assigned, a default assignment is made to the work area LO (list output).

# CHAPTER C

# SYSTEMS PROCESSORS

# SECTION I
# VULCANIZER

## 1-1 INTRODUCTION

The VULCANIZER is the processor which converts link modules (relocatable binary) output from processors such as the VASSEMBLER, FORTRAN, BASIC, SNOBOL, etc., into Program load modules. These program load modules are directly loaded via explicit JOB-CONTROL or Operator commands. The VULCANIZER will generate all types of program load modules and these are explained in section 1-6.2.

## 1-2 VULCANIZER FORMAT

The VULCANIZER generates two types of program formats. The first is for non-resident handlers and for monitor programs. This format consists of the program module in a contiguous string of words followed by a relocation map consisting of one bit for each word of the program. Following the last relocation vector word is a single word giving the starting address of the program. A sample non-resident handler program is described in Figure 1-1.

When non-resident handlers or monitor programs are loaded they must be relocated since their loading location is determined dynamically. These programs are never swapped out and run within the "monitor" and have monitor privileges.

The second format output by the VULCANIZER is for programs that execute in the user space. This includes control point programs, interactive programs and real time programs (both resident and non-resident). Re-entrant libraries, which are generated by the VULCANIZER, also have this format. The VULCANIZER outputs the load module in "page format". This consists of blocks of 1024 words being written to the disc area. Each block occupies 10 sectors of disc area. No relocation bits are required since the program load module is absolute within its address space. The program is subdivided into different types of pages; re-entrant and non-re-entrant. Figure 1-2 shows the general case of a program's logical address space layout. Every Vulcanized program may, potentially, contain one re-entrant library. If this option is utilized the first B logical pages (Figure 1-2) of the program will be devoted to the re-entrant library. Since the library generally will not completely fill an even number of 1024 word pages, there will be a wastage of B-A (Figure 1-2) words where B-A is less than 1024. The re-entrancy of re-entrant libraries is achieved by having one or more non-reentrant data pages associated with the re-entrant library. These pages (D-B in size) follow the re-entrant library in the

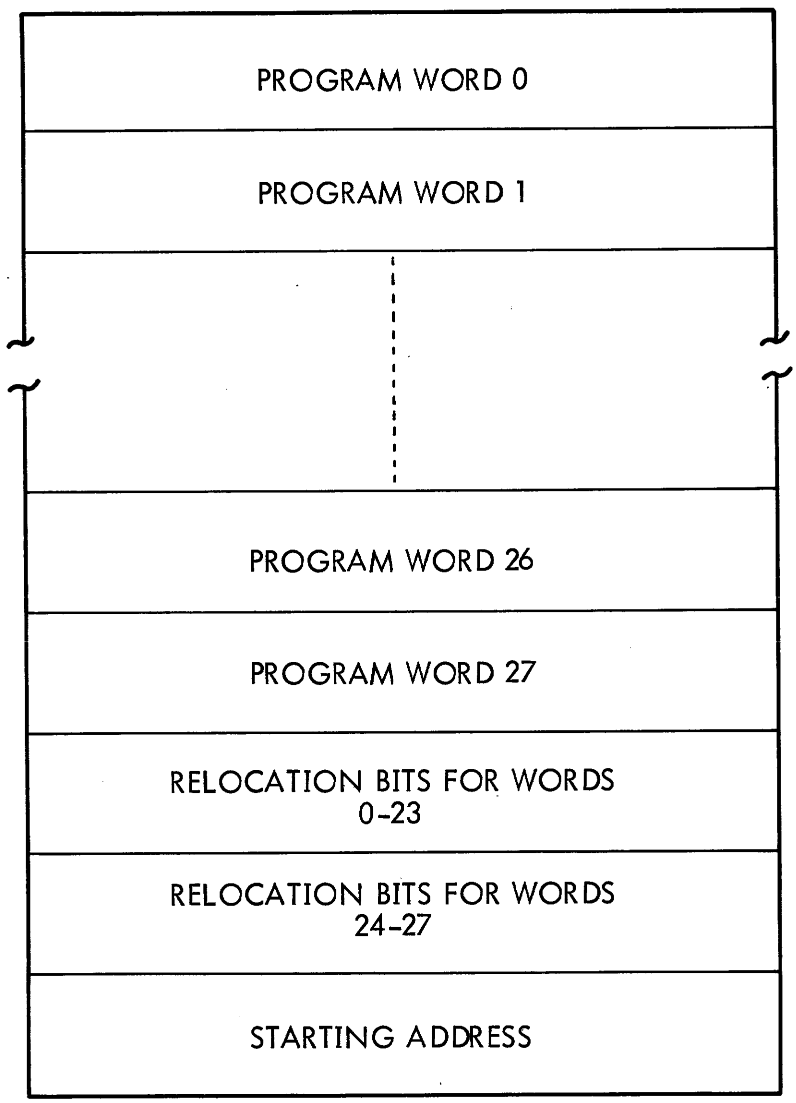



Figure 1-1. Sample Non-resident Handler Program

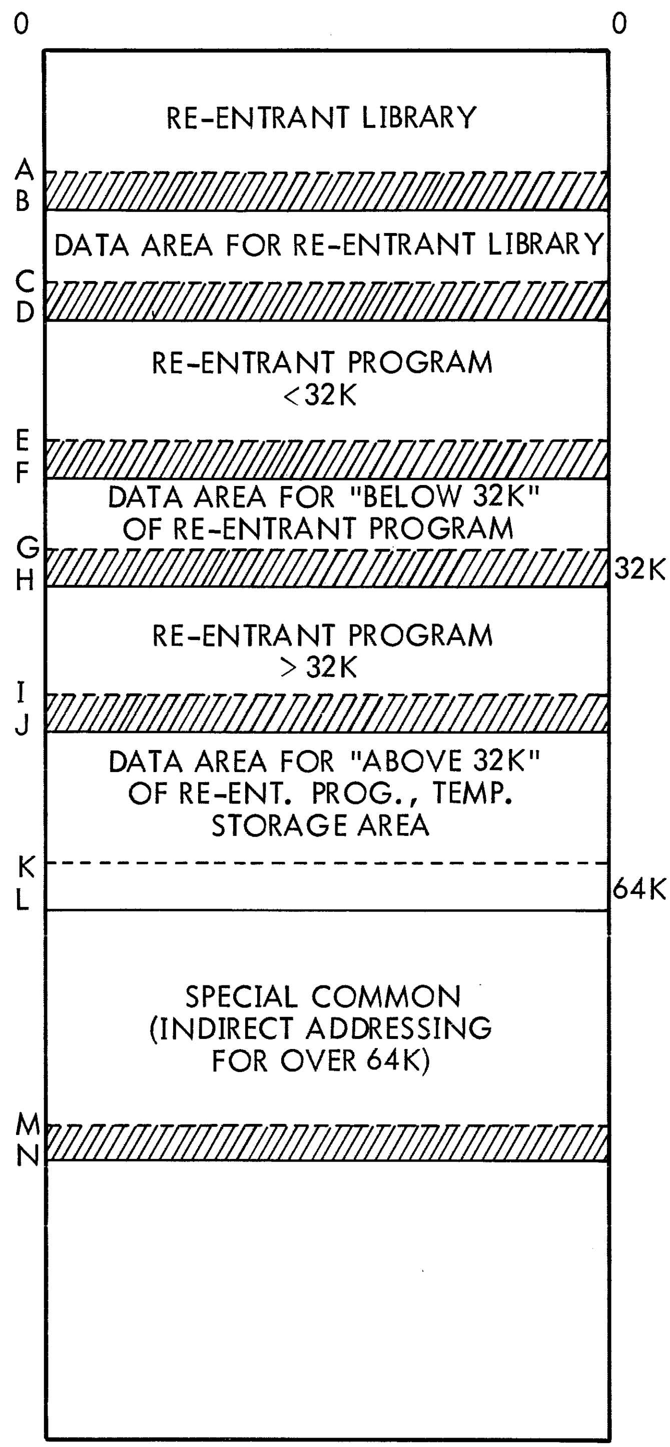



Figure 1-2. Logical Address Space of a User Program

logical address space of the program. A fraction of a page (D-C) wastage occurs if the program to be Vulcanized is re-entrant. The re-entrant program must start on a page boundary (D) and, if greater than 32K in size, is Vulcanized in two parts; below 32K and above 32K. The below 32K part of the program has its own data area to achieve re-entrancy (G-F) and to not require 16 bits logical addressing. The second part of the re-entrant program begins at the 32K boundary (H). A data area for the above 32K part of the program starts at page J and this data area also will include the temporary storage area space for monitor routine re-entrancy. The space above the temporary storage area to either the end of the logical address space of the program or the 64K boundary will be used for dynamic buffer space. This buffer space is either asked for directly by the user (via the $DCM service) or indirectly via blocked disc areas. Any areas defined as special common are Vulcanized above 64K in the program's logical address space and will be accessed from the program via indirect addressing. Figure 1-2 is the general case and, in the case of fractionation of the logical address space, is the worst case. When the program body is not re-entrant, the space C through J disappears and the program body is contained within the J through K area. If the program is greater than 32K, a break will still occur at the 32K boundary since individual routines may not be Vulcanized across this boundary. Sample disc area layouts for user space programs are shown in Figures 1-3 and 1-4.

## 1-3 RE-ENTRANT LIBRARIES

Re-entrant libraries allow an installation to collect together groups of routines which are commonly used across programs and Vulcanize them as a re-entrant Library program. This "program" may then be "included" in future programs that require these routines. Note that the difference between a re-entrant library and a disc library that is manipulatable via the library area editing commands (Chapter B) is that the whole of the re-entrant library is Vulcanized into the program whereas only those routines specifically requested are Vulcanized from the disc area library. However, regardless of how many programs are concurrently executing and require a re-entrant library routine only one copy of the library is ever in core at that time. When the disc library routines are required each one is uniquely Vulcanized into each program's logical address space. Thus, in execution, N programs each requiring the same disc library routine might have N copies of the same routine in memory at the same time. It is therefore suggested that each installation weigh the advantages of creating one or more re-entrant libraries which contain commonly used groups of routines. These routines must be written re-entrantly or have been created by the Fortran compiler and only one re-entrant library may be Vulcanized into each program.

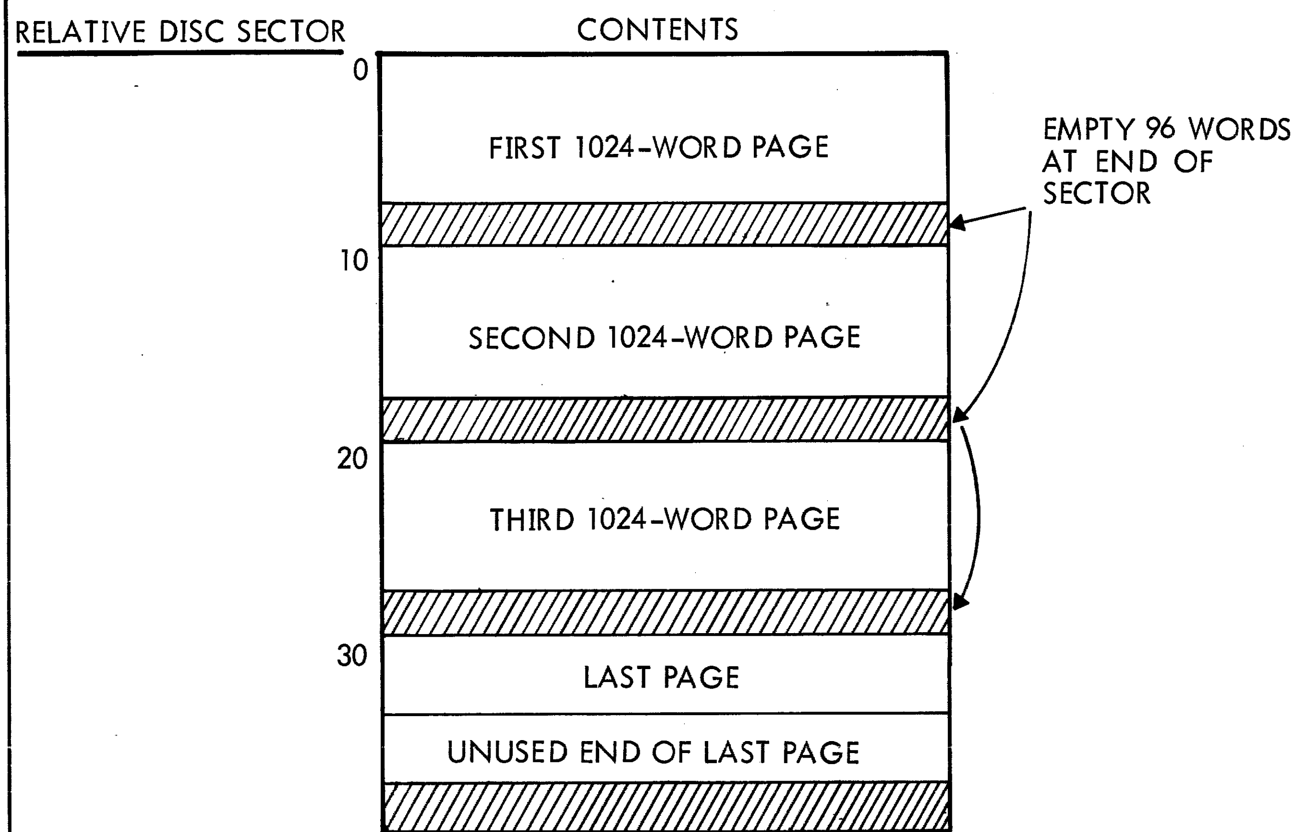



Figure 1-3. Sample Non-reentrant User Program

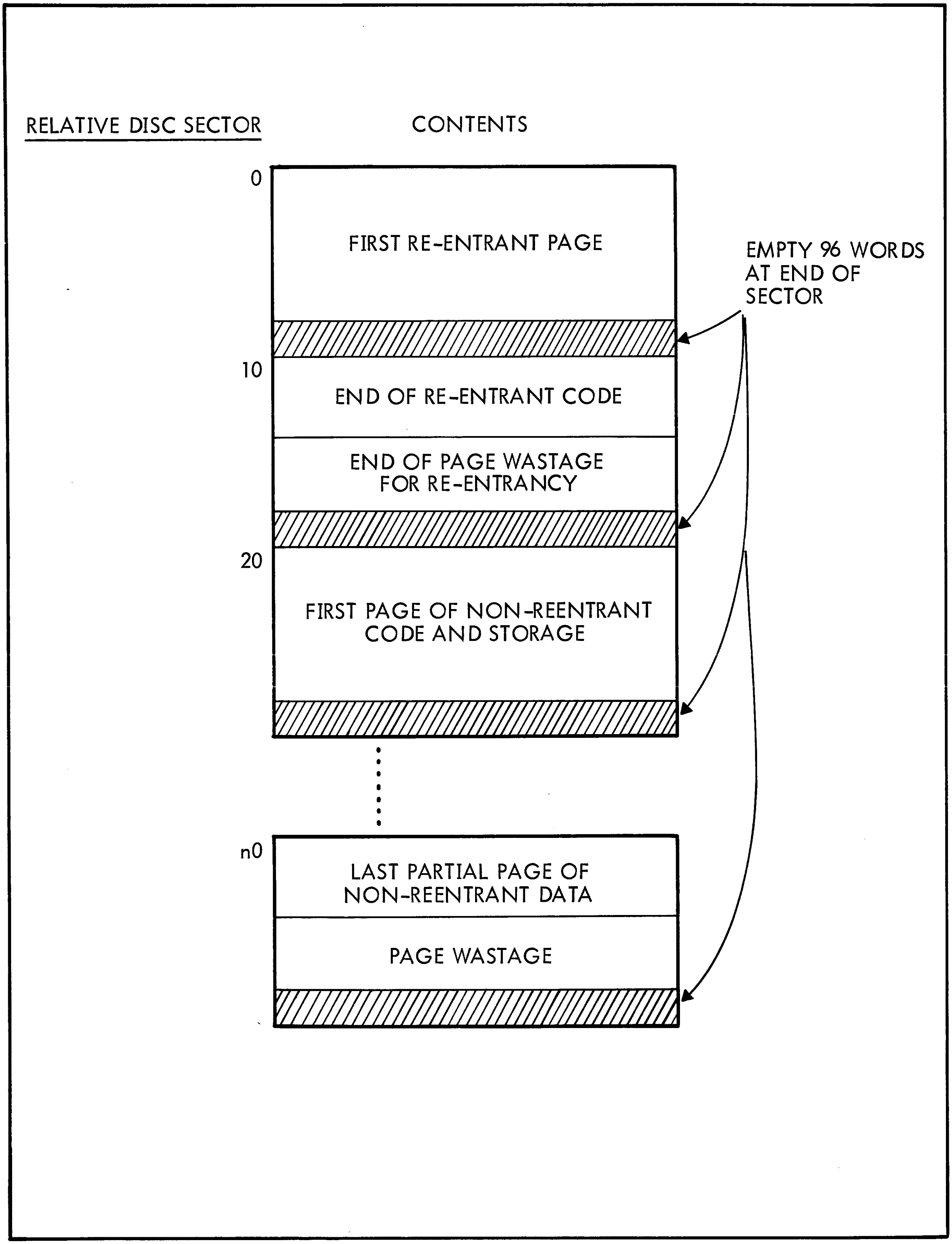



Figure 1-4. Sample Reentrant User Program

A re-entrant library is Vulcanized by linking all the required routines together. This is achieved by compiling them onto the LR (link ready) area or by specific Vulcanizer "LOAD" requests. A "TYPE" card of "RLI" is also required (Paragraph 1-6.2).

## 1-4 OPERATING PROCEDURE

The Vulcanizer processor must be loaded via Job Control. The generalized form of processing loading may be used:

$$*VULCANIZ. ABC,NAME,PARAMETERS

or a special Job Control statement may be issued which requests the loading of the Vulcanizer:

$VU. ABC,NAME,PARAMETERS

where A, B and C are Vulcanizer options and are specified in the next section. NAME is an optional parameter which, if included, is the disc area name of the program when Vulcanized. PARAMETERS are the optional parameters associated with the creation of the Program disc area NAME (see Section 1-6).

Since it is frequently desirable to be able to Vulcanize and execute the program, a "LOAD and GO" command is recognized by Job Control:

$VX. ABC,NAME,PARAMETERS

The "LOAD and GO processor" may be called as a regular processor via:

$$*VXECUTE. ABC,NAME,PARAMETERS

where A,B, C and NAME represent the same parameters as for the Vulcanizer loading request. If NAME is not supplied, the program is Vulcanized to the XE (execute) disc area. The program may subsequently be executed via the Job Control processor call:

$$XE

## 1-5 OPTIONS

The Vulcanizing options are detailed in this section. These options are supplied on the processor loading command. For example:

$VU. ABC

requests the Vulcanizer to be loaded and to set the A, the B and the C options. Note that some options are mutually exclusive.

1-5.1 A-option

The A-option allows the Vulcanizing process to continue despite what would normally be classified as fatal errors. This might be used to execute only a portion of a program which contains known errors.

1-5.2 D-option

The Dump module option allows a non-zero starting address to be specified for a non-resident handler or a monitor program. It is primarily of use for stand-alone programs which are dumped via the $DBOOT command (Chapter B).

1-5.3 I-option

The Include only option prevents any relocatable load modules from being Vulcanized from the link ready area. Thus, all modules to be Vulcanized must be explicitly loaded via the LOAD statement. This should be used if all modules which are to be used to construct the load module already exist as binary link modules in permanent disc areas.

1-5.4 M-option

The Multiple copies option allows more than one copy of the program to be in memory at a time. By default no multiple copies of non-reentrant programs will be loaded while a copy is already executing in memory.

1-5.5 N-option

The No listing option suppresses all the map information from the Vulcanizer.

1-5.6 O-option

The Original load option allows a user to specify that this is the first Vulcanizing of a program and, therefore, if a disc area of this name already exists it should not be eliminated. Without the O-option, the procedure is to eliminate the disc area of the given name if it exists before Vulcanizing the new program. With the O-option, an error will result if the program already exists.

1-5.7 R-option

The Re-entrant option enables a program to be Vulcanized as a re-entrant program. The program will be differently structured when this option is invoked since all procedure parts of routines will be built into the re-entrant part of the program and all data areas of routines are built into the non-reentrant part of the program. In a non-reentrant program (no R-option), the program is structured as procedure, data for one routine and then procedure, data for each successive routine.

### 1-5.8 S-option

The Short call option is used to specify to the Vulcanizer that no Vulcanizer commands will be supplied. The only information supplied to the Vulcanizer is through the options and, optionally the name of the program area to create. The VXECUTE commands (Section 1-4) have an implied S-option since no more Vulcanizer commands are expected.

### 1-5.9 U-option

The Update option is used to indicate that a re-entrant library is to be updated. That is, a new version of the library is to be created and replace the previous version. When a re-entrant library is originally created, a certain amount of data storage is required. The Vulcanizer creates the library with extra words of data storage to allow room for expansion. By default 100 extra words are included. This value may be changed by use of the TEMP command (Paragraph 1-6.8) when the library is Vulcanized. Thus, if the library procedure space does not exceed the number of pages which it used when originally created and the data area does not exceed the extra words allocated when it was originally created, the library can be updated. If either the procedure or data areas are too large, an error will result. Note that other than the U-option, the procedure to update a re-entrant library is identical to that of creating it originally. If the U-option is not supplied, the old library will be deleted and a new one, of the same name, created. This will necessitate all programs using the library to be Vulcanized again. If the Update option is successfully used, no updating of the programs using the re-entrant library is required.

### 1-5.10 W-option

The over-Write option allows a program to over-write the same disc sectors where the previous version resided. This option is only useful when modifying an already existing program and an error will result if the updated version of the program cannot be contained in the disc area allocated for the previous version.

### 1-5.11 X-option

The eXecute option requests that, on completion of the Vulcanizing process, the program be executed. It is implicit in the VXECUTE processor call (Section 1-4).

## 1-6 COMMAND STATEMENTS

When the S-option is not used either explicitly or by default the Vulcanizer expects to read commands from the Job Control input stream. The commands that may be input are detailed in this section. They are Vulcanizer commands and must follow a Vulcanizer processor load request. All commands given here may be abbreviated to their first two characters if desired.

### 1-6.1 NAME

The NAME statement is used to provide a name for the resultant program created by the Vulcanizer. The form of the command is:

NAME,AREANAME,PARAMETERS

where AREANAME is the disc area name and PARAMETERS are optional parameters associated with the creation of the area. The possible parameters are:

| | |
|---|---|
| PR,PW,PD,PX | indicating Public Read, Write, Delete and eXecute. |
| AR,AW,AD,AX | indicating Account Read, Write, Delete and eXecute. |
| OW,OD | indicating Owner Write and Delete. |
| CD,DD | specifying Core Directory or Disc Directory (default case). |
| A=N | where N is the Access level (less than or equal to the users access level). |
| P=N | where N is the Pack number on which to create the program. |
| S=N | where N is the Starting sector number if it is desired to have the program in a specific area on the disc. |
| E=N | where N is the number of days the area will remain on the disc before being Eliminated. |

More details of the format of these parameters is specified in Chapter B for the $GENERATE command. The parameters may be in any order.

When a NAME command is given it must immediately follow the Vulcanizer processor load request. The arguments AREANAME and PARAMETERS may be included on the Vulcanizer processor load card:

$VU. ABC,AREANAME,PARAMETERS

The form and definition of these arguments is the same as for the NAME command. Under these conditions a NAME command must not be given.

When no program area name is given either on the Vulcanizer processor loading request or by supplying a NAME command, the program is Vulcanized onto the XE (execute) area.

### 1-6.2 TYPE

The TYPE command specifies what kind of program is to be Vulcanized. This command must follow the NAME command (if one is given) or the Vulcanizer load request. The format of the command is:

TYPE,TEXT,PARAMETERS

where TEXT is one of the following:

| | |
|---|---|
| RT | indicating a Real Time program |
| RRT | indicating a Resident Real Time program |
| INT | indicating an Interactive or Control Point program |
| NRH | indicating a Non-Resident Handler |
| MON | indicating a Monitor program |
| RLIB | indicating a Re-entrant LIBrary is to be Vulcanized |

if the TEXT is RT or RRT a second argument, PARAMETERS, may be invoked. PARAMETERS consists of one or both of:

| | |
|---|---|
| PRIV | indicating the program may execute privileged instructions |
| NONACC | indicating the program should not generate accounting records during execution. |

If the TYPE command is absent the program is Vulcanized as "INT" (an interactive or control point program).

### 1-6.3 MODE

The MODE command defines the processing of contingencies for the program when it is in execution. The contingencies definable apply to SAU (Scientific Arithmetic Unit) and Fortran support libraries. The format of the command is:

MODE,PARAMETERS

where PARAMETERS consist of one or more of the following:

| | |
|---|---|
| SAUT | specifies that the program is to be executed with the SAU executive interrupt trap enabled. |
| NSAUT | specifies that the program is to be executed with the SAU executive interrupt trap disabled. |

| | |
|---|---|
| OA | specifies that the program is to be aborted if a floating point overflow occurs. (SAU or non-SAU) |
| NOA | specifies that the program is not to be aborted if a floating point overflow occurs. (SAU or non-SAU) |
| OM | specifies that a message is to be output to the diagnostic output device if a floating point overflow occurs. (SAU or non-SAU) |
| NOM | specifies that no message is to be output if a floating point overflow occurs. (SAU or non-SAU) |
| UA | specifies that the program is to be aborted if an underflow occurs. (SAU or non-SAU) |
| NUA | specifies that the program is not to be aborted if a floating point underflow occurs. (SAU or non-SAU) |
| UM | specifies that a message is to be output to the diagnostic output device if a floating point underflow occurs. (SAU or non-SAU) |
| NUM | specifies that no message is to be output if a floating point underflow occurs. (SAU or non-SAU) |
| IOA | specifies that the program is to be aborted if a FORTRAN error is detected during input or output. |
| NIOA | specifies that the program is not to be aborted if a FORTRAN error is detected during input or output. |
| CONV | specifies that special characters encountered during input are to be converted from 026 character set to an 026/029 compatible character set. |
| NCONV | specifies that no character conversion is to take place. |

The MODE specification PARAMETERS may be abbreviated to their first three characters. The default MODE values are:

MODE,SAUT,OA,OM,NUA,NUM,IOA,NCONV

1-6.4 MCOM

The MCOM command is only usable when the program to be Vulcanized contains monitor common blocks. The command allows a monitor common block name used with the program to be equivalenced to a real disc area which is a monitor common block. Since no qualifier specification is allowed with a monitor common block name in a program this command is necessary to allow monitor common blocks to be used that are not generated with the system default qualifier. For example, a reference in a program to monitor common block ABCD would, without an MCOM command, cause a monitor common disc area of 0000SYST*ABCD to be used. With an MCOM command ABCD may be equivalenced to any monitor common disc area. One form of the command is:

MCOM,BLOCKNAME=AREANAME,....

where BLOCKNAME is the name of the common block when internally referenced by the program and AREANAME is the disc area name to which the BLOCKNAME is to be equivalenced. The equal sign ( = ) is required to delimit the BLOCKNAME and AREANAME. More than one equivalence may appear on one MCOM command.

A second form of the command may be used:

MCOM,QUALIFIER

where QUALIFIER is a disc qualifier and will cause all monitor common block name references to be used as area names not with the system default qualifier but with QUALIFIER.

The two forms of the command may appear together in a string of Vulcanizer commands, the explicit monitor common equivalencing being unaffected by the default qualifier specification. Only one MCOM command of the second type is meaningful.

The use of Monitor Common is discussed further in Chapter A.

1-6.5 RLIB

The RLIB command is used whenever it is wished to Vulcanize a program with a re-entrant library. The form of the command is:

RLIB,AREANAME

where AREANAME is the disc area name of the re-entrant library that is to be included in the program. The disc area AREANAME must have previously been generated and Vulcanized as a re-entrant library (TYPE,RLIB).

### 1-6.6 PRIORITY

The PRIORITY command is used to set the execution priority of a program. The command format is:

PRIORITY,N

where N is a numeric value within the priority limits of the system. This command is only meaningful for programs Vulcanized as real time (TYPE,RT or TYPE,RRT).

### 1-6.7 ALLOCATE

The ALLOCATE command is used to modify the number of logical 1024 word pages that the program will require when executing. By default the Vulcanizer will give interactive and control point programs three more logical pages than the Vulcanized program size. This is for dynamic allocation of buffers during execution. It will not give any extra pages to real-time programs. To change the default values, the ALLOCATE command may be used:

ALLOCATE,N1

or

ALLOCATE,SIZE+N2

where N1 is the absolute number of pages the Vulcanized program will contain. If the program is larger than N1, an error will result. The second form of the command requests N2 more pages than the size of the Vulcanized program. SIZE+ may be any non-numeric character string starting with an "S". Note, that N1 and N2 are integer values of the number of logical pages, not words, that are required.

### 1-6.8 TEMP

The TEMP command allows the user to control the size of his temporary area. By default the Vulcanizer will give each program a default of 100 words of temporary storage space. This is sufficient for the worst case of "pushing down" due to nested monitor service routines. If a user has routines which use the "PUSH" and "POP" services as well as other system service calls it may be necessary to increase the temporary storage required. This is achieved by the Vulcanizer TEMP command:

TEMP,N

where N is an integer and is the requested size of the temporary storage area in words.

The temporary storage area is also used by the Vulcanizer to store information required when starting the program into execution. This information contains, among other things, all the Vulcanized resource and assign requests. If the default 100 words of temporary storage is not sufficient to contain this information, the default size will be increased to the size necessary to contain the loading information. However, if a TEMP command is used then an error will result if the size specified (N) is not large enough to contain all the required loading information.

### 1-6.9 ASSIGN

It is possible to set permanent assignments when Vulcanizing a program (any type). The ASSIGN command is the mechanism by which this is achieved. It is identical in format and result as the $ASSIGN Job Control command described in Chapter B. The only difference is that the dollar sign ( $ ) should not be present. A simple form of the command is:

ASSIGN,LFN=PDN,.....

where LFN is a logical file number through which input and/or output is accomplished internally in the program and PDN is the physical device to which the LFN is to be equivalenced.

### 1-6.10 RSOURCE

The RSOURCE command allows resource requests to be made at Vulcanization time. This command may not be used with interactive or control point programs (TYPE,INT) and must be used when Vulcanizing real time programs (TYPE,RT or TYPE,RRT) if the program is to use a resourceable device during execution. The form of the command is identical to that described in Chapter B for the Job Control $RSOURCE command. The only difference is that the dollar sign ( $ ) must not be present. A simple form of the command is:

RSOURCE,9=TAPE17

where logical file number 9 is to read and/or write a magnetic tape whose identifier is TAPE17. The tape options, since none are specified, are the system default values set at GENASYS time. Note that, since the program is not allowed into execution until all its resources are satisfied, the WAIT parameter (Chapter B, $RSOURCE command) is not required with the Vulcanizer RSOURCE command.

### 1-6.11 LIB

The LIB command is used to specify which disc libraries to search for undefined external references in the program. By default (that is, no LIB command) the system searches the system library (*LIBERY) only. If a LIB command is given and it is still wished to search the system default library then it, along with the other libraries to be searched, must be entered on the LIB command. The command format is:

LIB,AREANAME,....

where AREANAME is the disc area name of the library to be searched. The disc area must have been created as a disc library via the $GENERATE command (Chapter B) before being referenced. Several disc libraries may be specified either on one or several LIB commands. The order of searching is the order in which they are entered.

### 1-6.12 LOAD

The LOAD command requests particular relocatable binary modules (link modules) to be included into the Vulcanized program from a specified disc area. This disc area must consist of relocatable binary modules.

The form of the command is:

LOAD,AREANAME,XDEF1,XDEF2,....

where AREANAME is a disc area containing relocatable binary modules and each module it is desired to load is identified by an external definition (XDEF1,XDEF2, etc.). Only one external definition is required for each module that is to be loaded. The XDEF1, XDEF2 arguments are optional, if supplied only those modules explicitly referenced are loaded from the disc area AREANAME. All the relocatable binary modules are LOADed from AREANAME when XDEF1, XDEF2, etc., are omitted.

### 1-6.13 SSD

The SSD command enables a user to use his own version of the System Service Directory disc area. By default (no SSD command) the disc area *V:SSD contains the service directory. To use a non-standard SSD the command format is:

SSD,AREANAME

where AREANAME is the new system service directory disc area. The service directory defines the position of VULCAN service calls in the first 32 cells of memory (the BLU table).

### 1-6.14 INTERCEPT

the INTERCEPT command enables "BLU's" to be changed into "BLL's", or visa-versa, for a particular external reference. It also allows "BLU" references to be changed to different "BLU" references and "BLL" references to other "BLL" references. The command format is:

INTERCEPT,TYPE,OLDNAME,NEWNAME

where OLDNAME is the external reference in the binary modules and NEWNAME is the name to replace OLDNAME. In the absence of NEWNAME the OLDNAME is assumed.

TYPE specifies the type of branching instruction to search for and is one of the following four possibilities:

| | |
|---|---|
| LL | requests conversion of a BLL to a BLL |
| LU | requests conversion of a BLL to a BLU |
| UL | requests conversion of a BLU to a BLL |
| UU | requests conversion of a BLU to a BLU |

For example:

INTERCEPT,UL,AA,BB

will convert all BLU $AA instructions to BLL $BB. In the case of LL types of intercepts the external reference name does not have to be a BLL reference in order that the name be changed. It could be any other operation capable of referencing an external.

### 1-6.15 BEGIN

When all the Vulcanizer commands have been given the BEGIN command is recognized by the Vulcanizer as the terminating input command. The Vulcanizing process begins at this point.

## 1-7 RELOCATABLE BINARY FORMAT

Relocatable binary modules (link modules) are the standard output of all the compilers and the assembler. Each binary record consists of 36 words. The first word contains a "last record" identifier and a checksum. The second word contains 6 four-bit loader codes. The next 6 words are the data corresponding to each of the six loader codes. There are five groups of codes plus data. Figure 1-5 shows the form of the binary record. The record identifier (bits 23-16 of word zero) are '151 for all records except the "end of module record" which is a '161 code. Thus, the final record of a binary module is identifiable. The checksum (bits 15-0 of word zero) is calculated by the summation of words one through thirty-five. Bits 7-0 are exclusively or-ed into bits 15-8 and the checksum word shifted right logically eight bits. This is the checksum contained in bits 15-0 of word 0 of the record.

The loader codes are four bits in size and are given in Table 1-1. The codes are in octal. The special action codes are listed in Table 1-2.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| WORD | 0 | RECORD IDENTIFIER | | CHECKSUM | | | |
| | 1 | CODE 1 | CODE 2 | CODE 3 | CODE 4 | CODE 5 | CODE 6 |
| | 2 | LOAD WORD 1 | | | | | |
| | 3 | LOAD WORD 2 | | | | | |
| | 4 | LOAD WORD 3 | | | | | |
| | 5 | LOAD WORD 4 | | | | | |
| | 6 | LOAD WORD 5 | | | | | |
| | 7 | LOAD WORD 6 | | | | | |
| | 8 | CODE 7 | CODE 8 | CODE 9 | CODE 10 | CODE 11 | CODE 12 |
| | 9 | LOAD WORD 7 | | | | | |
| | 10 | LOAD WORD 8 | | | | | |
| | 34 | LOAD WORD 29 | | | | | |
| | 35 | LOAD WORD 30 | | | | | |

Figure 1-5. Binary Record Format

TABLE 1-1

RELOCATABLE BINARY CODES

| Code | Meaning |
|---|---|
| 00 | Direct Load |
| 01 | 15 bit Memory Reference relative to procedure counter (RORG) |
| 02 | 16 bit Memory Reference relative to procedure counter (RORG) |
| 03 | 18 bit Memory Reference relative to procedure counter (RORG) |
| 04 | RDAT (repetition of data groups). The Load word consists of:<br>Bits 23-16 as number of words in the group<br>Bits 15-0 as number of times to repeat the group |
| 05 | 15 bit Memory Reference relative to data counter (PORG) |
| 06 | 16 bit Memory Reference relative to data counter (PORG) |
| 07 | 18 bit Memory Reference relative to data counter (PORG) |
| 10 | External definition: uses three load words - the first contains the address associated with the name. The name is contained in the next two load words. If bit 23 of the first word is ON, the address is relative to the data counter (PORG). Bit 23 OFF indicates program counter (RORG). |
| 11 | 15 bit Common Reference, see 13 below. |
| 12 | 16 bit Common Reference, see 13 below. |
| 13 | 18 bit Common Reference. The 3 types of common reference use three load words. The first word contains the operation and a displacement relative to the beginning of the Common block. The next two load words contain the name of the referenced common block. |
| 14 | Special Action Code. Bits 23-18 are the Function (Table 1-2). |
| 15 | 15 bit external reference, see 17 below. |
| 16 | 16 bit external reference, see 17 below. |
| 17 | Conditional external reference (bit zero set to unity is a 16 bit reference, bit 0=0: 15 bit reference. A conditional reference differs from unconditional in that, if an unconditional request for the same external has not appeared, the request is modified to "BLU $ABORT". All external references take 3 words, the first being the operation and the last two the name referenced. |

TABLE 1-2

RELOCATABLE BINARY SPECIAL ACTION CODES

| Code | Meaning |
|---|---|
| 00 | RORG: use bits 15-0 value as program counter till otherwise indicated. |
| 01 | PORG: use bits 15-0 value as data counter till otherwise indicated. |
| 02 | END, transfer address is in the program area (bits 15-0). |
| 03 | END, transfer address is in the data area (bits 15-0). |
| 04 | END, no transfer address. |
| 05 | Internal string back, bits 15-0 contain address of first link in the chain to be strung. |
| 06 | External string back, bits 15-0 contain address of first link in chain, next two words contain external name. |
| 07 | Name definition, next two words contain the name to be associated with the relocatable binary module. |
| 10 | Common definition, bits 15-0 contain the size of the block, the next two words contain the common block name. |
| 11 | Special Common definition, next two words contain the name of the common block to be above 64K in the program's logical address space. |
| 12 | Monitor Common definition, next two words contain the name of the monitor common block. |
| 13 | Common Origin, bits 15-0 contain the displacement from the block name (contained in the next two words) into which data is to be loaded. |
| 14 | System Service Request, the next two words contain the requested external name. |
| 15 | Re-entrant Flag, indicates that this relocatable binary module is re-entrant. |
| 16 | Source Error, was a source error in the program. |
| 17 | External Equivalence definitions, the next two words contain the name of the equivalence and the third word the equivalence value. |

TABLE 1-2 (CONT'D.)

RELOCATABLE BINARY SPECIAL ACTION CODES

| Code | Meaning |
|---|---|
| 20 | External Equivalence reference, the next two words contain the name of the equivalence and the third word contains the instruction into which the external equivalence is to be or-ed. |
| 21 | Short code flag, indicates that this relocatable binary module will not run in a program greater than 32K in size. |

# SECTION II
# VASSEMBLER

## 2-1 INTRODUCTION

The VULCAN assembler (VASSEMBLER) is identical to the HARRIS Macro Assembler except that extensions have been incorporated to support VULCAN re-entrancy. Some modifications have been included in the Vassembler to change the operating procedure from the Macro Assembler.

## 2-2 OPERATION

The Vassembler may be loaded via a Job Control command:

$VA. OPTIONS,AREANAME

or by a regular processor call:

$$*VASSEM. OPTIONS,AREANAME

where VA is a recognized Job Control command and AREANAME is an optional disc area name which indicates from where the source statements for the Vassembler are to be read. OPTIONS are Vassembler options described in the next section.

## 2-3 OPTIONS

There are two possible options with the Vassembler, the "S" (scratch area) option and the "E" (errors only) option.

The E-option allows the user to suppress all Vassembler output to the list output unless there is an error associated with the statement. Thus, only the records that have errors in them are output when the E-option is set.

The S-option controls the use of the scratch output area (LFN 8). The Vassembler will, optionally, copy the source input to the scratch area during pass-1. If the S-option is set, this copying is always done. If the S-option is not set, the source input is only copied to the scratch area if the source input is from a physical device other than the disc.

## 2-4 ASSIGNMENTS

The Vassembler will use up to four logical files. These are logical file number 5, 6, 7 and 8.

LFN 5 is for the relocatable binary output. If LFN 5 is assigned when the Vassembler is loaded, the Vassembler will write its relocatable binary output to 5 without repositioning the logical file. If LFN 5 is not assigned, the Vassembler dynamically assigns LFN 5 to the LR (link ready) work area. Since LR is a work area, it will be automatically created by the assignment if it does not already exist. The assignment automatically rewinds the logical file.

Logical file 6 is for the list output. As for LFN 5, if LFN 6 is already assigned, the Vassembler writes its list output to LFN 6. For control point programs LFN 6 is assigned at initiation time and, without user re-assignment, will be assigned when the Vassembler is loaded. The interactive terminal programs do not have LFN 6 assigned and, therefore, without a specific user assignment, the Vassembler will make a default assignment for LFN 6 to the LO (list output) work area. LO will be created via the assignment, if not already in existence, and rewound. The assignment to LO will occur for control points in the event of LFN 6 not being assigned on entry to the Vassembler.

The source input statements are read from LFN 7. If the initiating command for the Vassembler contained an argument, the argument is the name of the disc area of the source statements to Vassemble and LFN 7 is assigned to this area. The assignment is released upon completion of the Vassembler execution. When no argument is supplied on the Vassembler load request, the source input statements will be read from LFN 7, if LFN 7 is currently assigned. If LFN 7 is not assigned, the Vassembler will indirectly assign LFN 7 to LFN 0 and read source statements from the command stream input.

The scratch output, a copy of the input source records, is optionally (see S-option, Section 2-3) output to LFN 8. When the scratch output area is used, the Vassembler will read LFN 8 for pass 2 for the source statements. If the scratch output is required, the Vassembler will output to LFN 8 which, if not currently assigned, is assigned to the work area W1. W1 will be created if it does not exist by the assign logic since it is a work area.

Examples:

The following four record program is to be built on area *DOG and Vassembled to the link ready area and listed to line-printer PDN 16:

```
TNK   '0122
BLU   $I/O
BLU   $EXIT
END$
```

Assuming the user has just logged on the terminal, the following sequence of commands could be used:

```
$GE,*DOG               or        $ASSIGN,6=16
$COPY,*,*DOG                     $VA.S
    TNK  '0122                       TNK  '0122
    BLU  $I/O                        BLU  $I/O
    BLU  $EXIT                       BLU  $EXIT
    END$                             END$
$EOF                             $FREE,6
$VA,*DOG                         $GE,*DOG
$FREE,6                          $COPY,8,*DOG,REW
$SPOOL,LO,:16
```

## 2-5 EXTENSIONS

Ten pseudo-operations have been added to the HARRIS Macro Assembler to form the Vassembler. Most of these pseudo-operations revolve around either program re-entrancy or the ability of programs under VULCAN to access locations beyond 64K.

### 2-5.1 Reentrancy

To achieve re-entrancy the Vassembler supports two location counters. These are referred to as RORG and PORG location counters. The RORG location counter is the location counter under which code, that will not be modified, is Vassembled. This is primarily instructions. Modifiable data and instructions are Vassembled under the PORG location counter. This does not preclude variable instructions and data from being Vassembled under the RORG locations counter. However, an attempt to execute this code as re-entrant will result in the program being aborted with a protection violation by VULCAN. The following pseudo-operations support VULCAN style re-entrancy:

RORG — The RORG pseudo-operation sets the location counter mode to the instruction counter. The argument supplied with the pseudo-operation is the value to set in the instruction counter. A "RORG *" pseudo-operation will set the instruction counter mode and the value will be the last RORG counter value. The Vassembler, upon starting execution, is in the RORG mode.

PORG — The PORG pseudo-operation sets the location counter mode to the data counter. The argument is the data counter value to be set. A "PORG *" pseudo-operation sets the PORG mode and the value to the last PORG counter value.

PDATA — This pseudo-operation is the equivalent of a DATA pseudo-operation if the location counter mode is PORG. When it is in the RORG mode, the pseudo-operation is equivalent to:

```
PORG  *
DATA  operand
RORG  *
```

PBLOK — The PBLOK pseudo-operation is the equivalent of a BLOK pseudo-operation if the location counter mode is PORG. When it is RORG, PBLOK is equivalent to:

```
PORG  *
BLOK  operand
RORG  *
```

PRDAT — The PRDAT pseudo-operation is the equivalent of a RDAT pseudo-operation if the location counter mode is PORG. When it is RORG, PRDAT is equivalent to:

```
PORG  *
RDAT  operand
RORG  *
```

REEN — The REEN pseudo-operation is used to flag a relocatable binary module as re-entrant. It is required by all binary modules that are to be Vulcanized into the procedure (RORG) part of a re-entrant program. There is no argument for this pseudo-operation.

LORG — The LORG pseudo-operation causes all literals which have been encountered up to the receipt of the LORG to be output under the RORG location counter. Reference to a literal after a LORG will generate a new literal even if the same literal had been referenced before the LORG. There is no argument for this pseudo-operation.

## 2-5.2 Addressing

Vulcan supports programs which have up to 64K of procedure space and up to 256K of data space. The upper 192K of memory is addressable via indirect LAC's and may be used to load, store or even execute a word. However, program execution can never exceed 64K since the program counter is limited to 16 bits. There are two pseudo-operations associated with accessing logical addresses above 64K.

LAC — The LAC pseudo-operation is used to generate a long address constant. The word generated, and form of the pseudo-operation, is identical to the DAC format (see Macro Assembler manual) except that bit 20 is set (indicating a LAC) and the address used is 18 bits rather than the 16 bits used by a DAC. Indirection and indexing may be used as for the DAC.

SCOM — The SCOM pseudo-operation is used to flag one or more common blocks as special common. The operand for this pseudo-operation consists of the common block names which are special common. These blocks are created via the regular Macro Assembler COMM pseudo-operation. Blank common is specified by a null name (two adjacent commas). Special common blocks are allocated above 64K and must be referenced indirectly through LAC pseudo-operations.

2-5.3 Monitor Common

Monitor common blocks may be Vulcanized into a program to provide a common data base between two programs. A pseudo-operation is provided to indicate to the Vulcanizer that certain common blocks are monitor common.

| | |
|---|---|
| MCOM | The MCOM pseudo-operation is used to flag common blocks as monitor common blocks. The operand of this pseudo-operation consists of the common block names which are monitor common blocks. The common blocks are created via the regular Macro Assembler COMM pseudo-operation. |

# SECTION III
# FORTRAN COMPILER

## 3-1 INTRODUCTION

The VULCAN Fortran compiler is identical to the Fortran used with other HARRIS operating systems. The only changes involve extensions for support of VULCAN re-entrant code and modifications to the operating procedure.

## 3-2 OPERATION

The Fortran compiler may be loaded via a Job Control command:

$FO. OPTIONS,AREANAME

or by a regular processor call:

$$*FORTRAN. OPTIONS,AREANAME

where FO is a recognized Job Control command and AREANAME is an optional disc area name which indicates from where the source for the Fortran compiler is to be read. OPTIONS are Fortran options described in the next section.

## 3-3 OPTIONS

There are ten possible options associated with the loading of the Fortran compiler. A more detailed explanation of all these options is contained in the Fortran manual.

### 3-3.1 A-option

The Abnormal (A) option is a specialized option associated with the use of index registers in the output code of the program. The Fortran compiler will, by default, attempt to optimize code by keeping index registers set across souce input records. If the Fortran code is doing abnormal functions with subscripts, particularly between subroutines, the A-option should be set to remove the optimization across Fortran input source records.

### 3-3.2 C-option

The Convert (C) option requests that input to the compiler, which is in the 026 card punch format, be converted to the 026/029 compatible set. The 026 characters ?@%&# are converted to " ' ( ) + =.

3-3.3 D-option

The Double precision (D) option requests that all implicit and explicit real variables and constants be double precision. Calls to Fortran real functions are replaced by calls to the double precision functions.

3-3.4 E-option

The Errors only (E) option suppresses the list output from the compilation unless the Fortran statement contained an error. Thus the only records listed when the E-option is set are error records.

3-3.5 L-option

The Long (L) option is used so that data in common above the 32K boundary will be accessed correctly. This option should be used for compilation of routines when they are to be included in a program greater than 32K.

3-3.6 M-option

The Map (M) option suppresses the map output after each END card. The map contains scalers, arrays and statement numbers and their relative addresses.

3-3.7 N-option

The sequence Number (N) option outputs sequences numbers for the list output from the compiler to the left of the line. It is separated from the compilation output line by a colon ( : ). This is useful when the output is to be output to a teletype or viewed via a CRT.

3-3.8 O-option

The Oject (O) option outputs the code generated by the compiler to the list output device. The object code consists of the relative address of the instruction, the instruction mnemonic, the operand, the Vulcanizer code, and the Vulcanizer load word (see Section 1).

3-3.9 T-option

The Triad (T) option outputs a symbolic dump of the internal compiler tables (TRIAD and AOIN) during expression scans. The option is only of value to system programmers.

3-3.10 W-option

For the Walk back (W) option the compiler sets mechanisms in the compiled program to provide execution error information. When an execution error occurs the subroutine linkages to get to the error point in the program and the record the error occurred in are output. The error record is denoted by the previous statement number in the routine plus a record count.

3-3.11 X-option

The eXtended object listing (X) option is of use only to system programmers. It allows the object listing associated with the O-option to be continued through the map output at the end of a sub-program. The information is of the same format as that described for the O-option.

3-4 ASSIGNMENTS

Three logical files may be used by the Fortran compiler, LFN's 5, 6 and 7.

Logical file 5 is for the binary output. It is normally assigned to the link ready area. When unassigned the compiler dynamically assigns LFN 5 to the link ready (LR) work area. If LR does not exist it is created by the assign request since it is a work area. No positioning of LFN 5 is undertaken by the Fortran compiler.

Logical file number 6 is for list output. It is normally assigned for control point programs. If it is unassigned Fortran will dynamically assign LFN 6 to the list output (LO) work area. If LO does not exist it is created by the assign request since it is a work area. LFN 6 is not positioned before being written to by the compiler.

Logical file number 7 is used for source input to the compiler. When the processor load request command card contains an argument (AREANAME in the examples of Section 3-2), the argument is the disc area name from which to read the source input statements. In this case LFN 7 will be assigned to this disc area and released upon completion of the compilation. When no argument on the load command card is supplied and LFN 7 is already assigned, the current assignment is used to read the source input cards. When no argument is supplied and LFN 7 is not assigned the compiler will assign LFN 7 indirectly to LFN 0, the command stream input.

Example:

The following input cards will all achieve the compilation of a program. Assume these commands are entered immediately after sign-on.

a) $FO,TESTFILE

b) $ASSIGN,7=TESTFILE

$FO

c) $FO

records in TESTFILE

.

.

.

END

where TESTFILE contains the source to compile and for case c) is entered on-line following the Fortran load request.

## 3-5 EXTENSIONS

The Fortran compiler is itself re-entrant when run under the VULCAN system. The code generated by the compiler is automatically re-entrant and can be used by the Vulcanizer to Vulcanize re-entrant programs. It may equally well be Vulcanized in non-re-entrant programs.

# SECTION IV
# VULCAN PROCESSORS

## 4-1 INTRODUCTION

This section briefly describes the operation of other processors available under the VULCAN operating system. These processors have all been made re-entrant for VULCAN operation. Details of the operation and use of the processor are obtained from the respective manuals.

## 4-2 BASIC

The Basic Compiler is loaded via:

$$*BASIC.OPTIONS,AREANAME

where AREANAME is an optional parameter and is the disc area name containing the input source for the Basic compilation. LFN 7 (source input) is assigned to this disc area. When AREANAME is absent the source input is from LFN 7 if assigned. If LFN 7 is unassigned it is assigned indirectly to LFN 0 so that the Basic statements are input from the command stream input. List output is written to LFN 6 which, if not assigned, is dynamically assigned to the work area LO (list output). Binary output is to LFN 5 which, if not assigned, is dynamically assigned to the work area LR (link ready).

The relocatable binary output from the BASIC compiler is re-entrant. Thus, Vulcanized Basic programs may be re-entrant.

There are five possible OPTIONS.

D-option is the Debug option. When the D-option is set a basic program is allowed to execute even though it contains assembly errors. The program aborts if an attempt to execute one of these errors occurs.

E-option is the Error only option. When the E-option is set only the records with errors in them are output (along with the error message) to the list out file.

I-option is the Inhibit run-time error messages option. Fatal errors will cause aborts but no run time library errors will be output when this option is set.

O-option is the Object list option. When set, the O-option produces an object listing of the program.

S-option is the Suppress error trace back option. When the S-option is set the Basic compiler does not produce the trace back code which determines the linkages gone through to get the error.

The Basic compiler acts as a regular compiler in that it outputs relocatable binary (object code) to LFN 5. To execute a Basic program the Vulcanizer is required to Vulcanize the program. The Basic run time library (*BASLIB) is a re-entrant library and should be requested in the Vulcanizing commands. A typical Vulcanizing command stream input is:

```
$VU.RX
RLIB,*BASLIB
BEGIN
```

## 4-3 SNOBOL

SNOBOL uses logical file number 7 for source input and LFN 6 for the list output. Both of these must be assigned before loading the SNOBOL processor via:

```
$$*SNOBOL
```

## 4-4 FORGO

The diagnostic Fortran, FORGO, uses logical file number 7 to read source input statements and LFN 6 to output the list output. Both of the assignments must be made prior to loading the FORGO processor. The loading is via:

```
$$*FORGO
```

to execute the compiled program one of the following commands may be used:

```
$CATGO
$LINKGO
$VX
```

Note that, although the $VX command is a regular Job Control command it is interpreted by the FORGO compiler and so may not be preceded by any Job Control commands.

## 4-5 CROSS REFERENCE

The cross reference processor uses logical file number 7 to read source input statements and writes its output to LFN 6. The source input must be assigned when the cross reference processor is loaded. If LFN 6 is not assigned it will, by default, assign LFN 6 to the work area LO (list output). The processor is loaded via the command:

$$*XREF

## 4-6 INDEXED SEQUENTIAL UTILITY

The index sequential utility package requires logical file number 6 to be assigned before being loaded. LFN 6 is used to list the commands from the processor and for any other list output. LFN 0 is used to read in the index sequential utility commands and these, by default will come from the same source as the loading request. Index sequential errors are output to LFN 3 (diagnostic output) which is defined as the terminal for interactive users and the list output device for control point users when the users sign-on. All the logical file numbers specified to the index sequential package are in decimal.

The processor is loaded via the command:

$$*ISUTIL

## 4-7 SORT/MERGE UTILITY

The sort/merge utility package uses LFN 6 for listing the commands and requires logical file number 6 to be assigned when the utility is loaded. The input commands are read from LFN 0 and, therefore, are from the same source as the loading request. Sort/merge errors are output to LFN 3 (diagnostic output) which is defined as the terminal for interactive users and the list output device for control point users when the users sign-on. All the logical file numbers specified to the sort/merge package are in decimal.

The processor is loaded via the command:

$$*MSUTIL

## 4-8 DEBUG

To debug programs the relocatable binary module *L. DEBUG may be included during the Vulcanizing procedure. The Vulcanizer commands to include the module are:

$VU

•

•

•

```
LOAD,*L. DEBUG
.
.
BEGIN
```

The debug module uses logical file number 3 for input and output and this is, by default, assigned to the terminal. The octal dump (OD) debug command writes to LFN 6 and, if this command is to be used, LFN 6 must be assigned prior to executing the program with *L. DEBUG included.

*L. DEBUG should not be used from Control Point programs.

# CHAPTER D

# SYSTEM SERVICES

# SECTION I
# GENERAL SERVICES

## 1-1 INTRODUCTION

System Services provide the interface between user programs and the VULCAN system. All are implemented through use of the Branch and Link Unrestricted (BLU) instruction, which is the only way a user program can enter or access the monitor. Fortran calls are implemented with library routines which make the BLU calls.

VULCAN BLUs are implemented by using the spare 7 bits of the instruction to define subtypes of BLU's. This provides a maximum of 2048 different BLU's, instead of the hardware limit of 32. Software interprets this subtype into seperated functions. Currently about 60 system services may be accessed, which are implemented through use of only about 16 BLU entries.

All of the System Services described in this and the following chapter are defined in terms of assembly language calls. Where they exist, the Fortran call is also included. It should be noted that unless otherwise stated, all system services destroy all user registers when called.

The services listed in this section provide the general interface capabilities required by all programs. This includes mostly those used to return to system control as needed.

## 1-2 $EXIT

Exit provides the mechanism for program to unload. All programs should have at least this statement in them. The calling sequences are as follows:

```
BLU $EXIT      (Assembly language)
```

or

```
CALL EXIT      (Fortran)
```

This will terminate the current program. If the program happens to be running at a control point or an interactive terminal then Job Control is loaded to read subsequent commands. Foreground programs will be removed from the system.

## 1-3 $SEXIT

Special Exit is the means by which a control point or interactive terminal terminates the job stream. It should not be used by user programs. If called by user programs, the call is converted into a regular Exit call.

## 1-4 $ABORT

A user program may terminate abnormally by calling the abort service, as follows:

BLU $ABORT

The address of the instruction following the abort call is output on the abort message unless the program is interactive or a control point, and the mode EM=OFF (refer to Chapter B) is in effect.

## 1-5 $SYSERR

A user program may output an error code message via the Syserr call. The calling sequence is as follows:

```
TLO     address
TOE     errorcode
BLU     $SYSERR
```

Control will be returned to the location following the BLU. No registers will be saved.

On entry, the K register is to contain the address to output with the message. K should be set to -1 if no address is to be output. The E register should contain the error code. If the program is interactive or a control point, and the mode EM=ON is in effect, the value in E will be used as a record number in the system error message disc area (*VULCMESS) in order to obtain a line of text to output with the error message.

Examples:

Assume:
a) Record 5678 of *VULCMESS is "EXAMPLE ERROR"

b) Mode EM=ON is in effect

c) Program is Job Control

Then

```
TOK   12345
TOE   5678
BLU   $SYSERR
```

Will Output: JOBCNTRL ER 5678 @12345: EXAMPLE ERROR

If K were negative, the above call would output:

JOBCNTRL ER 5678: EXAMPLE ERROR

If EM=OFF were in effect, the above call would output:

JOBCNTRL ER 5678

## 1-6 $DELAY

The delay service allows a user program to wait for a specified length of time before continuing to execute. The calling sequences are as follows:

| Assembly Language | Fortran |
|---|---|
| TOK n<br>BLU $DELAY | CALL TOADS ("WAIT", n) |

where "n" is the number of 120 Hz clock cycles (100 Hz on 50 cycle power supply systems) to delay for.

Upon return, the V, I, E and A registers are restored to their contents on the original call, and control is returned to the location following the BLU.

Example: To delay 3 seconds:

```
TOK   360        or        CALL TOADS ("WAIT", 360)
BLU   $DELAY
```

## 1-7 $HOLD

The hold service allows a program to output a message to the operator, and then be suspended from execution until released by the operator. It is called as follows from assembly language:

```
            TMK    PARLIST
            BLU    $HOLD
             .
             .
             .
PARLIST     'XX    MES            (generates XX8 in bits 23-18,
                                   address of "MES" in 15-0)
MES         DATA   "message text"
```

where: XX is the octal word count of the message (maximum = 20) message text is the message to be output.

The Fortran call is as follows:

```
PAUSE       •message text
```

Example:
```
TMK   A          .
BLU   $HOLD      .
                 .
                 .
                 .
A     '05        MES
MES   DATA       "EXAMPLE MESSAGE"
```
will output to the Operator:

```
programname:     EXAMPLE MESSAGE
```

# SECTION II
# BACKGROUND SERVICES

## 2-1 GENERAL

These services are usable only by interactive or control point programs. If called by real-time programs, the information returned is unspecified.

## 2-2 $NXTPRG

The Next-program service allows existing interactive or control point programs to "chain in" another specific program rather than having Job Control come in by default. The calling sequence is as follows:

```
                 TLO    PARLIST
                 BLU    $NXTPRG
                  .
                  .
                  .
LIST             DATA   2-word areaname
                 DATA   2-word qualifier
```

There is no return from this service. No registers are passed to the called program if it exists. This function performs an exit followed by a program load.

## 2-3 $OPTIONS

The Options service returns to the program the local option word associated with the loading request of the program. These local options are the option letters following the program name on the Job Control statement. The option letters A-X correspond to option bits 23-0 respectively. That is, option A is bit 23, etc.

The calling sequence for this service is as follows:

```
BLU     $OPTIONS
```

The local option word is returned in the E register.

Example:

Job Control Program Call: $$XYZ.ACE

Options word bits set are: 23, 21, 19

## 2-4 $LINES

The Lines service returns the number of lines per page that has been set either as the default 55 or by a Job Control mode statement. This may be used to control printer page spacing. The calling sequence is as follows:

```
BLU    $LINES
```

The number of lines per page is returned in the E register.

## 2-5 $UNWORK

The Unwork service is used to eliminate all of the batch work disc areas (such as W1, LR, LO etc.) for a particular control point or terminal, regardless of their access bits. This may be useful to ensure that no work areas were left over from a previous user of this terminal. The calling sequence is as follows:

```
BLU    $UNWORK
```

# SECTION III
# TIME/DATE SERVICES

## 3-1 $DATE

This service returns the current date and time in ASCII format. It is called via the following sequence:

```
TLO   PARLIST
BLU   $DATE
```

Upon return the following is set into PARLIST:

| | |
|---|---|
| PARLIST+0<br>+1<br>+2 | current Date in ASCII<br>Format is<br>dd Mon Yr. |
| PARLIST+3<br>+4<br>+5 | Current time based on<br>24-hour clock. As<br>HH:MM:SS |

Example:

If the current date and time were 11:24 PM on July 3, 1974, the service would return the following:

| | |
|---|---|
| PARLIST+0: | ƀ3ƀ |
| +1: | JUL |
| +2: | ƀ74 |
| PARLIST+3: | 23: |
| +4: | 24: |
| +5: | 00ƀ |

## 3-2 $TIME

This service returns the current date and time in binary format. It is called via the following sequence:

```
BLU   $TIME
```

Upon return, (A) = tenths of seconds since midnight; (E, bits 23-12) = year; (E, bits 11-0) = day of year, starting at 1, i.e., Jan. 1 is day 1, February 1 is day 32, etc.

3-3 $EXTIME

The Extime service returns the execution time of the current job or program. This is the CPU time that the program has used up to that point. It is called as follows:

BLU $ESTIME

Upon return the contents of the D register is the current execution time in T-register increments. This service may be used to return a value in milliseconds as follows:

BLU $ESTIME
DVO 133

3-4 $STRTIM

The start time service is used to return the date and time (in $TIME service format) at which this program, terminal, or control point began execution. It is called as follows:

BLU $STRTIM

Upon return, the contents of the E register is the starting date in $TIME format (Bits 23-12 = year; Bits 11-0 = day of year, 1-365). The A register is set to the starting time in tenths of seconds since midnight.

# SECTION IV
# TEMPORARY STORAGE SERVICES

## 4-1 GENERAL

For user or system subroutines to be re-entrant temporary storage is usually required. For this reason, all programs have cataloged into them a temp storage area, variable in size, the default size being 100 words. The system services and control routines utilize this area, which is also available to user routines.

## 4-2 $PUSH

In order to allocate temporary storage, one must "push down" on the storage stack. This is done via the Push service, as described in the following calling sequence:

```
TJK
BLU     $PUSH
DATA    n
```

where n is the number of temp cells to be allocated.

The K register value to PUSH is assumed to be the return address to the routine that called the service that is using the PUSH service and is saved as the return address for the subsequent $POP call. Return from PUSH is to the location following "DATA n" and the J register will be set to the value of the K register on input. The I register points to the first available temporary cell. The E register is saved across the call.

On each PUSH call, the system increases the number of cells requested by 2, to allow storage of internal pointers. An abort will result if an attempt is made to push more temp storage than has been allocated.

## 4-3 $POP

The POP service releases temporary service and returns to the routine that called the service (i.e., to the address passed in K on the corresponding PUSH call) and thus is the compliment of the matching PUSH call.

The A, E, and K registers remain intact through the POP call. The I register returns the current temp pointer as set after the storage has been released. The calling sequence is as follows:

```
BLU   $POP
```

## 4-4 $TEMP

The Temp service picks up the current value of the temproary storage pointer as determined by the preceding PUSH and POP calls. The calling sequence is as follows:

```
        BLU     $TEMP
```

The A, E, and K registers are not destroyed and the I register is returned as the current temporary storage value.

## 4-5 TEMP STORAGE EXAMPLE

The following example is a re-entrant user subroutine using temp storage to achieve re-entrancy. The subroutine is designed to save the calling parameter, which is a floating point number, multiply it by a constant and add in the value in temp, leaving the result in the X register.

```
                XDEF    SUB, ENTRY
ENTRY           GAP     1
                TIE                     Save Argument ptr
                TJK
                BLU     $PUSH
                DATA    2
                TEK                     (K) = Argument ptr
                TMX*    DAC0/K          Pick up argument.
                TXM     0, I            Save in temp.
                MMX     CONS
                AMX     0, I            Add in original Argument
                BLU     $POP

CONS            DATA    2.4D1
DAC0/K          LAC     0, K
                END
```

This routine would be called as follows:

```
                BLL     $SUB
                DAC     argument
```

# SECTION V
# MEMORY ALLOCATION SERVICES

## 5-1 GENERAL

Large blocks of memory may be allocated via the Dynamic Core Manager (DCM). These blocks are allocated from the program's logical address space and thus do not remain when a program exits or chains to another program.

The logical address space from which the blocks are allocated is that above all program code, data, and temp storage, up to the maximum number of pages allocated when the program is catalogued. This is normally three 1024-word pages beyond the program temp storage, but may be modified with the Vulcanizer Allocate statement (refer to Chapter C).

## 5-2 MEMORY ALLOCATION

The Dynamic Core Manager is used to allocate or deallocate buffer space within the logical address space of a program. To allocate regular dynamic core, the following calling sequence is used:

```
TOK   n
BLU   $DCM
DATA  1
```

where: n is the number of words to allocate.

On return, the K register will be set to the address of the core block allocated. If insufficient space exists to satisfy the request, K will be set negative.

To allocate a core block in High Speek (semi-conductor) memory, the following Calling Sequence is used:

```
TOK   n
BLU   $DCM
DATA  3
```

where: n is the number of words to allocate.

The K register is again set to the address of the block, unless insufficient High Speed Memory space exists, in which case K is set negative. K will also be set negative if insufficient High Speed Memory has been allocated via the Resource Commands (Chapter B or C).

## 5-3 MEMORY DEALLOCATION

The following sequence is used to deallocate a previously allocated block:

```
TOK   address
BLU   $DCM
DATA  2
```

where: address contains the address of the core block to be deallocated. There are no error return codes, as an invalid address causes the program to abort.

## 5-4 $LSPACE

The largest space available service returns to the user the size of the largest block of core that could currently be dynamically allocated. The calling sequence is as follows:

```
BLU  $LSPACE
```

On return, the E register contains the size of the largest core block available within the logical address space.

## 5-5 SYSTEM USAGE

The VULCAN system will use user DCM space for each blocked disc area which is open. The number of words allocated is the number of words in the blocking factor plus three. For example, a blocking factor of 2 will cause an allocation of 2X112 + 3 = 227 words when opened. With a maximum blocking factor of 7, the maximum blocked area DCM block is 787 words per blocked disc area open.

When using the $LSPACE service care should be taken to leave enough space for subsequent blocked area opens. Each $ADD card (refer to Chapter E) requires one additional blocking buffer area.

## 5-6 EXAMPLE

This example determines the largest available space, allocates it, and later returns the space.

```
BLU   $LSPACE
TEK
BLU   $DCM
DATA  1
CZK
```

```
BON   no space
TKM   BLOK
 .
 .
 .
 .
TMK   BLOK
BLU   $DCM
DATA  2
```

# SECTION VI
# DISC MANAGEMENT SERVICES

## 6-1 GENERAL

The VULCAN Disc Management Services are used to generate, eliminate, and otherwise change the status of disc areas. The relevant parameters are discussed further in Chapter A.

Disc Areas are classified according to types. These type fields are 8-bits long. Disc area types are broken down as follows:

Bit 7 = 1 = Program Area (may be created only by VULCANIZER)
0 = Data Area

Bit 6 = 1 = Directory entry to be core resident
0 = Directory entry to be disc resident only

FOR PROGRAM AREAS:

Bit 5: 1 = Reentrant
0 = Non re-entrant

Bits 4-2: 000 = Non-resident handler
001 = Re-entrant library
010 = Interactive/Control point program
011 = Monitor Program
100 = Non-resident real-time program
101 = Non-resident Monitor Common Block
110 = Resident real-time
111 = Resident Monitor Common Block

Bits 1: 1 = Multiple copies allowed of non re-entrant programs
0 = Multiple copies not allowed

FOR DATA AREAS:

Bit 5: 1 = Blocked

Bit 4-3 = 00 = Normal Blocked
01 = Spool out
10 = Spool in
11 = $Data area

Bit 5: 0 = Unblocked

Bit 4: 1 = Normal unblocked
0 = Random

Bits 2-0: Disc Area type. User controller.

## 6-2 DISC AREA GENERATION

Disc Areas may be created by using the generate service. The calling sequence for this service is as follows:

```
TLO   PARLIST
BLU   $GENERATE
```

The contents of the parameter list is as follows:

| | |
|---|---|
| PARLIST+0, +1 | Areaname. Format is truncated ASCII (8 6-bit characters). Minimum length is 2 characters. |
| +2, +3 | Qualifier. Format is truncated ASCII (8 6-bit characters). First four characters are account, second four are identifier. If both words are zero, then the system qualifier is used. If all 8 characters are blanks, then the user's sign-on qualifier is used. |
| +4 | Granule size in sectors. If zero, default granule size for specified disc pack is used. |
| +5 | Maximum size in sectors to which this disc area may expand. If zero, system default maximum area size is used. |
| +6 | Pack number. This pack must have been resourced if it is not a permanently mounted pack. If zero, system work pack is used. |
| +7 | Disc area type. See previous paragraph for definition of bits. |
| +8 | Purge date, entered as number of days the disc area will be saved. If zero, the system default is used. |
| +9 | Blocking factor. Sectors per block specification for blocked disc areas. Must be in range of 1-7. If zero, default for the disc pack is used. |
| +10 | Spool physical device. Normally used only on system generated spool disc areas. |

| | |
|---|---|
| PARLIST+11 | Access bits: |
| | Bit 6: 1 = public<br>0 = account |
| | Bit 5: set: public/account read<br>4: set: public/account write<br>3: set: public/account execute<br>2: set: public/account delete<br>1: set: owner write<br>0: set: owner delete |
| +12 | Access Level. (0-15) |
| +13 | Forced starting sector. This must be input as an absolute sector on the specified pack. If zero, disc area is generated wherever room exists. |

The following additional parameters may be used only by certain system processors for special purpose functions.

| | |
|---|---|
| PARLIST+14<br>+15<br>+16 | Special parameters for program areas for VULCANIZER |
| PARLIST+14<br>+15<br>+16<br>+17 | Area name of edit area used by Job Control when generating TP area during edit process. |

On return from the generate call, the status of the operation is returned in the A register as follows:

| | |
|---|---|
| (A) = 0 | Function performed as specified |
| 1 | Area Name already used on Master Disc |
| 2 | No room on disc pack |
| 3 | No room on master disc to create qualifier director |
| 4 | No room on satellite pack to create qualifier directory |
| 5 | Fatal Disc I/O prior somewhere in process |
| 6 | Area Name already used on satellite disc |
| 7 | Pack not resourced |
| 8 | Invalid access level requested |

| | | |
|---|---|---|
| A = | 9 | Spool PDN supplied on non-spool generate |
| | 10 | Invalid sectors per block specification |
| | 11 | First character of Name not alphabetic or name less than 2 characters long |
| | 12 | Old name for "TP" create does not exist |
| | 13 | Old area for "TP" create does not have write access |

Example:

| | | |
|---|---|---|
| Name | = | 1234ABCD* XYZ |
| Pack | = | 1 |
| Granule Size | = | 20 sectors |
| Default maximum size | | |
| Type | = | Blocked data, 2 sectors per block |
| Access | = | owner only |

The call would be as follows:

```
TLO     PARLIST
BLU     $GENERATE
BNZ     error
 .
 .
 .
DATA    T"XYZ ƀƀƀƀƀ "
DATA    T"1234ABCD"
DATA    20
DATA    0
DATA    1       pack = 1
DATA    '40     type = blocked
DATA    0
DATA    2       blocking factor
DATA    0
DATA    3       access
DATA    0
DATA    0
```

## 6-3 AREA ELIMINATION

Disc areas may be deleted by using the Eliminate service. This service is called as follows:

```
TLO     PARLIST
BLU     $ELIMINATE
```

where the parameter list is as follows:

| | |
|---|---|
| PARLIST+0, +1 | Areaname. 8 6-bit characters (Truncated ASCII). |
| +2, +3 | Qualifier. Truncated ASCII. First 4 characters are account, second four are identifier. If both words are zero, the system qualifier is used. If both are full of blanks, user's sign-on qualifier is used. |

Upon return from the Eliminate call, the A register is set to reflect the status as follows:

| | |
|---|---|
| A = 0 | Function performed as specified |
| 5 | Fatal disc I/O error |
| 7 | Pack not resourced |
| 20 | Disc area is open to another program |
| 21 | Disc area does not exist |
| 22 | User does not have delete access |

Example: To remove disc area 1234ABCD*XYZ:

```
                TLO     PARLIST
                BLU     $ELIMINATE
                BNZ     error
                 .
                 .
                 .
PARLIST         DATA    T"XYZ bbbbb"
                DATA    T"1234ABCD"
```

## 6-4 DISC AREA RENAMING

The name of a disc area, along with the qualifier, may be changed via the Rename service. Changing the name of a disc area requires that the user have delete access to the disc area. The calling sequence for Rename is as follows:

```
TLO     PARLIST
BLU     $RNAME
```

when the contents of PARLIST is defined as follows:

| | |
|---|---|
| PARLIST+0, +1 | Old areaname. 8 6-bit characters in truncated ASCII. |
| +2, +3 | Old qualifier. 8 6-bit characters in truncated ASCII. If both words are zero, the system qualifier is used. If all eight are blanks, the sign-on qualifier for the user is used. |
| +4, +5 | New areaname. 8 6-bit characters in truncated ASCII. Minimum length of 2 characters. The first character must be alphabetic. |
| +6, +7 | New qualifier. 8 6-bit characters in truncated ASCII. First four are account, second four are identifier. If both words are zero, the system qualifier is used. If both are blanks, the sign-on qualifier for the user is used. |

Upon return from this service, the A register is set to reflect the status of the operation as follows:

| | |
|---|---|
| A = 0 | Function performed as requested. |
| 1 | New area name and qualifier combination already exists on master disc. |
| 3 | No room on master disc to create qualifier directory. |
| 4 | No room on satellite pack to create qualifier directory. |
| 5 | Fatal disc I/O error. |
| 6 | New areaname and qualifier combination already exists on satellite disc. |
| 7 | Disc pack not resourced. |

A = 11 First character of new areaname not alphabetic or name less than 2 characters long.

20 Old disc area is in use by another program.

21 Old disc area does not exist.

22 User does not have delete access.

The reader will note that the error return codes for the Generate, Eliminate, and Rename services are identical, and a single error processor may therefore be used for them.

Example:

To change disc area 1234ABCD*XYZ to 1234FGHI*XYZ:

```
                TLO     PARLIST
                BLU     $RNAME
                BNZ     error
                 .
                 .
                 .
PARLIST         DATA    T"XYZ bbbbb"
                DATA    T"1234ABCD"
                DATA    T"XYZ bbbbb"
                DATA    T"1234FGHI"
```

## 6-5 DISC AREA RETYPING

The disc area type, access, and other parameters, may be altered for an existing disc area by using the Retype service. Retyping of a disc area may be done only by the original creator. The Retype service is called as follows:

```
TLO     PARLIST
BLU     $RTYPE
```

where the parameter list is defined as follows:

PARLIST+0, +1 } Areaname. Eight 6-bit characters in truncated ASCII.

| | |
|---|---|
| PARLIST+2 +3 | Qualifier. Eight 6-bit characters in truncated ASCII. If both words are zero, the system qualifier is used. If all blanks, the user's sign-on qualifier is used. |
| PARLIST+4 | If non-zero, the new area type. Only bits 6, 2, 1, 0 may be changed for data areas (see Section G-1). If zero type is unchanged. |
| PARLIST+5 | If non-zero, bits 6-0 are new area access bits to be or'ed into existing accesses. If bit 23 is set, then old bits are not used, and bits 6-0 become new access bits. See Paragraph 6-2 (Generate) for definition of bits. If word is zero, access left unchanged. |
| PARLIST+6 | If non-zero, new area maximum size in sectors. If zero, maximum remains unchanged. |
| PARLIST+7 | If non-zero, new purge date in days henceforth from current day. If zero, purge date remains unchanged. |
| PARLIST+8 | New access level. If zero, old access level remains. To change access level to zero, set bit 23 only. |
| PARLIST+9 | Used only by VULCANIZER for program area retyping. |

Upon return from this service, the A register is set to reflect the status of the retype operation. These codes are as follows:

| | |
|---|---|
| A = 0 | Operation performed as requested |
| 5 | Fatal Disc I/O error |
| 7 | Pack not resourced |
| 20 | Disc area in use by another program |
| 21 | Disc area does not exist |
| 22 | User is not the owner of the disc area. |

Example:

This example is used to change the access of disc area 1234ABCD*XYZ to Public Read, Owner Write, and Owner Delete.

```
                    TLO     PARLIST
                    BLU     $RTYPE
                    BNZ     error
                     .
                     .
                     .
PARLIST             DATA    T"XYZ ƀƀƀƀƀ"
                    DATA    T"1234ABCD"
                    DATA    0
                    DATA    B6B5B1B0
                    DATA    0
                    DATA    0
                    DATA    0
```

## 6-6 DISC AREA COMPRESSION

A disc area may be compressed to a smaller size via the squeeze service. This routine is used to remove granules from the end of the disc area.

Blocked areas are automatically compressed to the end of the valid data by this service. The user specifies the resultant size desired for unblocked areas.

The squeeze service requires that the user have delete access to the disc area.

The squeeze service is called as follows:

```
        TLO     PARLIST
        BLU     $SQUEEZE
```

where the parameter list is defined as follows:

| | |
|---|---|
| PARLIST+0, +1 | Areaname. 8 6-bit characters in truncated ASCII format. |
| PARLIST+2, +3 | Qualifier. 8 6-bit characters in truncated ASCII. If both are zero, the system qualifier is used. If blanks, the user's sign-on qualifier is accessed. |
| PARLIST+4 | For unblocked areas, this parameter is required to be the number of sectors to compress to. The disc area will be compressed to the next larger granule boundary above this size. This parameter is not required for blocked areas. |

Upon return from this service, the A register is set to reflect the status of the operation as follows:

| | | |
|---|---|---|
| A = | 0 | Operation performed as required |
| | 5 | Fatal Disc I/O error |
| | 7 | Disc pack not resourced |
| | 20 | Disc area in use by another program |
| | 21 | Disc area does not exist |
| | 22 | User does not have delete access to disc area. |

Example:

Assume disc area 1234ABCD*XYZ is unblocked and currently has 3 granules of 20 sectors each. Then, after the following call is made, it will have only two granules of 20 sectors each:

```
                TLO     PARLIST
                BLU     $SQUEEZE
                BNZ     error
                 .
                 .
                 .
PARLIST         DATA    T"XYZ ƀƀƀƀƀ"
                DATA    T"1234ABCD"
                DATA    30
```

## 6-7 DISC AREA INFORMATION SERVICES

The $DASAVE service is used to obtain disc area information on a single disc area or a group of areas.

The single disc area functions described below are those most likely to be used by user programs. Multiple save information is intended for use by Job Control MAP and KEEP processors, or user programs desiring similar features.

The calling sequence for this service is as follows:

```
TLO     PARLIST
BLU     $DASAVE
DATA    n
```

where the contents of the parameter list is determined by the specific function invoked, and "n" is the function code.

6-7.1 Directory Entry

Each of these services return the QDD (Qualifier Disc Directory) entry for one or more disc area. A QDD entry is eight words long. The contents are given in Table 6-1.

TABLE 6-1
QDD ENTRY

| WORD | CONTENTS |
|---|---|
| 0, 1 | Areaname. 8 6-bit characters in truncated ASCII |
| 2 | Starting sector number |
| 3 | Bit 23 = Generate/Eliminate pending flag<br>22-16 = Area Access bits (Refer to paragraph 6-2)<br>15-0 = Granule size in sectors |
| 4 | High 24 bits of user number |
| 5 | Bits 23-12 = Low 12 bits of user number<br>11-8 = Number of whole library pages before program<br>3-0 = Access Level |
| 6 | Bits 23-16 = Disc area type (Refer to paragraph 6-1)<br>15-0 = a) Temp address for program<br>b) Program size for non-resident handler<br>c) Size of temp area for re-entrant library |
| 7 | Bits 23-16 = Disc Pack number<br>15-8 = a) Number of temp cells for non-resident monitor program<br>b) Number of Virtual Address Registers required for other program types<br>c) Physical device number for spool disc areas<br>7-0 = a) Number of pages of program to read in for program and temp area<br>b) Sectors per block for blocked area (Bits 2-0 only) |

6-7.2 Single Disc Area Information

This function is used to obtain the area information of the specified disc area. This information will be returned only if the user has any valid access to the disc area. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $DASAVE
DATA    6
  .
  .
  .
```

where on entry:

| | |
|---|---|
| PARLIST+0, +1 | Areaname. 8 6-bit characters in truncated ASCII. |
| +2, +3 | Qualifier. 8 6-bit characters in truncated ASCII. |

If the disc area was located, the following information is returned.

| | |
|---|---|
| PARLIST+4 . . . +11 | Disc Area QDD entry. The user-number field will be set to zero. |
| +12, +13 | Qualifier. |
| +14 | Current disc area size in sectors. |
| +15 | Maximum disc area size in sectors. |
| +16, +17 | Purge date/time*. |
| +18, +19 | Generated date/time*. |
| +20, +21 | Last referenced date/time*. |
| +22, +23 | Last written date/time*. |

If the disc pack was not resourced, PARLIST+14 will be set to -1, and PARLIST+15-23 will not be returned.

* All date/time fields are in $TIME service format. The first word has the year in bits 23-12, the day of the year in bits 11-0. The second word is the time in tenths of seconds since midnight.

Upon return from this service, the A-register is set to reflect the status of the call as follows:

| | |
|---|---|
| A = -2 | Disc area not there |
| 0 | Function performed as requested |
| 3 | Disc read error in MDD |
| 4 | Disc read error in QDD |
| 5 | Disc read error in specified disc area |

Example:

To find out information about disc area 1234ABCD*XYZ:

```
                TLO     PARLIST
                BLU     $DASAVE
                DATA    6
                BNZ     error
                 .
                 .
PARLIST         DATA    T"XYZ ƀƀƀƀƀ"
                DATA    T"1234ABCD"
                BLOK    20
```

### 6-7.3 Privileged Disc Area Information

This service is identical to the service described above except that it will return the correct user number field. Additionally, only disc areas generated by the user may be accessed, unless the requesting program is Job Control, and the user has System Save access. The calling sequence is as follows:

```
                TLO     PARLIST
                BLU     $DASAVE
                DATA    7
```

The parameter list on entry, and the information returned is identical to the service above, except that the user number portion of the QDD entry is not set to zero.

## 6-7.4 Multiple Disc Area Information

The purpose of this function is to obtain the area names and corresponding parameters of all disc areas that match a specified parameter or group of parameters.

Information for up to 14 disc areas is returned on each call. Subsequent calls may be made to obtain further disc areas that match the previously specified parameters.

The calling sequence is as follows:

```
TLO     PARLIST
BLU     $DASAVE
DATA    n
```

where the matching function is determined by"n" as listed in the following chart.

| n | Function |
|---|---|
| <0 | Invalid. |
| 0 | Used for subsequent calls to continue returning information on previously specified parameters. |
| 1 | Find all disc areas on requested pack number supplied in PARLIST+0. |
| 2 | Find all disc areas with requested qualifier supplied in PARLIST+2, +3. |
| 3 | Find all disc areas on requested pack in PARLIST+0 that have requested qualifier in PARLIST+2, +3. |
| 4 | Find all disc areas with requested account number supplied in PARLIST+2. |
| 5 | Find all disc areas on specified pack in PARLIST+0 having requested account number in PARLIST+2. |
| 6 | -Single file information only- |
| 7 | -Single file information only- |
| 8 | Find all disc areas generated by current user. |
| 9 | Find all disc areas on specified pack in PARLIST+0 generated by current user. |
| 10 | Find all disc areas with specified qualifier in PARLIST+2, +3 generated by current user. |

| n | Function |
|---|---|
| 11 | Find all disc areas on specified pack in PARLIST+0 with specified qualifier in PARLIST+2, +3 generated by current user. |
| 12 | Find all disc areas with specified account number in PARLIST+2 generated by current user. |
| 13 | Find all disc areas on specified pack in PARLIST+0 with specified account number in PARLIST+2 generated by current user. |
| <13 | Invalid |

Access to disc areas is restricted to those generated by the current user unless the request comes from the Job Control program, and the user has the System Save access bit. Thus for regular user program requests, functions 1-5 are treated as if they were 9-13.

Upon return from this service, one or more 20-word disc area information blocks are returned starting at PARLIST+4, PARLIST+24, PARLIST+44, etc. Due to the nature of the disc optimization used by this service, between one and 14 blocks will be returned on each call. The word count returned (20 times the number of information blocks returned) will be returned in register E on each call. Subsequent calls should be made until the word count in E goes to zero (all disc areas located). Each call after the first should be with a function code of n=0, to continue scanning based on previously entered parameters.

The contents of the disc area information blocks will be as follows:

| Word | Contents |
|---|---|
| 0-7 | Disc area QDD entry |
| 8-9 | Disc area qualifier |
| 10 | Current file size in sectors |
| 11 | Maximum file size in sectors |
| 12-13 | Purge date/time* |
| 14-15 | Generated date/time* |
| 16-17 | Last referenced date/time* |
| 18-19 | Last written date/time* |

* Each date/time is in $TIME service format (Refer to paragraph 3-2).

If the disc pack containing the located disc areas has not been resourced, the word 10 is set to -1, and words 11-19 are not returned.

Upon return from this service, in addition to the word count in the E-register, the A register is set as follows:

| | | |
|---|---|---|
| A = | -1 | One or more disc areas are on unresourced packs |
| | 0 | Function performed as requested |
| | 1 | Invalid function requested |
| | 2 | No initial call made when n=0 |
| | 3 | Disc read error in MDD |
| | 4 | Disc read error in QDD |
| | 5 | Disc read error in user disc area |

Example:

The following routine will locate all disc areas generated by the current user located on disc pack 2:

```
START   TLO    BUF
        BLU    $DASAVE
        DATA   9
        CZA
        BNZ    error
LOOP    . . .  process block
        . . .
        TLO    BUF
        BLU    $DASAVE
        DATA   0
        CZA
        BNZ    error
        CZE
        BNZ    LOOP
        end  - all disc areas processed
BUF     DATA   2     Pack
        BLOK   3     not used
BLOCK   BLOK   280   (14 blocks max. times 20 words per block)
```

## 6-8 $DAASGN

The Disc Area Assign service is used to access a disc area which is to be saved. It is normally used by Job Control to assign to the disc area being accessed via the $KEEP command. However, user programs may access the service to provide special purpose assignments.

This service will assign Logical File Number 200 to the specified disc area, and will allow the disc area to be accessed in an unblocked mode regardless of the mode of the disc area. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $DAASGN
```

where the parameter list (PARLIST) points to a Disc Area Information Block (20 words) as output from the DASAVE service described above. The format of this block is given under paragraph 6-7.4.

The status of the call is reflected in the contents of the A register returned from the service as follows:

| | |
|---|---|
| A = 0 | Function performed as requested |
| 1 | User does not have valid access to disc area |
| 2 | Specified disc area does not exist |
| 3 | User specified QDD entry does not match actual entry |

## 6-9 $DAREST

The Disc Area Restore service is used to generate a disc area prior to restoring it. It is used normally by Job Control under the $FETCH command. Normal user programs should not have a requirement to use it.

The service will generate a disc area matching the specified parameters. It will then assign Logical File Number 200 to the disc area and provide unblocked access to the area regardless of the status of the area. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $DAREST
```

where the parameter list contains a Disc Area Information Block in exactly the format output by the DASAVE service, as given under paragraph 6-7.4.

The status of the call is returned in the A register as follows:

| | |
|---|---|
| A = 0 | Function performed as requested |
| 1 | User does not have valid access to disc area |
| 2 | Specified disc area already exists |
| 3 | Disc area cannot be generated |

# SECTION VII
# FORMAT SCANNER

## 7-1 GENERAL

VULCAN has, as a non-resident handler, a standarized format scanner. This scanner is used by Job Control, Operator Communications, Vulcanizer, and other system routines and processors. These routines provide the user the ability to analyze the same card format.

## 7-2 CARD FORMAT

The following conventions apply to input parameters to the scanner:

### 7-2.1 Delimiters

All blanks and commas are regarded as delimiters unless enclosed in quotes. A string of blanks is equivalent to one comma. Blanks following a comma are ignored. Default, or null, parameters may be specified by a pair of commas as delimiters.

Example:

The following four examples are all equivalent to two arguments separated by a delimiter:

A, B
A ƀ B
A ƀƀ B
A, ƀ B

The following four examples are all equivalent to the parameter A, followed by a null parameter, followed by the parameter B.

A,,B
A ƀ, B
A ƀ, ƀ B
A,,ƀ B

It should be noted that throughout this manual all scanner examples will use a single comma as a delimiter, to remove the ambiquity created by typewritten blanks.

### 7-2.2 Double Quotes

Double quotes ( " ) may be used to enclose a string of characters including blanks or commas. If a double quote is desired in such a string, it should be entered as a pair of double quotes.

If a double quote is not starting a string, then it is considered as a character in that string.

Example:

| Input Card Format | Actual Argument | Mode of Argument |
|---|---|---|
| ,ABCD, | ABCD | text |
| ,A"BCD, | A"BCD | text |
| ,"ABCD", | ABCD | text |
| ,"A""BCD", | A"BCD | text |
| ," A,B,C,D", | A,B,C,D | text |
| ,"3", | 3 | text |
| ,"3", | 3" | text |
| , 3, | 3 | numeric |

### 7-2.3 Single Numeric Values

All numeric values are assumed to be decimal unless preceded by a single quote, which denotes them to be octal.

Example:

, '77, converts to decimal 63

, 77, converts to decimal 77

7-2.4 Ranges of Numeric Values

A range of numbers may be input by placing a dash between two numbers. This range must not be decending. A single quote in front of the first number, denotes the pair to be octal.

Example:

| | |
|---|---|
| ,1-3, | converts to 1 with a range of 3 |
| ,4-4, | converts to 4 with a range of 1 |
| ,'10-12, | converts to 8 with a range of 3 |
| ,'4-14, | converts to 4 with a range of 9 |

7-2.5 Negative Numbers

A negative number may be input by preceding the parameter with a dash.

Example:

| | |
|---|---|
| ,-'77, | translates to -63 |
| ,-77, | translates to -77 |

7-2.6 Arguments with #

The character # must not be the first character in any parameter input to the scanner. This sequence is reserved for a future capability and will cause an abort currently.

7-2.7 Qualifiers

Standard VULCAN disc area qualifiers may be input in truncated ASCII with a special call described later. Qualifiers are of the form "nnnnxxxx" where "nnnn" are 1 to 4 digits specifying account number, and "xxxx" are one to four characters, starting with an alphabetic character, specifying identifier. Leading zeros are appended on the account numbers as needed.

Example:

| | |
|---|---|
| ,1234ABCD, | is translated as a qualifier to 1234ABCD |
| ,123ABD, | is translated as 0123ABCƀ |
| ,1A, | is translated as 0001A ƀƀƀ |

## 7-2.8 Disc Area Names

Disc area names are input with the scanner via a special call described later. The format of these parameters should be qualifier*areaname where qualifier is as discussed above, and areaname is 2-8 characters, the first of which is alphabetic.

Examples of area names:

| Input | Resultant Qualifier | Areaname |
|---|---|---|
| 1234ABCD*XYZ | 1234ABCD | XYZ |
| 1A*XY | 0001Ab̶b̶b̶ | XY |
| XYZ | b̶b̶b̶b̶b̶b̶b̶b̶ | XYZ |
| *XYZ | 0000SYST | XYZ |
| A | -Invalid - error - (only one character) | |

## 7-3 $SCINIT

The Scan Initialize service is used to initialize the scanner, and pass a buffer to be scanned. Only one buffer may be processed by the scanner at a time.

The calling sequence is as follows:

```
        TLO     PARLIST
        BLU     $SCINIT
```

where the parameter list is a two word list as follows:

```
PARLIST     DATA    buffer length in words
            DAC     buffer address
```

There are no error returns from the SCINIT service.

## 7-4 $GTHEAD

The Get Header service is to be used in conjunction with the $STHEAD service described below. They are used to save and restore the scanner buffer pointers. This allows concurrent scanning of multiple buffers by saving and restoring the buffers as needed.

The GTHEAD service copies the four-word scanner pointer block into the user's buffer. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $GTHEAD
```

where PARLIST is the first word of a 4-word buffer in which to save the pointers.

## 7-5 $STHEAD

The Set Header service is used to restore the pointers saved via a GTHEAD call as described in the preceding paragraph. This block must be unmodified from that returned by the GTHEAD service.

The calling sequence is as follows:

```
TLO     PARLIST
BLU     $STHEAD
```

where PARLIST is the first word of a 4-word block as saved by GTHEAD.

Example:

To save buffer pointers, re-initialize to another buffer, and then later continue scanning the original buffer:

```
                TLO     BUF
                BLU     $GTHEAD               save pointer.
                TLO     PL                    initialize to
                BLU     $SCINIT               new buffer.
                ---     process new buffer

                TLO     BUF                   restore original
                BLU     $STHEAD               pointers.
                ---     continue scanning first buffer

BUF             BLOK    4
PL              DATA    WC
                DAC     buffer address
```

## 7-6 $GTDISP

The Get Displacement service enables the current buffer displacement to be saved thus allowing a return to this buffer position in the event of a search for an unsuccessful type of argument. The calling sequence is as follows:

```
BLU     $GTDISP
```

Upon return, the current displacement is returned in the K register. This is an integer value from 0-n specifying the buffer position in terms of columns.

## 7-7 $STDISP

The Set Displacement service enables a new value to be set as the current scanner displacement value. This is normally a value output from a previous $GTDISP call, but may be any column number within the range of the buffer. The calling sequence is as follows:

```
TMK     buffer displacement
BLU     $STDISP
```

where buffer displacement is normally the output of a previous GTDISP call.

Example:

To reset scanner to start of current buffer, set displacement to column 0.

```
TZK
BLU     $STDISP
```

## 7-8 $CHAR

The Character service obtains the next character in the buffer, regardless of parameters, delimiters, etc., and updates the buffer displacement pointer.

The ASCII character is returned in bits 7-0 of the A register, with the rest of the register set to 0. When the buffer has been completely scanned, A will be returned as -2 (the end of buffer code). The calling sequence is as follows:

```
BLU     $CHAR
```

## 7-9 $BKCHAR

The back character is used to pick up the preceding character in the buffer (i.e., the last character scanned).

The ASCII character is returned in bits 7-0 of the A register, with the rest of the register set to 0. If the buffer has just been initialized, and there is no previous character, then the A register is set to -2.

The buffer displacement pointer is not modified by this call. Hence, successive Back Character calls will not backspace through the buffer, but will merely continually return the same character.

The calling sequence is as follows:

```
BLU     $BKCHAR
```

## 7-10 $NXCHAR

The Next Character service looks ahead and returns the next character that will be picked up. It is a look ahead feature in that the buffer displacement pointer is not modified.

The ASCII character picked up is returned in bits 7-0 at the A register, with the remainder of A set to zero. If the end of buffer is reached, A is set to -2. The calling sequence is as follows:

```
BLU     $NXCHAR
```

Example:

The following example demonstrates the function of the character input routines described above. The underscore is the buffer displacement pointer.

| Buffer before call | call | Character returned |
|---|---|---|
| $\underline{A}BCD$ | BKCHAR | none : (-2) |
| $\underline{A}BCD$ | NXCHAR | A |
| $\underline{A}BCD$ | CHAR | A |
| $A\underline{B}CD$ | CHAR | B |
| $AB\underline{C}D$ | BKCHAR | B |
| $AB\underline{C}D$ | BKCHAR | B |
| $AB\underline{C}D$ | CHAR | C |
| $ABC\underline{D}$ | NXCHAR | D |

## 7-11 $NXPARM

The Next Parameter service skips over redundant blanks and sets the buffer pointer such that the next character picked up is the first non-blank encountered.

The A register is set to zero upon completion of this call, unless the end of buffer was detected, in which case A is set to -2.

The calling sequence is as follows:

```
BLU     $NXPARM
```

Example: The underscore (____) is buffer displacement pointer.

| Buffer before call | Buffer after NXPARM call |
|---|---|
| A$\underline{,}$B,C | A, $\underline{\text{B}}$, C |
| A$\underline{,}$ ƀƀƀB | A, ƀƀƀ $\underline{\text{B}}$ |
| $\underline{\text{A}}$BC | A$\underline{\text{B}}$C |
| A $\underline{\text{ƀ}}$ƀƀƀ B | A ƀƀƀƀ $\underline{\text{B}}$ |

## 7-12 $STCHAR

The Set Character service allows a special delimiter character to be given to the Format Scanner. This delimiter is in addition to the standard blank or comma as a delimiter.

Note that this character will remain a delimiter until either overwritten by another Set Character call or an initialize of the scanner.

The calling sequence is as follows:

```
TOK     character
BLU     $STCHAR
```

Example:

If the following call were made:

```
TOK     "="
BLU     $STCHAR
```

then the character = would also delimit parameters. Thus the input text

AB = CD

would be input as the two parameters AB and CD.

## 7-13 $NUMBER

The Number service is used to input a numeric parameter. The buffer is scanned for the next argument and, if numeric, the binary value is returned to the calling program. The calling sequence is as follows:

```
BLU     $NUMBER
```

Upon return, the A register is set either to the binary value, or an error code as listed below:

| | |
|---|---|
| A = $\geq 0$ | Binary value scanned |
| -1 | Wrong type of argument was found |
| -2 | End of buffer |
| -3 | Null argument (default case) |
| -4 | Negative number encountered, value in K register, E is still length of range |

The E register is set to the length of a group of numbers. E is zero if only one number is input.

Example:

Given the following text inputs, the register values returned after a NUMBER call would be as follows:

| Text | A after call | E after call |
|---|---|---|
| ,32, | 32 | 0 |
| ,32-32, | 32 | 1 |
| ,32-41, | 32 | 10 |
| ,32A, | -1 | unspecified |
| ,'32, | 26 | 0 |
| ,, | -3 | unspecified |
| ,"32", | -1 | unspecified |
| ,-32 | -4 (K= -32) | 0 |
| ,-32-30 | -4 (K= -32) | 3 |

## 7-14 $SPNUMB

The Special Number service is used to input a number with embedded blanks. The service is identical to the $NUMBER service described above except that embedded blanks are ignored, and the scanning is terminated by any non-blank non-digit character. The calling sequence is as follows:

```
BLU     $SPNUMB
```

The return conditions are identical to those presented above under the $NUMBER service.

Example: Assume a $SPNUMB call.

| Text | A after call |
|---|---|
| ,32ƀ, | 32 |
| ,3ƀ2, | 32 |
| ,32A | 32 |
| ,1A2B, | 1 (pointer set to "A") |
| ,ƀ3ƀ2ƀƀA, | 32 |

## 7-15 $ONENUM

The One Number service is useful if a group of numbers would be an incorrect argument.

The returns from the $ONENUM service are identical to those produced by $NUMBER described above, except that the A register is set to -5 if a group of numbers was encountered.

In this case the K register is set to the first number of the group. The calling sequence is as follows:

```
BLU     $ONENUM
```

Examples:

| Input Text | A after call | E after call |
|---|---|---|
| ,32, | 32 | 0 |
| ,32-32, | -5 (K=32) | 1 |
| ,32-41, | -5 (K=32) | 10 |
| ,32A, | -1 | unspecified |

## 7-16 $NUMTEX

The Number-text service is used to return an argument starting with a numeric digit and after one or more such characters, one or more non-digit non-delimiter characters are encountered before reaching the delimiter. The calling sequence is as follows:

```
BLU     $NUMTEX
```

Upon return, the A register is set to the numeric value found, and E is set to the first three characters found after the numeric portion of the argument. In an error condition, the A register is set as follows:

| | | |
|---|---|---|
| A = | -1 | Wrong type of argument |
| | -2 | End of buffer condition |
| | -3 | Null argument (Default case) |

Examples:

Assume the indicated text is scanned using the $NUMTEX service:

| Input text | A register after call | E register after call |
|---|---|---|
| ,32AB, | 32 | ABƀ |
| ,3A, | 3 | A |
| ,3ABCD, | 3 | ABC |
| ,3, | -1 | unspecified |
| ,A, | -1 | unspecified |
| , 1-3AB, | -1 | unspecified |
| ,A3, | -1 | unspecified |
| ,, | -3 | unspecified |

## 7-17 $TEXNUM

The Text-number service is the reverse of the $NUMTEX service. Non-numeric characters are expected followed by an uninterrupted string of digits until a delimiter is encountered. The calling sequence is as follows:

```
BLU     $TEXNUM
```

Upon return, the E register is set to the first 3 non-digit characters found, and A is the integer value following the text part of the argument. In case of an error, the A register is set as follows:

| | | |
|---|---|---|
| A = | -1 | Wrong type of argument |
| | -2 | End of buffer |
| | -3 | Null argument (Default case) |

Examples:

Assume the indicated text is scanned using the $TEXNUM service:

| Input text | A register after call | E register after call |
|---|---|---|
| AB3 | 3 | ABƀ |
| ABCD32 | 32 | ABC |
| A3 | 3 | Aƀƀ |
| 3A | -1 | unspecified |
| A | -1 | unspecified |
| 3 | -1 | unspecified |

## 7-18 $TEXT

The text service picks up the first 6 characters in the next argument and returns them in the E, K registers. For less than 6 characters, E and K are blank filled with the text left justified, first in E, then in K. The A register is set to the number of characters in the argument just encountered. The calling sequence is as follows:

```
BLU     $TEXT
```

Upon return, the E register has the first three characters picked up, the K register has the second 3 picked up, and A has the number of characters encountered. In case of error, the A register is set as follows:

| | | |
|---|---|---|
| A = | -2 | End of buffer |
| | -3 | Null argument (Default case) |

Examples:

The following shows the results of $TEXT calls assuming the given text:

| Input text | A | E | K |
|---|---|---|---|
| ,ABCDEF, | 6 | ABC | DEF |
| ,AB, | 2 | ABƀ | ƀƀƀ |
| ,A, | 1 | Aƀƀ | ƀƀƀ |
| ,, | -3 | unspecified | unspecified |
| ,ABCDEFGH, | 8 | ABC | DEF |

## 7-19 $LTEXT

The Long Text service is the same as the $TEXT service except that a parameter list is supplied which enables an argument of size larger than six characters to be returned to the user. The calling sequence is as follows:

```
        TLO     PARLIST
        BLU     $LTEXT
```

Where PARLIST is a two word parameter list as follows:

```
PARLIST         DATA    word count of buffer
                DAC     buffer address
```

On return, A is set to number of characters picked up. If this is greater than the number of characters the user supplied buffer will hold, only those that can be held in this buffer are returned. Otherwise A is the number of characters stored in the buffer. The last word accessed for storage in the user's buffer is blank filled.

In case of error, the A register is set to -2 if end of buffer was found or -3 if a null argument (two adjacent commas) was encountered.

Example:

If the following call were made:

```
        TLO     PL
        BLU     $LTEXT
         .
         .
         .
```

```
PL      DATA    3
        DAC     BUF
BUF     BLOK    3
```

then the contents of BUF given the sample input would be as follows:

| Input text | A register | BUF |
|---|---|---|
| ,ABCDEFGHI, | 9 | ABC/DEF/GHI |
| ,ABCDEFG, | 7 | ABC/DEF/Gbb |
| ,AB, | 2 | ABb/?/?/ |
| ,ABCDEFGHIJ, | 10 | ABC/DEF/GHI |
| ,, | -3 | ?/?/? |

## 7-20 $AREANM

The Areaname service is used to input disc area names, which consist of a qualifier and an areaname field. The results are stored into a 4-word user supplied buffer, with the 8-character qualifier in truncated ASCII (6-bit characters) occupying the first two words, and the 8-character Areaname occupying the last 2 words. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $AREANM
```

where PARLIST is a user buffer at least 4 words long which will contain the results of the scanner input. On return the A register will be set to be the total number of characters picked up by the call. In case of an error, A is set as follows:

| | | |
|---|---|---|
| A = | -1 | Incorrect argument format |
| | -2 | End of buffer detected |
| | -3 | Null argument (consecutive commas) |

All areanames must be at least 2 characters long, the first of which must be alphabetic. The first four characters of the qualifier are the account number. The $AREANM service will right justify the account digits and zero fill the qualifier. An asterisk (*) must be used to delimit the qualifier from the areaname.

Examples:

Given the specified input text, a $AREANM call will produce the following outputs:

| Input text | Contents of PARLIST |
|---|---|
| ,1234ABCD*XYZ, | 1234/ABCD/XYZƀ/ƀƀƀƀ |
| ,12AB*XYZ, | 0012/ABƀƀ/XYZƀ/ƀƀƀƀ |
| ,*XYZ, | 0000/SYST/XYZƀ/ƀƀƀƀ |
| ,XYZ, | ƀƀƀƀ/ƀƀƀƀ/XYZƀ/ƀƀƀƀ |

## 7-21 $QUAL

The Qualifier service is used to pick up an argument that is a qualifier. This is effectively the same service as the first half of the $AREANM service. A 2-word buffer must be supplied by the user to store the argument. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $QUAL
```

The qualifier will be returned into PARLIST and PARLIST+1. Up to eight characters will be picked up and returned in truncated ASCII. The A register will be set to the number of characters scanned for this argument. For error conditions the A register will be set as follows:

| | | |
|---|---|---|
| A = | -1 | Invalid argument format |
| | -2 | End of buffer encountered |
| | -3 | Null argument (default case) |

Example:

Assume the following inputs were scanned with the $QUAL service.

| Input text | Contents of PARLIST |
|---|---|
| ,1234ABCD, | 1234/ABCD |
| ,12AB, | 0012/ABƀƀ |
| ,1A | 0001/Aƀƀƀ |
| ,A, | 0000/Aƀƀƀ |
| ,1, | 0001/ƀƀƀƀ |

## 7-22 SCANNER EXAMPLE

The following is a scanner example which will process a $VA (Vassembler) statement in Job Control. The basic format is as follows:

$VA. options     input areaname

Both the options field and input areaname fields are optional.

```
                TLO   PL                     initialize scanner
                BLU   $SCINIT
                TOK   " . "                  use  .  for delimiter
                BLU   $STCHAR
                BLU   $TEXT
                BON   error
                LRD   8
                COB   "$"                    ignore leading $
                BNZ   *+2
                LRD   8
                LRD   16
                TOB   " ƀ "
                CMA   ="VAƀ"
                BNZ   error
                BLU   $BKCHAR                test delimiter
                COB   " . "
                BNZ   NOOPTS
OPTS            BLU   $CHAR                  input an option
                BON   ENDOPTS
                COB   " ƀ "
                BOZ   NOOPTS                 end of options
                ...   process option letter
                BUC   OPTS
ENDOPTS         AOA   2
                BOZ   ENDCARD
                BUC   error
NOOPTS          TLO   NAME
                BLU   $AREANM                input areaname
                BOP   process name           process n areaname
                AOA   2
```

```
            BNZ     error                  no areaname
ENDCARD     . . .   continue processing
PL          DATA    27
            DAC     BUF
BUF         BLOK    27                     input buffer
NAME        BLOK    4
```

# SECTION VIII
# REAL TIME SERVICES

## 8-1 GENERAL

The Real-Time services discussed below are designed to provide the user with a flexible scheme for controlling real-time programs under VULCAN. The services are designed to effect the operation of real-time programs only. However, the services may be invoked by any program executing under VULCAN. Additionally Operator Communications provides access into most of these services, allowing the operator to control the real-time system.

### 8-1.1 Sleep State

All real-time and Monitor programs may "go to sleep" via the sleep service. The sleep state is a self-invoked suspension. Once a program has gone to sleep, it will not continue in execution unless awakened by another program, the operator, a timer schedule activation, or an external interrupt. This provides a means for a program to rapidly go into execution following an external stimulus, rather than waiting for a complete load operation.

### 8-1.2 Timer Schedule

The VULCAN Timer Schedule services provide means for activating tasks at specified intervals, or at a specific time. Timer scheduled events are based on the 120 Hertz clock (100 Hertz on 50 Hz power systems), and may be specified in terms of clock ticks. There are thus 120 clock ticks per second.

Timer Scheduling may be used for two functions. These are program initiation and program wakeup. The use of the Wakeup and Sleep services is recommended for any event with a frequency of greater than once every two seconds. This ensures that a program will be ready when called upon. Due to the nature of real-time program load and unload, it is not recommended to initiate any program with a frequency greater than once every two seconds. For this reason, the Initiate service is not available at greater frequencies.

Two types of schedule entries are available. These are permanent and temporary. Temporary schedule requests will be lost when a system is booted from disc. Permanent entries will remain through system loads (they are kept on a disc area) and will be recalibrated to the current time each time the system is booted. Events scheduled to occur while the system is not operating are ignored.

### 8-1.3 External Interrupt Control

Any external interrupt other than those used by standard I/O devices may be used for program control through use of the external interrupt services. These services provide a means for any real-time program to be awakened from the sleep state when a specified interrupt occurs. Additional controls allow selected interrupts to be armed or disarmed.

## 8-2 $INIT

The Initiate service is used to initiate real-time programs at a specified time and date, and with any specified repeating frequency. In all cases, the program will be loaded from disc and placed into execution. The service is called as follows:

```
TLO     PARLIST
BLU     $INIT
```

where the parameter list is defined as follows:

| | | |
|---|---|---|
| PARLIST | +0, +1 | 8-character program areaname in truncated ASCII. |
| | +2, +3 | 8-character program qualifier in truncated ASCII. |
| | +4 | Execution priority (0-63). |
| | +5 | Initiation parameter. Passed to program at initiation and loaded in its K register. |
| | +6 | Days in future from current day to start program. |
| | +7 | Hour of day to start program. *(0-23) |
| | +8 | Minutes of hour to start program. *(0-59) |
| | +9 | Seconds of minute to start program. *(0-59) |
| | +10 | Ticks of 120 Hz clock after specified second to start program. *(0-119) |
| | +11 | Period for re-initiation in days. |
| | +12 | Period for re-initiation in hours. |
| | +13 | Period for re-initiation in minutes. |
| | +14 | Period for re-initiation in seconds. |

| | | |
|---|---|---|
| PARLIST | +15 | Period for re-initiation in 120 Hz clock ticks. |
| | +16 | Zero for temporary schedule entry, non-zero for permanent entry. |

*If these words are negative, the current value in the time-of-day clock is used.

Upon exit from this service, the A register will be set as follows:

| | | |
|---|---|---|
| A = | 0 | Operation performed as requested |
| | 1 | Invalid priority |
| | 2 | Initiation frequency less than 2 seconds requested |
| | 3 | Specified starting time has already passed |

Four FORTRAN calls are available to perform the above functions. These are as follows:

a) To initiate a program only once:
CALL TOADS ('INITIATE', name, priority, param, istat)

b) To initiate a program only once at some time in the future:
CALL TOADS ('FSTART', name, priority, param, dd, hh, mm, ss, tt, perm, istat)

c) To schedule a program for periodic initiation:
CALL TOADS ('PSTART', name, priority, param, rdd, rhh, rmm, rss, rtt, perm, istat)

d) To schedule a program for periodic initiation at some time in the future:
CALL TOADS ('DSTART', name, priority, param, dd, hh, mm, ss, tt, rdd, rhh rmm, rss, rtt, perm, istat)

The parameters are specified as follows:

| | |
|---|---|
| "name" | is a 17-character hollerith string giving the qualifier and areaname of the program to initiate. |
| "priority" | is the priority at which the program will execute. |
| "param" | is the initiation parameter which is passed to the program in the K register when it is initiated. |
| "dd" | is the number of days in the future before the program is to be initiated. |

"hh" is the time in hours (24 hour clock) when the program is to be initiated.

"mm" is the time in minutes when the program is to be initiated.

"ss" is the time in seconds when the program is to be initiated.

"tt" is the time in timer counts (120 per second) when the program is to be initiated.

"rdd" is the number of days between program re-execution.

"rhh" is the number of hours between program re-execution.

"rmm" is the number of minutes between program re-execution.

"rss" is the number of seconds between program re-execution.

"rtt" is the number of timer counts (120 per second) between program re-execution.

"perm" is a flag to indicate whether or not the program is to be permanently scheduled. If a program is permanently scheduled, the scheduling information will be retained between re-boots of the system. To indicate that the program is to be permanently scheduled, perm should be set to a non-zero value.

Examples:

a) To initiate program 1234ABCD*CAT immediately just once, with parameter-2 at priority 50:

```
                TLO     PARLIST
                BLU     $INIT
                 .
                 .
                 .
PARLIST         DATA    T"CATbbbbb"
                DATA    T"1234ABCD"
                DATA    50
                DATA    -2
                DATA    0
                DATA    -1, -1, -1, -1
                DATA    0, 0, 0, 0, 0, 0
```

OR

CALL TOADS ('INITIATE', 17H1234ABCD*CATƀƀƀƀƀ, 50, -2, ISTAT)

b) To initiate program 1234ABCD*CAT at exactly 11 A.M. every day after today, using parameter -2 at priority 50.

```
                    TLO     PARLIST
                    BLU     $INIT
                     .
                     .
                     .
PARLIST             DATA    T"CATƀƀƀƀƀ"
                    DATA    T"1234ABCD"
                    DATA    50
                    DATA    -2
                    DATA    1                tomorrow
                    DATA    11
                    DATA    0
                    DATA    0                11 A.M.
                    DATA    0
                    DATA    1
                    DATA    0, 0, 0, 0
                    DATA    -1               (permanent)
```

OR

CALL TOADS ('DSTART', 17H1234ABCD*CATƀƀƀƀƀ, 50, -2, 1, 11, 0, 0, 0, 1, 0, 0, 0, 0, -1, ISTAT)

## 8-3 $WAKEUP

The Wakeup service is used to wakeup or trigger a sleeping program at a specified time and date and with an optional repeating frequency. In all cases, the program must be sleeping for this operation to have any effect. The service is called as follows:

```
        TLO     PARLIST
        BLU     $WAKEUP
```

where the parameter list, which is exactly as under $INIT above, is defined as follows:

| | |
|---|---|
| PARLIST +0, +1 | 8-character program areaname in truncated ASCII. |
| +2, +3 | 8-character program qualifier in truncated ASCII. |
| +4 | -Not used- |
| +5 | Parameter to be loaded in the program's K register when it returns to execution. |
| +6 | Days in future from current day to wakeup program. |
| +7 | Hour of day to wakeup program. *(0-23) |
| +8 | Minutes of hour to wakeup program. *(0-59) |
| +9 | Second of minute to wakeup program. *(0-59) |
| +10 | Ticks of 120 Hz clock after specified second to wakeup program. *(0-119) |
| +11 | Period for re-wakeup in days. |
| +12 | Period for re-wakeup in hours. |
| +13 | Period for re-wakeup in minutes. |
| +14 | Period for re-wakeup in seconds. |
| +15 | Period for re-wakeup in 120 Hz clock ticks. |
| +16 | Zero for temporary schedule entry, non-zero for permanent entry. |

*If these words are negative, the current value in the time-of-day clock is used.

Upon exit from this service, the A register will be set as follows:

| | |
|---|---|
| A = 0 | Operation performed as requested. |
| 3 | Specified first wakeup program has already passed. |

Four FORTRAN calls are used to access the above functions.

a) To wakeup a program just once:
CALL TOADS ('WAKEUP', name, param, istat)

b) To wakeup a program at some time in the future:
CALL TOADS ('FWAKEUP', name, param, dd, hh, mm, ss, tt, perm, istat)

c) To schedule a program for periodic wakeup:

CALL TOADS ('PWAKEUP', name, param, rdd, rhh, rmm, rss, rtt, perm, istat)

d) To schedule a program for periodic wakeup beginning at some time in the future:

CALL TOADS ('DWAKEUP', name, param, dd, hh, mm, ss, tt, rdd, rhh, rmm, rss, rtt, perm, istat)

The parameters are defined as follows:

"name" is a 17 character hollerith string containing the qualifier and name of the program to be awakened.

"param" is the parameter to be passed to the program in its K register when it returns to execution.

"istat" is an integer variable. Upon return from the TOADS service, it will be set as defined by the A-register returns above.

"dd" is the number of days in the future before the program is to be awakened.

"hh" is the time in hours (24 hour clock) when the program is to be awakened.

"mm" is the time in minutes when the program is to be awakened.

"ss" is the time in seconds when the program is to be awakened.

"tt" is the time in timer counts (120 per second) when the program is to be awakened.

"rdd" is the number of days between program re-execution.

"rhh" is the number of hours between program re-execution.

"rmm" is the number of minutes between program re-execution.

"rss" is the number of seconds between program re-execution.

"rtt" is the number of timer counts (120 per second) between program re-execution.

"perm" is a flag to indicate whether or not the program is to be permanently scheduled. If a program is permanently scheduled, the scheduling information will be retained between re-boots of the system. To indicate that the program is to be permanently scheduled, perm should be set to a non-zero value.

Example:

To wakeup program 1234ABCD*CAT immediately and once every Second thereafter:

```
                TLO     PARLIST
                BLU     $WAKEUP
                 .
                 .
                 .
PARLIST         DATA    T"CATbbbbb"
                DATA    T"1234ABCD"
                DATA    0
                DATA    -2                  parameter
                DATA    0                   first wakeup
                DATA    -1, -1, -1, -1      is now
                DATA    0, 0, 0
                DATA    1                   1 second
                DATA    0
                DATA    0                   (temporary)
```

OR

```
CALL TOADS ('PWAKEUP', 17H1234ABCD*CATbbbbb, -2, 0, 0, 0, 1, 0, 0,
ISTAT)
```

## 8-4 $TERMIN

The Terminate service is used to remove a program entry from the periodic schedule queue (Initiate or Wakeup). The calling sequence is as follows:

```
                TLO     PARLIST
                BLU     $TERMIN
                 .
                 .
                 .
```

where the parameter list contains the program disc areaname as follows:

PARLIST +0, +1 — 8-character program areaname in truncated ASCII.

PARLIST +2 }
+3 } 8-character program qualifier in truncated ASCII.

The entry for the specified program is removed regardless of whether the entry was temporary or permanent.

The corresponding FORTRAN call is as follows:

CALL TOADS ('TERMINATE', name)

Example:

To remove program 1234ABCD*CAT from the permanent wakeup list.

```
                    TLO     PARLIST
                    BLU     $TERMIN
                     .
                     .
                     .
PARLIST             DATA    T"CATƀƀƀƀƀ"
                    DATA    T"1234ABCD"
```

OR

CALL TOADS ('TERMINATE', 17H1234ABCD*CATƀƀƀƀƀ)

## 8-5 $SLEEP

The Sleep service is the means by which a program places itself in the sleep state. It will remain in this state until aborted or triggered by a wakeup request by another program, timer schedule, operator, or external interrupt.

The E register is preserved through the sleep call. The K register will return the Wakeup parameter passed by the program making the wakeup call. The service is called as follows:

```
BLU     $SLEEP
```

When awakened, execution continues at the instruction following the BLU $SLEEP.

The FORTRAN call is as follows:

CALL TOADS ('SLEEP')

## 8-6 $DEXIT

The Delay Exit service is a mechanism by which a program exits from the system for a specified time interval, and is then re-initiated. The calling sequence is as follows:

```
TOK     n
BLU     $DEXIT
```

where n is the number of 120 Hz clock cycles to delay before initiating.

When the call is made, the program is entirely unloaded from core, and all resources allocated are removed. When the specified interval has elapsed, a new copy of the program is loaded from disc just as any other program initiation.

This service may be used only by real-time or monitor programs.

Example:

To exit for 5 seconds and then be reloaded.

```
TOK     600
BLU     $DEXIT
```

## 8-7 $SUSP

The Suspend service enables one program to suspend another program from execution for an indefinite period of time. A release program command from the operator or another program is required to continue execution. The service is called as follows:

```
TLO     PARLIST
BLU     $SUSP
```

where the parameter list is defined as follows:

| | | |
|---|---|---|
| PARLIST | +0<br>+1 | 8-character program areaname in truncated ASCII. |
| | +2<br>+3 | 8-character program qualifier in truncated ASCII. |
| | +4 | Physical device number used to distinguish multiple copies of the same program. This is the diagnostic PDN of the desired program. If zero, the highest priority program with the specified name is suspended. |

Upon return from this service, the A register is set as follows:

| | | |
|---|---|---|
| A = 0 | | Operation performed as requested |
| | negative | Program not found |
| | positive | Program found, but already suspended |

The corresponding FORTRAN call is as follows:

CALL TOADS ('SUSPEND', name, pdn, istat)

where "name" is a 17 character hollerith string giving qualifier and program name to suspend.

"pdn" is the terminal number as discussed above.

"istat" is set to reflect the A register as above.

Example:

To suspend program 1234ABCD*CAT executing from terminal 42:

```
                TLO     PARLIST
                BLU     $SUSP
PARLIST         DATA    T"CATƀƀƀƀƀ"
                DATA    T"1234ABCD"
                DATA    42
```

OR

CALL TOADS ('SUSPEND', 17H1234ABCD*CATƀƀƀƀƀ, 42, ISTAT)

## 8-8 $RSTRT

The Restart service is used to release a suspended program. The program will continue execution at the point it was suspended. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $RSTRT
```

where the parameter list, exactly as defined for the Suspend service, is as follows:

PARLIST +0, +1 } 8-character program areaname in truncated ASCII.

+2, +3 } 8-character program qualifier in truncated ASCII.

PARLIST +4 Physical device number used to distinguish multiple copies of the same program. This is the diagnostic PDN of the desired program. If zero, the highest priority program with the specified name is released.

Upon return from this service, the A register is set as follows:

| | |
|---|---|
| A = 0 | Operation performed as requested |
| negative | Program not found |
| positive | Program found but not suspended |

The FORTRAN call is as follows:

CALL TOADS ('RESTART', name, pdn, istat)

where "name" is a 17 character hollerith string containing qualifier and areaname of the program to be released.

"pdn" is the terminal as discussed above.

"istat" is an integer variable which is set to reflect the A register as returned above.

## 8-9 $QSTAT

The Query Status service is used to examine the status of another executing program. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $QSTAT
```

where the parameter list is defined as follows:

PARLIST +0, +1 — 8-character program areaname in truncated ASCII.

+2, +3 — 8-character program qualifier in truncated ASCII.

+4 Physical device number used to distinguish multiple copies of the same program. This is the diagnostic PDN of the desired program. If zero, the highest priority program with the specified name is tested.

Upon return from this service, the A register is used to return the status of the examined program as follows:

| | |
|---|---|
| A = -1 | Program not found |
| +1 | Program is suspended |
| +2 | Program is aborting |
| +3 | Program is loading |
| +4 | Program is exiting |
| +5 | Program is sleeping |
| +6 | Program is waiting for I/O transfer |
| +7 | Program is in execution |

In the event of more than one of the above conditions being true, the smallest number possible will be returned.

The FORTRAN call is as follows:

CALL TOADS ('STATUS', name, pdn, istat)

where "name" is a 17 character hollerith string containing the program qualifier and areaname.

"pdn" is the optional terminal number or zero as discussed above.

"istat" returns the status of the specified program as shown above for the A register.

Example:

To examine the status of program 1234ABCD*XYZ.

```
                TLO     PARLIST
                BLU     $QSTAT
                 .
                 .
                 .
PARLIST         DATA    T"XYZƀƀƀƀƀ"
                DATA    T"1234ABCD"
                DATA    0
```

OR

CALL TOADS ('STATUS', 17H1234ABCD*CATƀƀƀƀƀ, 40, ISTAT)

## 8-10 $PABORT

The Program Abort service enables one program to remove another program from the system. The calling sequence is as follows:

```
TLO    PARLIST
BLU    $PABORT
```

where the parameter list, which is identical to that described for QSTAT service is as follows:

| | | |
|---|---|---|
| PARLIST +0, +1 | | 8-character program areaname in truncated ASCII. |
| +2, +3 | | 8-character program qualifier in truncated ASCII. |
| +4 | | Physical device number used to distinguish multiple copies of the same program. |

Upon return from this service, the A register is used to return the status of the call as follows:

| | |
|---|---|
| A = 0 | Operation performed as specified |
| negative | Program not found |
| positive | Program found and already aborting |

The corresponding FORTRAN call is as follows:

CALL TOADS ('ABORT', name, pdn, istat)

where "name" is a 17 character hollerith string giving the program qualifier and areaname to be aborted.

"pdn" is the optional terminal number or zero as discussed above.

"istat" is an integer variable which is set to reflect the status of the operation as returned in the A register above.

## 8-11 $PRIOR

The change priority service allows a program to change its own priority or the priority of another program. The calling sequence is as follows:

```
TLO    PARLIST
BLU    $PRIOR
```

where the parameter list is defined as follows:

| | | |
|---|---|---|
| PARLIST | +0 +1 | 8-character program areaname in truncated ASCII whose priority is to be changed. If both words are zero, the calling program's priority is changed. |
| | +2 +3 | 8-character program qualifier in truncated ASCII. |
| | +4 | Physical device number used to distinguish multiple copies of the same program. If zero, the highest priority program of the specified name is changed. |
| | +5 | New priority value (0-63). |

Upon return from this service, the A register is set negative if the program is not found, and non-negative if found.

The FORTRAN call is as follows:

CALL TOADS ('PRIORITY', name, pdn, new-priority, istat)

where "name" is a 17 character hollerith string containing the program qualifier and areaname which is to be changed.

"pdn" is the optional terminal number or zero as discussed above.

"new-priority" is the new priority value.

"istat" is an integer variable which is set to reflect the status of the call as returned in the A register.

Example:

To change the priority at the calling program to 40.

```
                    TLO     PARLIST
                    BLU     $PRIOR
                     .
                     .
                     .
PARLIST             DATA    0, 0
                    BLOK    23
                    DATA    40
```

(Assume calling program is 1234ABCD*CAT)

CALL TOADS ('PRIORITY', 17H1234ABCD*CATbbbbb, 0, 40, ISTAT)

## 8-12 $CONNECT

The Connect service is used to connect up a real-time program to an unused external interrupt, and optionally arm and enable that interrupt. When the interrupt occurs, the program will be awakened from the sleep state. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $CONNECT
```

where the parameter list is defined as follows:

| | |
|---|---|
| PARLIST +0, +1 | 8-character program areaname in truncated ASCII. |
| +2, +3 | 8-character program qualifier in truncated ASCII. |
| +4 | External interrupt number. Group 1 is numbered 0-23, with group 2 numbered 24-47. |
| +5 | Parameter to be passed to program when awakened in K register. |
| +6 | -1: temporary, arm and enable interrupt immediately. |
| | 0: temporary, do not restore on reload of system, do not arm and enable interrupt. |
| | 1: permanent, restore on reboot. |
| | 2: permanent, restore on reboot and arm and enable interrupt. |

Upon exit from the service, the A register will be set as follows:

| | |
|---|---|
| A = 0 | Operator performed as requested |
| 1 | Interrupt level does not exist |
| 2 | Interrupt level is used by standard system I/O device |
| 3 | Interrupt level is already connected to a program |

The FORTRAN calling sequence is as follows:

CALL TOADS ('CONNECT', name, interrupt, parameter, perm, istat) where "name" is a 17 character hollerith string containing the program name and qualifier, interrupt is the interrupt level to be connected to (0-47).

"parameter" is the value to be loaded in the programs K register when awakened.

"perm" is the flag set to indicate the type of operation.

-1: temporary, do not reload on reboot, arm and enable interrupt immediately.

0: temporary, do not arm and enable interrupt.

1: permanent, restore entry on reload of VULCAN, but do not arm and enable interrupt.

2: permanent, and arm and enable interrupt immediately and on each system load.

"istat" is an integer variable which is set to the contents of the A register following the operation as discussed above.

Examples:

a) To connect program 1234ABCD*CAT up to interrupt level 5 on group 2, with parameter -2:

```
                TLO     PARLIST
                BLU     $CONNECT
                 .
                 .
                 .
PARLIST         DATA    T"CATbbbbb"
                DATA    T"1234ABCD"
                DATA    29
                DATA    -2
                DATA    0
```

OR

CALL TOADS ('CONNECT', 17H1234ABCD*CATbbbbb, 29, -2, 0, ISTAT)

## 8-13 $DISCONNECT

The Disconnect service is used to remove the program connected to the specified external interrupt. The interrupt is also disarmed by this call. The calling sequence is as follows:

```
TOK     interrupt
BLU     $DISCONNECT
```

where "interrupt" is the external interrupt number (0-23 for group 1 and 24-47 for group 2) from which the program is to be disconnected.

Upon return from this service, the A register is set as follows:

| | |
|---|---|
| A = 0 | Operation performed as requested. |
| 1 | Invalid interrupt designation. |
| 2 | Specified interrupt level does not have a program connected to it. |

The FORTRAN calling sequence is as follows:

CALL TOADS ('DISCON', interrupt, istat)

where "interrupt" is the interrupt level to be disconnected as discussed above.

"istat" is an integer variable which is set to the contents of the A register as discussed above following the operation.

Example:

To disconnect the program from interrupt level 5 on group 2:

```
TOK     29
BLU     $DISCONNECT
```

OR

CALL TOADS ('DISCON', 29, ISTAT)

## 8-14 $ENABLE

The Enable service is used to arm and enable a specified interrupt level. A program must be connected to the level as discussed under paragraph 8-12. The calling sequence is as follows:

```
TOK     interrupt
BLU     $ENABLE
```

where "interrupt" is the external interrupt number. Group 1 levels are numbered 0-23 and group 2, 24-47.

Upon return from the service the A register is set as follows:

| | |
|---|---|
| A = 0 | Operation performed as requested. |
| 1 | Interrupt invalid or does not exist. |
| 2 | No program is connected to specified interrupt level. |

The FORTRAN calling sequence is as follows:

CALL TOADS ('ENABLE', interrupt, istat)

where "interrupt" is the interrupt level as defined above.

"istat" is an integer variable which is set to the contents of the A register given above upon completion of the operation.

Example:

To enable interrupt level 5 on group 2:

```
TOK     29
BLU     $ENABLE
```

OR

CALL TOADS ('ENABLE', 29, ISTAT)

## 8-15 $INHIBIT

The Inhibit service is used to disarm and inhibit a specific external interrupt level. A program must be connected to the interrupt level as discussed in paragraph 8-12. The calling sequence is as follows:

```
TOK     interrupt
BLU     $INHIBIT
```

where "interrupt" is the external interrupt level designation. Group 1 levels are numbered 0-23 and group 2 levels are 24-47.

Upon return from this service, the A register is set to reflect the sets of the operation as follows:

A = 0 Operation performed as requested.

| | | |
|---|---|---|
| A = 1 | | Interrupt invalid or does not exist. |
| 2 | | No program is connected to the specified interrupt level. |

The FORTRAN calling sequence is as follows:

```
CALL TOADS ('INHIB', interrupt, istat)
```

where "interrupt" is the interrupt level as defined above.

"istat" is an integer variable which is set to the contents of the A register given above upon completion of the system call.

Example:

To disable level 5 on group 2:

```
TOK    29
BLU    $INHIBIT
```

OR

```
CALL TOADS ('INHIB', 29, ISTAT)
```

# SECTION IX
# RESOURCE SERVICES

## 9-1 GENERAL

The Resource services listed below are not normally required by user programs. They are listed below for reference only.

## 9-2 $TAPEOP

The Tape Option service is used to obtain the default system tape option word. This word is set by the system generation system (GENASYS) and is used on resource commands by Job Control when no specific tape option information is included. The service is called as follows:

```
BLU     $TAPEOP
```

Upon return, the E register is set to the default tape option word, as shown in Table 9-1. The A register is set to the default tape type, as shown below:

| | | | |
|---|---|---|---|
| A: | Bit 0 = | 1: | 9 track |
| | | 0: | 7 track |
| | Bit 1 = | 1: | High Speed (greater than or equal to 75 IPS) |
| | | 0: | Low Speed (less than 75 IPS) |

## 9-3 RESOURCE SERVICE

The Resource Service is used for resource allocation. It will allocate four types of resources. These are:

1) Disc Packs
2) Mag Tapes
3) Physical Devices
4) High Speed Memory

The calling sequence is as follows:

```
TLO     PARLIST
BLU     $RESORC
DATA    n
```

TABLE 9-1

TAPE OPTION WORD FORMAT

| | | | | | | |
|---|---|---|---|---|---|---|
| Bits 16 and 14: | 0 = odd parity with no conversion. | | | | | |
| | 1 = even parity with BCD/EBCDIC conversion. | | | | | |
| Bits 13-12: | Density: | | | | | |
| | | 13 | 12 | | | |
| | | 0 | 0 | = | 200 | BPI |
| | | 0 | 1 | = | 556 | BPI |
| | | 1 | 0 | = | 800 | BPI |
| | | 1 | 1 | = | 1600 | BPI |
| Bits 11-10: | Characters per word: | | | | | |
| | | 11 | 10 | | | |
| | | 0 | 0 | = | 1 | CPW |
| | | 0 | 1 | = | 2 | CPW |
| | | 1 | 0 | = | 3 | CPW |
| | | 1 | 1 | = | 4 | CPW |
| All other bits are zero. | | | | | | |

where the parameter list is as defined below, and "n" is the function as follows:

n = 1 — Allocate resource. This code is used to pass an initial resource request list. This is the only function needed for non-interactive programs.

n = 2 — Test allocation. Used only by interactive programs to test the status of the specified resource allocation.

n = 3 — Wait for allocation. Used only by interactive programs to wait for completion of specified allocations. (Wait is implicit in calls by Real-Time programs, and Control Points may never wait.)

Upon return from the call, the A register is used to reflect the status of the request as follows:

| | |
|---|---|
| A = 0 | Resources allocated as specified. |
| 1 | For Control Points: Requested resources not available. For interactive Terminals: One or more resources not yet allocated; not returned for Real-Time programs. |
| 2 | Non-existent physical device requested. |
| 3 | Non-allocatable physical device requested. |
| 4 | Disc I/O error on *V:PACK area. |
| 5 | Disc pack non-existent to this user. |
| 6 | Resource request for this device already entered. |
| 7 | No functional disc drives available for specified pack. |
| 8 | No functional tape drives of requested type are available. |
| 9 | Non-existent High Speed memory requested. |
| 10 | More High Speed memory requested than is available. |

Real-Time programs need only make an initial (n=1) call, and they will be suspended until the requests can be satisfied. Control Points will never wait; if one or more resources cannot be immediately allocated, all will be returned, and Status returned to calling program. Interactive requests are queued while program execution continues. In order to access the requested device, a "Wait" request (n=3) must be made at some time following the initial (n=1) call.

The format of the parameter list is a series of resource entries, terminated by a zero word. The size and content of the resource entry is dependent on the device type, as follows:

a) Disc Pack: One word entry, having pack number in Bits 7-0, and other bits 0.

b) Physical Device: One word entry, having Bits 7-0 set to physical device, Bits 15-8 set to Logical File Number which is to be assigned to the device when allocated, and Bits 23-16 set to a 2 (Bit 17 on only).

c) Magnetic Tape Drive: 5 word entry:
Word 1 has Logical File Number which is to be assigned to the tape drive in Bits 15-8, and Bits 23-16 = 1 (Bit 16 on).

Words 2 and 3 have tape name in ASCII (6 characters). If both are zero, scratch is assumed.

Word 4 is set to the tape option word (see Table 9-1) with Bit 23 set if write access is required (Write ring to be inserted).

Word 5 has either two physical device numbers of possible tape drives (if Bit 23 set) in Bits 7-0 and 15-8, (use only 7-0 if only one Specific PDN) or, a tape type specification in Bits 7-0. These are:

Bit 0: 1 = 9 track
0 = 7 track

Bit 1: 1 = High Speed
0 = Low Speed

d) High Speed Memory Request: One word entry with Bits 7-0 having number of pages required, Bits 23-16 = 4 (Bit 18 on only), and Bits 15-8 as follows:

0 = Regular request

1 = Contiguous pages required

>1 = Contiguous pages required starting at physical memory page number specified by this field.

Example:

For an interactive program to assign LFN 8 to a 9-track Low speed drive, to read tape "ABCD" which is 800BPI, 3 characters per word ASCII, and wait for the tape, the following call is required:

```
TLO     PARLIST
BLU     $RESORC
DATA    1
BOZ     * + 3
COB     1
BNZ     error
TLO     PARLIST
```

```
          BLU     $RESORC
          DATA    3
          TNK     '1013            open 8
          BLU     $I/O
           .
           .
           .
PARLIST   DATA    B16B11           LFN 8 (Bit 11) and Bit 16
          DATA    "ABCDbb"
          DATA    B13B11           tape option
          DATA    1                Low speed, 9 track
          DATA    0                List termination
```

# SECTION X
# MISCELLANEOUS SERVICES

## 10-1 $PTYPE

The Program Type service allows the calling program to determine if it is a control point, interactive, or real-time program. The calling sequence is as follows:

```
BLU     $PTYPE
```

Upon return, the condition code and E register are set as follows:

| | | |
|---|---|---|
| E = | negative | Real-time program |
| | zero | Interactive program |
| | positive | Control Point program |

## 10-2 $SPOOL

The Spool service is used to place a disc area on the spool-out queue for a particular device. The calling sequence is as follows:

```
TLO     PARLIST
BLU     $SPOOL
```

where the contents of the parameter list are as follows:

| | |
|---|---|
| PARLIST +0, +1 | 8-character disc areaname in truncated ASCII |
| +2, +3 | 8-character qualifier in truncated ASCII |
| +4 | Physical Device Number to which the disc area is to be sent |

Upon return from this service, the A register is set as follows:

| | |
|---|---|
| A = 0 | Operation performed as requested |
| 1 | Requested physical device does not exist |

Example:

To print disc area 1234ABCD*XYZ on printer 6:

```
                TLO     PARLIST
                BLU     $SPOOL
                 .
                 .
                 .
PARLIST         DATA    T"XYZƀƀƀƀƀ"
                DATA    T"1234ABCD"
                DATA    6
```

## 10-3 $IJOB

The Insert Job service is used to place a disc area containing a control point job on the jobs-to-be-run queue. The service is called as follows:

```
TLO     PARLIST
BLU     $IJOB
```

where the contents of the parameter list are defined as follows:

| | |
|---|---|
| PARLIST +0, +1 | 8-character disc areaname |
| +2, +3 | 8-character qualifier in truncated ASCII |
| +4 | If zero, run the job as soon as possible, if non-zero a 20 second delay is placed on the job entry, to delay execution. This is used by Job Control to hold the execution of jobs requiring resources which are not yet available. |

There are no return codes from the $IJOB service.

## 10-4 $USERNO

The User number service is used to access and validate user numbers. It is used in two different ways: a) Job Control uses it to handle sign-on; and b) any user program may use it to return the name of the current user.

The service is called as follows:

```
TLO     PARLIST
BLU     $USERNO
```

where the parameter list is a buffer in which the user name is left, as follows:

| | |
|---|---|
| PARLIST +0, +1 | User number to be signed on, for Job Control only. |
| +2, +3 | Qualifier to be signed on, for Job Control only. |
| +4, +5, +6, +7 | 12-character user name returned here in ASCII, blank filled. |

Example:

To return the current user name into BUF:

```
            TLO     BUF-4
            BLU     $USERNO
             .
             .
             .
BUF         BLOK    4
```

## 10-5 $OPCOM

The Opcom service may be used to pass an Operator command to be processed. The valid commands accepted from user programs, and their format, is discussed in Chapter F.

Returns from the command will consist of A = 0 for valid commands, and A = Opcom error number for Opcom errors.

Commands producing output information will write the lines to Logical File Number 3.

The calling sequence is as follows:

```
TLO     PARLIST
BLU     $OPCOM
```

where PARLIST is a 24-word buffer containing the requested command in ASCII, 3-characters per word.

Example:

To query the status of terminal 42 and return the output to LFN 3:

```
             TLO     BUF
             BLU     $OPCOM
              .
              .
              .
BUF          DATA    "QPb42b"
             RDAT    22 ("bbb")
```

## 10-6 $PACK

The Pack service returns the system default work pack number, along with the Pack number of the disc containing the swapping disc area. The calling sequence is as follows:

```
             BLU     $PACK
```

The system work pack number (default for disc area generation) is returned in the E register. The system swapping pack number is returned in the A register.

# CHAPTER E

# INPUT/OUTPUT

# SECTION I
# INPUT OUTPUT STRUCTURE

## 1-1 GENERAL

Input/Output under VULCAN is performed via system services that overlap I/O transfer operations with program execution. If a program must wait for an operation to complete, it is placed in a "wait state" and another program is executed. Access to these services is made through use of logical assignments. A user program can "assign" a number called a Logical File Number (LFN) to a particular physical device or disc area via services described below. He then transacts his I/O via this logical file number.

## 1-2 DISC AREAS

Data files are stored on disc storage units in "disc areas". One or more disc areas are stored on disc packs. A disc area may not overlap disc packs. Disc areas stored on permanent disc packs such as fixed head discs or fixed portions of cartridge discs are always available and may be accessed via logical assignments. The system master pack and any removable pack which has been marked as permanently resident by the operator, are also always available. To reference disc areas on removable packs, the disc pack must first be resourced. This causes the system to allocate a disc drive for the pack when it becomes available. The computer operator then mounts the pack and all of the disc areas on it are available for access.

### 1-2.1 Disc Area Names

Disc Area Names consist of an 8-character qualifier and an 8-character areaname. These names are normally handled internally in truncated ASCII notation (6 bits per character). Externally they appear as QUALIFIER*AREANAME.

The Qualifier consists of a four digit account and a four character identifier. For example:

1234ABCD is under account 1234 with the identifier ABCD

### 1-2.2 Disc Area Assignments

Disc Area Assignments are made by specifying a logical file number and disc area name. This is discussed in Paragraph 2-3. When this assignment is made, first the qualifier is located in the Master Disc Directory (MDD). This entry then points to a Qualifier Disc Directory (QDD), which contains the entry for the disc area. Thus a normal assignment takes two disc accesses. However, once a particular qualifier has been located, its MDD entry remains in core for that user until he terminates his program or job stream. Thus successive assignments for a particular qualifier require only a single access.

Additionally, a disc area can be specified as having a "core directory". In this case, the entire entry (both MDD and QDD) remains in core at all times, and no disc accesses are required to assign the disc area. However, use of this feature should be minimized since system memory space is required for each such core directory entry.

### 1-2.3 Disc Area Catagories

Three types of disc areas exist under VULCAN. The first of these is unblocked disc areas. In unblocked disc I/O all transfers begin on disc sector boundaries and no data compression takes place. For example, writing symbolic 27-word records to an unblocked disc area will cause one record followed by 85 words of zero to be written to each sector of the disc.

The second type of disc area is a blocked area. I/O transfers to blocked areas are done via the system blocking handler. Multiple records are compressed and packed into a sector. In addition, in symbolic records, strings of blanks are compressed into a single blank-count character. This allows for a much higher packing density per disc sector. Typical packing might be 8-10 card images per 112-word disc sector.

The third catagory of disc areas is a random disc area. Random areas are accessed exactly as unblocked areas discussed above. The single exception is that multiple programs may simultaneously read or write the disc area, which is not possible with unblocked or blocked disc areas. With block or unblocked areas, if any program is writing to the area, no other program is allowed to access it. Similarly, if any program or programs are reading the area, no program may begin writing to it.

## 1-3 PHYSICAL DEVICES

Each Physical Device excluding disc units is assigned a unique Physical Device Number (PDN) when the system is generated. These numbers are then used to refer to the device during normal VULCAN operation.

Physical Device I/O is dependent on the device type, as discussed in the following paragraphs.

### 1-3.1 Magnetic Tape Files

Magnetic Tape Drives are an allocatable system resource under VULCAN, and as such must be allocated via a resource request. These may be done via Job Control statements or via requests cataloged into real-time programs.

To resource a magnetic tape, the user must supply a logical file number which will be assigned to the tape drive by the system, a tape name, and various tape options. When the system finds a free tape drive of the correct type, the operator is told to mount the specific tape. When the tape is mounted, the assignment is made and the user program may access the tape via the logical file number. If additional LFN's assigned to the tape are required, the indirect assign feature discussed below may be used.

All tapes are treated as unlabeled under VULCAN. The tape name described above is an external name only. Automatic conversion between BCD and ASCII for 7-track drives, and EBCDIC and ASCII for 9-track units is available as an option when the resource request is made.

1-3.2 Card Readers

All card reader input is automatically spooled to the disc under VULCAN. No program is allowed to read directly from the card reader. Two types of input are available from card readers: control point jobs and program decks, containing binary or symbolic data.

1-3.2.1 Job Input

Jobs to be run at VULCAN Control Points may be submitted from card readers directly. The first card of the deck must be a $JOB card as discussed in Paragraph 8-2.

1-3.2.2 Data Input

Data Files destined for user programs may be read in from card readers by placing a $DATA card (Paragraph 4-6) on the front of the deck, which, identifies the user and program to which it is destined. This disc area will be automatically generated on the work disc pack. At a later time when the user program makes an assignment to the physical device number which represents the card reader, his request is modified such that the assignment is made to the disc area containing the card images.

1-3.3 Line Printers, Plotters, Card Punches and Paper Tape Punches

Under VULCAN these output devices are treated as spooled devices only. All I/O is done via system spool files. Direct user I/O with these devices is not allowed.

When a user program assigns a logical file number to a physical device which is a line printer, plotter, card punch, or paper tape punch, the system creates a spool disc area for the device thus enabling the device to always be "available". The LFN is assigned to the disc area. When de-assigned, the spool disc area is queued for output to the appropriate device.

1-3.4 Teletypes and CRT's

All teletypes and CRT's, whether local or over modems, are treated as interactive terminals under VULCAN. The only exception is that the operator CRT is not usable as an interactive terminal.

Terminals may be accessed in three different ways: As an interactive terminal, logging on to the terminal causes it to be allocated to that user. Any normal I/O function may be performed to it, and any logical file number may be assigned to the terminal. When the user signs off, the terminal again becomes free.

The second way to access a terminal is to allocate it as a resource. When the terminal becomes free it is allocated to the highest priority program requesting it as a resource. Once the resource has been satisfied, the terminal may be referenced via standard I/O calls using the logical file number specified on the resource request.

Additionally, output may be spooled to terminals. This is done by requesting a spooled assign when assigning a logical file number to the physical device. The logical file is then assigned to a spool disc area, which is spooled to the terminal when the terminal becomes free and the spool area is closed.

On Model 2200 teletypes, (TI 733, "Silent 700") if the ASR cassette units are available, they are treated as separate devices within the teletype physical device. Cassette one is referred to as "T1" and Cassette two as "T2". When making assignments to a particular cassette unit, the physical device number precedes the Tn. For instance, cassette 2 of physical device 43 is referred to as 43T2.

On other teletypes the paper tape reader/punch shares the same physical device number as the keyboard/printer.

1-3.5 Paper Tape Readers

High Speed Paper Tape readers are considered allocatable devices under VULCAN, and as such, must be resourced before being accessed. This is done by providing a Logical File Number, and the Physical Device Number of the Tape Reader desired. When the device becomes free, the LFN is assigned and program I/O may be transacted.

## 1-4 INDIRECT ASSIGNMENTS

During certain processing, it may be necessary to assign a Logical File Number (LFN) to the same place that another LFN is currently assigned so that they follow each other along in the file. For instance, it may be necessary to have two LFN's referencing the same job stream file, one to read commands and another to read data. This may be accomplished via the Indirect Assign.

To make an indirect assign, the user supplies the usual logical file number being assigned, and additionally specifies another LFN to which the initial LFN's is being assigned. For example, if LFN 4 is assigned to disc area CAT, then Logical File 8 may be assigned to the same place in CAT by assigning 8 to LFN 4.

There are two types of indirect assigns. The first is the "follows" assign. In this case, the indirectly assigned LFN follows its object LFN wherever it is assigned even if de-assigned. Given the above example, if 8 were "follows" assigned to 4, and 4 re-assigned to disc area DOG, then LFN 8 would also be assigned to DOG.

The second type is the "permanent" indirect assign. In this case, the indirectly assigned LFN stays assigned to the object physical device or disc area name regardless of any reassigns of the object LFN. Given the above example, if 8 were "permanently" assigned to 4, and 4 re-assigned to disc area DOG, the LFN 8 would still remain assigned to disc area CAT.

## 1-5 FILE INSERTION FEATURE ($ADD)

A special disc area insertion feature is available for Control Point or Interactive Terminal programs. This allows a disc area to be effectively inserted in an input stream without modifying the actual input stream being read.

This is accomplished through use of the reserved command $ADD. "$ADD" must be the first four columns of the record. The format is as follows:

$ADD disc-area-name

If the Job Control Mode $ADD is off (Chapter B), this record is treated just as any other symbolic record. No special system action is done. This allows $ADD images to be set into disc areas initially as needed.

If Job Control Mode $ADD is on, then whenever a record of this type is read, system intervention automatically transfers control to read the current input request from the first record of the "disc-area-name" on the $ADD card. Subsequent reads will read from the Add Disc Area until another nested $ADD card is encountered, or an end-of-file or end-of-disc-area is detected, at which point control reverts to the original input stream.

The $ADD mode will work on any logical file. For example, if a $ADD were entered from a terminal and the ADD mode were enabled, then it would appear to the calling program as if the entire disc area were typed in at that point. No more than 20 $ADD cards may be concurrently in use by any program.

The following example illustrates the operation of the $ADD command.

a) Assume disc area *CAT contains:
   LINE 1
   LINE 2
   LINE 3
   end-of-file

b) disc area *DOG contains:
   LINE A
   $ADD *CAT
   LINE B
   end-of-file

c) Mode $ADD = ON

d) Then reading the following stream will yield the listed records:

| Input Stream | Records transferred |
|---|---|
| LINE X | LINE X |
| $ADD *DOG | LINE A |
| LINE Z | LINE 1 |
| | LINE 2 |
| | LINE 3 |
| | LINE B |
| | LINE Z |

# SECTION II
# LOGICAL ASSIGNMENT SERVICES

## 2-1 GENERAL

The services discussed in the following paragraphs are designed to allow the user to easily and effectively control logical file number assignments under VULCAN.

## 2-2 LOGICAL FILE NUMBERS

Logical File Numbers are of the range 0-255 (8 bits). Table 2-1 reflects the logical file number usage.

Certain logical file numbers are used as default LFN's for certain functions. These are listed in Table 2-2.

TABLE 2-1
LOGICAL FILE NUMBER USAGE

| LFN | USAGE |
|---|---|
| 0 through 99 | Regular LFN's. May be used anywhere. Use caution when using a number that may be a default for some operation (See Table 2-2). |
| 100 through 199 | Fortran Translate Area. May be used directly but FORTRAN execution routines may also use them. |
| 200 through 219 | Temporary LFN's used and destroyed by system processors. |
| 220 through 255 | May be used only by Job Control. |

TABLE 2-2

STANDARD LFN's

| LFN | USAGE |
|---|---|
| 0 | Job Stream. Used to read Job Control commands for Control Points & Terminals. |
| 3 | Diagnostic Output. Assigned to List Output for Control Points and to terminal for interactive terminal users. This should normally never be re-assigned, as doing so may cause the terminal or job to be terminated prematurely. |
| 4 | Used by KEEP/FETCH routines. Must be assigned to device or disc area to KEEP to or FETCH from. (Chapter B) |
| 5 | Binary Output. Used for Link Code from Assembler, Fortran Compiler, etc. Default assignment is to disc area LR. |
| 6 | List Output- Used by Compilers, Assembler to produce listings. Default assignment for Control Points is to List Output Area (LFN 3), for interactive to disc area LO. |
| 7 | Source Input. Compilers, Assembler read source from here. Default assignment is indirect to LFN 0 (to read from Job Stream). |
| 8 | Scratch Output. Used for intermediate output by Assembler. Default assignment is to work disc area W1. |
| 9 | Dump Output. This is the LFN to which the $DLOAD and $DBOOT statements write their dump output. |
| 10 | Library File input. This is the LFN from which the Library File Edit Commands read their modules to add, replace, etc. Default disc area is LR. |

## 2-3 ASSIGN SERVICE

The system service $ASSIGN is used to assign logical file numbers to physical devices, disc areas, or other LFN's. It is accessible from Job Control Statements or from user programs via the following assembly language sequence:

```
                      TLO    PARLIST
                      BLU    $ASSIGN
                        .
                        .
                        .
PARLIST               DATA   function code
                      DATA   logical file number
                      DATA   2-word areaname (8 characters-truncated
                             ASCII) or physical device number or logical
                             file number for indirect assigns
                      DATA   2-word qualifier (8 characters-truncated ASCII)
                             may be used if areaname supplied above
```

The function code word is defined as follows:

Bits 7-0 = 0 — Assignment to Disc Area. The areaname is taken from words PARLIST + 2,3. If bit 23 of the function code word is zero, the sign-on qualifier for the user is used. If bit 23 is set, the qualifier is taken from words PARLIST + 4,5.

If this qualifier field is all zero, then the system qualifier 0000SYST is used. If the qualifier field is all truncated ASCII blanks, then the default sign-on qualifier for the user is used.

If the assignment is made to a batch/interactive terminal work disc area which does not exist, it will be dynamically generated by the Assign Logic.

1 — Spool Assign. Used to assign output to spool areas destined for interactive terminals. The terminal need not be allocated as a resource in this case. A spool disc area will be dynamically generated.

2 — Physical Device Assign. Used to make an assignment to the physical device specified in word PARLIST +2. Word PARLIST + 3, 4, 5, are not used. If the physical device is an output only device, a spool disc area is generated and the assignment is made to it. If the physical device is a card reader, the data input queue is searched for the disc area destined for this program.

| | | |
|---|---|---|
| Bit 7-0 = | 2 | If the specified physical device is a Model 2200 teletype (TI 733), then bits 9 and 10 of the function code are used to assign to cassettes one and two respectively. Both bits should not be on together. |
| | 3 | Indirect (Logical File) Assign. If bit 8 of the function code word is off (0) then this is a "follows" indirect assign (see paragraph 1-4). If bit 8 is set (1) then this is a permanent indirect assign. The object LFN is specified in word PARLIST + 2. Words PARLIST + 3, 4, 5 are not used. |
| | Any other Value | Invalid call - an abort will result. |

Upon return from the Assign call, the A register will be set to reflect the status of the requested assign, and the condition is set to match the A register. The A register values are as follows:

| | | |
|---|---|---|
| A = | 0 | Valid assignment. |
| | 1 | Referenced object does not exist. For disc area assigns: the disc area is either not there or is not accessible to this user.<br><br>For Card Reader assigns: no input data file for this program.<br><br>For other PDN assigns: referenced device does not exist. |
| | 3 | Physical device not ready. |
| | 4 | Spool assign allowed only to interactive devices. |
| | 5 | Resource not allocated.<br><br>For disc area assigns: required disc pack is not resourced or mounted.<br><br>For PDN assigns: requested device has not been resourced. |

| | | |
|---|---|---|
| A = | 6 | User does not have required access bit to access this physical device. |
| | 7 | Batch/interactive work file cannot be generated, and does not exist for this user. |
| | 8 | Spool pack full or not mounted.<br>Cannot create spool area for spool assign.<br>Cannot create spool area for output physical device assign. |

Examples of Assigns:

a) To assign LFN 8 to physical device 6:

```
            TLO    PARLIST
            BLU    $ASSIGN
             .
             .
             .
PARLIST     DATA   2
            DATA   8
            DATA   6
```

b) To assign LFN 8 to disc area 1234ABCD*CAT:

```
            TLO    PARLIST
            BLU    $ASSIGN
             .
             .
             .
PARLIST     DATA   B23
            DATA   8
            DATA   T"CATƀƀƀƀƀ"
            DATA   T"1234ABCD"
```

c) To assign LFN 8 to cassette two of teletype PDN 23:

```
            TLO    PARLIST
            BLU    $ASSIGN
             .
             .
             .
PARLIST     DATA   '2002
            DATA   8
            DATA   23
```

## 2-4 LFN SERVICE

This service returns the type of assignment made on a specific logical file number. It is called as follows:

| | |
|---|---|
| TOK | Logical File Number |
| BLU | $LFN |

Upon return, the A register is set to reflect the type of assignment as follows:

| | | |
|---|---|---|
| A = | -1 | LFN assigned to disc area. |
| | 0 | LFN is unassigned. |
| | positive | LFN is assigned to physical device. In this case (A) is set to describe the physical device type exactly as returned by the PDN Service described in the following paragraph. |

Note that the condition register is NOT set upon return from this service.

## 2-5 PDN SERVICE

This service returns information about a physical device. It is called as follows:

| | |
|---|---|
| TOK | Physical Device Number |
| BLU | $PDN |

Upon return from this call, the A register is set as follows:

| | | |
|---|---|---|
| A = | 0 | Specified physical device does not exist. |
| | 1 | Specified physical device was zero, or call made to LFN service with logical file number that was assigned to 0. |
| | >1 | Physical device type: |

Bit 22 set if device is allocatable to a program.
Bit 21 set if device is an interactive terminal.
Bit 20 set if device is input only (i.e. card reader).
Bit 19 set if device is output only (i.e. line printer).
Bit 18 set if device requires resource call to be used.
Bit 17 set if device is connected via CBC channel.
Bits 15-8 contain model information based on particular device type. (See Table 2-3)

A = >1 Bits 7-0 contain device type as listed in Table 2-3.

Note that the condition register is NOT set upon return from this service.

TABLE 2-3
DEVICE TYPES

| Type | Device Name | Information in Bits 15-8 of PDN or LFN Service Call. |
|---|---|---|
| 2 | Teletype | |
| 3 | CRT | |
| 4 | Paper Tape Reader | |
| 5 | Paper Tape Punch | |
| 6 | Line Printer | Model: 1 = Analex |
| | | 2 = Data Products |
| | | 3 = Data Printer |
| | | 4 = Potter |
| | | 5 = CDC |
| | | 6 = Tally |
| | | 10 = Versatec |
| 7 | Card Reader | |
| 8 | Card Punch | |
| 9 | Mag Tape Drive | Bit: 8 = 9-track |
| | | 9 = Hi-speed |
| | | 10 = 200BPI |
| | | 11 = 556BPI |
| | | 12 = 800BPI |
| | | 13 = 1600BPI |
| 10 | Synchronous Interface for RJE to "Host" Computer | |

## 2-6 LFNAME SERVICE

The LFNAME service is used to find the name of the disc area to which a logical file number is assigned. It may also be used to find the physical device number if the LFN is assigned to a physical device. It is called as follows:

```
                    TLO     PARLIST
                    BLU     $LFNAME


PARLIST             DATA    logical file number
                    BLOK    3
```

Upon exit, the A register is set to reflect the type of assignment:

| | | |
|---|---|---|
| A = | -1 | LFN not assigned. |
| | 0 | LFN assigned to disc area. |
| | +1 | LFN assigned to physical device. |

The contents of the "PARLIST" is also modified depending on the assignment:

If A register was positive (PDN assign), then on exit, "PARLIST" is set to the physical device number.

If register A was 0 (Disc area assign) then on exit "PARLIST" is set to the disc area as follows:

PARLIST+0,1 = areaname

PARLIST+2,3 = Qualifier

Programming example:

To find out what LFN 8 is currently assigned to, and assign LFN 9 to that same device or disc area independently of LFN 8 (to work in same file position, use indirect assign feature).

```
                    TOA     8
                    TAM     PAR2
                    TLO     PAR2
                    BLU     $LFNAME
                    TME     =B23
                    CZA
                    BON     not assigned
```

```
          BOZ    * + 2
          TOE    2
          TEM    PAR1
          TLO    PAR1
          BLU    $ASSIGN

PAR1      DAC    0
          DATA   9          LFN for Assign
PAR2      DAC    0
          BLOK   3
```

# SECTION III
# I/O SERVICES

## 3-1 GENERAL

This section describes the user interface services necessary to do input and output under VULCAN.

## 3-2 I/O SERVICE

The system service $I/O is used to communicate with the I/O handlers. It is called in two different ways. The first method is used to initiate I/O transfers or pass special information to the handlers. It is called as follows:

```
                TLO     PARLIST
                BLU     $I/O

PARLIST         DATA    'XXXYY
                DATA    WORD-COUNT
                DATA    USER-BUFFER-ADDR
```

where XXX is the octal Logical File Number which must have been previously assigned.

YY is the octal function code as described in Table 3-1.

This form of the I/O call is referred to as a "long call".

The second form of $I/O call, or "short call", is used when no information other than function code need be passed. It is used on function codes 0, '12-'22, and '24. It is called as follows:

```
                TNK     'XXXYY
                BLU     $I/O
```

where XXX and YY are as defined above.

Upon return from this service, no user registers are saved, and the A register will be set to reflect the status of the operation, and the condition register is set to reflect the condition of A. These codes are as follows:

| | | |
|---|---|---|
| A = | 0 | Operation performed as requested. |
| | 2 | Disc area being written to by another program. Open call must be made again. |
| | 3 | Disc area in use by another program. This write operation ignored. |
| | 4 | Chained I/O (Paragraph 4-2) not allowed across multiple granules of a disc area. |
| | 5 | Fatal disc I/O error. |

## 3-3 FUNCTION CODES

The octal I/O function codes are summarized in Table 3-1 and discussed below.

### 3-3.1 Status - 00

The Status function is used to return information about the previously initiated I/O transfer. It will return in the A register the following information, with additional information available depending on the particular device:

Bit 22 set if the requested word count is not complete.

Bit 21 set if an end-of-file is read.

Bit 20 set if an end-of-tape or end-of-disc area is encountered.

Bits 15-0 = the word count transferred on last operation.

### 3-3.2 Symbolic Read - 01

This function is used to initiate the transfer of a symbolic record in ASCII from an I/O device. Conversion from the device media to ASCII is accomplished if necessary. The unused portion of the user's buffer will be set to blanks. If an end-of-file record is encountered, no words are transferred, and the end-of-file status bit is set. If an end-of-tape or end-of-disc area is read, then again no words are transferred, and both end-of-file and end-of-tape status bits are set.

3-3.3 Symbolic Write - 02

This function is used to initiate the transfer of a symbolic record to a physical device. Conversion from internal ASCII to the appropriate form for the device is accomplished if necessary.

3-3.4 Binary Read - 03

This function is used to initiate the transfer of a binary record from the device. Normally no conversion from external representation is done. If an end-of-file record is encountered, no words are transferred and the end-of-file status bit is set. If an end-of-tape or end-of-disc area is read, again no words are transferred and both end-of-file and end-of-tape status bits are set.

3-3.5 Binary Write - 04

This function is used to initiate the transfer of a binary record to an I/O device. Normally no conversion from internal representation is done.

3-3.6 Special Codes - 05, 06, 07, '10, '11

These codes are used for initiating special purpose input and output transfers. The exact use depends on the particular device, and is described in the following sections.

3-3.7 Write End-of-file - '12

This function is used to write an end-of-file mark on the physical device or disc area. The short call may be used.

3-3.8 Open - '13

This function is used to open a logical file. It is required after an Assignment, and prior to the first reference of a logical file in every program executed. Extra opens are always allowed and ignored. The normal form of the open is to use the short call; however, a long call is available for special purpose open functions. In this case, the word count should be 1, and the buffer address parameter contains the special open codes as listed below:

1 = Open with write access.
Opens LFN and sets the "in use for writing" status which keeps other programs from accessing the disc area.

3-3.9 Close - '14

This function code is used to close a logical file. It dumps all working buffers, releases, resources, and removes the logical file number assignment.

A close function code is allowed if the file has not been opened. The system will generate a close for all assigned logical files when a program exits.

3-3.10 Reposition File - '15

This function code will backspace the physical device or scan backwards through the disc area until an end-of-file mark is found, or the beginning of the area or device is located. If a file mark was located the logical file is then positioned forward over it, to be at the start of the current file.

3-3.11 Advance File - '16

The physical device or disc area is positioned forward until an end-of-file mark is found or the end-of-tape or end-of-disc area is reached. The file then remains positioned following the file mark if found, or at end-of-area if not found.

3-3.12 Backspace File - '17

This function code will backspace the physical device or scan backward through the disc area until an end-of-file is found. If found, the logical file is left positioned in front of the file mark so that the next read will detect an EOF. If no file mark is found, the logical file is rewound.

3-3.13 Advance Record - '20

The physical device or disc area is positioned forward one logical record. If a file mark is detected, the end-of-file status bit will be set. If an end-of-tape or end-of-disc area is detected, the end-of-tape and end-of-file status bits will be set.

3-3.14 Backspace Record - '21

The physical device or disc area is positioned backward one logical record. If already in the rewound (BOT) position, it is left unchanged.

3-3.15 Rewind - '22

The physical device is rewound, or the disc area is positioned to the front of the first record.

## 3-3.16 Set Current Record Address - '23

This function is applicable only to disc areas. The second parameter (word count) is used as the new current record number, and the disc area is positioned to that record. End-of-area status may be set if the requested record is not in the disc area, in which case the disc area is positioned to the end of the disc area. A subsequent write will append the record to the last record of the area.

## 3-3.17 Dump Buffer - '24

This function is used to unload blocking buffers on blocked disc area I/O. It is null on other devices. The logical file must be re-opened after doing a Dump Buffer operation. The system uses this function to return the dynamic memory used for blocked disc area blocking operation, between terminal and control point programs.

# 3-4 I/O EXAMPLES

Below are presented a few examples of doing I/O from assembly language:

## 3-4.1 Rewind, and Read First Record

Assume LFN = 8:

```
                TNK    '1013                   Open
                BLU    $I/O
                BNZ    file cannot be opened
                TNK    '1022                   Rewind
                BLU    $I/O
                TLO    PARLIST                 initiate transfer
                BLU    $I/O
                TNK    '1000                   wait
                BLU    $I/O
                LLA    2
                BON    end-of-file or end-of-
                       area detected
                 .
                 .
                 .
PARLIST         DATA   '1001
```

```
                 DATA    27
                 DAC     BUF

BUF              BLOK    27
```

3-4.2 Position to End of Disc Area and Append Additional Record

Assume LFN = 8:

```
                 TNK     '1013                       Open
                 BLU     $I/O
                 BNZ     file cannot be accessed
                 TLO     SETBIG                      SET CRA to
                                                     end-of-area
                 BLU     $I/O
                 TLO     PARLIST                     initiate write
                 BLU     $I/O
                 BNZ     can't write                 some other program
                                                     is using disc area
                 TNK     '1000
                 BLU     $I/O
                 LLA     3
                 BON     can't expand any larger
                  .
                  .
                  .
SETBIG           DATA    '1023
                 DATA    '37777777
PARLIST          DATA    '1002
                 DATA    27
                 DAC     BUF
BUF              BLOK    27
```

TABLE 3-1
I/O FUNCTION CODES

| OCTAL FUNCTION CODE | USE |
|---|---|
| 00 | Status |
| 01 | Symbolic Read |
| 02 | Symbolic Write |
| 03 | Binary Read |
| 04 | Binary Write |
| 05 | Special Read |
| 06 | Special Write |
| 07 | Special Read |
| 10 | Special Write |
| 11 | Special |
| 12 | Write End-of-File |
| 13 | Open |
| 14 | Close |
| 15 | Reposition File |
| 16 | Advance File |
| 17 | Backspace File |
| 20 | Advance Record |
| 21 | Backspace Record |
| 22 | Rewind |
| 23 | Set Current Record Address |
| 24 | Dump Buffer |

# SECTION IV
# DISC I/O

## 4-1 GENERAL

This section describes the I/O accessing methods for disc areas under VULCAN. Included is individual device media formats, special device dependent information, and a description of each of the I/O function codes available under the $I/O system service, for discs.

Disc I/O under VULCAN is transmitted via the resident disc queue,based on priority, to each disc controller on the system. As a controller finishes an operation it removes the next entry from its queue based on the following algorithm:

a) Real-time entries for programs above priority 32 are handled on a priority basis. Entries of equal priority are taken in order of entry (First in, first out).

b) Interactive entries, for programs between priorities 16 and 31, are handled on a first in first out method regardless of priority.

c) Control point entries, for programs below priority 16, are handled based on minimum access time. All of the entries in this range are scanned (after all catagory a and b entries are removed) and the one closest to the current arm position for that disc is taken.

If a hardware error condition exists on a disc operation, it is attempted a total of 3 times. If the error condition still exists, the program is aborted and the operator notified of the type and location of the error.

Interaction between different programs reading and writing a disc area simultaneously is inhibited for blocked and unblocked areas as discussed earlier. This works as follows:

a) When a program opens the disc area, a check is made to ensure that no other program is writing to the area. If it is, an error code is returned for real-time and interactive programs, and the logical file must be re-opened. For control point programs, the program is suspended until the disc area becomes available.

b) When the first write to the disc area is attempted, or an open with write access request is made, a check is made to ensure that no other program has opened the disc area. If it has, an error code is returned to interactive and real-time programs, and the write request must be reissued. For control point programs, the program is suspended until it can do the write.

Automatic disc area extension occurs when writing in a sequential manner. That is, as continual writing is done to a disc area additional granules are allocated and appended to the disc area until either the disc area maximum size is reached, or the disc pack has no more space available. At this time an end-of-disc-area condition is returned to the calling program. An unblocked disc area cannot be extended by doing a Set Current Record Address to a record beyond the current end-of-disc area and then doing a write. It must be done via sequential writing.

Disc I/O varies slightly between blocked and unblocked disc area in the areas of special function codes and packing formats. Hence, two discussions are provided below:

## 4-2 UNBLOCKED DISC AREA I/O FUNCTIONS

### 00 - Status

Returns the following:

(E) = Current Relative Sector

(A) = Bit 22 - Word Count Not Complete on last operation
Bit 21 - End of file detected on last operation
Bit 20 - Disc Area bounds exceeded on last operation
Bit 19 - File is open
Bit 18 - Always a 1
Bit 15-0 - Word Count transferred on last operation

### 01 - Symbolic Read

Reads into the specified user buffer the specified word count. Transfer is initiated at the start of the Current Relative Sector Number and is terminated either when the word count is complete or when an End-of-file is read. Upon completion, the Current Relative Sector Number is set to the next sector of the disc area.

02 - Smybolic Write

Writes from the specified user buffer the specified word count to the disc at 112 words per sector, zeroing any unused words in the last sector. Transfer is initiated at the start of the Current Relative Sector Number and is terminated either when the word count is complete or when the physical end of the disc area is reached. When this occurs, and the disc area cannot be expanded further, status Bit 20 is set and the Word Count transferred reflects that which was written to the disc. Upon completion of the operation, the Current Relative Sector Number is set to the next sector of the disc area.

03 - Binary Read

Same as 01 - Symbolic Read.

04 - Binary Write

Same as 02 - Symbolic Write.

05 - Special Read

Same as 01 - Symbolic Read.

06 - Special Write

Same as 02 - Symbolic Write.

07 - Chain Read

This function allows a user program to utilize the automatic restart/command chain features of the Chained Block Controller (CBC). Its use is recommended only for special purpose applications; i.e., where quick access is required for multiple operations as in Real-time applications. Chained I/O is done via the following sequence:

```
                TLO     PARAM
                BLU     $I/O
                 .
                 .
                 .
PARAM           DATA    'XXXYY
                DATA    1st Word Count
                DAC     1st Buffer Address with Bit 23 set for
                        additional parameters, and Bit 22 set
                        for Command Chain.
```

| | |
|---|---|
| DATA | relative sector number for next operation if Bit 22 set in above word. If Bit 22 is not set, this word is absent from the parameter list. |
| DATA | 2nd Word Count |
| DAC | 2nd Buffer Address. Bits 23, 22 may be set for additional continuation. Etc. |

Examples:

1) To read two sequential disc sectors into different user buffers:

```
PARM      DATA   'XXX07
          DATA   112
          DAC*   BA1      (Bit 23 set for continue)
                          first buffer address
          DATA   112
          DAC    BA2      second buffer address
```

2) To read disc sectors 0 and 3 into a buffer without reading the intervening 2 sectors:

```
PARAM     DATA   'XXX07
          DATA   112
          DAC*   BA1, J   (Bits 23, 22 set for command chain)
          DATA   3
          DATA   112
          DAC    BA1+112
```

'10 - Chain Write

This function is identical to Chain Read (07) except that it writes data to the disc.

'11 - Invalid Function Code

The calling program is aborted.

'12 - Write End-of-File

Writes one End-of-File sector to the disc file at the Current Relative Sector Number.

'13 - Open

Opens the file for subsequent access. The first open of a logical file will rewind the disc area.

'14 - Close

Closes the disc file and removes the assignment from the program's Assignment List.

'15 - Reposition File

Reads each sector from the Current Relative Sector Number backward until an end-of-file sector is reached, at which point the Current Relative Sector Number is set to the sector following the end-of-file sector. If none is found, the file is rewound.

'16 - Advance File

Reads from the Current Relative Sector Number forward until an end-of-file sector is encountered or the end-of-disc area is reached. The Current Relative Sector Number is set to the sector following the end-of-file sector if found. If not found, the end-of-disc area status is set and the Current Relative Sector Number is set to the last sector of the disc area.

'17 - Backspace File

Same as Reposition File (15) except that the Current Relative Sector Number is set to the end-of-file sector upon completion.

'20 - Advance Record

Increments the Current Relative Sector Number by one.

'21 - Backspace Record

Decrements by one the Current Relative Sector Number, and if negative, the disc area is rewound.

'22 - Rewind

The Current Relative Number is set to 0, the start of the disc area.

'23 - Set Current Record Address

The Current Relative Section Number is set to the second parameter in the parameter list (word count). If this number is not within the disc area, the Current Relative Sector Number is set to the end of the disc area and the End-of-Disc Area status is set.

'24 - Dump Buffer

Null.

## 4-3 BLOCKED DISC AREAS

Blocked disc areas contain multiple records packed into one disc sector, thereby creating a large saving in disc space. Binary records are packed multiple per sector with one gap word between each record. Symbolic records are packed as above and in addition multiple blanks are compressed into one blank-count byte. This format is discussed below in Paragraph 4-4.

Blocked I/O Functions

00 - Status

Returns the following:

(E) = Current Record Number

| | | | |
|---|---|---|---|
| (A) = | Bit 22 | - | Word Count not complete on last operation |
| | 21 | - | End-of-file on last operation |
| | 20 | - | End-of-disc area on last operation |
| | 19 | - | File is open |
| | 18 | - | Always a 1 |
| | 15-0 | - | Word Count transferred on last operation |

01 - Symbolic Read

Transfers the next record of the file into the users' buffer. If the user requests more words than are in the record being read his buffer is blank filled. If an end-of-file record is read, no words are transferred, and the end-of-file status is set. If the end-of-disc area is detected, no words are transferred and both the end-of-file and the end-of-disc area status bits are set.

02 - Symbolic Write

Transfers the specified word count of the user's buffer to the next record address of the area. Multiple blanks are compressed into a single blank-count byte. If the physical end-of-disc area is detected, the record will not be written and the end-of-disc area and word-count-not-complete status bits are set.

03 - Binary Read

Transfers the next record of the area into the requested number of words of the user's buffer. If end-of-file or end of disc area conditions are detected, the appropriate status bits are set as discussed under Symbolic Read (01).

04 - Binary Write

Transfers the specified word count of the user's buffer to the next record address of the file. No blank compression is done. If the end of disc area is detected, the end-of-disc area and word-count-not-complete status bits are set.

05 - Special Read

Same as 03 - Binary Read.

06 - Special Write

Same as 04 - Binary Write.

07 - Invalid Function Code

The calling program is aborted.

'10 - Invalid Function Code

The calling program is aborted.

'11 - Continue Write

This function appends the new record as specified by the user's buffer address and word count to the previously written record. This function code must follow an 02, 04 or 06 function code (except for intervening status calls) and automatically assumes the same mode (Symb, Bin, etc.) as the previous function. Records written with continue write are just extensions of the previous record and do not increment the Current Record Number pointer. This function is useful if a long record is desired on a disc area but cannot be conveniently built up in memory for a single write operation.

'12 - Write End-of-file

An end-of-file record is written at the Current Record Number. Note that this is a software End-of-file and a hardware mark is not written. An end-of-file is encoded as a zero length record.

'13 - Open

Opens the specified disc area and allocates a block of user core utilizing the Dynamic Core Manager service. This block is the length of the disc area's blocking factor in words plus 3 words. For example a blocking factor of 2 will require a DCM buffer of 227 words.

'14 - Close

Closes the disc area, returns the dynamic core buffer, and removes the Logical File Assignment from the program's assignment list.

'15 - Reposition File

Reads records backward from the Current Record Number until an end-of-file record is encountered at which point the Current Record number is set to the record following the end-of-file record. If no end-of-file record is found, the file is rewound.

'16 - Advance File

Reads records forward until an end-of-file record is found, or the end of the disc area is reached. When found, the Current Record Number is set to the record following the end-of-file record or the end of disc area.

'17 - Backspace File

Same as Reposition File except that the Current Record Number is set to the end-of-file record.

'20 - Advance Record

Advance the Current Record Number and pointers to the next record of the area.

'21 - Backspace Record

Decrements the Current Record Number and pointers to the record immediately preceding the current one. When the current record number is zero, the area has been rewound.

'22 - Rewind

Rewinds the file and sets the Current Record Number to zero.

'23 - Set Current Record Address

Sets the Current Record Number to that specified in the second parameter (word-count) of the user's parameter list. The requested record is located by an iterative search technique rather than by reading single records. If the requested record number is greater than the number of records in the area, the Current Record Number is set to the next available position following the last record of the area.

'24 - Dump Buffer

The user's Dynamic Core Buffer allocated on the open request is returned. Another open function is required to continue accessing the logical file.

## 4-4 BLOCKING FORMAT FOR BLOCKED DISC AREAS

Blocked disc areas are divided into a number of blocks. The size of these blocks, which varies from one to seven disc sectors (112 to 784 words) is called the blocking factor. The blocking factor is specified when a disc area is generated, and the granule size is made an even multiple of it to minimize disc accesses.

For normal I/O transactions, one block of the area is in memory. Records are unpacked from this block as reading is going on, and a new block read when the current one is exhausted. When writing, records are packed until the block is full at which time it is written to disc, and a new block begun. Multiple records are packed into a block. They are separated by a one-word inter-record gap. The format of these gap words is as follows:

Bits 23-22 Mode of following record:

00 = Symbolic

01 = Binary

10 = Special

11 = Continuation of previous record (gap is to be ignored)

Bits 21-11 Word Count of previous record (Backward word count)

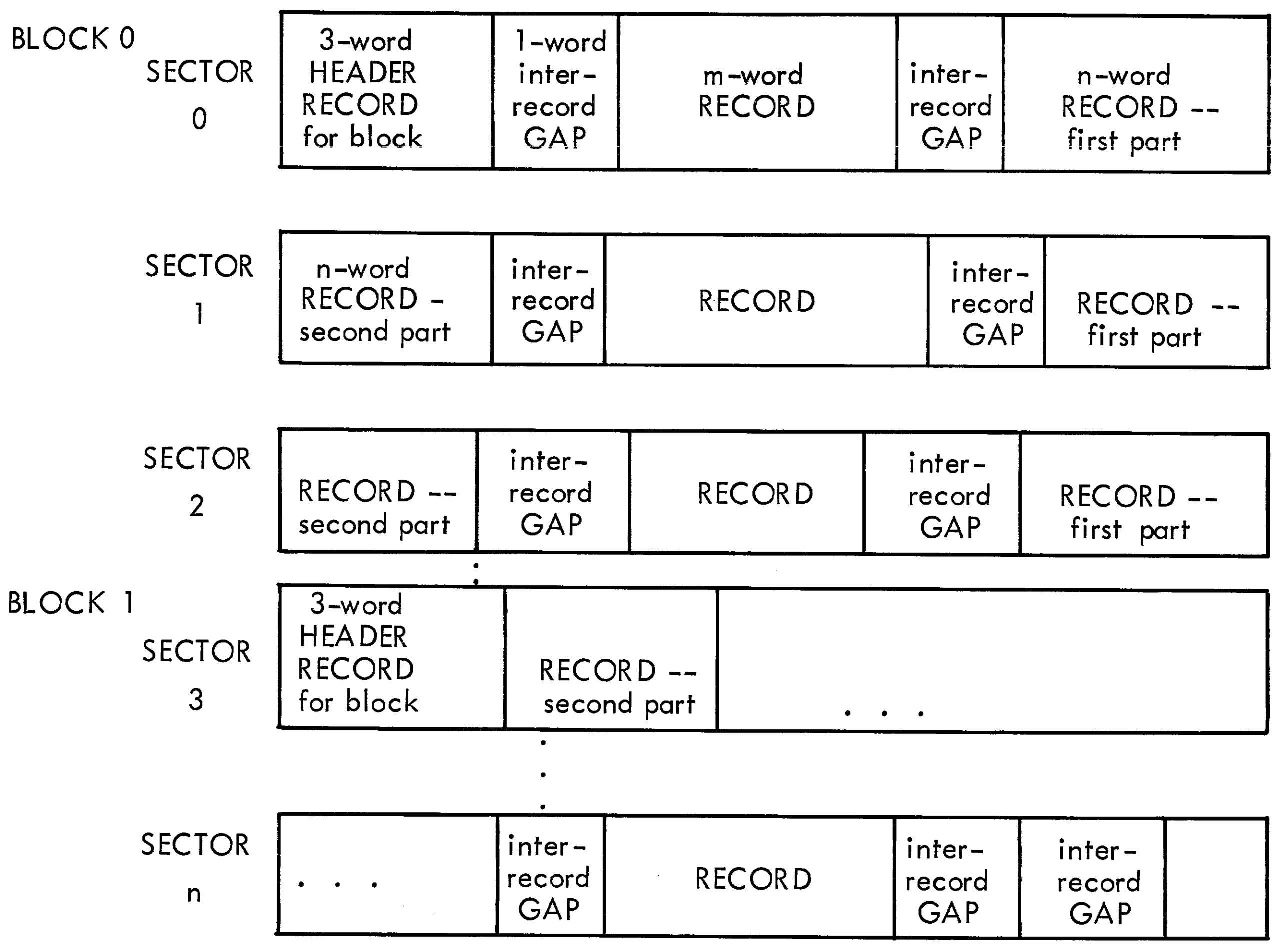
Bits 10-0 Word Count of following record (Forward word count)

An end-of-file record is encoded as a zero length record, and thus consists of two consecutive gap words, the first having a forward word count of zero, and the second having a backward word count of zero.

The first three words of each block are reserved for special pointers. The first word (word 0) is normally the relative record number of the first record in the block. For the first block of an area (block 0), word 0 is the relative sector number of the first sector of the last block of the area. This is sometimes referred to as the LRS (last Recorded Sector) value. The second header word (Word 1) is the relative record number of the last record in the block. The third head word (Word 2) is divided as follows:

| | |
|---|---|
| Bits 23-12 | Relative word position from start of block of first inter-record gap in this block. |
| Bits 11-0 | Relative word position from start of block of last inter-record gap in this block. |

Below is shown a sample blocked disc area. It has a blocking factor of 3.




This is an EOF record.

# SECTION V
# CRT I/O

## 5-1 GENERAL

The Model 9050 (TEC-425) CRT is used as the Operator Communications Device, and may also be used as an interactive terminal. The same accessing methods apply in both cases, except that the Operator Device is not allocatable, and may be accessed by more than one program concurrently.

The page size of the Model 9050 is normally 24 lines by 80 characters per line. Under VULCAN, the top line is used for symbolic input and the other 23 lines for symbolic output. Any line or character position may be used for Special I/O via the Edit Read/Write functions. Lines are number 0-23, and columns are numbered 1-80.

The symbolic output mode may be set to one of three different methods by special input keys. By entering the sequence " ∧ S", the CRT is put into the "scroll" mode. In this mode, when symbolic output reaches the bottom line, the entire screen is shifted up one line, and the next symbolic line is again written on the bottom line. This continues until the current symbolic output line number is reset. The sequence " ∧ P" places the CRT in the page mode. In this mode when the symbolic output page is filled, the entire screen is reased and output begins again from line one. The third mode, or "Wait Mode" is activated by entering the sequence "∧W". This mode is identical to the page mode except that prior to erasing the page, the character ∧ is placed in the home position, and the CRT waits for the user to depress the "XMIT" key, indicating the next page should be output. The default mode is the scroll mode.

## 5-2 INTERACTIVE CRT FUNCTION CODES

<u>00 - Status</u>

Returns the following:

(A) = Bit 21 set if $EOF (end-of-file) record encountered

Bit 15-0 = Word Count Transferred

(E) = Current Status of Data Panel lights. Bits 1-16 are on to correspond to lights 1-16. The other bits are zero.

01 - Smybolic Read

Transfers into the user buffer the specified word count from the top line of the display when the operator has pressed XMIT to signal the input is read. If the top line is ready when the Symbolic Read request is received, it is transferred immediately. If not, the INPUT REQUEST light (#16) is turned on and the transfer is made on the next I/O call, after the user has typed the line and pressed XMIT.

The sequence $EOF in colunms 1-4 is used to enter an end-of-file.

02 - Smybolic Output

Transfers one line of data from the users buffer to the current smybolic output line. This line number is incremented for each symbolic write. The first column output is treated as a Carriage Control character and ignored.

03 - Edit Read

This function uses a special I/O calling sequence utilizing a fourth parameter (after buffer address) specifying initial position on the screen to read from. For this function mode, the user's word count is treated as a character count and the specified number of characters are packed into the users buffer at 3 characters per word until the character count is complete. The I/O call is set up as follows:

```
                    FORM    8,8,8
        PARLIST     DATA    'XXXYY
                    DATA    Character Count
                    DAC     Buffer address
                    DATA    /0, line, column/
```

04 - Edit Write

This function utilizes the same calling sequence as Edit Read above, and the specified character count is written to the screen starting at the specified cursor address.

05 - Wait for XMIT

The calling program is held in a WAIT state until the CRT operator depresses the XMIT key.

06 - Set Tabs

Tab stop characters are set at the designated positions on the screen, and are replaced at these positions after every Symbolic Input Request until changed or removed. If this function is called with a word count of 0, all previously defined tab stops are removed. If the word count is non-zero, it is taken as a character count of the number of tabs contained in the user's buffer. Tab positions are to be packed 2 per word (12 bits each) in the user's buffer.

For example, to set tab stops at columns 9, 15 and 30:

```
             FORM   12, 12
PL           DATA   'XXX06
             DATA   3
             DAC    BUF
BUF          DATA   /9, 15/
             DATA   /30, 0/
```

07 - Get Cursor Address

The current position of the cursor is returned in the first word of the user's buffer. The line is returned in bits 15-8 and the column in bits 7-0.

'10 - Set Cursor Address

The cursor is set to the position specified in the first word of the user's buffer. The line is taken from bits 15-8 and the column from 7-0.

'11 - Set Data Panel Lights

The first word of the user's buffer is used to control the Data Panel lights. Bits 1-15 are used to control lights 1-15 (the top light corresponds to bit 1). If the bit is on, the corresponding light is turned on. If the bit is off, the light is turned off. Light #16 (Bottom) is reserved for use by the handler as a symbolic input request light.

'12 - Write-End-of-File

The message EOF.. is written on the Current Symbolic Output line.

'13 - Open

Opens the logical file and sets the current symbolic output line to one.

'14 - Close

Closes the logical file and removes the assignment from the program's assignment list.

'15 - '20

These are invalid function codes. The calling program is aborted.

'21 - Backspace Record

Sets the backspace flag such that the next Symbolic Read request will transfer the same image that was transferred on the last Symbolic Input.

'22 - Rewind

Clears the screen and sets the Current Symbolic Output Line to one.

'23 - Set Output Line Number

Sets the Current Symbolic Output Line to that specified as the second parameter (word count) in the user's I/O parameter list.

'24 - Dump Buffer

Null.

# SECTION VI
# TELETYPE I/O

## 6-1 TELETYPE TERMINALS

All interactive terminals other than Model 9050 CRT's are treated as teletypes under VULCAN. Certain teletypes may have special features such as paper tape (ASR feature) or Magnetic Cassettes ( on Model 2200 "Silent 700").

## 6-2 TELETYPE I/O

Normal I/O is done one line at a time from the keyboard and to the printer unit. All characters input and recognized by the system are output back to the printer unit, unless it is in the non-echo mode (see below). The output line length is normally 72 characters, however the first character is treated as carriage control when output as a symbolic record. A "1" in this position causes 3 line-feeds, to simulate top of page. A "0" in the first column causes two line-feeds for double spacing. A "+" causes no line feeds for over printing. Any other character is ignored and a single line-feed is output. The carriage control character (first character of the buffer) is never printed.

## 6-3 PAPER TAPE I/O

Paper tape I/O is dependent on the type of teletype being used. On local Harris modified teletypes, where tape control is achieved through computer commands, tape input must be requested by the device handler. To read in a symbolic paper tape the key "TAPE" should be sent to activate the reader. The reader stays enabled until the ~~TAPE~~ is entered. On local teletypes the tape punch must be manually activated when it is desired to punch the output. All output is also printed.

On standard teletypes, the punch is turned on under device handler control as needed.

Binary tapes are assumed to be in the following format:

a) First character is line feed ('212)

b) Data at 4 characters per word, 6-bits per character, most significant bits first.

c) End of record indicator is Carriage Return('215)

An end-of-file record is encoded as an EOT ('204) followed by carriage return ('215).

## 6-4 CASSETTE I/O

The Magnetic Tape Cassettes available on the Model 2200 teletypes are accessed much as magnetic tape units. Tapes may be advanced, backspaced, rewound just as magnetic tape drives.

Binary Records are encoded in the following manner:

a) No start-of-record character.

b) Data at 4 characters per word, 6-bits per character, most significant bits first. An octal 40 is added to each character before being written to tape to ensure that it is a valid ASCII character. When read back in binary, the bias of '40 is removed.

c) End-of-record indicator is carriage return.

d) An end-of-file record occupies a whole tape block, and is encoded as a Line Feed ('212), an EOT ('204), followed by carriage return ('215), followed by 83 NUL's (000).

e) NUL (000) or RUBOUT ('377) characters may exist between records on tape. These must be ignored.

Symbolic records have no leading character, and are terminated by carriage return ('215).

## 6-5 SPECIAL PURPOSE KEYS

The following keys are used for special purpose functions on teletypes.

| Regular TTY | Model 2200 | Function |
|---|---|---|
| ALT MODE | ESC | Line Cancel. Outputs a " ! " and carriage return. Allows re-entry of entire line. |
| ← | US or ____ (underscore) | Character Cancel. Last character entered is removed and input buffer backspaced one character. |

| Regular TTY | Model 2200 | Function |
|---|---|---|
| X-OFF | DC3 | Abort program currently executing at that terminal. |
| CTRL Z | SUB | Enter non-echo mode. Characters input from keyboard are not printed until cancelled by CTRL W or carriage return. |
| CTRL W | ETB | Return to normal echo mode. |
| WRU (CTRL E) | ENQ | System identification is output. Functions only when no input is pending. |
| TAPE | N/A | Changes input from keyboard to paper tape and turns on paper tape reader. |
| $\overline{\text{TAPE}}$ | N/A | Turns off paper tape reader, and returns to keyboard control. |

## 6-6 TELETYPE I/O FUNCTION CODES

### 00 - Status

Returns the following in the A register.

| | |
|---|---|
| Bit 21 | set if end-of-file encountered. |
| Bits 15-0 | set to word count transferred. |

### 01 - Symbolic Input

Accepts one line of data at 3 characters per word into the user's buffer. Line is terminated by a carriage return. The characters $EOF as the first characters in the line are treated as an end-of-file record.

### 02 - Symbolic Output

Writes one line of data from the user's buffer until the specified word count is complete. Only 72 characters at most will be output. The first character is treated as carriage control as discussed above.

03 - Binary Input

The paper tape reader is turned on if possible and a binary record is input to the user's buffer at 4 characters per word until the word count is complete or an end-of-record is encountered.

On cassettes, the input of the specified cassette is activated and one binary record is transmitted assuming the format discussed above.

04 - Binary Output

On paper tape teletypes the tape punch is turned on where possible and the data is output according to the binary format described above. If this is the first binary write following an open, 6 inches of leader are punched prior to outputting the record.

On cassettes, the record is output according to the binary format discussed above. No leader is written.

05 - Special Input

Reads data and places the characters one per word into Bits 7-0 of the user's buffer until the specified word count is complete. No special characters are recognized.

06 - Special Output

Outputs data from Bits 7-0 of the user's buffer at one character per word until the specified word count is complete. The characters are output unmodified with no header or trailer codes.

07 - '11 - Invalid Function Codes

The program is aborted.

'12 - Write End-of-File

The message EOF.. is output, along with an EOT ('204) code for paper tape end-of-file code. On cassettes, an entire block is reserved for the end-of-file record.

'13 - Open

Opens the logical file.

'14 - Close

Closes the logical file and removes the assignment from the program assignment list.

'15 - Reposition File

The message RPF . . . is output and the program is put into a hold condition until a "TAPE" key is depressed. The user should appropriately re-position the paper tape to the start at the current file.

On cassettes, the tape is backspaced so as to be positioned just following the preceding file mark block.

'16 - Advance File

The message ADF. . is output and the program is put into a hold condition until the "TAPE" key is depressed. The user should advance the tape to the next file mark. On cassettes, data is read until an "EOT" code (ASCII '204) is located.

'17 - Backspace File

The message BSF. . is output and the program is put into a hold condition until the "TAPE" key is depressed. The operator should backspace the paper tape to the previous file mark. On cassettes, the tape is backspaced until a file mark block is detected.

'20 - Advance Record

The message ADR. . is output and the program is put into a hold condition until the "TAPE" key is depressed. The operator should advance one record on the paper tape. On cassettes, one record is bypassed (until next carriage return).

'21 - Backspace Record

A flag is set such that the next Symbolic Input request will reread the record transferred on the last Symbolic Read request.

For cassettes, one tape block is backed over.

'22 - Rewind

On cassettes, the specified cassette is rewound. Otherwise, the function is null.

'23 - Set Current Record Address

No action.

'24 - Dump Buffer

No action.

# SECTION VII
# MAGNETIC TAPES

## 7-1 GENERAL

Magnetic tape drives are system resources and as such must be allocated via the resource commands. These commands are used to specify tape options, densities, drive speeds, and number of tracks.

## 7-2 MAGNETIC TAPE I/O FUNCTIONS

00 - Status

Returns the following:

| | | | |
|---|---|---|---|
| (A) | = | Bit 22 | set if word count not complete |
| | | Bit 21 | set if end-of-file record |
| | | Bit 20 | set if end-of-tape detected |
| | | Bits 15-0 are word count transferred on last operation | |

01 - Smybolic Input

Inputs into the user's buffer the specified number of words until an end-of-record is detected. If conversion was requested on the resource call, it is done prior to releasing control back to the user program.

02 - Symbolic Output

If conversion was requested on the resource call, the data in the user's buffer is converted in his buffer; then the data is written to the tape until the specified word count is complete.

03 - Binary Input

Inputs into the user's buffer the specified number of words until an end-of-record is detected. No conversion is done on the data.

04 - Binary Output

Outputs data from the user's buffer to the tape until the word count is complete. No conversion is done.

05 - Special Input

Same as 03 - Binary Input.

06 - Special Output

Same as 04 - Binary Output.

07 - '11

Invalid Function codes. The calling program is aborted.

'12 - Write End-of-File

An end-of-file record is written to the tape.

'13 - Open

Opens the logical file. The first open on a drive will rewind it.

'14 - Close

Closes the logical file and removes the assignment. If no other assignments are made to the drive by this program, then the message TAPE $_{xx}$ FREE is output to the operator, with "$_{xx}$" being the physical device number, the tape is rewound, and the drive is made available for resourcing by another program.

'15 - Reposition File

The tape is backspaced until either an end-of-file record is encountered or the beginning of the tape is detected. If an end-of-file was read, the tape is positioned forward over the end-of-file record.

'16 - Advance File

Tape moves forward until an end-of-file record is detected.

'17 - Backspace File

The tape is backspaced until either an end-of-file record is encountered or the beginning of tape is detected. If an end-of-file was detected, the tape is left positioned in front of the end-of-file.

'20 - Advance Record

The tape is advanced one record.

'21 - Backspace Record

The tape is backspaced one record.

'22 - Rewind

The tape is rewound.

'23 - Set Current Record Address

Invalid. The calling program is aborted.

'24 - Dump Buffer

No action.

# SECTION VIII
# CARD READER I/O

## 8-1 GENERAL

Card Readers are spooled input only devices under VULCAN. No program may communicate directly with them. Input from card readers can be broken down into either Control Point jobs, or program data decks.

## 8-2 JOB INPUT

Jobs to be run at Control Points may be read in from card readers providing they have a valid $JOB card on the front. The format of this card is as follows:

$JOB jobname qualifier userno parameters

where: "jobname" is a 1 through 12 character job identifier.

"qualifier" is the valid user's sign-on qualifier.

"userno" is the user's user number.

"parameters" consist of the following:

| | |
|---|---|
| OUT = device<br>OUT = areaname | Sets Diagnostic Output (LFN 3) and List Output (LFN 6) to the specified device or disc area. |
| TIME = n | Sets time limit for job execution to "n" seconds. |
| LINES = n | Sets spool output limit to "n" records. |
| SIZE = n | Sets default background size to "n" pages. |
| PRI = n | Sets priority of execution to "n" (0-15). |

Control point jobs are terminated with a $EOJ card, preceding the required EOT card. (6/9 punch in column one, used to close out spool area)

## 8-3 DATA INPUT

Data files to be read by any program may be input on cards provided they are preceded by a valid $DATA card. The format of this card is as follows:

$DATA programname userno

where: "programname" is the disc area name of the program which will read the data.

"userno" is the user number of the user who will read the data.

When the specified program and user make an assignment to the particular card reader device, the Logical File Number will be automatically assigned to the system disc area containing the card deck images. This disc area will be eliminated when the logical file is closed.

## 8-4 CARD READER CONTROLS

All job and data decks read in through the card reader must be followed with an EOT card. This card has a 6/9 punch in colunm 1 and columns 2-4 must be blank.

The following card images are reserved for special functions as described below:

| | |
|---|---|
| $026 | This sets the 026 card conversion mode. See Table 8-1. |
| $029 | This returns to the default 029 conversion mode. See Table 8-1. |
| $EOF<br>8/9 in Col. 1 | Either of these images are used as an end-of-file record. |
| $BIN | This card indicates that subsequent cards (until another $ card reader control card) are to be interpreted as binary records (two 12-bit columns make one word, columns 1-6 contain no data). The Harris Corp. Disc Monitor System (DMS) binary format is assumed. (37 words per record) |
| $UNF | This card indicates that subsequent cards (until another $ card reader control card) are to be interpreted as unformatted binary records (40 words per card, 2 columns per word). |
| $ABS | This card indicates that the following cards make up a $DLOAD format module. (6-word header record followed by binary load module). See Chapter B. |

Standard VULCAN binary cards have a unique punch configuration in the first column and are automatically recognized as binary records.

## 8-5 CARD READER I/O FUNCTION CODES

The following function codes are honored when doing I/O from a card reader input file:

<u>00 - Status</u>

Returns the following:

(E) = Current Record Number.

(A) = Bit 21 set if end-of-file encountered on last operation.

Bit 20 set if end-of-card-deck encountered on last operation.

Bits 15-0=word count transferred on last input.

<u>01 - Symbolic Read</u>

Transfers the next record of the data deck to the user's buffer. The record is assumed to be a symbolic card image. If more than 27 words are requested, the remainder of the user's buffer is blank filled.

<u>03 - Binary Read</u>

Transfers the next record of the data deck to the user's buffer until the word count is complete or the end-of-record is encountered. Normally a single read will input one card, but a single binary read may transfer a multi-card DMS binary ($BIN) record or a $DLOAD ($ABS) module. (A $DLOAD module is 2 records)

<u>05 - Special Read</u>

Same as binary read above.

<u>'13 - Open</u>

Opens the logical file.

<u>'14 - Close</u>

Closes the logical file, removes the LFN assignment, and eliminates the disc area containing the card images.

'16 - Advance File

Advance records until an end-of-file record is encountered, or the end of the card deck is reached.

'20 - Advance Record

Bypasses one card image.

All other function codes are invalid and will abort the calling program.

TABLE 8-1

CARD-CODE CONVERSION TABLE

Where two punch codes are shown, the first is preferable but either is accepted.

| 029 Punch | 026 Punch | Character | ASCII OCTAL |
|---|---|---|---|
| 4-8 | 11-2-8 | @ | 100 |
| 12-1 | 12-1 | A | 101 |
| 12-2 | 12-2 | B | 102 |
| 12-3 | 12-3 | C | 103 |
| 12-4 | 12-4 | D | 104 |
| 12-5 | 12-5 | E | 105 |
| 12-6 | 12-6 | F | 106 |
| 12-7 | 12-7 | G | 107 |
| 12-8 | 12-8 | H | 110 |
| 12-9 | 12-9 | I | 111 |
| 11-1 | 11-1 | J | 112 |
| 11-2 | 11-2 | K | 113 |
| 11-3 | 11-3 | L | 114 |
| 11-4 | 11-4 | M | 115 |
| 11-5 | 11-5 | N | 116 |
| 11-6 | 11-6 | O | 117 |
| 11-7 | 11-7 | P | 120 |
| 11-8 | 11-8 | Q | 121 |
| 11-9 | 11-9 | R | 122 |
| 0-2 | 0-2 | S | 123 |
| 0-3 | 0-3 | T | 124 |
| 0-4 | 0-4 | U | 125 |
| 0-5 | 0-5 | V | 126 |
| 0-6 | 0-6 | W | 127 |
| 0-7 | 0-7 | X | 130 |
| 0-8 | 0-8 | Y | 131 |
| 0-9 | 0-9 | Z | 132 |
| 12-7-8 | 7-8 | [ | 133 |
| 12-2-8 | 12-2-8 | \ | 134 |
| 11-7-8 | 0-2-8 | ] | 135 |
| 0-2-8 | 11-5-8 | ↑ | 136 |

TABLE 8-1 (Cont'd.)

CARD-CODE CONVERSION TABLE

| 029 Punch | 026 Punch | Character | ASCII OCTAL |
|---|---|---|---|
| 0-5-8 | 6-8 | ← | 137 |
| --- | --- | (blank) | 40 |
| 11-2-8 or 11-0 | 11-0 | ! | 41 |
| 7-8 | 0-6-8 | " | 42 |
| 3-8 | 0-5-8 | # | 43 |
| 11-3-8 | 11-3-8 | $ | 44 |
| 0-4-8 | 12-5-8 | % | 45 |
| 12 | 0-7-8 | & | 46 |
| 5-8 | 4-8 | ' | 47 |
| 12-5-8 | 0-4-8 | ( | 50 |
| 11-5-8 | 12-4-8 | ) | 51 |
| 11-4-8 | 11-4-8 | * | 52 |
| 12-6-8 | 12 | + | 53 |
| 0-3-8 | 0-3-8 | , | 54 |
| 11 | 11 | - | 55 |
| 12-3-8 | 12-3-8 | . | 56 |
| 0-1 | 0-1 | / | 57 |
| 0 | 0 | 0 | 60 |
| 1 | 1 | 1 | 61 |
| 2 | 2 | 2 | 62 |
| 3 | 3 | 3 | 63 |
| 4 | 4 | 4 | 64 |
| 5 | 5 | 5 | 65 |
| 6 | 6 | 6 | 66 |
| 7 | 7 | 7 | 67 |
| 8 | 8 | 8 | 70 |
| 9 | 9 | 9 | 71 |
| 2-8 | 5-8 or 2-8 | : | 72 |
| 11-6-8 | 11-6-8 or 12-7-8 | ; | 73 |
| 12-4-8 | 12-6-8 | < | 74 |
| 6-8 | 3-8 | = | 75 |
| 0-6-8 | 11-7-8 | > | 76 |
| 0-7-8 or 12-0 | 12-0 | ? | 77 |

# SECTION IX
# CARD PUNCH I/O

## 9-1 GENERAL

The card punch is a spooled device under VULCAN and as such all I/O is transacted via system disc areas.

## 9-2 CARD PUNCH I/O FUNCTION CODES

The following function codes apply to output to the card punch.

00 - Status

Used only to wait for completion of previous transfer. No special returns.

02 - Symbolic Output

This function is used to punch one symbolic card image. 029 conversion is assumed, until a $026 record (Columns 1-4) is encountered. The $026 record is punched and subsequent cards will use the 026 conversion. A $029 image will be punched if encountered, with subsequent conversion reverting back to 029 mode.

04 - Binary Output

Data is assumed to be standard VULCAN binary and is punched unformatted at 2 columns per word until the word count is complete. If necessary, multiple card images may be punched.

06 - Special Output

A binary record in the Disc Monitor System (DMS) format is output. Multiple cards may be punched until the specified word count is complete. Column 1 is used for end-of-record control, and columns 2-5 are blank.

'12 - Write End-of-File

A single card containing only an 8/9 punch in column 1 is written.

'13 - Open

The logical file is opened.

'14 - Close

The logical file is closed, the LFN assignment removed from the user's program, and the system disc area containing the card images is queued for output to the specified card punch. It will be punched as soon as the device becomes available.

All other function codes are invalid, and will abort the calling program.

# SECTION X
# LINE PRINTER I/O

## 10-1 GENERAL

All line printers are spooled devices under VULCAN and accordingly all output is transmitted via system disc areas. The Models 47xx (Versatec) Printer-Plotters function slightly differently and are discussed in the next section.

The first column output to the line printer is a carriage control character, and interpreted according to the following table:

TABLE 10-1
LINEPRINTER CARRIAGE CONTROL

| Column 0 | Action |
|---|---|
| + <br> @ | No space, over print this line over previous line. |
| A <br> (blank) | Single space. |
| B <br> 0 | Double space. |
| C | Triple space. |
| 1 <br> P | Skip to channel 1 of carriage tape (Top of form). |
| Q | Skip to channel 2. |
| R | Skip to channel 3. |
| S | Skip to channel 4. |
| : (colon) | Output contents of line starting with character after colon to operator and place printer in "Hold" mode until released by operator. This is used to request special forms control or other special operator action. |
| Any other character | Single space. |

The carriage control character is never printed.

## 10-2 LINE PRINTER I/O FUNCTION CODES

00 - Status

No action other than to wait for previous transfer to terminate.

02 - Symbolic Output

Data is transferred from the users buffer in order to print one line until the word count is complete. Excess characters over the line length are ignored. The first character is carriage control.

04 - Binary Output

Same as 02 - Symbolic Output.

06 - Special Output

Same as 02 - Smybolic Output.

'12 - Write End-of-File

The message EOF. . is printed on one line, single spaced.

'13 - Open

The logical file is opened.

'14 - Close

The Logical File is closed, the LFN assignment for the program is removed, and the system disc area continuing the print images is queued for output to the appropriate printer.

'24 - Dump Buffer

Null.

All other function codes are invalid, and the calling program is aborted.

## 10-3 MODEL 47xx (VERSATEC) PRINTERS

The printer/plotter is treated as a spooled output only device under VULCAN and as such all output is transmitted via system spool disc areas.

The printer/plotter operates in two modes, print and plot. In the print mode, which is accessed via symbolic output requests, it behaves as does an ordinary line printer. In the plot mode, which is accessed via binary output, the user's buffer is treated as a bit stream, and designates which bits in that row are to be printed.

## 10-4 PRINTER/PLOTTER I/O FUNCTION CODES

The following function codes are honored.

00 - Status

No action except to wait for completion of last I/O transfer.

02 - Symbolic Output

Data is taken from the user's buffer to fill one print line. The user's buffer is assumed to be three characters per word ASCII. The first column output is assumed to be carriage control and not printed. The carriage control functions are as follows:

| Column 0 | Action |
|---|---|
| 1, P | Skip to top of form. |
| 0, B | Skip one line before printing (Double Space). |
| Any other character | Single Space. |

04 - Binary Output

Data in the user's buffer is assumed to be a bit stream at 24 bits per word. The most significant bits of each word (leftmost) correspond to lower numbered column positions. A bit in a position in the word signifies that the "nib" or dot at that position is to be printed. Multiple lines may be plotted in this manner but care should be taken because at the end of a line, the first bit of the next character position (8-bit byte) starts the subsequent line. Because of this, the safest plotting mode is to write each binary record of length to hold exactly a multiple of 3 lines.

06 - Special Output

Same as 04 - Binary Output.

'12 - Write End-of-File

The message EOF. . is printed.

'13 - Open

The logical file is opened.

'14 - Close

The logical file is closed, the LFN removed from the program, and the system disc area is queued for output to the device.

'24 - Dump Buffer

Null.

All other function codes are invalid and will abort the calling program.

# SECTION XI
# PAPER TAPE I/O

## 11-1 PAPER TAPE READERS

High speed paper tape readers are a system resource under VULCAN and therefore must be allocated via a resource command before using.

Certain backspacing functions which cannot be performed by the device, cause a message of the following form to be output to the operator:

programname : XXX nn

where: "programname" is the program using the physical device "nn",

and "XXX" is the function.

Upon completing the required function, the operator should release the program to continue its execution.

## 11-2 PAPER TAPE READER I/O FUNCTION CODES

The following function codes are honored:

<u>00 - Status</u>

Returns the following:

(A) = Bit 21 set if end-of-file detected.
Bits 15-0 = word count transferred.

<u>01 - Symbolic Input</u>

Leader is bypassed until a non-zero character is detected, leading line feed codes ('212) are ignored, and then data is transferred at 3 characters per word until the word count is complete or an end-of-record code ('215) is detected.

<u>03 - Binary Input</u>

Characters are bypassed until the start-of-record code (Line feed = '212) is detected, the data is transferred beginning with the following character at 4 characters per word into the user's buffer until the word count is complete or an end-of-record code (carriage return - '215) is encountered.

05 - Special Input

No characters are ignored, and data is transferred at 1 character per word into Bits 7-0 of the user's buffer until the requested word count is complete.

'13 - Open

The logical file is opened.

'14 - Close

The logical file is closed, the LFN removed from the program's assignment list, and the device is freed to allow resource allocation by another program.

'15 - Reposition File

The message RPF n is output to the operator and the program suspended pending operator action. "n" is the physical device number.

'16 - Advance File

The tape is spaced forward until an end-of-file code ('204) is detected.

'17 - Backspace File

The message BSF n is output to the operator and the program suspended pending operator action. "n" is the physical device number.

'20 - Advance Record

One logical record (until a carriage return '215) is bypassed.

'21 - Backspace Record

On models having backspace capabilities, one record is read backwards until a start-of-record code ('212) is detected. Otherwise, the message BSR n is output to the operator and the program suspended pending operator action. "n" is the physical device number.

'22 - Rewind

The message REWIND n is output to the operator and the program suspended pending operator action. "n" is the physical device number.

'24 - Dump Buffer

No action.

All other function codes are invalid, and the calling program will be aborted.

## 11-3 PAPER TAPE PUNCHES

High speed paper tape punches are treated as spooled devices and correspondingly all output is transacted via system disc areas.

## 11-4 PAPER TAPE PUNCH I/O FUNCTION CODES

<u>00 - Status</u>

No action except to wait for completion of last operation.

<u>02 - Symbolic Output</u>

Data is transferred from the user's buffer at 3 characters per word until the word count is complete. A start-of-record code (Line feed - '212) is placed on the front of the record and a carriage return ('215) terminates it.

<u>04 - Binary Write</u>

A leading line feed ('212) is punched, followed by the data from the user's buffer, taken at four 6-bit characters per word, until the word count is complete. A carriage return code ('215) terminates the record.

<u>06 - Special Write</u>

Data is taken unmodified at one character per word from Bits 7-0 of the user's buffer until the word count is complete. No other special characters are punched. The specified word count is thus the character count punched.

<u>'12 - Write End-of-File</u>

The character sequence EOT ('204) and carriage return ('215) are punched.

<u>'13 - Open</u>

Opens the logical file. The first such open for the LFN will punch 18 inches of blank leader.

'14 - Close

Closes the logical file, removes the LFN from the programs assignments, and queues the spool disc area for output to the paper tape punch.

18 inches of trailer are punched after all of the user's records.

'24 - Dump Buffer

No action.

All other function codes are invalid, and the calling program is aborted.

# CHAPTER F

# SYSTEM OPERATION

# SECTION I
# OPERATING VULCAN

## 1-1 GENERAL

Operator interaction with VULCAN is achieved through use of a dedicated Model 9050 (TEC 425) Interactive CRT display. This allows the operator additional flexibility through use of full page displays available virtually momentarily. For this reason, the operator CRT is dedicated to operator functions and not available as an interactive terminal. Only Model 9050 CRT's may be used as operator devices.

The operator may enter commands and display system messages on the console CRT. Additionally, user programs may transact I/O via this device, which is available to any program without being resourced. The console CRT is always Physical Device 1.

## 1-2 LOADING VULCAN

This section describes the procedure necessary to load VULCAN into memory for operation.

### 1-2.1 Initial Procedure

a) The CPU should be in the Master Clear state, which is activated by raising and then returning the switch labeled MSTR CLR. The Virtual Memory System (VMS) should be enabled by turning the PROG REST (Program Restrict) key clockwise.

b) The VULCAN Master disc pack should be loaded on the appropriate disc drive. VULCAN will execute regardless of the location of the master pack. The normal location is that which is referenced by the hardware disc boot strap.

c) The operator CRT must be powered on, and any other devices activated that are needed for system operation.

d) The appropriate boot-in options should be set in the console and sense switches as listed in Table 1-1.

e) VULCAN is then loaded into memory via the hardware disc bootstrap. This may be activated by lowering only the Disc Bootstrap switch and then depressing the Boot Enable switch.

TABLE 1-1

VULCAN BOOTSTRAP SENSE SWITCH OPTIONS

| SENSE SWITCH | FUNCTION |
|---|---|
| 1 | Set-use Physical Device number specified in Control Switches 0-7 as new operator CRT. This must be a TEC 425 CRT, and on line.<br><br>Reset - Use default (sysgen) operator device. |
| 2 | Set - Halt after loading VULCAN from disc. This allows patches to be made by Systems Programmers for diagnostic purposes only.<br><br>Reset - Do not halt after loading VULCAN. |
| 3 | Set - Select options set in Control Switches 8-23 as further boot options. See Table 1-2.<br><br>Reset - Ignore Control Switches 8-23. |
| 4 | Set - Inhibit 120 Hz clock from being turned on. Warning: This should be used with caution as VULCAN requires a functioning clock for numerous operations. This switch is intended for debugging by systems programmers only.<br><br>Reset - Enable 120 Hz clock. |

TABLE 1-2

CONTROL SWITCH OPTIONS AT BOOT-IN,

ASSUMING SENSE SWITCH 3 IS SET

| SWITCH | FUNCTION |
|---|---|
| 10 | Set - Do not write date and time on Opcom Disc areas.<br><br>Reset - Write current date and time record on Opcom Input area and Opcom Message Area. (See paragraph 6-1.) |
| 11 | Set - Do not update entries on Permanent Schedule disc area. (Refer to Chapter A.)<br><br>Reset - Update entries on permanent schedule area to current times. |
| 12 | Set - Do not initiate "Ten-second program". This is useful for systems programmers only.<br><br>Reset - Initiate "Ten-second program". |
| 13 | Not currently used. |
| 14 | Set - Do not eliminate left-over input spool disc areas. These contain $DATA and $JOB files. Refer to Chapter A.<br><br>Reset - Eliminate input spool disc areas containing $DATA and $JOB files. |
| 15 | Set - Do not re-output spool disc areas.<br><br>Reset - All output spool areas will be re-queued for output to their appropriate device. |
| 16 | Set - Eliminate Output Spool areas. If this switch is on, switch 15 is ignored.<br><br>Reset - Do not eliminate output spool disc areas, and test switch 15. |
| 17 | Set - Do not eliminate left-over Control Point work disc areas.<br><br>Reset - Eliminate any remaining Control Point work disc area. |

TABLE 1-2
CONTROL SWITCH OPTIONS AT BOOT-IN,
ASSUMING SENSE SWITCH 3 IS SET (CONT'D.)

| SWITCH | FUNCTION |
|---|---|
| 18 | Set - Do not clean up disc areas that are in a pending state.<br><br>Reset - All disc areas that were in process of Generation Elimination, or Renaming are cleaned up as best as possible to allow the operation to be repeated. |
| 19 | Set - Inhibit loading and Initiation of Resident Real-time programs.<br><br>Reset - Initiate all Resident Real-time programs on disc. |
| 20 | Set - Do not load entries marked Core Dictionary into memory.<br><br>Reset - All QDD entries marked as Core Dictionary are loaded into core. |
| 21 | Set - do not load Resident Monitor Common blocks.<br><br>Reset - Load Resident Monitor Common blocks as specified in QDD entry. |
| 22 | Set - Do not load resident monitor services.<br><br>Reset - Load resident monitor and non-resident handler services as specified by Sysgen. |

1-2.2 System Initialization Procedure

After loading from the disc VULCAN configures itself and initializes the system as specified by the System Generation options and the console switches discussed above.

VULCAN will initially output the following message:

```
* * * START VULCAN        00A:DEC    74  * * *
```

Where the "00A:DEC 74" is the version number of the VULCAN system. The VULCAN version numbers are discussed in Chapter A. If this message is not output, and the display lights are flashing in a circular motion, then the bootstrap has loaded in a disc pack which is not the master pack.

The following message may be output next:

TURN ON PROG REST KEY.

If it is output, then the PROG REST Key was not enabled, or there is a problem in the VMS hardware. The boot procedure should be repeated after the problem has been corrected.

Next VULCAN will ask for the date and time. The following message will be output:

ENTER DATE & TIME

The operator should then enter the date and time as follows:

a) First, enter in Column 1 of the top row or the home position, an STX character (displayed as ⊥).

b) Follow this with the last two digits of the year, the month (1-12) and the day of the month separated by blanks or a comma.

c) Next, still on the same line, comes the current time, based on a 24-hour clock, entering separately the hours, minutes and seconds.

For example, 3:30 PM on November 20, 1974 would be encoded as follows:

⊥ 74,11,20,15,30,0

If the format of the data is invalid, or some invalid date or time is entered, the following message is output:

RE-ENTER DATE AND TIME ; FORMAT IS ⊥ YY MM DD HH MM SS.

If the format of the input is good but seemingly inconsistent with the last time and date stored on an accounting disc area, the date and time just entered, along with the following message, is output:

ARE YOU SURE?

If the displayed date and time is indeed valid, the operator should respond with a YES, as follows:

⊥ YES

and then press the XMIT Key.

If the displayed information is not correct, the operator may respond as follows to allow re-input of the date and time:

1 NO

and then depress the XMIT Key.

When VULCAN initialization is complete, the following message is output:

***VULCAN READY***

VULCAN is now operational.

# SECTION II
# OPERATING INFORMATION

## 2-1 GENERAL

This section describes some of the important procedures for operating VULCAN.

## 2-2 OPERATOR COMMANDS

One of the prime capabilities of the operator is to enter the operator commands. These are described in Section III.

In order to enter these commands, an operator must be "signed on". This procedure ensures that an operator has access to only those capabilities deemed necessary for him. For example the master system operator may wish to enter and remove users from the system, and he may refrain from giving this access to any other operator. The operator signs on with the "ON" command. The format of this command is as follows:

ON usernumber

Where usernumber is the user's usernumber or password. If accepted Opcom responds with the following message:

HELLO, username

Where username is the ASCII name stored for the user.

The "ON" command requires access bit 0, and therefore a user may be prevented from entering Opcom commands by not giving him access bit 0 when he is added to the system.

Once signed-on, an operator may enter those commands to which he has access. The Opcom command access bits are listed in summary table 3-1.

When finished, an operator should sign-off with the OFF command. The next operator must then sign-on.

## 2-3 OPERATOR MESSAGES

All system operator messages are transmitted via Opcom. These messages are written to the Opcom Message Disc Area (0000SYST*OPCOM. MA). This disc area may be displayed via the operator commands VM (View Messages) or DM (Display Messages). Following the input of a Display Message command all subsequent messages sent to Opcom are immediately displayed on the CRT screen, in addition to being written to the message disc area. These messages are written to the bottom of the screen and the remainder of the screen scrolled up one line. This allows multiple messages to remain displayed until seen by the operator.

Following the entry of a View Message command, the messages will go only to the disc area and not be displayed immediately. In this case, the "Message" light on the Data-Panel will be turned on when a new message is received, indicating there is a new message that has not yet been seen by the operator. This light is turned off following the VM or DM which outputs the messages.

The major VULCAN messages are listed in Section XVI.

## 2-4 OPERATOR DISPLAY LIGHTS

All VULCAN CRT's should be equipped with the Data-Panel option. This is particularly important for the operator CRT. The positions and colors of each light in this panel are shown in Table 2-1. Unused lights may be accessed by user programs. The meaning of the assigned lights are as follows:

ERROR- This indicates that the last Opcom command entered had an error on it. The light is turned off when a valid command is entered.

ACTIVE- This indicates that Opcom is active. Entering another command during this period will have no effect. The light goes out when Opcom completes its current activity.

MESSAGE- As discussed in Paragraph 2-3 this light indicates that there is an undisplayed Operator message pending on the disc area. The light will go out when the message area is displayed.

TAPE REQUEST- This indicates that there is a request to mount a magnetic tape that is still pending operator action. The light will go out when all tape mount requests have been satisfied.

DISC REQUEST- This indicates that there is a request still pending operator action to mount a disc pack. The light will go out when all disc pack requests have been satisfied.

DEVICE TROUBLE- This indicates that there is a physical device requiring operator intervention. This light will go out when all devices are trouble-free.

INPUT REQUEST- This indicates that some program is requesting input from the operator terminal. Program input must be preceded with a STX ( ⊥ ) character to distinguish it from OPCOM input.

TABLE 2-1
DATA PANEL LIGHT FORMAT

| Light | Color |
|---|---|
| ERROR | red |
| ACTIVE | green |
| | |
| | |
| MESSAGE | clear |
| | |
| TAPE REQ. | orange |
| DISC REQ. | blue |
| | |
| | |
| | |
| I/O TROUBLE | red |
| | |
| | |
| | |
| INPUT REQ. | yellow |

## 2-5 CONTROLLING CARD READERS

Batch card readers are controlled by the operator. In order to read cards the appropriate card reader handler must be loaded. This may be accomplished in one of two ways. a) by declaring the device to be RESIDENT at system generation time, or b) by loading the handler with an Initiate Input (II) command (Section 7).

A card reader may be de-activated through use of the Terminate Input (TI) command.

Once loaded the operator should ensure that each deck input has either a $DATA or $JOB card as the first card. Each deck must have an EOT card (6/9 punch in column 1) as the last card.

If a deck has not been terminated with an EOT card a hopper empty message is output to the operator when the happer becomes empty. This indicates a partial deck has been input.

## 2-6 CONTROLLING TERMINALS

All interactive terminals must be initialized prior to being used. This may be accomplished in one of two ways: a) by declaring the terminal to be RESIDENT at System Generation time or b) by having the operator initialize it with the Initialize Terminal (IT) command. The latter form may be desirable if it will ever be desired to run without terminals for a period of time after booting the system, if for example, it is necessary to run a particular batch job in the minimum amount of time.

## 2-7 RESOURCE CONTROL

One of the major functions of the VULCAN operator is to interact with the automatic resource allocation system. There are two types of devices that the operator will control. These are Magnetic Tapes and Disc Packs. Normally the system will output a message to the operator when a manual function is required requesting the tape or disc drive on which to mount the appropriate volume. The appropriate "TAPE REQ" or "DISC REQ" light is turned on, and the indicated drive awaits operator action.

At this point there are two actions possible. If the requested tape or disc pack is already mounted on the indicated drive the operator should respond with a Tape Ready (TR) or Pack Ready (PR) command (Section VIII) to indicate this to the system. The VULCAN system will then continue execution of the requesting program and extinguish the corresponding REQ light.

If the requested tape or pack is not mounted the operator should mount the requested volume. If the requested item is not available, the operator may abort the program with an Abort Program (AP) keyin. The specific program requesting the device may be determined by entering a Peripheral Status (PS) command for the particular device. (See Section 5.)

As the operator mounts the requested volume the system, in the form of the "ten-second" program notes that the specified device goes off-line for a time, and back on-line when the volume is ready. Once back on-line, VULCAN assumes the requested disc or tape is mounted and program execution continues. The disc pack is checked to ensure the correct pack is mounted. Tapes are checked to ensure that the write ring is inserted if and only if requested. There is no way to ensure that the correct unlabeled tape is mounted. Care should be taken when mounting tapes.

# SECTION III
# OPERATOR COMMANDS

## 3-1 GENERAL

The following sections list the operator commands available under VULCAN Opcom. Table 3-1 summarizes these commands into an alphabetic list for reference. Also listed in this table is the access bit needed to use the command.

All of the commands described are available via the operator CRT. Additionally certain of these, as shown in Table 3-1, are available from interactive terminals and user programs.

## 3-2 COMMAND FORMAT

All operator commands conform to the standard VULCAN Format Scanner's free format input: blanks and commas are used as delimiters. A text parameter including blanks or commas may be input by enclosing the text in double quotes, for example "A,B".

Numeric parameters are assumed decimal, unless preceded by an apostrophe ( ' ) which indicates octal.

All commands require the first two characters. Additional characters are ignored.

TABLE 3-1

ALPHABETIC OPCOM COMMAND SUMMARY

| Command Format | Name | Access Bit | Available From Terminal(X) | See Section |
|---|---|---|---|---|
| AD, pack, disc, parameters | Add Pack | 0 | | 8 |
| AL | Allocator List | 0 | | 14 |
| AO,Pdn | Advance Output | 0 | | 9 |
| AP,program | Abort Program | 17 | X | 4 |
| AQ,user,qualifier | Add Qualifier | 1 | | 12 |
| AS,service | Add Service | 0 | | 11 |
| AU,user,qualifier,parameters | Add User | 1 | | 12 |
| BO,pdn,lines | Backup Output | 0 | | 9 |
| CD,disarea | Clear Disc Area | 0 | | 15 |
| CL | Control point List | 0 | | 10 |
| CX,program parameters | Connect Program to interrupt | 16 | X | 13 |
| CP,number | Control Point Numbers | 0 | | 10 |
| DA | Disc Area Map | 0 | | 5 |
| DD,area,sectors | Disc Dump | 0 | | 14 |
| DD,sector,pack | | | | |
| DM | Display Messages | 0 | | 6 |
| DT | Date/Time | 14 | X | 5 |
| DX,program,parameters | Disconnect program from interrupt | 16 | X | 13 |
| EI,n,pdn | Eliminate Input | 0 | | 6 |
| EM,n,pdn | Eliminate Messages | 0 | | 6 |
| ES,service | Eliminate Service | 0 | | 11 |
| EX,interrupt | Enable External Interrupt | 16 | X | 13 |
| GA,address | Get Address | 0 | | 14 |
| HO,pdn | Hold Output | 0 | | 9 |
| HP,program | Hold Program | 15 | X | 4 |
| II,pdn | Initialize Input | 0 | | 7 |
| IP,program,parameters | Initiate Program | 16 | X | 4 |
| IT,pdn | Initialize Terminal | 0 | | 7 |
| IX,interrupt | Inhibit External Interrupt | 16 | X | 13 |
| JQ | Job Queue display | 0 | | 10 |
| KO,pdn | Keep Output | 0 | | 9 |
| LS,service | Load Service | 0 | | 11 |
| MD,sector,word,old,new | Modify Disc | 0 | | 14 |
| MO,pdn,copies | Multiple Outputs | 0 | | 9 |
| MP,program,word,old,new | Modify Program | 15 | X | 14 |
| NP,pack,model,parameter | New Pack | 0 | | 0 |
| OFF | Sign-off | 0 | | 2 |
| ON,user | Sign-on | 0 | | 2 |
| PA,address | Patch Address | 0 | | 14 |
| PD,pdn | Peripheral Down | 0 | | 7 |
| PL | Program List | 0 | | 7 |

TABLE 3-1 (Cont'd.)

ALPHABETIC OPCOM COMMAND SUMMARY

| Command Format | Name | Access Bit | Available From Terminal(X) | See Section |
|---|---|---|---|---|
| PR,packnumber | Pack Ready | 0 | | 8 |
| PS,pdn | Peripheral Status | 0 | | 5 |
| PV,pdn | Peripheral Up | 0 | | 7 |
| QP,program | Query Program | 14 | X | 4 |
| QS,service | Query Service | 0 | | 11 |
| RO,pdn | Release Output | 0 | | 9 |
| RP,program | Release Program | 15 | X | 4 |
| RQ,user,qualifier | Remove Qualifier | 1 | | 12 |
| RS,service,service | Rename Service | 0 | | 11 |
| RU,user | Remove User | 1 | | 12 |
| SD,sector<br>SD,areaname,sector | Sector Display | 0 | | 14 |
| SL | Service List | 0 | | 11 |
| SQ,pdn | Spool Queue | 0 | | 9 |
| TI,pdn | Terminate Input | 0 | | 7 |
| TO,pdn | Terminate Output | 0 | | 9 |
| TP,program | Terminate Scheduling | 16 | X | 4 |
| TR,pdn | Tape Ready | 0 | | 8 |
| TS,n | Time Slice | 0 | | 15 |
| US,service | Unload Service | 0 | | 11 |
| VD | View Down | 0 | | 6 |
| VI | View Input | 0 | | 6 |
| VL | View List | 0 | | 6 |
| VM | View Messages | 0 | | 6 |
| VU | View Up | 0 | | 6 |
| WP,program | Wakeup Program | 16 | X | 4 |

# SECTION IV
# PROGRAM CONTROL COMMANDS

## 4-1 GENERAL

These commands are designed to provide the basic control capabilities. All commands listed in this section are re-entrant and thus available from terminals or user programs also. The only restriction is that they may be used on only those programs initiated at the terminal making the call.

## 4-2 ABORT PROGRAM

The Abort Command is used to abort any program, either by name, terminal, or control point designation. The format is one of the following:

a) AP,program,terminal

b) AP,pdn

c) AP,x

Format a) is used to abort any program. The program name field must be a standard disc area name. The terminal designation is used to distinguish programs if there exist more than one with the same name. The highest priority program with a matching name is aborted if the terminal identification is not supplied.

Format b) is used to abort whatever program is on the specified terminal. The pdn field must be a valid Physical Device Number.

Format c) is used to abort whatever program is on the specified Control Point. The x field must be a valid control point identification letter.

Examples:

```
AP,1234ABCD*XYZ
AP,42              (Terminal 42)
AP,A               (Control point A)
AP,*JOBCNTRL,42
```

## 4-3 QUERY PROGRAM STATUS

The Query Program command is used to test the status of any program, either by name, by terminal, or by control point designation. The formats are as follows:

a) QP,program,terminal

b) QP,pdn

c) QP,x

Format a) is used to test any specified program name. The program field must be a standard disc area name format. The terminal designation is an optional peripheral device number which is used to distinguish programs if more than one exist with the given name. If no terminal designation is supplied the highest priority program with a matching name is queried.

Format b) is used to test the highest priority program connected to the specified terminal peripheral device number. The pdn field must be a valid terminal peripheral Device Number.

Format c) is used to test whatever program is currently executing at the specified Control Point. The x field must be a valid control point identification letter.

The output of the query is a single line written to Logical File Number 3 of user programs, or to the top line of the CRT for Opcom. This output consists of the following:

1) Program name

2) Diagnostic PDN or Control Point letter.

3) Priority

4) Status codes, as listed below:

| | |
|---|---|
| RUN- | Ready to execute |
| WAIT- | Waiting for a flag word to become non-negative. The monitor address of and contents of this flag word is output on the line following the status codes. |
| DISC- | Waiting for system disc transfer to complete. |
| SUSP- | Suspended, Requires operator release. |
| DELAY- | Waiting for clock interval to elapse before continuing. |
| ABTIN- | Program is abort-inhibited. |
| ABORT- | Program is pending an abort. |

| | |
|---|---|
| ABOK- | Program is in a wait state that allows aborting unless ABTIN is set. |
| ABTDN- | Program has aborted. |
| ABAB- | Program has aborted during an abort sequence. |
| ABTX- | Do not exit after abort. Used to return control to interactive terminals following an abort. |
| RSRC- | Program has allocated a resource. |
| LOAD- | Program is in start-up sequence. |
| EOP- | Program is in exit sequence. |
| EXIT- | Program is in final stages of exit sequence. |
| SAU- | Program has caused SAU overflow/underflow trap. |
| EXTRP- | Program has caused Executive Trap (stall alarm, limit violation, instruction trap). |
| SPACT- | Program will execute special code based on the SAU or EXTRP status. |

5) If in wait state, address and contents of wait flag word.

Examples:

Sample commands:

```
QP,1234ABCD*XYZ
QP,42         (Terminal 42)
QP,B          (Control Point B)
QP,*JOBCNTRL,42
```

The following are sample output lines:

```
JOBCNTRL   TR=42   PR=30   WAIT   AB-OK   31715   42000052
XYZ        TR=1    PR=54   RUN
VASSEM     TR=A    PR=10   LOAD
```

## 4-4 HOLD PROGRAM

The Hold command is used to suspend the execution of the specified program, terminal, or control point, by placing the suspend bit on the program. The program requires a release command to continue execution. The formats are as follows:

HP,programname,terminal
HP,pdn
HP,x

Format a) is used to suspend any specified program by name. The programname field must be of standard disc area name format. The terminal designation is an optional peripheral device number which is used to distinguish programs if more than one exist with the given name. The highest priority program with a matching name is held if the terminal designation is absent.

Format b) is used to suspend the highest priority program connected to the specified terminal peripheral device number. The pdn field must be a valid terminal Peripheral Device Number.

Format c) is used to suspend whatever program is currently executing at the specified Control Point. The x field must be a valid control point identification letter.

Examples:

HP,1234ABCD*XYZ
HP,*JOBCNTRL,42
HP,42 (Terminal 42)
HP,A (Control Point A)

## 4-5 RELEASE PROGRAM

The Release command removes the suspended status of the specified program, terminal, or control point, and allows execution to continue.

It is used to remove the effect of the Hold command described above, and is also used if another system routine or the program itself sets the suspend status. The formats are as follows:

RP,programname,terminal
RP,pdn
RP,x

Format a) is used to release any specified program by name. The programname field must be of standard disc area name format. The optional terminal designation is a peripheral device number which is used to distinguish programs if more than one exist with the given name. If the terminal designation is not supplied the highest priority program with a matching name is released.

Format b) is used to release the highest priority program connected to the specified terminal peripheral device number. The pdn field must be a valid Peripheral Device Number.

Format c) is used to release whatever program is currently executing at the specified Control Point. The x field must be a valid control point identification letter.

Example:

```
RP,1234ABCD*XYZ
RP,*JOBCNTRL,42
RP,42          (Terminal 42)
RP,C           (Control point C)
```

## 4-6 INITIATE PROGRAM

The Initiate command is used to place the specified monitor or real-time program into execution. The command allows specification of starting time and date, and repetitive frequency for initiation. The format is as follows:

IP,programname,parameters

The "programname" field is the program name which is to be initiated, in standard disc area name format.

The "parameters" are as follows:

PRI=n — Specifies the execution priority (0-63). If not specified catalogued priority is used.

PAR=n — Initiation parameter. Any 24-bit integer. If not specified, zero is used.

FREQ=nM — Frequency of re-initiation. It is specified as n (integer) units of M, where M one of the following:

TICKS
SECONDS
MINUTES

HOURS
DAYS

| | |
|---|---|
| | If not specified, no repeat initiation will be done. |
| DAYS=n | Number of days to wait for first initiation. If not specified, 0 is assumed. |
| HO=n | Hour of day for first initiation. If not specified, current hour is used (0-23). |
| MI=n | Minute of hour for first initiation. If not specified, current minute is used (0-59). |
| SE=n | Second of minute for first initiation. If not specified, current second is used. (0-59) |
| PERM | Permanent flag. The specified schedule operation will be saved across system loads. |

Examples:

a) To initiate 1234ABCD*XYZ immediately at its catalogued priority.

IP,1234ABCD*XYZ

b) To initiate 0000SYST*CAT at noon today and permanently every hour thereafter, at priority 43:

IP,*CAT,PR=43,HO=12,MI=0,SE=0,FREQ=1HOUR,PERM

## 4-7 WAKEUP PROGRAM

The Wakeup command is used to trigger the specified monitor or real-time program out of the sleep state into execution. The command allows specification of wakeup time and date, and optional repetitive frequency of wakeup. The format is as follows:

WP, programname, parameters

The "programname" field is the programname to be awakened. It is in standard disc area name format. If the specified program is not executing or not sleeping when the trigger occurs, the wakeup is ignored.

The parameters are as follows:

| | |
|---|---|
| PAR=n | Specifies the parameter to be passed to the program when awakened, as any 24-bit integer. If not specified, zero is used. |

| | |
|---|---|
| FREQ=nM | Frequency of repeat wakeups. It is specified as n units of M, where M is one of the following:<br>TICKS<br>SECONDS<br>MINUTES<br>HOURS<br>DAYS<br>If not specified, no repeat wakeup is done. |
| DAYS=n | Number of days to wait for first wakeup. If not specified zero is assumed. |
| HO=n | Hour of day for first wakeup. If not specified current hour is used.(0-23). |
| MI=n | Minute of hour for first wakeup (0-59). If not specified, current minute is used. |
| SE=n | Second of minute for first wakeup. If not specified current second is used (0-59). |
| PERM | Permanent flag. Indicates that the specified operation will be saved across system loads. |

Examples:

a) To wakeup program 1234ABCD*XYZ immediately:

    WP,1234ABCD*XYZ

b) To wakeup program 0000SYST*CAT every 2 seconds beginning immediately:

    WP,*CAT,FREQ=2SEC

## 4-8 TERMINATE SCHEDULING

The Terminate command is used to remove the entry in the schedule table for the specified program. This will remove the effect of the previous Initiate (IP) or wakeup (WP) command for the specified program. The format is as follows:

    TP,programname

Where "programname" is the disc area name of the specified program. The entry is removed regardless of its permanent status.

Examples:

TP,1234ABCD*XYZ

TP,*CAT

# SECTION V
# SYSTEM STATUS COMMANDS

## 5-1 GENERAL

These commands are for the operator's use only to query the status of various system components. Only the Date/Time command listed below is available from user programs or terminals.

## 5-2 PROGRAM LIST

The Program List command displays to the CRT screen a list of all of the currently executing programs. The format is as follows:

PL

Each executing program produces one line of output consisting of the following:

1. Program qualifier
2. Program areaname
3. Terminal or diagnostic PDN
4. Priority
5. Current executing address (P-register value)
6. Program status word
7. Program Control Area Address

The interpretation of the status word is done via the Query Program (QP) command (Section 4). It should be noted that a status word of all zeroes is an executing program. The P-register value and program Control Area address are supplied for systems programmers only.

Example:

The following is a sample program list. The circled numbers refer to items listed above.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0000SYST | V:OPCM·V | 1 | 60 | P=016164 | 00000000 | PCA=001410 |
| 0000SYST | V:TENS:V | 1 | 59 | P=016512 | 00000200 | PCA=046310 |
| 0000SYST | V:EXEC:V | 1 | 34 | P=016164 | 00000200 | PCA=001312 |
| 0000SYST | JOBCNTRL | 42 | 30 | P=016164 | 00000200 | PCA=032627 |
| 0000SYST | V:IDLE:V | 1 | 255 | P=001606 | 00000000 | PCA=001506 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

## 5-3 DISC AREA MAP

The Disc Area Map command outputs a listing of all disc areas in the Master Disc Directory, by qualifier. The basic format is to list all of the corresponding disc areas under the respective qualifier. The listing is written to the System List Device. The command format is as follows:

DA

Items for each area include starting sector number, granule size, access bits, type (T:nnn), and pack (P: n). The meaning of the various bits in the access bits and type field is normally of use only to systems programmers.

## 5-4 DATE/TIME

The Date and Time command is used to output the current date and time in ASCII. It is output to the top line of the Opcom CRT. For user programs or terminals, the command will output to LFN 3. The format is as follows:

DT

The output will be in the following format:

dd Mon YY HH:MM:SS

For example,

17 JUL 74 13:30:00

## 5-5 PERIPHERAL STATUS

The Peripheral Status command is used to output the status of a selected physical device. The format is as follows:

PS,pdn

where "pdn" is a physical device number.

For interactive terminals devices, the output is one of the following:

a) "FREE" if the terminal is not in use

b) a user name of the user signed-on to the terminal if in use.

For resource devices, the output will be either "FREE" if not in use, or one of the following if allocated:

a) If allocated to a real-time program, the program name is output

b) If allocated to an interactive terminal, the output will be "TERMnn" where nn is the terminal PDN.

c) If allocated to a control point, the control point identification letter and job name are output.

For all other devices, the output will be either "FREE" if not in use, or "ACTIVE" if the specified device is busy.

# SECTION VI
# OPCOM DISPLAY COMMANDS

## 6-1 GENERAL

These commands are used to display and control the various Opcom disc areas. There are 3 basic disc areas.

The Input area, 0000SYST*OPCOM. OA, contains a record of each Opcom command entered from the operator CRT. The disc area continually expands until shortened by an Eliminate Input (EI) command.

The Message area, 0000SYST*OPCOM. MA, contains each of the messages written to the operator. It, too, grows continually until shortened by an Eliminate Message (EM) command. When the VULCAN system is loaded from disc, the date and time entered is also recorded on these areas for reference purposes.

The List area, 0000SYST*OPCOM. LA, contains the output of certain Opcom commands which might produce too much output to be seen on the CRT at a single time. These commands include the Allocator List (AL), Job Queue (JQ), and Spool Queue (SQ) displays.

## 6-2 VIEW INPUT

The View Input command will display to the screen the last 23 lines of the Opcom Input area, containing the operator commands that have been input. The format is as follows:

VI

The repeater command (Section 6-8) is then set to a View Down (VD).

## 6-3 VIEW MESSAGES

The View Messages command will display to the screen the last 23 lines of the Opcom Message area, containing the operator messages. The format is as follows:

VM

The repeater command (Section 6-8) is then set to a View Down (VD).

## 6-4 DISPLAY MESSAGES

The Display Message command will display to the screen the last 23 lines of the Opcom Message area, exactly as the View Message command. Additionally however, it sets a mode as discussed in Section 2-3 such that subsequent messages will be immediately displayed in addition to being written to the message disc area. The format is as follows:

DM

The repeater command (Section 6-8) is then set to a View Down (VD).

## 6-5 VIEW LIST

The View List command displays the last 23 lines of the Opcom List area, containing the output from long display commands. The format is as follows:

VL

The repeater command (Section 6-8) is then set to a View Down (VD).

## 6-6 VIEW DOWN

The View Down command will display to the CRT the 23 lines previous to those displayed by the preceding view or display command. It is used to back down a disc area, in order to view it in reverse. The format is as follows:

VD

The repeater command (Section 6-8) is set to a View Down also, unless the front of the disc area was encountered, in which case the repeater is set to a View Up (VU).

## 6-7 VIEW UP

The View Up command will display to the CRT the 23 lines immediately following those displayed by the preceding view or display command. It is used to display forward through a disc area. The format is as follows:

VU

The repeater command (Section 6-8) is set to a View Up, unless the end of the disc area was detected, in which case the repeater is set to a View Down (VD).

## 6-8 REPEATER

The Repeater command, which is entered as a blank line, is used to continue scanning through an Opcom disc area. It always represents either a View Up or View Down as discussed above. When an end of the specified disc area is detected, the command is changed to view in the other direction, as for View Down and View Up above.

## 6-9 ELIMINATE INPUT

The Eliminate Input command is used to remove some or all of the records of the Opcom Input area. Optionally, the removed records may be copied to an output device. This removal should be done occasionally as needed to prevent the disc area from becoming too large. The format is as follows:

EI,n,pdn

where "n" is the number of lines to be eliminated, and

"pdn" is an optional specification. If present it is the physical device to which the records are copied as they are deleted. For example, a printer for hard copy records.

Example:

To remove the first 200 lines of the Input area and copy them to printer 6:

EI,200,6

## 6-10 ELIMINATE MESSAGES

The Eliminate Message command works similar to the Eliminate Input command described above except that it is used to remove some or all of the records of the Opcom Message area. Optionally, the removed records may be copies to an output device. This elimination process should be done occasionally to prevent the disc area from becoming too large. The format is as follows:

EM,n,pdn

where "n" is the number of lines to be eliminated, and

"pdn" is an optional specification giving the physical device to which the records are to be copied as they are deleted.

Example:

To remove the first 300 lines of the Message Area, and print them on printer 6:

EM,300,6

# SECTION VII
# GENERAL I/O CONTROL COMMANDS

## 7-1 GENERAL

These commands are used by the operator for general I/O device control.

## 7-2 PERIPHERAL DOWN

The Peripheral Down command is used by the Operator to mark a device down and unavailable for general use. It generally requires a Peripheral Up command to clear it. The format is as follows:

PD,pdn

where "pdn" is the specific Physical Device Number.

If a resource type device (i.e., magnetic tape drive) is marked down, it becomes unavailable for any future allocation. If an output device (i.e., line printer) is marked down, output destined for it is held, unless it can be output to another device through use of the Alternate PDN capability (Chapter A).

## 7-3 PERIPHERAL UP

The Peripheral Up command is used to return a "down" device to its normal available status.

The format is as follows:

PU,pdn

where "pdn" is the specific physical device number.

## 7-4 INITIATE INPUT

The Initiate Input command is used to start input spooling transmissions from card readers. This command ensures that the card reader handler is loaded, and initiates I/O transfers for the specified device(s). The formats are as follows:

a) II,pdn,pdn,...

b) II,ALL

Format a) is used to initiate one or more specific pdn's, as listed on the command. Format b) is used to initiate all card reader devices.

Examples:

II,7 (PDN 7)
II,ALL
II,7,8 (PDN's 7,8)

## 7-5 TERMINATE INPUT

The Terminate Input command is used to terminate input operations from card readers. This command terminates the current spool input operation immediately, and unloads the programs doing the input from the specific device(s). The formats are as follows:

a) TI,pdn,pdn
b) TI,ALL

Format a) is used to terminate one or more specific peripheral devices, as listed on the command. Format b) is used to terminate input from all card reader devices, and unload the card reader handler.

If input is terminated while a card deck is being read, it is lost.

## 7-6 INITIATE TERMINALS

The Initiate Terminal command is used to enable one or more interactive terminals (Teletypes or CRT's) for interactive operation under VULCAN. The required handlers are loaded, and the specific device(s) reset and enabled. The formats are as follows:

a) IT,pdn,pdn,...
b) IT,ALL

Format a) is used to initiate one or more specific physical devices, as listed on the command. Format b) is used to initiate all interactive terminal devices generated into the system.

Example:

IT,20,21 (Terminals 20, 21)

IT,ALL (All terminals)

# SECTION VIII
# RESOURCE CONTROL COMMANDS

## 8-1 GENERAL

These commands are used by the system operator to interact with the automatic resource allocation mechanisms. Refer also to paragraph 2-7, which further discusses resource control.

## 8-2 PACK READY

The Pack Ready command is used to notify the resource control routines that the requested disc pack is already mounted on the indicated drive. This command is not needed if the operator mounts the pack. The format is as follows:

PR,discdrive

where "discdrive" is the disc number which is requesting the new pack.

Example:

PR,2 for disc 2

## 8-3 TAPE READY

The Tape Ready command is used to notify the resource control routines that the requested tape is already mounted on the indicated physical device. This command is not needed if the operator actually mounts the tape. The format is as follows:

TR,pdn

where "pdn" is the Physical Device on which the tape is requested and ready.

## 8-4 NEW PACK

The New Pack command is used to initialize a new disc pack for VULCAN usage. The command allocates a drive, clears the pack, and catalogues the pack number in the disc pack identification disc area. The format is as follows:

$$\text{NP,pack,} \begin{bmatrix} \text{disc drive} \\ \text{model} \end{bmatrix} \text{,parameters}$$

| | | |
|---|---|---|
| where | "pack" | is the disc pack number to be initialized. |
| | "disc drive" | is used when the desired pack is already mounted. The format is<br>D=n<br>where "n" is the disc drive number. |
| | "model" | is used to specify to the system the type of disc drive required to mount the pack. The entry is a 4-digit disc model number. |
| | the "parameters" | consist of the following: |

8-4.1 Permanent

If the New Pack is to be permanently mounted on the system the permanent parameter is supplied. The format is as follows:

P

8-4.2 User Restricted

If only one user is allowed to access the new pack then the user restriction parameter must be used:

U=usernumber

where "usernumber" is the user number of the desired user. The user restriction parameter cannot be used if the account restriction parameter (below) is used.

8-4.3 Account Restricted

If it is desired to restrict access to the disc pack to users with a specific account the account restriction parameter is used:

A=account

where "account" is a 4-digit account number that users must be signed-on under in order to access the pack. The account restriction parameter may not be used when the user restriction parameter is specified.

8-4.4 Examples

a) Initialize pack 3 which is on disc 4.

NP,3,D=4

b) Initialize a 10 MB cartridge disc pack as pack 5 and restrict usage to members of account 2200.

NP,5,5208,A=2200

c) Initialize the fixed platter of a cartridge disc, which is to be pack 3 on disc 3.

NP,3,D=3,P

## 8-5 ADD PACK

The Add Pack command is used to bring a disc pack from another VULCAN system into the current one. This pack may be from another site, or merely a prior VULCAN system. The command allocates a drive, or locates the pack if mounted, and merges the entries in the pack disc directories into the master disc directories. The format is as follows:

$$\text{AD,pack,} \begin{bmatrix} \text{disc drive} \\ \text{model} \end{bmatrix} \text{,parameters}$$

where "pack" is the disc pack number to be added.

"disc drive" is used when the desired pack is already mounted. The format is

D=n

where "n" is the disc drive number on which the pack is mounted.

"model" is used to specify to the system the type of disc drive required to mount the pack. The entry consists of a 4-digit disc model number.

"parameters" are discussed in the following paragraphs:

### 8-5.1 Permanent

If the pack being added is to be permanently mounted on the system the permanent parameter is supplied. The format is as follows:

P

8-5.2 User Restricted

If only one user is allowed to access the specified pack then the user restriction parameter must be used:

U=usernumber

where "usernumber" is the user number of the desired user. The user restriction parameter cannot be used if the account restriction parameter (below) is specified.

8-5.3 Account Restricted

If it is desired to restrict access to the disc pack being added to users with a specific account, the account restriction parameter is used:

A=account

where "account" is a 4-digit account number that users must be signed-on under in order to access the pack. The account restriction parameter may not be used when the user restriction parameter is entered.

8-5.4 Examples

a) Add pack 3 which is on disc 3 currently.

AD,3,D=3

b) Add pack 5 for user XYZ only, which is a 10 MB cartridge pack.

AD,5,5208,U=XYZ

# SECTION IX
# OUTPUT SPOOL CONTROL COMMANDS

## 9-1 GENERAL

These commands are used to control output spooling functions.

## 9-2 HOLD OUTPUT

The Hold Output command is used to suspend the output spool operation on the specified physical device. If currently active, output is held after the end of the current record. If free, no new operation is begun. The format is as follows:

HO,pdn

where "pdn" is the Peripheral Device Number of the device to hold.

Example:

Hold output on printer 6:

HO,6

## 9-3 RELEASE OUTPUT

The Release Output is used to continue suspended output spool operations on the specified physical device. This may be to continue operations suspended either by the Hold Output command, or a device invoked hold state. The format is as follows:

RO,pdn

where "pdn" is the Peripheral Device Number to release.

## 9-4 TERMINATE OUTPUT

The Terminate Output is used to terminate the output spooling operation for the current area being output to the specified device. The spool area is eliminated if appropriate, and the output of the next area, if there is one is begun. The format is as follows:

TO,pdn

where "pdn" is the Physical Device Number for which to terminate the output.

## 9-5 ADVANCE OUTPUT

The Advance Output command is used to bypass a number of records being output to the specified physical device. The format is as follows:

AO,pdn,lines

where "pdn" is the physical device number and

"lines" is the number of records to skip. If skipping causes the entire remainder of the area to be bypassed, the effect is that of the Terminate Output command above.

## 9-6 BACKUP OUTPUT

The Backup Output command is used to re-output a number of records from the specified physical device. The number of records specification is approximate. The format is as follows:

BO,pdn,lines

where "pdn" is the physical device to back up, and

"lines" is the approximate number of records to be re-output. If "lines" is not entered, the entire disc area is output. To accurately reposition a spool output area, a Hold Output (HO) followed by a Backup Output (BO) will restart the disc area. Then an Advance Output (AO) with a specified number of records to advance will accurately position the output area.

Examples:

a) Reprint approximately 100 lines on printer 6:

BO,6,.100

b) Repunch the entire card deck being output on card punch 8:

BO,8

## 9-7 KEEP OUTPUT

The Keep Output command is used to prevent a spool disc area from being eliminated after it has been output. The format is as follows:

KO,pdn

where "pdn" is the physical device that is currently outputting the spool area to be preserved.

## 9-8 MULTIPLE COPIES

The Multiple Output command is used to output multiple copies of the current file being output to the specified device. The format is as follows:

MO,pdn,c

where "pdn" is the physical device that is currently outputting the disc area of which multiple copies are desired.

"c" is the desired number of copies, including the current one. If this command is repeated after some copies have been output, "c" extra copies will be output. Thus, whenever the command is entered, "c"-1 additional copies are output.

This command has no effect on spool operations to interactive terminals.

Examples:

To print 4 copies of the current area on printer 6:

MO,6,4

## 9-9 SPOOL QUEUE DISPLAY

The Spool Queue command causes the contents of the spool output queue for either a specified device or for all devices to be output to the Opcom List Area. The last 23 lines of this list are then displayed and the repeater command set to continue displaying the entries. The format is as follows:

SQ,pdn

where "pdn" is an optional specification indicating the physical device for which the spool queue entries are to be displayed. If the command has no parameters (no "pdn") the entire spool queue for all devices is output.

The output consists of the following:

1) Physical device to which area is destined.
2) Disc area name to be spooled.
3) Originating program, terminal number, or control point job name.
4) User name.

# SECTION X
# CONTROL POINT CONTROL COMMANDS

## 10-1 GENERAL

These commands are used to interact with the control point programs in VULCAN.

## 10-2 SET NUMBER OF CONTROL POINTS

The Control Point Number command is used to interact with the automatic control point determination logic. It may turn this mechanism on or off, or it may select the manual number of control points. The format is as follows:

CP,n

where "n" is as follows:

if n is not specified (a null field), then the automatic control point mechanism will determine the number of control points.

if n is specified as zero, then no control points will be run.

if n is positive, then this is the maximum number of control points that will be run.

Examples:

a) To run a maximum of 3 control points:

CP,3

b) To let the automatic control point system determine the number of control points:

CP

## 10-3 CONTROL POINT LIST

The Control Point List command is used to display the status of control points. The format is as follows:

CL

The output consists of the following:

1. Control Point Letter
2. Priority of jobs which can be run by this control point
3. Job name if the control point is active
4. User name if the control point is active
5. Status of the program currently executing at that control point

Also, output is the number of jobs waiting in the jobs-to-be-run queue, the number of control points allowed, and whether this number has been manually entered or automatically determined by the system.

## 10-4 JOB QUEUE DISPLAY

The Job Queue command is used to display the contents of the Control Point jobs-to-be-run queue. The format is as follows:

JQ

The contents of the job queue are written to the Opcom List Area, and the last 23 lines are automatically displayed using the View List (VL) command. This queue listing consists of job name, user name, and priority of the Job.

# SECTION XI
# NON-RESIDENT MONITOR SERVICE CONTROL COMMANDS

## 11-1 GENERAL

These commands are used to query and control the non-resident monitor handlers and monitor service routines, such as device handlers and background services. All such routines have names which have "V:" as the first two characters and ":V" as the last two characters.

All service names listed below are four ASCII characters. The "V:" and ":V" portions of the disc area name should not be included. A complete list of system services is provided in Chapter G.

## 11-2 QUERY SERVICE

The Query Service command is used to display the status of a specific service. The format is as follows:

QS,service

where "service" is the specified service name. The output lists the status of the service, and, if loaded, the memory location and a list of the programs using it.

Example:

To test the Mag Tape Handler:

QS,MAGT

## 11-3 SERVICE LIST

The Service List command is used to list all services currently loaded, and the memory address at which they are loaded. Two columns are output, with service name followed by address in each column. The command format is as follows:

SL

## 11-4 ADD SERVICE

The Add Service command is used to locate a service on the disc and set up the internal pointers necessary so that it can be loaded from disc via the normal mechanisms. The service must be catalogued on the disc as a Non-Resident Handler or a Monitor Program. The format is as follows:

AS,service

where "service" is the specific service name.

## 11-5 ELIMINATE SERVICE

The Eliminate Service command is used to remove the disc pointers to a service from the resident list, and eliminate the service from disc. This will have no effect on the copy currently in memory, if any. The format is as follows:

ES,service

where "service" is the service to be eliminated.

## 11-6 RENAME SERVICE

The Rename Service command is used to change both the resident name and disc area name of a service. The resident pointers to the disc area, and the version loaded in memory if any, are not effected. The old service name becomes undefined. The format is as follows:

RS,oldservice,newservice

the "oldservice" name is changed to "newservice".

Example:

To change the name of monitor service MAGT to OMAG:

RS,MAGT,OMAG

## 11-7 LOAD SERVICE

The Load Service command causes the specified service to be loaded into memory, if not already resident, and remain resident until unloaded by the operator. The format is as follows:

LS,service

where "service" is the service to be loaded.

## 11-8 UNLOAD SERVICE

The Unload Service command is used to release a service previously loaded so that it can be unloaded when no longer needed by any program which may be currently using it. It removes the effect of a Load Service command. The format is as follows:

US,service

## 11-9 CHANGING A NON-RESIDENT HANDLER

The following procedure may be used to generate a new version of an existing non-resident service.

a) Enter Opcom command LS,service to ensure any references to it are satisfied while it is being changed.

b) Save the old copy with the command: RS,service,newname.

c) Generate a new copy of "service" under the qualifier 0000SYST and catalogue as a Non-Resident Handler or Monitor Program on the master disc pack.

d) Cause the system to find the new version of the service with the command: AS,service.

e) Unload the old version which is still resident with the US command. This allows the new version to be used.

Example: To change the Magnetic tape Handler:

a) LS,MAGT

b) RS,MAGT,OMAG

c) generate new 0000SYST*V:MAGT:V on pack 1

d) AS,MAGT

e) US,MAGT

# SECTION XII
# USER NUMBER CONTROL COMMANDS

## 12-1 GENERAL

It is recommended that these commands be used only by the Master System operator to add new users to the system, to change the user's privileges, and to remove users.

User numbers are used to achieve privacy of disc areas and access capabilities under VULCAN. Each user must have a unique number which identifies him to the system, and which defines his qualifiers and access information.

A user number can have one of two forms. First it can be a string of from one to six ASCII characters, the first of which must be alphabetic. Examples: ABC, A123, U123, A1B2C3.

The second form is a binary integer of from one to twelve decimal digits. Examples: 1, 12345678, 389495145. It is recommended that a site choose one system or the other as random conflicts may occur if user numbers are chosen from both systems.

## 12-2 ADD USER

The Add User command is used to sign a new user on to the system. The user is added with a qualifier and access parameters as discussed below. The format is as follows:

AU,userno,qualifier,name,parameters

where: "userno" is the user's user number. This should be kept secret to achieve maximum security. Each user number is unique in a system.

"qualifier" is the user's sign-on qualifier, consisting of from 1 to 4 decimal digits representing the account followed by 1 to 4 alphabetic characters representing the identifier.

"name" will be the internal ASCII name stored for the user. A maximum of 12 characters is stored.

"parameters" give access information as follows:

| | |
|---|---|
| L=n | Maximum access Level will be "n". Range is 0-15. Default is 0. |
| A=n | User's Access bit mask will be "n". The default is '77777777. |
| S=n | The maximum program Size for this user will be "n" pages. Default is 64. |
| T=n | The maximum Time-limit for this user's control point jobs is "n" seconds. Default is the time-limit sysgen parameter. |
| O=n | The maximum spool Output lines for this user is set to "n" lines. The default is the output lines sysgen parameter. |
| P=n | The maximum Priority this user may execute at is set to "n". The default is 63. |

Examples:

AU,123,2200TEST,SMITH,A='77777700,T=120

AU,PASS,1234ABCD,"MR JONES",A='00707177

AU,U,12XY,"G. HARRIS".

## 12-3 ADD QUALIFIER

The Add Qualifier command is used to add a sign-on qualifier to a specified user. The qualifier must consist of a numeric account and an alphabetic identifier. The format is as follows:

AQ,user,qualifier

where "user" must be a user number that is currently on the system

"qualifier" is the qualifier to be added

Example:

AQ,123,1200TEST

AQ,PASS,1234VWXY

AQ,U,1AB

## 12-4 CHANGE USER

The Change User command is used to change any of the access parameters for a specified user. The format is as follows:

CU,userno,parameters

where "userno" is the user number of the specific user and

"parameters" are as listed below (Same as for Add User).

| | |
|---|---|
| L=n | Changes access Level to "n". Range is 0-15. |
| A=n | Changes the user's Access bit mask to "n". |
| S=n | Changes the maximum program Size for the user to "n" pages. |
| T=n | Changes the maximum Time-limit for the user to "n" seconds. |
| O=n | Changes the maximum spool Output line limit to "n" records. |
| P=n | Changes the maximum Priority at which this user may execute to "n". Range is 0-63. |

Examples:

CU,1,T=300,A='76777700

CU,PASS,A='007777177

CU,U,T=120, 0=5000, P=60

## 12-5 REMOVE QUALIFIER

The Remove Qualifier command is used to remove a sign-on qualifier from a specified user. The format is as follows:

RQ,userno,qualifier

where "userno" is the user's user-number or password

"qualifier" is the qualifier to be removed.

Examples:

RQ,123,2200TEST

RQ,PASS,1234VWXY

RQ,U,1AB

## 12-6 REMOVE USER

The Remove User command removes a user and all of his sign-on qualifiers. The format is as follows:

RU,userno

where "userno" is the user's user number.

Examples:

RU,123

RU,PASS

RU,U

# SECTION XIII
# INTERRUPT CONTROL COMMANDS

## 13-1 GENERAL

These commands are intended to allow the operator to control program connection to external interrupts not used by standard system devices. All of these commands may be passed from a user program or interactive terminal. Additionally, the operator only may enable and disable interrupt levels used by the system I/O devices. To do this, however, requires a verification response as discussed below.

## 13-2 CONNECT PROGRAM

The Connect Program command is used to connect a real-time program to an unused external interrupt such that when the interrupt occurs the program will be awakened from the "sleep" state. The format is as follows:

CX,int,programname,parameters

where "int" — is the interrupt level to which the program is to be connected. Group 1 levels are entered as to 23 and group 2 levels as 24-47.

"programname" — is the program name to be awakened when the interrupt occurs. It is in the standard qualifier and areaname format. If the program is not sleeping when the interrupt occurs, the trigger is ignored.

"parameters" — consist of the following:

PAR=n — This specifies a parameter which is to be set into the program's K register when it is awakened. If not specified, zero is used.

PERM — If entered, this parameter indicates that the connection is to be re-established each time VULCAN is loaded from disc for execution. If not entered, the entry is lost when VULCAN is rebooted.

ENABLE — If entered, this parameter indicates that the specified interrupt level is to be armed and enabled when the connection is made. The level will also be armed and enabled on each VULCAN reboot if the PERM flag is entered. This is the default.

INHIBIT — If entered, this indicates that the interrupt level is not to be armed and enabled when the connection is made or when VULCAN is rebooted. The default is ENABLE.

Example:

Connect program 1234ABCD*CAT to interrupt level 5 on Group 2 permanently and enable the interrupt:

CX,29,1234ABCD*CAT,PERM

## 13-3 DISCONNECT PROGRAM

The Disconnect Program command is used to disconnect a program from an external interrupt which has been previously set up via a Connect Program operation. The specified interrupt level is also disarmed. The format is as follows:

DX,int

where "int" is the external interrupt level designation. Group 1 levels are numbered 0-23 and group 2 levels 24-47.

Example: To disconnect a program from interrupt level 5 on group 2:

DX,29

## 13-4 ENABLE INTERRUPT

The Enable External Interrupt command is used to arm and enable an interrupt level which has previously been connected to a program via a connect command. In addition, the system operator may enable any system interrupt level as discussed below. The format of the command is as follows:

EX,int

where "int" is the external interrupt level designation. Group 1 levels are numbered 0 to 23 and Group 2 levels are 24 to 47.

If the command is input by the operator and the level is a standard system I/O device level, Opcom will output the following message:

"SYSTEM INTERRUPT LEVEL nn?"

where "nn" is the interrupt level. If this is correct, the command should be re-entered as specified to perform the operation. If in error, the proper command should be entered.

Example: To Enable interrupt level 5 on group 2:

EX,29

## 13-5 INHIBIT INTERRUPT

The Inhibit External Interrupt command is used to disarm and inhibit an external interrupt level which has been connected to a real-time program via a connect operation. In addition, the system operator may inhibit any system interrupt level provided proper verification is established as discussed below. The command format is as follows:

IX,int

where "int" is an external interrupt level designation. Group 1 levels are numbered 0 to 23 and group 2 levels are 24 through 47.

If the command is input by the operator, and the level is a standard system I/O device level, Opcom will output the following message:

"SYSTEM INTERRUPT LEVEL nn?"

where "nn" is the interrupt level. If this is correct, the command should be re-entered as specified in order to disable the level. If in error, the proper command should be entered and re-verification will take place.

# SECTION XIV
# SYSTEM DIAGNOSTIC AID COMMANDS

## 14-1 GENERAL

These commands are used by system programmers to aid VULCAN development. Their use is intended for persons familiar with the structure and operation of the VULCAN system.

## 14-2 ALLOCATION LIST

The Allocator List writes out to the Opcom List disc area a dump of the monitor memory allocation linkages. Two columns each containing three values are output; the first value being the block starting address, the second the size of the allocated block, and the third the unused fragment between this and the next block. The command format is as follows:

AL

## 14-3 PATCH ADDRESS

The Patch Address commands is used to modify selected absolute memory locations. The command formats are as follows:

PA,n,a
PA,n-m,b,c,d,c. . .

| | |
|---|---|
| where "n"(or n-m) | is the memory location(s) to change, and |
| "a"(or b,c,d,. . . ) | is the new value(s). |

Examples:

| | |
|---|---|
| PA,0,1 | Change location 0 to a 1 |
| PA,0-2,1,2,3 | Change locations 0, 1, 2 to 1, 2, 3 respectively |
| PA,'32637,'21046325 | Change octal Location 32637 |

## 14-4 GET ADDRESS

The Get Address Command is used to display selected absolute memory locations. The formats are as follows:

GA,n,m
GA,K-L

The contents of "m" locations starting with location "n" are displayed via the first command. If "m" is not entered, 8 locations are displayed. The second format will display locations "K" through "L" inclusive. If 8 or less locations are displayed, output is to top line of CRT (line 0), and the remainder of the screen is left intact. If more than 8 locations are output, the screen is cleared and the locations are displayed beginning with line 1 of the CRT.

An asterisk ( * ) may precede the first location specification using the first format ("n"). This implies that an indirect level should be used to obtain the starting address. For example, the contents of location "n" is obtained to determine the starting memory location to output.

Examples:

| | |
|---|---|
| GA,0 | (Display 0-7) |
| GA,0,7 | (Display 0-6) |
| GA,0-7 | (Display 0-7) |
| GA,*0,7 | (Fetch 0, and display fetched location and next 7) |
| GA,*'363,64 | (Display the memory location and the next 63 pointed to by location octal 363) |

## 14-5 SECTOR DISPLAY

The Sector Display command is used to display to the CRT screen the octal contents of one or more disc sectors. The formats are as follows:

a) SD,n,P=X

b) SD,n-m,P=X

c) SD,disc-area-name,n

d) SD,disc-area-name,n-m

Formats a) and b) are used to display absolute disc sector "n" (or "n" through "m") of disc pack "X". If the "P=" specification is not made, pack one is assumed.

Formats c) and d) are used to display relative sector "n" (or "n" through "m") of disc area "disc-area-name". If no sector specification is made, the first sector (0) is displayed.

Examples:

SD,0

SD,1234ABCE*CAT

SD,1234ABCD*XTZ,5

SD,0,P=2 (sector 0 of pack 2)

## 14-6 DISPLAY REPEATER COMMANDS

Three commands are used to scan sequential memory locations or disc sectors in accordance with the two commands discussed above. They are as follows:

| | |
|---|---|
| + | To display next unit |
| = | To display current unit |
| - | To display previous unit |

If the last system display command was a Get Address, then these commands will be used to display the next eight, same eight or previous eight memory locations from those last displayed. If the last system display command was a Sector Display, these commands are used to display the next disc sector, repeat the current disc sector, or display the previous disc sector.

Note that these commands have no effect on the Opcom Disc Area display commands, View Up and View Down.

Examples:

| | |
|---|---|
| SD,*CAT | Displays sector 0 of CAT |
| + | Displays sector 1 of CAT |
| = | Displays Sector 1 of CAT |
| + | Displays Sector 2 of CAT |
| - | Displays Sector 1 of CAT |
| - | Displays Sector 0 of CAT |

## 14-7 DISC DUMP

The Disc Dump Command is used to produce an octal and ASCII dump of specified disc sectors to the System List Device. The formats, which are identical to the Sector Display command, are as follows:

a) DD,n,P=x

b) DD,n-m,P=x

c) DD,disc-area-name,n

d) DD,disc-area-name,n-m

Formats a) and b) are used to dump absolute disc sector "n" (or "n" through "m") of disc pack "x". If the "P=" specification is absent, pack 1 is assumed (Master pack).

Formats c) and d) are used to dump relative sector "n" (or "n" through "m") of the specified disc area "disc-area-name". If no sector specification is made, the entire disc area is dumped.

Examples:

DD,0

DD,0,P=2

DD,1234ABCD*CAT

## 14-8 MODIFY DISC

The Modify Disc command is used to change the contents of an absolute disc sector. This change must be made one word at a time, giving the old contents of the word to lower the probability of error. The format is as follows:

MD,sector,P=x,word,old,new

where "sector" is the absolute disc sector to modify

"x" is the pack number of the specified sector. If "P=x" is not entered, pack 1 is assumed

"word" is the relative word number from the start of the sector to modify (Words are numbered 0 through 111).

"old" is the old contents of the specified word

"new" is the value to be placed in the specified word, if the "old" value matches.

Examples:

a) First word of sector 2 from 34 to 56:
MD,2,0,34,56

b) Last word of sector '115 of pack 3.
MD,'115,P=3,111,'21042535,'21042564

# SECTION XV
# MISCELLANEOUS COMMANDS

## 15-1 TIME SLICE

The Time Slice command is used to specify the time slice interval in 120 Hz clock counts, for programs of equal priority. The format is as follows:

TS,n

If "n" is zero, time-slicing for equal priority programs is turned off. If "n" is positive, the time slice period before changing to another program of equal priority will be "n" 120 Hz clock counts. The default is set via a system generation parameter.

## 15-2 CLEAR DISC AREA

The Clear Disc command is used to zero specified sectors of a specified unblocked disc area. The format is as follows:

CD, disc-area-name, n

The first "n" sectors of the specified disc area "disc-area-name" are cleared to zeroes. For correct operation the disc area should be unblocked.

## 15-3 MODIFY PROGRAM

The Modify Program command, which is available from user programs and interactive terminals, is used to change selected words of a catalogued program. The program may be of any type (Interactive/Control Point, Real-time, Non-resident Handler, etc.). The user must have write access to the program area to perform the modification.

The specified location is entered as a relative number, but since all programs start at location zero, the location is also absolute for the program. The location thus corresponds to that output by the Vassembler, added to the load address output on the Vulcanizer map. The format is as follows:

MP,disc-area-name,location,old,new,R

The specified "location" of the program is checked against the "old" specification, and if a match occurs, the value "new" replaces it.

For non-resident handlers and monitor programs only, the R parameter should be entered if the "new" value is a relocatable value (contains a relative location in the program). This is not needed for any other type of program.

Examples:

a) To change the second location of program *XYZ from a NOP to a TOA 6:

MP,*XYZ,1,'62000000,'62500006

b) To change location 5 of non-resident monitor program V:ABCD:V from a "TOA 6" to a "TOA *":

MP,*V:ABCD:V,5,'62500006,'62500005,R

# SECTION XVI
# OPERATOR MESSAGES

The following table is a partial list of the message which VULCAN may output to the operator via the Opcom Message Area.

| | |
|---|---|
| ***date/time | System boot message. The date and time of each system boot is written to the Message Area. |
| nnnnnnnnAB m @xxxxxxxx | Program nnnnnnnn Aborted and either it was initiated by the operator, or its diagnostic output file was unavailable. |
| DEVICE n NOT ON LINE | Physical device "n" is not on-line when it should be. System recovery is automatic. |
| DEVICE n TROUBLE | There is trouble on physical device "n". Operator intervention is required. System recovery is automatic. |
| HOPPER EMPTY ON CARD READER n | A hopper empty condition has been detected on card reader physical device "n", prior to reading an EOT card. |
| MOUNT PACK n ON DISC m | The operator should mount disc pack "n" on disc number "m". If already mounted, the "PR" command may be used. (Section VIII.) |
| MOUNT TAPE xxxxxx ON PDN:n | Mount the tape named "xxxxxx" (SCRTCH is output for a scratch tape) on physical device "n". If already mounted, the "TR" command may be used. (Section VIII.) |

| | |
|---|---|
| PRINTER n:xxxxxxxxxxx | Forms Control Message. The output spool area on printer "n" has output the message xx...xx and is awaiting operator action. When ready, the "RO" (release output) command should be given to continue printing. |
| TAPE n FREE | The tape on physical device "n" is no longer needed and may be removed. |

# SECTION XVII
# ACCOUNTING

## 17-1 GENERAL

Accounting is an integral part of VULCAN Operation. VULCAN maintains accounting information on system functions, peripheral device, and CPU usage, and outputs it to accounting disc areas at appropriate intervals. Through use of this information, the system manager can evaluate system use and charge users as needed.

## 17-2 ACCOUNTING STRUCTURE

Accounting records are accumulated by user programs, system programs, and device handlers into internal blocks. When a block is filled it is written to an accounting disc area by the "Ten Second" Program. Each of these unblocked disc areas is 100 sectors long, with every two sectors holding 12 accounting records. Partial disc areas are terminated with an end-of-file sector. These accounting areas are under qualifier 0000ACNT and have areaname AC##0nnn where "nnn" is the accounting area number.

Whenever one of these areas becomes filled, another system program is initiated which copies the accounting area to a public read disc area. It replaces the user numbers, which have been stored in the accounting records for identification purposes, with the appropriate user names. These disc areas are under qualifier 0000ACSM and have areanames AC##0nnn where "nnn" is an accounting area number.

An accounting utility program, *ACUTIL, is available to process records from the accounting areas. This program is described in Section 18.

## 17-3 ACCOUNTING RECORD FORMAT

Accounting summary records are 21 words in length. Approximately 600 records are written to each accounting summary area, which is a blocked area. There is an end-of-file record at the end of each disc area. The format of the records is as follows:

| | | |
|---|---|---|
| Word 0: | Device Type and Number: | |
| | Bits 23-18: | Device type |
| | 7-0: | Physical Device Type |
| 1<br>2 | Starting date and time in $TIME service format. * | |

Word 3, 4: Ending date and time in $TIME service format. *

5, 6, 7, 8: User-name - 12 ASCII characters.

9: Mode word:

| Bits <u>9</u> <u>8</u> | |
|---|---|
| 0 0 | Real-time or Monitor Program. |
| 0 1 | Control point program. |
| 1 0 | Interactive terminal program. |
| 1 1 | -invalid- |
| Bits 5-0: | Program Priority |

10, 11:

a) For real-time or monitor programs, 8-character program areaname in truncated ASCII.

b) For Control points, first 6 characters of job name in regular ASCII.

c) For Interactive Terminals, word 10=zero and word 11 is the terminal PDN.

12, 13: For Control points, last 6 characters of Job Name in regular ASCII.

14, 15: Sign-on Qualifier for user in truncated ASCII. First word is 4-digit account number, second word is 4 character identifier.

16-20: Information Particular to the device type as listed in Table 17-1.

*$TIME format has year in Bits 23-12 of the first word, the day of the year in Bits 11-0, and the time of day in tenths of seconds since midnight in the second word.

TABLE 17-1

ACCOUNTING INFORMATION

| Device Type | Device Name | | Record Contents |
|---|---|---|---|
| '01 | PDN (Bits 7-0) = 0 | CPU Time | Words 16-17=CPU time in T-register<br>Word 18 = Number of program context switches in program execution (billing) mode. |
| '01 | PDN (Bits 7-0) = 1 | CPU Overhead | Words 16-17: CPU Time spent in Paging overhead in T-register counts.<br>Word 18: Number of program context switches in overhead mode.<br>Word 19: Number of demand page requests. |
| '01 | PDN (Bits 7-0) = 2 | System Idle | Words 16-17: CPU time in T-register counts absorbed in idle loop.<br>Word 18: Total number of context switches.<br>Word 19: Total number of non-redundant context switches. |
| '02 | | Teletype | Word 16: Records input.<br>Word 17: Records output. |
| '03 | | CRT | Word 16: Records input.<br>Word 17: Records output. |
| '04 | | Paper tape reader | Word 16: Records transferred. |
| '05 | | Paper tape reader | Word 16: Records output. |
| '06 | | Line Printer | Word 16: Lines printed.<br>Word 17: Lines plotted (Versatec printer/plotters only). |
| '07 | | Card Reader | Word 16: Cards read. |

TABLE 17-1 (Cont'd.)
ACCOUNTING INFORMATION

| Device Type | Device Name | Record Contents |
|---|---|---|
| '10 | Card Punch | Word 16: Cards punched. |
| '11 | Magnetic Tape | Word 16: Records input.<br>Word 17: Records output.<br>Words 18-19: Words transferred. |
| '77 | Disc | Word 16: Regular Disc requests.<br>Word 17: System disc requests.<br>Words 18-19: Disc words transferred. |

# SECTION XVIII
# ACCOUNTING UTILITY PROGRAM

The VULCAN Accounting Utility, *ACUTIL, is an interactive/control point processor which analyses accounting records to produce summaries and usage reports. It includes features which allow CPU usage analysis, I/O device usage analysis, and individual billing information based on user, qualifier, or account.

## 18-1 PROGRAM OPERATION

ACUTIL is initiated at a control point or an interactive terminal via the statement:

$$*ACUTIL

Once initiated, it reads commands as discussed below from the Job Stream (Logical File 0). When all commands have been input, the information is obtained from disc, calculations made, and the results printed. The end of input is noted by either an end-of-file or the following command:

BEGIN

## 18-2 SUMMARY PERIOD COMMANDS

These commands are used to specify the period over which the analysis is to be done.

### 18-2.1 Start

The Start command is used to specify an inclusive initial date and time. If no Start is entered, analysis begins with the earliest accounting information located in the accounting areas. The format is as follows:

START,mm/dd/yy,hhmm

where "mm" is the month, "dd" is the day, and "yy" is the year.

The time field is optional and consists of hours (hh) minutes (mm), in 24-hour clock format.

### 18-2.2 End

The End command is used to specify an inclusive terminating date and time. If no End is entered, the analysis will end with the largest dated accounting record found. The format is as follows:

END,mm/dd/yy,hhmm

where the date and time fields are as described above for the Start commands.

18-2.3 Period

The Period command is used to determine which hours of the day, or which days of the week the analysis is to cover. The format is as follows:

PERIOD,days,hours,days,hours

where "days" consists of a single letter or "range" (M-N) of letters where letter specifies day of week as follows:

M-Monday
T-Tuesday
W-Wednesday
R-Thursday
F-Friday
S-Saturday
N-Sunday

and "hours" consists of a range of 24-hour clock times (the values separated by a dash.

Days and hours specifications must normally be paired, except that a single "days" specification implies all hours, and a single "hours" specification implies all days.

Examples:

| | |
|---|---|
| PERIOD,M-F | all hours, Monday through Friday |
| PERIOD,800-1700 | 8:00-5PM every day |
| PERIOD,M-F,800-1700,S,800-1200 | 8AM-5PM on weekdays, and 8AM through noon on Saturdays |
| START,12/01/74<br>END,12/31/74<br>PERIOD,M-F,1200-1300 | Noon to 1PM every day in December |
| START,12/01/74,1200<br>END,12/02/74,1200 | Noon on Dec. 1 to Noon on Dec. 2 |

## 18-3 TITLE COMMAND

The Title command is used to specify a title line to be printed on the top line of each page of output. The format is as follows:

TITLE,Title-line

A maximum of 60 characters can be stored as a title. If blanks or commas are to be included, the entire title field should be enclosed in double quotes.

## 18-4 GROUPING COMMANDS

These commands are used to determine what accounting records are to be included in the analysis. Only one of the following commands may be entered. If none are specified, all records encountered are processed.

### 18-4.1 User

The User command is used to list individual users to be included. The format is as follows:

USER, user-name, user-name, ...

where "user-name" is a 1-12 character field giving the usernames. These names should match exactly that stored for the user when he is added to the system.

Multiple User commands may be entered if all user-names cannot be contained on one card.

### 18-4.2 Qualifier

The Qualifier command is used to specify sign-on qualifiers to be included. The format is as follows:

QUALIFIER , qualifier, qualifier, ...

where "qualifier" is a 8-character qualifier consisting of account and identifier fields.

Multiple Qualifier commands may be entered if needed.

### 18-4.3 Account

The Account command is used to specify which accounts, based on sign-on qualifiers, are to be included. The format is as follows:

ACCOUNT, account, account, ...

where "account" is a 1 through 4 digit account number. Multiple Account commands may be entered if needed.

## 18-5 OUTPUT ORDERING COMMANDS

These commands are used to specify the ordering and grouping of the output information. Only one of the following three commands may be entered, but the Total command may always be used. If none is entered, a single total analysis is output.

### 18-5.1 Order by User

The Byuser command indicates that the output is to be done by individual user. Users are listed alphabetically, and grouped by qualifiers. This command is entered as follows:

BYUSER

### 18-5.2 Order by Qualifier

The Byqualifier command indicates that the output is to be grouped by sign-on qualifier, ordered by account. The format is as follows:

BYQUALIFIER

### 18-5.3 Order by Account

The Byaccount command indicates that the output is to be grouped by account number only, arranged in numeric account number order. The format is as follows:

BYACCOUNT

### 18-5.4 Total

The Total command, which may be used with any one of the above three, indicates that totals are to be accumulated for each group of individual outputs, along with a total of all charges output. For example, if a Total command were entered along with a Byqualifier command, then totals would be output for each account, and a total of all accounts. The format is as follows:

TOTAL

## 18-6 PRIORITY CATAGORIES

Priorities of execution can be broken down on output through use of the Priorities command. This breaks down CPU usage for all user's into the priority catagories specified. The command format is as follows:

PRIORITIES, range 1, range 2, ...

where "range" fields consist of two numbers in ascending order separated by a dash. For all charges to be output, all priorities from 0 to 63 should be covered by the specifications on the Priority command. If no priority command is entered, all CPU usage is listed in a single catagory.

Examples:

PRIORITIES,0-15,16-32,33-63

PRIORITIES,0-31,32-63

## 18-7 LIST OPTION

The List option allows all of the accounting records used in the analysis to be listed prior to being output. The command format is as follows:

LIST

## 18-8 DEVICE UTILIZATIONS

A summary of CPU and device utilizations is output when the Utilizations command is specified. This summary lists the total usage for the device, along with the time allocated and the total time available. The format is as follows:

UTILIZATIONS

## 18-9 RATES

Charge rates for individual billing may be specified by use of the RATES command. The basic format is as follows:

RATES,device=charge,device=charge,

where "charge" is a decimal number specifying the charge for units of the specified "device" in dollars. "Device" specifies the device type, optionally the physical device number, and item type. The "device" field is as follows:

dev,nn/t

where "dev" is the device name as listed below, and

"nn" is optional. If specified it is the specific physical device number for which the change applies, and

"t" is the type of item as shown below.

For CPU time, the "nn" field is used to specify a range of priorities for that change. In this case, the specifications on the Priorities command are not required and, if present, are ignored.

| Device | Device Type Field |
|---|---|
| CPU time | CPU |
| Disc | DSC |
| Teletype | TTY |
| CRT | CRT |
| Card Reader | CRD |
| Card Punch | CPH |
| Paper Tape Punch | PTP |
| Paper Tape Reader | PTR |
| Line Printer | LPR |
| Magnetic Tape | MGT |

| Item Type | Command Code | Units of Billing |
|---|---|---|
| Words transferred | W | Hundreds |
| Time | T | Seconds |
| Input records | I | Records |
| Output records | O | Records |
| Input & Output records | R | Records |

Examples of Rates cards:

a) 1/10 cent per CPU second

RATES,CPU-T=.001

b) 2 cents per card punched

RATES,CPH-R=.02

c) 3/10 cent per teletype record input, and 1/10 cent per teletype connects second

RATES,TTY-I=.003,TTY-T=.001

d) 1/10 cent per line on printer 6, 2/10 cent per line on printer 8.

RATES,LPR6-R=.001,LPR8-R=.002

# CHAPTER G

# SYSTEM GENERATION

# SECTION I
# INTRODUCTION

## 1-1 GENERAL

GENASYS is the VULCAN system generation program. It has as its input a standard VULCAN sysgen tape, which has no configuration information on it. The tape is merged with user input defining his configuration to produce a functional VULCAN on a disc pack. User input is designed to be in a flexible, readable form.

## 1-2 MINIMUM CONFIGURATION

GENASYS need not be run on a Slash 4 VMS computer. This allows a system to be generated on a disc on one system and the disc pack then taken to a Slash 4 VMS system for execution. It may be run on any Slash 4 or Slash 5 computer having the following features:

1. Bit Processor Option
2. Magnetic Tape unit
3. Disc Drive on which to generate system
4. Teletype or CRT for input

Additional useful peripherals might be a card reader and line printer for user input and output.

GENASYS does not require any external interrupts.

## 1-3 CAPABILITIES

GENASYS is designed to always generate an entire new master disc pack.

User disc areas on the master pack should be saved prior to executing GENASYS. Disc areas on all other packs remain intact and may be merged into the Master Disc Directory using Operator Commands. This procedure is discussed in Section XII.

## 1-4 INPUT DEVICES

GENASYS configuration information may be input on punched cards, punched paper tape, or may be entered by the operator via a teletype or CRT. This information is in a flexible free format designed to minimize input. Knowledge of the internal structure of VULCAN is not required.

## 1-5 OUTPUT DEVICES

GENASYS will produce output listings of the parameters input and disc areas restored. This may be written to a teletype, CRT or line printer. Additionally a summary of Physical Devices Assignments, Hardware Interrupts, and Hardware Channel/Unit Assignments is available.

# SECTION II
# LOADING GENASYS

## 2-1 GENERAL

GENASYS must be loaded from the appropriate media by either activating the hardware Bootstrap switch or by entering the appropriate bootstrap for that device. Once GENASYS is loaded it immediately reads the console switch register to determine the type and location of the terminal through which the operator is to communicate with GENASYS. This section describes the procedures.

## 2-2 LOADING THE DEVICE BOOTSTRAP

If the /4 computer system has a hardware bootstrap for the GENASYS media, this section need not be performed. The bootstrap program should be entered using the following procedure:

1) Raise and return the Master Clear switch
2) Lower only the M and OR display switches
3) Depress the "AUTO" button
4) Depress "ENTER REG"
5) Enter the first instruction of the bootstrap into the Operand Register by depressing the appropriate buttons on the top row.
   The first instruction is:

   62500406 (for Mag Tape input)
6) Depress "ENTER REG"
7) Repeat steps 5 and 6 for the remaining bootstrap instructions as listed below:

| 7 Track Mag Tape | 9 Track Mag Tape | Location (Octal) |
|---|---|---|
| 00700700* | 00700700* | 2 |
| 62500012 | 62500012 | 3 |
| 00714700* | 00714700* | 4 |
| 05000016 | 05000016 | 5 |
| 00700700* | 00700700* | 6 |
| 22600006 | 22600006 | 7 |

| 7 Track Mag Tape | 9 Track Mag Tape | Location (Octal) |
|---|---|---|
| 00730700* | 00730700* | 10 |
| 22600010 | 22600010 | 11 |
| 00110204 | 00110204 | 12 |
| 22000020 | 22000020 | 13 |
| 22600010 | 22600010 | 14 |
| 21000020 | 21000021 | 15 |
| 40036506 | 40034506 | 16 |

* The appropriate device channel/unit number must be set into the low order 11 bits of these instructions. (The "standard" Mag Tape channel/unit number is '0700).

8) Depress "ENTER Pa"

9) Depress "HALT/RUN" once
The bootstrap program is now ready for operation

It should be noted that GENASYS assumes a particular format for the bootstrap program, and only those listed above should be used if the hardware bootstrap is not available.

## 2-3 DEFINING OPERATOR TERMINAL FOR GENASYS

The operator terminal for GENASYS must be defined by raising the appropriate switches in the console switch register as defined below.

### 2-3.1 Channel/Unit

The hardware channel number must be entered switches 10-6, and the unit number in switches 3-0.

### 2-3.2 Device Type

If the console device is a TEC 425 CRT, switch 23 should be raised. Otherwise, the operator terminal is assumed to be a teletype or teletype compatible device, in which case switch 23 must be lowered (OFF).

### 2-3.3 Multiplexor

If the terminal is connected via a 9010 communications Multiplexor, switches 22 and 21 should be raised. In addition, switches 15-12 should be set to contain the position within the MUX that this device occupies. Lastly, the Unit Number of the MUX itself on the 8-bit channel must be set into switches 19-16. In this case, the unit number in switches 3-0 is the unit number which is used for the connected devices.

### 2-3.4 Interface

If the terminal is connected via an Asynchronous Communications Interface ( or 9050 Dual Asynchronous or Multiplexor) switch 22 must be raised.

### 2-3.5 Console Switch Examples

Given a TEC 425 CRT on a multiplexor, the multiplexor being on channel 2 with the control logic on unit 0 and the data logic on unit 1, and the CRT plugged into priority slot 3 of the MUX, the switches should be set as follows:

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | X | X | | | | | | | | X | X | | | | | X | | | | | | | X |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Given a TEC 425 CRT on a CRT controller on Channel 0, unit 0, the switches should be set as follows:

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Given a teletype on channel 5, unit 3, the switches should be set as follows:

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | X | | X | | | | | X | X |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

## 2-4 EXECUTING GENASYS

Once the console switch register has been set, GENASYS is ready to be executed. First ensure that GENASYS is loaded on the appropriate device and that device is ready.

If the bootstrap program is used, depress HALT/RUN again to begin execution.

If the hardware bootstrap is to be used, raise and return the Master Clear switch, lower the appropriate device bootstrap switch, and depress "Boot Enable".

When GENASYS begins execution the message

VULCAN GENASYS XX. XXXXXXX

is output. The XX. XXXXXXX is the version number and date of the GENASYS/ VULCAN system being used.

# SECTION III
# GENASYS OPERATION

## 3-1 GENERAL

This section describes the operator interaction with GENASYS, along with the operational capabilities of the system.

## 3-2 COMMUNICATING WITH GENASYS

Operator interaction with GENASYS is controlled via the established operator device as determined by setting the console switches as discussed in the previous section. All operator messages are output to this device as are error messages and requests for input. Input is determined by the type of device.

### 3-2.1 TEC 425 Interactive CRT Input

If the operator terminal is an interactive 425 CRT, input is from the top line of the display. When the line is correct, the XMIT key is depressed to signal GENASYS to input the top line of the display. Any editing of the input on the top line may be done via local editing keys on the 425 prior to pressing XMIT. The CRT will display the line just input at the bottom of its display.

The operator is signalled that GENASYS is awaiting input by noting either that the flashing cursor mark is on the left edge of the top line, or that the red "WAIT" is off, (both will occur), or the "INPUT REQ" light is on.

### 3-2.2 Teletype Input

With all non-CRT devices (teletypes, or teletype compatable), input is required and may be entered when a line feed is output to the device. Normally this will follow some type of question or request from GENASYS. Characters should be typed until the line is complete, at which time the "RETURN" key (carriage return) is depressed to signal GENASYS to process the line.

If an error is made while entering a line, entering a backspace character, which is typed as a " ← " (Shift 0) on standard teletypes, and as a "____" (underscore, or US) on TI-700 teletypes, will erase the preceding character. The entire line may be cancelled and re-entered by typing the "ALT MODE" key, or ESC (Escape) on TI-700's.

### 3-2.3 Error Messages

Under most circumstances, unless otherwise noted in this manual, when an error is detected by GENASYS, an appropriate message is output to the operator terminal, preceded by a copy of the input line containing the error.

The operator should respond to error conditions by re-entering the corrected line on the operator device, which then allows GENASYS to continue processing the regular input stream.

## 3-3 GENASYS INPUT DEVICE

The second message output by GENASYS will be:

"ENTER INPUT DEVICE"

This is a request to the operator to indicate which system input device is to be used to input the system configuration information. This may be a card reader, paper tape reader, or the operator TTY or CRT. One of the following responses should be made:

| | |
|---|---|
| CRT | Indicates that the data input will be from the same CRT that is currently the operator device. |
| CRT CU='ccuu | Indicates that the data input will be from a CRT, with hardware channel/unit number "ccuu". This should only be used when a teletype is the regular operator device. |
| TTY | Indicates that the data input will be from the same TTY that is currently the operator device. |
| TTY CU='ccuu | Indicates that the data input will be from a Teletype, with hardware channel/unit number "ccuu". This should only be used when a CRT is the regular operator device. |
| CR CU='ccuu | Indicates that the data input will be from a card reader with hardware channel/unit number "ccuu". |
| PTR CU='ccuu | Indicates that the data input will be from a paper tape reader, with hardware channel/unit number "ccuu". |

## 3-4 GENASYS OUTPUT DEVICE

Next GENASYS will output the following message in order to determine the list output device:

ENTER OUTPUT DEVICE

A listing of all GENASYS messages along with a copy of the records from the input device is output to the output device. In addition, Physical Device Summary Information, is optionally output to this device.

The output device may be a line-printer, CRT, or teletype as determined by one of the following responses:

| | |
|---|---|
| LP#mmmm CU='ccuu | Indicates output device is line printer, with model number "mmmm" as defined in Table 5-2. The hardware channel/unit number is "ccuu". |
| CRT | Indicates output device is operator CRT. |
| CRT CU='ccuu | Indicates output device is a CRT, having hardware channel/unit number "ccuu". The operator device must be a teletype in this case. |
| TTY | Indicates output device is operator teletype. |
| TTY CU='ccuu | Indicates output device is a teletype, having hardware channel/unit number "ccuu". The operator device must be a CRT in this case. |

## 3-5 MODES OF GENASYS OPERATION

As a complete system generation system for VULCAN, GENASYS assumes no valid VULCAN information resides on the disc unit, and a complete new system is written over the disc, nullifying any previous information stored on disc. Of course, user data files may be saved (refer to Chapter B) to another mass storage device (cards, magnetic tape, etc.) for later restoration prior to clearing the disc.

GENASYS will output the following request;

DO YOU WISH TO PRESERVE DISC CONTENTS?

The operator response to this message must be

NO

Any other input may cause unspecified results.

### 3-5.1 Clear Disc Option

GENASYS has an option of zeroing the entire disc pack before using it. This must be done if the disc unit has not been entirely written upon at least once.

GENASYS will output the following message:

DO YOU WANT ENTIRE PACK ZEROED?

The operator responses to this message are:

NO --to clear only the disc directory areas of the pack

YES --to clear the entire pack

### 3-5.2 Normal Operation

At this point, GENASYS will output the message

******

to indicate that it is switching to the specified input and output devices determined above. Under normal conditions, no further input is needed from the operator terminal (except if it is the input device or an error occurs), and all output will be sent to the defined output device.

# SECTION IV
# REQUIRED SYSTEM PARAMETERS

## 4-1 GENERAL

This section describes the input of those parameters which are required regardless of configuration.

## 4-2 MASTER DISC DEFINITION

The disc drive onto which the VULCAN system is to be generated must first be defined to GENASYS. This should be done after the following message is output:

"ENTER MASTER PACK TYPE AND LOCATION"

The following parameters should be entered. They must be contained on one source image from the input device.

TYPE=MMMM --where MMMM is the model number of the disc unit. See Table 4-1.

CU='ccuu --where ccuu is the channel/unit of the disc dontroller for the master disc drive.

DRIVE=N --This parameter is optional where N is the drive number of the particular drive being used. If this parameter is not entered, drive 0 is assumed. On cartridge disc systems (Model numbers 52xx) the drive number should be multiplied by two, and the platter number added. (The fixed platter is "0" and the removable platter is "1".) For instance, the fixed platter of drive 0 is entered a "0", the top (removable) platter of drive 0 is entered as "1", the fixed platter of drive 1 is entered as "2" and the top (removable) platter of drive 1 is entered a "3".

Examples: To generate fixed platter of a cartridge disc on channel 5:

TYPE=5208 CU='0500

To generate a removable platter of a cartridge disc on channel 5 drive 3:

TYPE=5208 CU='0500 DRIVE=7

GENASYS will then ensure the master pack is on-line and if not, an appropriate error message is output.

TABLE 4-1
DISC MODEL NUMBERS

| Model # | Type | Cylinders | Default Granule Size |
|---|---|---|---|
| 5102 | Moving Head | 203 | 20 |
| 5104 | Moving Head | 406 | 40 |
| 5202 | Cartridge Only | 204 | 10 |
| 5204 | Cartridge Fixed | 204 | 10 |
| 5206 | Cartridge Only | 406 | 20 |
| 5208 | Cartridge Fixed | 406 | 20 |
| 5404 | Fixed Head | 8 heads | 1 |
| 5406 | Fixed Head | 16 heads | 2 |
| 5408 | Fixed Head | 32 heads | 4 |
| 5410 | Fixed Head | 64 heads | 8 |
| 5413 | Fixed Head | 128 heads | 10 |
| 5415 | Fixed Head | 256 heads | 20 |
| 5500 | Disc Storage Module | 411 | 40 |
| 5501 | Disc Storage Module | 411 | 40 |
| 5510 | Disc Storage Module | 823 | 40 |
| 5511 | Disc Storage Module | 823 | 40 |

## 4-3 MASTER DISC ALLOCATION PARAMETERS

The parameterization of the master disc is done next, prior to GENASYS initializing the master disc. The following message is output:

"ENTER MDD PARAMETERS"

The following parameters may be optionally input, on one or more lines, terminated by the single parameter:

END

A further discussion of disc structure parameters is presented in Chapter A.

4-3.1 Entries Per Master Disc Directory

The following parameter defines the approximate size of the initial Master Disc Directory. This is the number of Qualifiers that may be held prior to a system generated rehash of the MDD and resultant increase in size. The parameter is entered as follows:

E/MDD=nnnn

If this parameter is not entered, the system default is 196.

4-3.2 MDD First Sector

The absolute sector number on which the Master Disc Directory is to start may be set by this parameter. It is entered as follows:

MDDFS=nnnn

If this parameter is not entered, the system will allocate the Master Disc Directory immediately following the Space Allocation Map at the front of the disc.

4-3.3 Space Allocation Map First Sector

The first absolute sector number for the master disc's Space Allocation Map may be set by this parameter. It is entered as follows:

SAMFS=nnnn

If this parameter is not entered, GENASYS will allocate the first available space on disc, following the two fixed sectors allocated to the Absolute Disc Loader, unless this conflicts with other Master Disc parameters.

4-3.4 Sectors Per Space Allocation Bit

The Sectors per Space Allocation Bit parameter defines how many disc sectors of space are represented by a single bit in the Space Allocation Map. For instance, if this parameter were set at two, all disc areas would start on even-numbered sectors, potentially wasting one sector of the end of each area. Small values of this parameter reduce disc storage waste but tend to increase the time spent searching for disc space when allocating disc areas. The system default for this parameter is 2. It may be specified as follows:

S/SAB=n

4-3.5 Default Granule Size

This parameter defines the default granule size for allocation of disc areas on the master pack. A granule is a contiguous protion of disc space. Regular disc areas are made up of one or more granules. Additional granules are automatically added by the System as the disc area increases. Program disc areas, however, always occupy a single granule of the size necessary to contain the program. The default size in sectors is dependent upon the disc being used and is shown in Table 4-1. This parameter may be entered as follows:

GRANULE=nn

4-3.6 Sectors Per Block

This parameter is used to specify the default blocking factor used with blocked disc areas created on the master disc. The blocking factor is the number of sectors which are used to hold one block. One block of user address space is used by the blocked disc area handler when the file is open. See Chapter E. Blocking factors must be in the range of 1-7. GENASYS will use a default blocking factor of two if none is entered. It may be entered as follows:

S/B=n

4-4 SYSTEM INPUT

Following the above parameterization, GENASYS copies the resident VULCAN from the binary input device to the disc and sets up the Master Directories and Space Allocation Map. Next the required system parameters are input, as noted by the message:

"REQUIRED SYSTEM PARAMETERS"

4-5 MEMORY SIZE

The maximum available Memory Size of the particular Slash 4 configuration should be entered following the message:

"MEMORY SIZE= "

If a particular bank of memory is currently not available, or not working, it should be included in this parameter and may be excluded as discussed in Paragraph 7-3.

The memory size may be entered in one of three different ways. It may be a number of 1024 word pages, followed by the letter "K"; it may be an octal integer giving the size in words; or it may be a decimal number of words. For example, a memory size of 65,536 words may be entered as any of the following:

64K
'200000
65536

4-6 CLOCK INTERRUPT LEVEL

The interrupt level number for the 120 Hz Clock must be defined next, when the following message appears:

"120 HZ CLOCK INTERRUPT LEVEL=?"

Interrupt levels are entered as a number between 0 and 47, inclusive. Group 1 interrupts are numbered 0 through 23 and group 2 levels are 24-47. For instance, Group 2, level 21 would be entered in GENASYS as 45.

4-7 SOFTWARE INTERRUPT LEVELS

Two software controlled interrupts (Interrupt levels not connected to any external device) are required by VULCAN, and they must be the lowest priority levels in the system. They should be entered after the following message appears:

"SYSTEM INTERRUPT LEVELS (2) = "

Enter the two level numbers, as discussed above, on the same line, in any order.

4-8 HARDWARE OPTIONS

The presence or absence of three hardware options must be specified next, after the following message has been output:

"ENTER HARDWARE OPTIONS (SAU, TIMER, 50HZ)"

These parameters, if used, must all be entered on one line separated by blanks or a comma. "SAU" should be specified if the slash 4 VM system has a Scientific Artihmetic Unit. "TIMER" should be specified if the Slash 4 VM system has the T-register option, and it is desired that VULCAN utilize it for accounting purposes. Without the timer, VULCAN cannot maintain program execution times for accounting purposes. "50HZ" should be specified if the Slash 4 VM system is designed to run on 50HZ power. This allows VULCAN to assume a 100 Hz

Clock instead of the 120 Hz clock. If this parameter is not entered, 60 Hz power is assumed.

4-9 EXAMPLE OF REQUIRED INPUT

The following might be a sample input card deck for the items described in Section 4:

TYPE=5208 CU='0500

| | |
|---|---|
| END | . Assume Default mdd parameters |
| 80K | . Memory size |
| 21 | . 120 Hz Clock interrupt |
| 22,23 | . System interrupts |
| SAU, TIMER | . Specifies SAU hardware, interval timer to be used for accounting. |

# SECTION V
# PERIPHERAL DEVICE DEFINITIONS

## 5-1 GENERAL

The next input to VULCAN must be the physical device definitions. This is done after the following message is output:

"ENTER PHYSICAL DEVICE DEFINITIONS"

The physical devices should be defined, as discussed in the following paragraphs, by entering one device per line, followed by a record containing only "END". If it is necessary for a device to be continued onto a second record, a continuation may be indicated by placing a semi-colon (;) at the end of the first record (but before column 72) and continuing on the next record. The semi-colon should be placed between parameters, separated by blanks.

On each record the first parameter must be the device name, followed by the other parameters in any order except as noted below.

### 5-1.1 Physical Device Numbers

Each peripheral in a VULCAN system is assigned a unique physical device number (PDN), except discs, each of which have a unique disc number (DISCNO). Physical device numbers must be between 2 and 254, and may not be duplicated. They need not be assigned in any order or consecutively.

### 5-1.2 Channel Unit Numbers

Each peripheral device in a VULCAN system has associated with it a hardware channel and unit number. Channels are numbered 0-37 $_{(8)}$ and units are 0-17 $_{(8)}$. These two are normally represented as the character string CCUU for channel and unit. For instance, channel 3 unit 2 might be entered as '0302. A channel/unit representation must be given for each device.

### 5-1.3 Interrupts

VULCAN uses interrupt controlled input/output. Therefore each peripheral requires an external interrupt. All devices require only one interrupt. VULCAN does not support 2 separate interrupts for any standard device. Thus the input and output interrupts must be or-ed together at the hardware level on devices having both.

There are a maximum of 48 interrupts on a Slash 4 VMS. This would be 24 levels on each of two groups. Group 1 interrupts are represented as 0-23 (0 is highest priority) and group 2 interrupts correspondingly as 24-47.

5-1.4 Resident Flag

The resident flag may be entered for most devices (except discs, which are implicitly resident). The parameter is used to cause the specified device handler to be loaded into memory when VULCAN is loaded from disc, and remain until unloaded by the operator. This reduces disc transfers for loading of system handlers. Additionally, card readers and interactive terminals will be initialized and armed for input if declared resident. This allows terminals to immediately sign-on, and card readers to read cards without operator intervention.

5-2 DISC DRIVES

Disc drives are entered using the following parameters. Double platter cartridge discs are treated with one input, and some special parameters as discussed below.

The first parameter must be the Word DISC followed by the model number as follows:

DISC# nnnn

where nnnn is the model number as given in Table 4-1.

The following parameters may also be input, as required:

DISCNO=n — Disc Number. Each disc drive must have a unique integer assigned. This parameter is required. On double platter cartridge discs, two numbers must be entered, the first for the fixed platter, the second for the cartridge, as follows:

DISCNO=n,m

CU=ccuu — Channel/Unit. The Hardware channel/unit designation is a required parameter.

INTERRUPT=n — Interrupt. The hardware interrupt level (0-47) of the disc ready interrupt must be entered on this required parameter.

DRIVE=n — Drive. This parameter is required when the drive number of the particular disc unit is non-zero. The drive number should be entered as a single digit from 0-3.

Examples:

a) Cartridge Disc, Drive 0 on channel 5

DISC#5208 DISCNO=1,2 CU='0500 INTERRUPT=14

b) 2 Cartridge Discs on channel 5

DISC#5208 DISCNO=1,2 CU='0500 INTERRUPT=14

DISC#5208 DISCNO=3,4 CU='0500 INTERRUPT=14 DRIVE=1

c) 40 MB Storage Module as Master pack

DISC#5500 DISCNO=1 CU='0500 INTERRUPT=16

## 5-3 MAGNETIC TAPE UNITS

Magnetic tape drives are entered using the following parameters. Each drive is entered as a separate record. However, the first drive entered for a particular controller will define that controller.

The first parameter must be the word MAGTAPE followed by the model number as follows:

MAGTAPE# nnnn

where nnnn is the model number as defined in Table 5-1. Note that the model number used should correspond with the drive number as noted in the table. This is to ensure that there is one and only one drive with controller defined for each different Mag Tape system.

The following parameters may also be input, as required:

PDN=n — Peripheral Device Number. The peripheral device number of this mag tape drive is a required parameter. It must be between 2 and 254.

CU='ccuu — Channel/Unit. This is the hardware channel/unit designation of the controller for the device. It is required.

INTERRUPT=n — Interrupt level. This is the hardware interrupt level of the controller ready interrupt. It is required only on the definition of the first drive for the controller.

DRIVE=n — Drive Number. This is the hardware drive number of this particular drive. If not entered, drive 0 is assumed.

| | |
|---|---|
| 556BPI | This parameter should be entered if the drive has a special 556 BPI density option. This should be done only if it is not listed as a standard density in Table 5-1. |
| ACCESS='nnnnnnnn | Access Mask. This is the VULCAN access bit mask for this particular Drive. A user may not access this drive unless he has an access bit that matches one of these. The default is '77777777. |
| RESIDENT | Resident Flag. This parameter is used to indicate that the handler for mag tapes is to remain permanently resident, instead of being loaded only when used. |

Examples:

a) Single 200 IPS Vacuum tape drive

```
MAGTAPE#6202   PDN=11  CU='0700           INTERRUPT=15
```

b) Dual low speed tapes on single controller

```
MAGTAPE#6630   PDN=11  CU='0700           INTERRUPT=15
MAGTAPE#6631   PDN=12  CU='0700  DRIVE=1  INTERRUPT=15
```

c) Dual 1600 BPI 200 IPS drives on single controller with access bit 7 required.

```
MAGTAPE#6212   PDN=20  CU='0700           INTERRUPT=16 ACCESS='200
MAGTAPE#6215   PDN=21  CU='0700  DRIVE=1  INTERRUPT=16 ACCESS='200
```

TABLE 5-1

MAGTAPE MODEL NUMBERS

| Model No. | Speed (ips) | Tracks | Density | Drive Numbers |
|---|---|---|---|---|
| 6200 | 100 | 9 | 800 | 0 |
| 6201 | 150 | 9 | 800 | 0 |
| 6202 | 200 | 9 | 800 | 0 |
| 6203 | 100 | 9 | 800 | 1-3 |
| 6204 | 150 | 9 | 800 | 1-3 |
| 6205 | 200 | 9 | 800 | 1-3 |
| 6210 | 100 | 9 | 1600 | 0 |
| 6211 | 150 | 9 | 1600 | 0 |
| 6212 | 200 | 9 | 1600 | 0 |
| 6213 | 100 | 9 | 1600 | 1-3 |
| 6214 | 150 | 9 | 1600 | 1-3 |
| 6215 | 200 | 9 | 1600 | 1-3 |
| 6220 | 100 | 9 | 800/1600 | 0 |
| 6221 | 150 | 9 | 800/1600 | 0 |
| 6222 | 200 | 9 | 800/1600 | 0 |
| 6223 | 100 | 9 | 800/1600 | 1-3 |
| 6224 | 150 | 9 | 800/1600 | 1-3 |
| 6225 | 200 | 9 | 800/1600 | 1-3 |
| 6230 | 100 | 7 | 556/800 | 0 |
| 6231 | 150 | 7 | 556/800 | 0 |
| 6232 | 200 | 7 | 556/800 | 0 |
| 6233 | 100 | 7 | 556/800 | 1-3 |
| 6234 | 150 | 7 | 556/800 | 1-3 |

TABLE 5-1 (Cont'd.)
MAGTAPE MODEL NUMBERS

| Model No. | Speed (ips) | Tracks | Density | Drive Numbers |
|---|---|---|---|---|
| 6235 | 200 | 7 | 556/800 | 1-3 |
| 6500 | 45 | 7 | 200/800 | 0 |
| 6501 | 45 | 9 | 800 | 0 |
| 6502 | 45 | 9 | 1600 | 0 |
| 6504 | 45 | 7 | 200/800 | 1-3 |
| 6505 | 45 | 9 | 800 | 1-3 |
| 6506 | 45 | 9 | 1600 | 1-3 |
| 6630 | 45 | 7 | 556/800 | 0 |
| 6631 | 45 | 7 | 556/800 | 1-3 |
| 6640 | 45 | 9 | 800 | 0 |
| 6641 | 45 | 9 | 800 | 1-3 |
| 6650 | 45 | 9 | 800/1600 | 0 |
| 6651 | 45 | 9 | 800/1600 | 1-3 |
| 6670 | 75 | 9 | 800/1600 | 0 |
| 6671 | 75 | 9 | 800/1600 | 1-3 |

## 5-4 TELETYPES & CRT'S

Because teletypes and CRT's are treated similarly in VULCAN, they use basically the same Sysgen parameters. Differences are noted below. The first parameter must be the device type as follows:

CRT (for TEC-425 interactive)

or

TTY (for all other teletype compatible devices)

The following parameters may also be input, as required:

| | |
|---|---|
| PDN=n | Peripheral Device Number. The peripheral device number of the terminal is a required parameter. It must be between 2 and 254. |
| CU='ccuu | Channel/unit. This is the hardware channel/unit designation of this teletype or CRT terminal. If the terminal is on a Multiplexor (MUX), then this is the channel/unit of the data unit of the MUX. |
| INTERRUPT=n | Interrupt level. This is the hardware interrupt level (0-47) of this device. |
| MUXCU='ccuu | MUX channel/unit. This parameter is used if, and only if, the terminal is on a multiplexor. This is the hardware channel/unit number of the control unit of the MUX. |
| MUXUNIT=nn | MUX unit number. This parameter is used if, and only if, the terminal is on a multiplexor. It defines which slot number on the MUX (0-15) this terminal is connected to. |
| HALFDUPLEX | Half-duplex flag. If it is desired to run this terminal in half-duplex mode (default=full duplex), this parameter should be entered. If entered, the characters input will not be echoed back to the print unit. |
| ASYNC | Asynchronous flag. If the terminal is connected via an asynchronous interface other than the MUX, this parameter must be entered. |
| MODEM | Modem flag. If the terminal is connected to the Slash 4 system via a Modem, this parameter is required. |
| AUTOANSWER | Auto-answer mode. If it is desired to have VULCAN automatically answer an incoming call for this terminal, and the modem is so equipped, this parameter should be entered. |

| | |
|---|---|
| PRIOR=n | Priority. This parameter defines the default execution priority for programs run from this terminal. It must be between 16 and 32. |
| ACCESS='nnnnnnnn | Access bit mask. This is the VULCAN access mask for this particular device. A user may not access this terminal unless he has an access bit that matches one in this mask. The default is '77777777. |
| CASSETTES | Cassettes model. This parameter may be used only on teletypes and if present indicates that the TTY is a Model 2200 (TI-733) with cassettes (ASR). |
| RESIDENT | Resident flag. This parameter should be used if it desired to to have this terminal always initialized and ready for users. If not entered, an "IT" (Initiate Terminal) Operator Command will be required to initialize the terminal after each re-boot. |
| OPTERM | Operator terminal. This parameter may be used only on interactive CRT's. The device is defined to be the default operator CRT. Only one device may have an OPTERM specification. If not defined, the first CRT specified becomes the operator device. |

Examples:

a) Standard CRT to be operator terminal:

```
CRT   PDN=40   CU='0000   INTERRUPT=8   OPTERM
```

b) Model 2200 teletype with cassettes:

```
TTY   PDN=41   CU='0001   INTERRUPT=9   CASSETTES RESIDENT
```

c) Two CRT's on slots 0 and 1 of multiplexor:

```
CRT PDN=42 CU='0101 MUXCU='0100 MUXUNIT=0 INTERRUPT=9
CRT PDN=43 CU='0101 MUXCU='0100 MUXUNIT=1 INTERRUPT=9
```

d) Standard teletype device connected via autoanswer modem on multiplexor:

```
TTY PDN=44 CU='0101 MUXCU='0100 MUXUNIT=2; AUTOANSWER   ;
MODEM   INTERRUPT=9
```

e) Half-duplex teletype connected via modem:

TTY PDN=45 CU='0002 ASYNC MODEM INTERRUPT=10 HALFDUPLEX

5-5 LINE PRINTERS

Line Printers are defined using the following parameters. The first parameter must be the word LINEPRINTER followed by the model number, as follows:

LINEPRINTER #nnnn

where nnnn is the model number as defined in Table 5-2.

The following parameters may also be input, as required.

| | |
|---|---|
| PDN=n | Peripheral Device Number. The peripheral device number of this Line Printer is a required parameter. It must be between 2 and 254. |
| CU='ccuu | Channel/Unit. The hardware channel/unit designation for this Line Printer is a required parameter. |
| INTERRUPT=n | Interrupt Level. The hardware interrupt level (0-47) is a required parameter. |
| ALTPDN=n,m | Alternate PDN's. This optional parameter may specify one or two additional PDN's to which this Line Printer will respond. (Refer to Chapter A) They may be shared with other devices' ALTPDNs. |
| LISTDEVICE | List Device Flag. This parameter may be present on at most one line-printer definition. If specified, it indicates that this printer is the system default list-out device. |
| ACCESS=nnnnnnnn | Access Mask. This is the VULCAN access bit mask for this particular printer. A user may not access this printer unless he has an access bit that matches a bit in this word. The default is '77777777. |
| RESIDENT | Resident Flag. If entered, this parameter indicates that the handler for this printer is to remain core-resident at all times. |

CBC — CBC Flag. This parameter should be used only on VERSATEC Printer/Plotters (Models 74xx), and if present indicates that the printer is on a CBC channel rather than a 24-bit IOC.

Examples:

a) Versatec printer/plotter on CBC channel:

```
LINEPRINTER#4707   CU='0401   PDN=6   INTERRUPT=13   CBC   ;
LISTDEVICE
```

b) Two printers on same system sharing a common PDN for output spooling:

```
LINEPRINTER#4044  PDN=2  CU='0401  INTERRUPT=13  ALTPDN=6

LINEPRINTER#4044  PDN=4  CU='0001  INTERRUPT=14  ALTPDN=6
```

TABLE 5-2
LINE PRINTER MODEL NUMBERS

| Model No. | Manufacturer |
|---|---|
| 5300 | Analex |
| 4005 | Potter |
| 4006 | Potter |
| 4010 | Dataprinter |
| 4020 | Dataproducts |
| 2042 | CDC |
| 4043 | CDC |
| 4044 | CDC |
| 4046 | CDC |
| 4440 | Tally |
| 4444 | Tally |
| 4704 | Versatec #1100A |
| 4705 | Versatec #1110A |
| 4714 | Versatec #2000A |
| 4715 | Versatec #2030A |

## 5-6 CARD READERS

Card readers are defined using the following parameters. The first parameter must be the device name as follows:

CARDREADER

The following parameters may also be input, as required:

| | |
|---|---|
| PDN=n | Peripheral Device Number. The peripheral device number of the card reader is a required parameter. It must be between 2 and 254. |
| CU='ccuu | Channel/unit. The hardware channel/unit designation for this card reader is a required parameter. |
| INTERRUPT=n | Interrupt level. The hardware interrupt level (0-47) is a required parameter. |
| OUTDEV=n | Output device. The default list output device associated with this card reader is specified as this parameter. This should be the PDN of some output device such as a line printer. If not entered, the system list device is used. |
| ACCESS='nnnnnnnn | Access Mask. This is the VULCAN access bit mask for this particular device. A user may not read $DATA input decks from this card reader unless he has an access bit that matches a bit in this word. The default is '77777777. |
| RESIDENT | Resident Flag. If this parameter is entered, it indicates that it is desired to have this card reader initially ready to accept input when the system is loaded from disc. If not entered, the Operator Command "II" (Initiate Input) will be required to start up card input from this device. |

Examples:

a) Card reader with associated output device of printer 6:

CARDREADER PDN=7 CU='0400 INTERRUPT=6 OUTDEV=6

b) Card reader to be always resident:

CARDREADER PDN=7 CU='0400 INTERRUPT=7 RESIDENT

## 5-7 CARD PUNCHES

Card Punches are defined using the following parameters. The first parameter is the device name, as follows:

CARDPUNCH

The following parameters may also be input, as required.

| | |
|---|---|
| PDN=n | Peripheral Device Number. The peripheral device number of this card punch is a required parameter. It must be between 2 and 254. |
| CU='ccuu | Channel/unit number. The hardware channel/unit designation is a required parameter. |
| INTERRUPT=n | Interrupt Level. The hardware interrupt level for this card punch (0-47) is a required parameter. |
| ALTPDN=n,m | Alternate PDN's. This optional parameter may specify one or two additional PDN's to which this Card Punch will respond. They may be shared with other devices ALTPDNs. See Chapter A. |
| ACCESS='nnnnnnnn | Access Mask. This is the VULCAN access bit mask for this particular card punch. A user may not access this card punch unless he has an access bit that matches a bit in this word. The default is '77777777. |
| RESIDENT | Resident Flag. If this parameter is entered, it indicates that it is desired to have the handler for this card punch remain core-resident at all times. |

Example: Card punch on group 2 level 3, with shared output as PDN 10:

```
CARDPUNCH  PDN=8  CU='0402  INTERRUPT=27  ALTPDN=10  ;
ACCESS='101
```

## 5-8 PAPER TAPE PUNCHES

Paper Tape Punches are defined using the following parameters. The first parameter is the device name, as follows:

TAPEPUNCH

The following parameters may also be input, as required:

| | |
|---|---|
| PDN=n | Peripheral Device Number. The peripheral device number of this paper tape punch is a required parameter. |
| CU='ccuu | Channel/unit number. The hardware channel/unit designation is a required parameter. |
| INTERRUPT=n | Interrupt Level. The hardware interrupt level for this paper tape punch (0-47) is a required parameter. |
| ALTPDN=n,m | Alternate PDN's. This optional parameter may be used to specify one or two additional PDN's to which this tape punch will respond. They may be shared with other devices' ALTPDNs. See Chapter A. |
| ACCESS='nnnnnnnn | Access Mask. This is the VULCAN access bit mask for this particular paper tape punch. A user may not access this tape punch unless he has an access bit that matches a bit in this word. The default is '77777777. |
| RESIDENT | Resident Flag. If this parameter is entered, it indicates that it is desired to have the handler for this tape punch remain core-resident at all times. |

Example: Tape punch with access bits 7 or 0, with handler to be core-resident:

```
TAPEPUNCH PDN=5 CU='0002 INTERRUPT=10 ACCESS='201 ;
RESIDENT
```

## 5-9 PAPER TAPE READERS

Paper Tape Readers are defined using the following parameters. The first parameter must be the word TAPEREADER followed by the model number, as follows:

TAPEREADER #nnnn

where nnnn is the model number as defined in Table 5-3.

The following parameters may also be input, as required:

| | |
|---|---|
| PDN=n | Peripheral Device Number. The peripheral device number of the tape reader is a required parameter. It must be between 2 and 254. |

| | |
|---|---|
| CU='ccuu | Channel/Unit. The hardware channel/unit designation for this paper tape reader is a required parameter. |
| INTERRUPT=n | Interrupt Level. The hardware interrupt level (0–47) is a required parameter. |
| ACCESS='nnnnnnnn | Access mask. This is the VULCAN access bit mask for this particular device. A user may not resource this device unless he has an access bit that matches a bit in this word. The default is '77777777. |
| RESIDENT | Resident flag. If this parameter is entered, it indicates that the I/O handler for this paper tape reader will remain core-resident at all times. |

Examples:

a) Remex reader with access bits 23 or 22 required to reference device:

TAPEREADER#2005 PDN=4 CU='0001 INTERRUPT=13 ;
ACCESS='60000000

b) Digitronics reader with no access restrictions:

TAPEREADER#6002 PDN=4 CU='0200 INTERRUPT=18

TABLE 5-3

PAPER TAPE READER MODEL NUMBERS

| Model No. | Manufacturer |
|---|---|
| 2005 | Remex |
| 2015 | Remex |
| 2020 | Remex |
| 6002 | Digitronics |

## 5-10 TERMINATING I/O DEVICES

The final input to the Physical Device definitions is the single parameter

END

When this has been entered, GENASYS prepares to output the Physical Device Summary. This is a list of

a) PDN assignments

b) Channel/unit assignments

c) External interrupt assignments

If the output device is a line printer, the summary is automatically output. If the output device is a CRT or a TTY GENASYS will output the following message to the operator device:

"OUTPUT PHYSICAL DEVICE SUMMARY?"

To this the operator should respond with a YES if output to the CRT or TTY is required or a NO if the Physical Device Summary should be suppressed.

# SECTION VI
# WORK AREAS

## 6-1 GENERAL

Work areas are used by control point and interactive programs for scratch disc space and are unique for each individual user. All work area names are two characters (such as "LR"). When a user program first references one of these work areas, his terminal number is automatically appended to the work area name to generate a unique name on disc. Additional information is presented in Chapter A.

## 6-2 SYSTEM DEFAULT WORK AREAS

Certain work areas are automatically supplied by VULCAN. These are listed below:

| Name | Type (Blocked/Unblocked) | Granule Size (sectors) |
|---|---|---|
| LR | B | * |
| XE | U | ** |
| LO | B | * |
| ED | U | 20 |
| RC | B | 40 |
| ER | U | 2 |
| TP | B | ** |
| W1 | B | * |
| OA | B | * |

* - System Default for the disc
** - Generated as needed by system

## 6-3 SPECIFYING WORK AREAS

Additional work areas, in addition to those listed above, may be defined after the following message is output.

"ENTER INTERPOINT WORK-FILE NAMES, ONE PER LINE"

Work areas may then be defined, as discussed below, one area-name per record, terminated by the required record:

END

The format of an input record is as follows:

areaname type granule-size

where areaname = 2 character name

type = optional type specification, either B for blocked, or U for unblocked, default being blocked.

granule-size = granule size of the area in sectors, default being the system default for the work pack. (See Section 9-3)

## 6-4 EXAMPLES

The following is sample work-area input:

```
W1  B  20
W2  U
W3  30
END
```

(creates work areas W1, W2, W3; W1 and W3 are blocked, W2 is unblocked; W1 granule size=20 sectors, W2=system default, W3=30 sectors)

# SECTION VII
# MEMORY CONFIGURATION OPTIONS

## 7-1 GENERAL

This section describes how to specify semi-conductor (Hi-speed) memory and how to indicate to VULCAN memory which is either non-existant or not available. The memory configuration information is input after the following message has been produced:

"ENTER MEMORY CONFIGURATION OPTIONS"

Parameters may then be input on one line, as specified below, and terminated with the required parameter "END".

## 7-2 HI-SPEED MEMORY

Hi-Speed memory is the multiport semi-conductor memory available on the Slash 4 systems. It must be specified to GENASYS in order that VULCAN may use it as an allocatable system resource. This allows user programs to request hi-speed memory for rapid I/O transfers.

If the system has hi-speed memory, but it is desired to treat it as normal memory, with no special allocation features, it need not be specified to GENASYS.

Hi-speed memory is specified following the parameter "HSMEM".

## 7-3 UN-AVAILABLE MEMORY

Unavailable memory is that which is not physically in the machine, or not available for VULCAN use. For instance, if a certain 8K bank of memory is unusable, it may be included in the total memory available and then specified as unavailable memory.

Unavailable memory is specified following the parameter "NONMEM".

## 7-4 SPECIFYING MEMORY AREAS

Memory areas may be specified after the above parameters in either of two ways. One is by giving the individual page (page=1024 words) numbers, the other by giving two page numbers separated by a dash to indicate a group of pages. Page numbers must be between 0 and 255. However, the first 32 pages must be available for VULCAN use. For example, the following two specifications are equivalent and are used to specify that memory locations '120000 through '1377777 are hi-speed memory.

```
HSMEM     40-47
HSMEM     40,41,42,43,44,45,46,47
```

# SECTION VIII
# PRIORITY PARAMETERS

## 8-1 GENERAL

This section describes the parameters used to change the priority structure in VULCAN. These parameters allow the system user to increase or decrease the number of unique priorities in a particular area, and to change the priority of certain system programs. Care must be taken, however, as these parameters are critical to the proper operation of VULCAN.

## 8-2 SPECIFYING PARAMETERS

The priority parameters are to be input after the output of the following message:

"ENTER PRIORITY PARAMETERS"

Parameters listed below may then be input, on one line, and terminated with the required record "END".

## 8-3 PARAMETERS

The following parameters may be input:

| | |
|---|---|
| PHIGH=n | Priority-high. This sets the maximum priority of any program in VULCAN. It must be between 1 and 127. The default is 63. |
| INTPRT=n | Interactive Priority. This parameter sets the upper limit for interactive terminal programs. It must be less than "PHIGH" and greater than "BATCHP" (below). The fault is 31. |
| BATCHP=n | Batch Priority. This parameter sets the highest priority for a batch program. It must be less than "INTPRI". The default is 15. |
| OPCOMP=n | Opcom Priority. This is the priority at which Operator Communications commands will be executed. For proper system operation, it should be among the highest priorities in the system. The default is 60. |

| | |
|---|---|
| EXECPR=n | Exec Priority. This is the priority of the resident VULCAN executive program, which is used to initiate batch, interactive, and spooling programs in VULCAN. The default is 34. |
| INPUTPRI=n | Input Priority. This is the priority of the input spooling programs. (Card Reader handlers) The default is 33. |
| OUTPRI=n | Output Priority. This is the priority of the output spooling programs (Line Printer, card punch, tape punch handlers). The default is 33. |

# SECTION IX
# MISCELLANEOUS SYSTEM PARAMETERS

## 9-1 GENERAL

This section describes the remaining parameters used to specify miscellaneous items throughout VULCAN.

## 9-2 SPECIFYING PARAMETERS

The miscellaneous parameters are to be input after the output of the following message:

"ENTER SPECIAL SYSTEM PARAMETERS"

Parameters listed below may then be input, one or more per line, and terminated with the required record "END".

## 9-3 PARAMETERS

The parameters listed below may be input as required:

| | |
|---|---|
| SWAPAREA=n | Swap Area. This parameter defines the size of the page swapping area in sectors. It should be a multiple of 10. This parameter also defines the number of VAR's (Virtual Address Registers) to be used by VULCAN, (a tenth of the swap area size). The default is 2000 sectors for 200 VAR's. |
| SWAPPACK=n | Swap Pack. This is the disc pack on which the swapping area will be generated. It must be a permanently resident disc pack. The default is pack 1. |
| T/S=n | Time-slice. This is the number of 120 Hz clock increments between time slices for programs of equal priority. The default is 2. This value may also be changed dynamically by the operator. |
| WORKPACK=n | Work area pack. This is the disc pack number on which system work areas (see Section VI) and spool areas are created. It is also the default pack used for file creation when none is specified by a user. This should specify a permanently resident pack. The default is pack 1. |

ACPACK=n — Accounting pack. This is the disc pack number on which system accounting areas will be generated. It should be a permanently resident pack. The default is pack 1.

CPMAX=n — Control Point Maximum. This is the maximum number of control points that can ever be concurrently active in VULCAN. For proper identification, it should be less than 27. The default is 16.

DAMAX=n — Default disc area maximum. This is the default maximum size in sectors to which a disc area can expand if not specified when generated. The default is 8388607 sectors.

LISTDEVICE=n — List out device. This sets the system default list device "n". "n" must have been previously defined as a physical device. No LISTDEVICE parameter may have been used in the physical device definitions. If no LISTDEVICE is defined anywhere, then the first line printer specified becomes the list device. If no line printers were defined then the operator CRT is the list device.

TIMELIMIT=n — Default Time Limit. This parameter defines the default time limit in seconds for background jobs. The default is 8388607 seconds.

LINELIMIT=n — Default Line Limit. This parameter defines the default spool-out line limit in records for background jobs. The default is 8388607 lines.

TUNE 4=n — This parameter is used with the automatic control point algorithm. It sets the minimum number of control points to run. The default is 1. (Refer to Chapter A.)

TUNE 5=n — This parameter is used with the automatic control point algorithm. It defines the desirable number of pages to be left for each active interactive terminal. The default is 4. (Chapter A)

TUNE 6=n — This parameter is used with the automatic control point algorithm. It defines the desirable number of pages to be left for each active control point. The default is 6. (Chapter A)

TAPEOP parameters — This parameter is used to set the system default tape option word for resourcing magnetic tapes.

The parameters are defined as follows:

| | | |
|---|---|---|
| TRACKS = 7, 9 | | 7 or 9 track drive |
| BPI = 200, 556, 800, 1600 | | density |
| CPW = 1, 2, 3, 4 | | characters per word |
| BCD, EBCDIC | | for conversion |
| LOW SPEED, HIGH SPEED | | to select default tape drive speed. Low is less than 70 ips. |

If the TAPEOP parameter is not entered, the default is as follows:

TAPEOP TRACKS=9 BPI=800 CPW=3 LOWSPEED

# SECTION X
# PERMANENTLY RESIDENT SERVICES

## 10-1 GENERAL

This section describes the input of non-resident service names, which allows the VULCAN user to specify that any of these services are to remain permanently resident in memory instead of being loaded as needed. This has the same effect as the "LS" operator command.

## 10-2 SERVICE NAME INPUT

Services to be marked permanently resident should be input after the following message is output:

"ENTER PERMANENTLY RESIDENT SERVICES"

Service names may then be input, on one line, exactly as listed below, and terminated by the required parameter "END".

## 10-3 SERVICES

The following list represents the non-resident services supplied with VULCAN. Additional user services may be created by users as needed.

| | |
|---|---|
| OPC0 | OPCOM Zero. Operator Communications command interpreter. |
| OPC1<br>OPC2<br>OPC3<br>OPC4<br>OPC5<br>OPC6<br>OPC7<br>OPC8<br>OPC9<br>OPCA | Operator Communications segments. See Chapter F for commands handled by each segment. |
| DSC1<br>DS1X | Disc Area Creation Services (Also used for Rename) |

| | |
|---|---|
| DSC2 | Disc Area Rename and deletion services |
| DSC3 | Used by OPCOM on AD command |
| DSC4 | Disc Area Retype and squeeze processor |
| REHH | Disc Directory Rehash service |
| DSPC | Disc Space Allocator |
| USPC | Disc Space De-allocator |
| BLAH | Blocked Disc Area Handler |
| MAGT | Mag Tape Handler |
| CRDH | Card Reader Handler |
| TTYH | Teletype Handler |
| CRTH | CRT Handler (automatically resident) |
| LP1H | Universal Line Printer Handler |
| CPH | Card Punch Handler |
| PTPH | Paper Tape Punch Handler |
| PTRH | Paper Tape Reader Handler |
| CEAS | EBCDIC:ASCII Conversion |
| CBAS | BCD:ASCII Conversion |
| RSRC | Resource Allocation Service |
| RSC2 | Resource Allocation Phase 2 |
| RSEX | Resource De-allocation Service |
| TENS | 5-second System Check Program |
| SERV | Background Services |
| TOAD | Special System Services ($SPOOL, $JOB) |
| USER | User-number lookup service |
| DSAV | Background service for KEEP/FETCH/MAP |
| SYI1<br>SYI2<br>SY25<br>SYI3<br>SYI4 | System Initialization Phases |
| RTEX | Real-time executive program |

# SECTION XI
# DISC AREA RESTORATION

## 11-1 DISC AREA RESTORATION

The final process in GENASYS is to copy from the binary input device to the system master disc pack, all disc areas on the system generation stream. These files include system handlers, system processors, and other disc areas as supplied with the VULCAN system. As each service is input, it is listed on the GENASYS output device.

## 11-2 ADDITIONAL USER TAPES

If the individual installation has saved additional master pack entries on Magnetic Tape via the "KEEP" command in Job Control, these may be input after the system files have been restored. GENASYS will output the following message:

"ANY MORE KEEP/FETCH TAPES TO INPUT"

If there are no additional user tapes, the user should respond with "NO". If this is done, file restoration is terminated. If there are additional tapes, the user should respond with "YES". When this is done, GENASYS will output the following message:

"MOUNT NEW TAPE AND RESPOND WHEN READY"

At this time, the next KEEP/FETCH tape should be mounted, and either XMIT or Return (as appropriate to the input device) be depressed. Disc Areas will be restored to the master pack until a double EOF is encountered.

## 11-3 MASTER OPERATOR SIGN-ON

If the system user identification areas were not restored, then VULCAN will ask for the master system operator via the following message:

"DEFINE MASTER SYSTEM OPERATOR"

The following parameters should then be entered:

usernumber qualifier username

where "user-number" is either a 1-6 character or 12-digit user number for the master user.

"qualifier" is an eight character sign-on qualifier for the master user (4-digit account, and 4-character identifier).

"user-name" is a 1-12 character name for the master system operator. Refer to Chapter A for further discussions of user parameter formats.

The master user is given total access to all VULCAN features.

## 11-4 END OF GENASYS

When GENASYS is complete, the following messages are output:

"VULCAN IS HOT!!"

"BOOT VULCAN?"

The user may respond YES or NO to the "Boot Vulcan?" message. A YES will load and execute the newly generated VULCAN system.

# SECTION XII
# RE-SYSGEN PROCEDURE

This section describes the procedures necessary to perform a system generation when a VULCAN system has been functioning and it is desired to keep user date files and other information.

a) Using Job Control $KEEP command, save all user disc areas on the master disc (pack 1) that must be saved for use on the future system. These should be all disc areas not contained on the GENASYS tape. This may be done by using group "keeps" on all accounts other than 0000. (See Chapter B)

b) Onto another tape, using the same tape options and parameters as the GENASYS tape, save the following disc areas via the $KEEP statement.

*V:USER
*V:UDD

c) Perform the system generation as discussed in Sections II through Section XI.

d) When GENASYS asks:

"ANY MORE KEEP/FETCH TAPES TO INPUT ?"

answer: YES.

e) Remove the GENASYS tape, and on the same drive mount the tape containing the *V:USER and *V:UDD disc areas.

f) Respond to GENASYS to indicate the tape is ready (XMIT or Carriage Return).

g) Using the Add Pack (AD) operator communications command, locate disc areas on satellite packs.

h) Using the Job Control $FETCH command, reload the user disc areas from disc 1 back onto the new master pack.

i) The system is now ready.

# SECTION XIII
# SAMPLE GENASYS INPUT

This section lists a sample GENASYS run. The notation in the left-hand column is the source of the record: "GEN" is GENASYS output to operator, "OP" is Operator input and "CR" is card reader input. A sample line printer output for this run follows.

```
GEN:  VULCAN GENASYS    00. DEC  74
GEN:  ENTER INPUT DEVICE
OP:   CR    CU='0400
GEN:  ENTER OUTPUT DEVICE
OP:   LP#4044    CU='0001
GEN:  DO YOU WISH TO PRESERVE DISC CONTENTS?
OP:   NO
GEN:  DO YOU WANT ENTIRE PACK ZEROED?
OP:   NO
GEN:  *******

CR:   TYPE=5208    CU='0500
CR:   END
CR:   65K
CR:   21
CR:   22, 23
CR:   SAU, TIMER
CR:   CRT    PDN=40    CU='0000    INTERRUPT=8    OPTERM
CR:   TTY    PDN=41    CU='0002    INTERRUPT=9    RESIDENT
CR:   TTY    PDN=42    CU='0101    MUXUNIT=0    ;
CR:   MUXCU='0100    INTERRUPT=10    CASSETTES
```

```
CR:   LINEPRINTER#4044   PDN=6   CU='0001
CR:   INTERRUPT=11   LISTDEVICE
CR:   CARDREADER   PDN=7   CU='0400   INTERRUPT=7
CR:   DISC#5208   DISCNO=1,2   CU='0500   INTERRUPT=14
CR:   DISC#5208   DISCNO=3,4   CU='0500   INTERRUPT=14   ;
CR:            DRIVE=1
CR:   MAGTAPE#6640   PDN=11   CU='0700   INTERRUPT=15
CR:   MAGTAPE#6641   PDN=12   CU='0700   INTERRUPT=15   ;
CR:                      DRIVE=1
CR:   MAGTAPE#6202   PDN=13   CU='1000   INTERRUPT=16
CR:   END
CR:   W2
CR:   U1   U
CR:   END
CR:   END
CR:   OUTPRI=20   END
CR:   TAPEOP   CPW=3   BPI=800   TRACKS=9   LOWSPEED
CR:   LINELIMIT=1000   END
CR:   TENS   RTEX     END
GEN:  ANY MORE KEEP/FETCH TAPES TO INPUT?
OP:   NO
GEN:  DEFINE MASTER SYSTEM OPERATOR
OP:   ABC.1   0000SYST   MASTER-OP
GEN:  VULCAN IS HOT!
GEN:  BOOT VULCAN?
OP:   YES
```