CDC® VSOS VERSION 2

FOR USE WITH
CYBER 200 SERIES
COMPUTER SYSTEM

Volume 2 of 2

REFERENCE MANUAL

CONTROL
DATA

CDC® VSOS VERSION 2

FOR USE WITH
CYBER 200 SERIES
COMPUTER SYSTEM

Volume 2 of 2

REFERENCE MANUAL

CONTROL
DATA

60459420

REVISION RECORD

REVISION	DESCRIPTION
A (04-16-82)	Manual released.
B (10-15-82)	Manual revised to reflect VSOS 2.0 corrective code release at PSR level 575.
C (07-29-83)	Manual revised to reflect VSOS 2.1 at PSR level 592. New features documented include support of on-line magnetic tapes, IQM performance improvements, and channel utilization statistics. This edition obsoletes all previous editions. Due to extensive reorganization, change bars and dots are not being used at this revision and all pages reflect the current revision level.
D (03-30-84)	Manual revised to reflect VSOS 2.1.5 at PSR level 607. Changes include documentation of adding the job name to the account record and various other technical and editing changes.
E (10-31-85)	Manual revised to reflect VSOS 2.2 at PSR level 644. Changes include documentation of dynamic file allocation, project tracking, multiple batch jobs per user and various other technical and editing changes. Due to extensive changes, change bars and dots are not used and all pages reflect the latest revision level. This edition obsoletes all previous editions.
F (04-18-86)	Manual revised to reflect VSOS 2.2.5 at PSR level 654. Changes have been made to descriptions of I/O connectors for explicit and implicit I/O and the explicit I/O system message has been updated. Various other technical and editing changes have also been made.
G (12-05-86)	Manual revised to reflect VSOS 2.3 at PSR level 670. Various technical and editing changes have been made.
H (10-23-87)	Manual revised to reflect VSOS 2.3.5 at PSR level 690.

Publication No.
60459420

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

© 1982, 1983, 1984, 1985, 1986, 1987
by Control Data Corporation
All rights reserved
Printed in the United States of America

Address comments concerning this manual to:

Control Data Corporation
Technology and Publications Division
4201 North Lexington Avenue
St. Paul, Minnesota 55126-6198

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Front Cover	-
Title Page	-
2	H
3	H
4	H
5	F
6	E
7	H
8	H
9	G
10	H
11	H
12	E
1-1	F
1-2	F
1-3	F
1-4	G
1-5	H
1-6	H
1-7	F
2-1	F
2-2	H
2-3	H
2-4	H
2-5	H
2-6	H
2-7	H
2-8	H
2-8.1/2-8.2	H
2-9	E
2-10	F
2-11	E
2-12	E
2-13	E
2-14	E
2-15	H
2-16	E
2-17	E
2-18	H
2-19	G
2-20	F
2-20.1/2-20.2	G
2-21	G
2-22	G
2-23	E
2-24	E
2-25	G
2-26	G
2-27	E
2-28	G
2-29	E
2-30	F
2-31	E
2-32	E
2-33	E
2-34	F
3-1	F
3-2	H
3-3	F

PAGE	REV
3-4	F
3-5	F
4-1	E
4-2	E
4-3	E
4-4	E
4-5	E
4-6	E
4-7	E
4-8	E
4-9	E
4-10	E
4-11	E
4-12	E
4-13	E
4-14	E
4-15	E
4-16	E
5-1	G
5-2	F
5-3	F
5-4	F
5-5	F
5-6	F
5-7	F
5-8	E
5-9	G
5-10	H
5-11	F
5-12	F
5-13	E
5-14	G
5-15	G
5-16	G
5-17	F
5-18	F
5-19	F
5-20	E
5-21	F
5-22	H
5-23	E
5-24	E
5-25	E
5-26	F
5-27	E
5-28	E
5-29	E
5-30	F
5-31	G
5-32	G
5-33	H
5-34	H
5-35	H
5-36	G
5-37	E
5-38	F
5-39	F
5-40	E
5-41	E

PAGE	REV
5-42	E
5-43	H
5-44	H
5-45	H
5-46	H
5-47	F
5-48	F
5-49	E
5-50	F
5-51	H
5-52	E
5-53	H
5-54	H
5-55	E
5-56	H
5-57	E
5-58	H
5-59	E
5-60	E
5-61	E
5-62	E
5-63	E
5-64	E
5-65	F
5-66	F
5-67	E
5-68	E
5-69	H
5-70	E
5-71	E
5-72	F
5-73	E
5-74	E
5-75	E
5-76	E
5-77	E
5-78	E
5-79	F
5-80	E
5-81	E
5-82	E
5-83	E
5-84	E
5-85	E
5-86	E
5-87	E
5-88	E
5-89	E
5-90	E
5-91	H
5-92	E
5-93	E
5-94	E
5-95	H
5-96	H
5-97	E
5-98	E
5-99	H
5-100	H

PAGE	REV
5-101	H
5-102	H
5-102.1/ 5-102.2	H
5-103	H
5-104	H
5-105	H
5-106	H
5-107	H
5-108	H
5-109	H
5-110	H
5-111	H
5-112	H
5-113	H
5-114	H
5-115	H
5-116	H
5-117	H
5-118	H
5-119	H
5-120	H
5-121	H
5-122	H
5-123	H
5-124	H
5-124.1	H
5-124.2	H
5-125	H
5-126	H
5-126.1/ 5-126.2	H
5-127	H
5-128	F
5-129	E
5-130	H
5-131	H
5-132	H
5-133	E
5-134	F
5-135	H
5-136	F
5-137	H
5-138	F
5-139	H
5-140	F
5-141	G
5-142	E
5-143	F
5-144	E
5-145	F
5-146	E
5-147	E
5-148	F
5-149	E
5-150	E
5-151	E
5-152	F
5-153	H

PAGE	REV
5-154	H
5-155	H
5-156	H
5-157	H
5-158	H
5-159	H
5-160	H
5-160.1	H
5-160.2	H
5-161	H
5-162	H
5-163	H
5-164	H
5-165	H
5-166	H
5-167	H
5-168	F
5-169	E
5-170	F
5-171	E
5-172	E
5-173	E
5-174	E
5-175	E
5-176	F
5-177	E
5-178	E
5-179	E
5-180	E
5-181	E
5-182	E
5-183	G
5-184	E
5-185	E
5-186	E
5-187	E
5-188	E
5-189	F
5-190	H
5-191	H
5-192	E
5-193	H
5-194	F
5-195	F
5-196	F
5-197	F
5-198	F
5-199	F
5-200	F
5-201	F
5-202	F
5-203	F
5-204	F
5-205	F
5-206	F
5-207	F
5-208	F
5-209	F
5-210	F

PAGE	REV
5-211	F
5-212	F
5-213	F
5-214	F
5-215	F
6-1	E
6-2	E
6-3	E
6-4	E
6-5	E
6-6	E
7-1	G
7-2	G
7-3	G
8-1	E
8-2	E
8-3	H
8-4	E
8-5	E
8-6	F
8-7	H
8-8	E
8-9	H
8-10	H
8-11	E
8-12	F
8-13	E
8-14	E
8-15	F
8-16	E
8-17	E
8-18	H
8-19	H
8-20	H
8-20.1/8-20.2	F
8-21	E
8-22	F
8-23	F
8-24	E
8-25	F
8-26	E
8-27	F
8-28	F
8-29	E
8-30	E
8-31	E
8-32	E
8-33	F
8-34	E
8-35	F
8-36	E
8-37	H
8-38	E
8-39	F
9-1	H
9-2	E
9-3	E
9-4	E
9-5	H
9-6	H
9-6.1/9-6.2	H
9-7	E
9-8	E
9-9	E
9-10	E
9-11	E
9-12	E
9-13	G
9-14	E
9-15	G
9-16	F
9-17	G
9-18	E
9-19	E
9-20	E
9-21	E
9-22	E
9-23	G
9-24	E
9-25	E
9-26	E
10-1	F
10-2	F
10-3	E
10-4	E
10-5	E
10-6	E
10-7	E
10-8	G
10-9	F
10-10	G
10-11	F
10-12	F
10-13	F
10-14	E
10-15	G
10-16	E
10-17	G
10-18	G
10-19	E
A-1	E
A-2	E
A-3	E
A-4	E
B-1	F
B-2	F
B-3	F
B-4	F
B-5	E
B-6	E
C-1	F
C-2	G
C-3	F
C-4	F
C-5	F
C-6	F
C-7	F
C-8	F
D-1	E
D-2	E
D-3	E
D-4	E
D-5	G
D-6	E
D-7	G
D-8	F
D-9	F
E-1	H
E-2	E
E-3	E
E-4	E
F-1	E
F-2	E
G-1	E
G-2	E
G-3	E
G-4	E
G-5	E
G-6	E
Index-1	F
Index-2	H
Index-3	H
Index-4	H
Index-5	F
Index-6	F
Comment Sheet	H
Back Cover	-

PREFACE

This manual describes the CDC® Virtual Storage Operating System (VSOS) for the CONTROL DATA® CYBER 200 Series Computer System. This manual is published in two volumes:

- Volume 1 describes system utilities and system interface language (SIL) subroutines. It also contains a general description of CYBER 200 hardware and operating system software, file concepts, and task execution. It is written primarily for applications programmers.
- Volume 2 describes system messages and job management tables. It also describes system accounting file formats, common execute line routines, and loader conventions. It is written primarily for systems programmers.

RELATED PUBLICATIONS

Related information can be found in the following publications.

Control Data Publication	Publication Number
CYBER 200 Maintenance Software System Reference Manual	60457200
CYBER 200 Model 205 Computer System Hardware Reference Manual	60256020
CYBER 200 Model 205 Troubleshooting Guide	60430060
RHF Usage	60460620
VSOS User's Guide for Fortran 200 Programmers	60455390
VSOS Site Manager's Handbook	60461490
VSOS Version 2 Reference Manual, Volume 1	60459410
VSOS Version 2 Operator's Guide	60459430
VSOS Version 2 Installation Handbook	60459440
FORTRAN 200 Reference Manual	60485000
CYBER 200 Assembler Version 2 Reference Manual	60485010
RHF Application-to-Application Interface Specification	ARH #4260

Control Data manuals can be ordered from:

Literature and Distribution Services
STP005
308 North Dale Street
St. Paul, Minnesota 55103

DISCLAIMER

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

Control Data does not support the station communication software described in this manual. It supports only the LCN/RHF communication software.

Control Data does not support the FORTRAN 66 compiler.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

NOTATIONS USED IN THIS MANUAL

UPPERCASE	Words or character strings that must be entered as shown. They must be spelled correctly including any = or / shown.
UNDERLINED UPPERCASE	Words or character strings that can be abbreviated to the number of underlined characters.
Lowercase	Generic terms which represent the words parameters or character strings supplied by the programmer. When generic terms are repeated in a format, a number or letter might be appended.
[ ] Brackets	Optional portion of a format. All parameters enclosed within the brackets can be omitted at programmer option. The brackets are editorial conventions only; they are not part of the format.
{ } Braces	Portion of a format in which only one of the vertically stacked items can be used. The braces are editorial conventions only; they are not part of the format.
. . . Ellipses	Repetition indicator. The portion of the format immediately preceding can be repeated at programmer option.
Δ	Blank indicator. In a format, this character indicates that a blank or space should appear.
#	Numbers used in this manual are decimal unless noted as hexadecimal. Hexadecimal numbers are prefixed by the # character.

Punctuation characters shown within the formats are required unless the text indicates another punctuation character can be substituted.

INTRODUCTION TO VSOS 1

The Virtual Storage Operating System (VSOS) consists of a central operating system, which runs in the central processing unit (CPU), and a peripheral operating system, which runs in the network access devices (NADs). The operating system consists of a resident system and a nonresident set of callable tasks.

CENTRAL OPERATING SYSTEM

The central operating system can be divided into three parts:

- The resident system runs in a hardware mode called monitor mode. It is always resident in memory and references memory by absolute address rather than through the virtual paging mechanisms. When the resident system is running, interrupts are inhibited and some extra instructions are enabled.
- The virtual system runs in a hardware mode called job mode. It consists of a pageable set of subroutines that perform such functions as controlling entry of users into the system, file management, and terminal message handling. Virtual system tasks communicate with the resident system by using resident system calls. The virtual system can modify system tables directly.
- Privileged system tasks run in the hardware mode called job mode and perform many of the same functions as virtual system tasks. Privileged system tasks can issue privileged resident system calls to communicate with the resident system. However, the only privileged system tasks that can modify system tables directly are Input Queue Manager (IQM) and Interactive Transfer Facility Servicer (ITFS).

CYBER 200 hardware modes are described in the CYBER 200 Hardware Reference Manual.

RESIDENT SYSTEM

The resident portion of the central operating system contains:

- KERNEL, which handles time-slicing and message communication.
- PAGER, which is responsible for main memory allocation and page swapping.

All communications between the various portions of the system are by means of system messages. These messages either pass through KERNEL, which in this case acts as a message switcher, or are processed directly by KERNEL. User jobs, privileged tasks, and virtual system tasks communicate messages to KERNEL through the exit force instruction (a machine language instruction). PAGER communicates messages to KERNEL by direct subroutine calls. The peripheral system communicates with KERNEL by setting pointers in the station queuing structure; KERNEL communicates with the peripheral system by setting pointers and station channel flags.

The time-slice management portion of KERNEL is controlled by a loop over the alternator table that acts as a circular table with various indicators in each table entry. These indicators include a pointer to a minus page table entry, a descriptor block table entry, and three sets of flag bits that define the status of each alternator entry.

A unique entry in the alternator is shared by all virtual system tasks; to prevent two routines from modifying the same system table simultaneously, only one virtual system task is allowed to run at a time. This system alternator slot has the highest priority; it is always run unless blocked for I/O or PAGER action.

All memory access interrupts, as well as certain messages dealing with physical memory allocation, are conveyed by KERNEL to PAGER for processing. PAGER dynamically allocates both large and small pages, and performs all implicit I/O necessary to free memory pages and obtain the pages caused by memory access interrupts.

That portion of memory that is not permanently occupied by the resident system and its tables is available for allocation to executing system and user tasks, including the virtual system. This allocatable memory is either allocated space (space reserved for use by a specific task) or free space (space not allocated to any task).

VIRTUAL SYSTEM

The virtual system contains routines for system functions such as file management, explicit I/O, message handling, and CPU scheduling. Only that part of the virtual system that is needed at any one time is in physical memory. The virtual system is assigned tasks by KERNEL and is initiated by KERNEL to do one type of task only. It must finish one task before it begins another.

Queuing of Virtual System Tasks

For virtual system demand tasks, which are critical to the efficient working of the operating system, queuing occurs if:

- Bits are set in one or more alternator slots to indicate that virtual system action is required.
- PAGER requests KERNEL to queue a virtual system demand task.

For periodic virtual system tasks, which are not considered critical, queuing occurs if:

- A communication from a peripheral processor requires activity.
- A user job issues a message that requests a system service not provided by the resident system.
- An entry in the periodic table indicates it is time to run a virtual system task.

Scheduler Interaction with PAGER

The CPU paging processor (PAGER) interacts with the CPU scheduler and the input queue scheduler to regulate the processing load on the CPU. The process is:

- When PAGER determines that the load on the system is excessive, it notifies the virtual system routine LOAD to suspend certain tasks (refer to the following) and/or disconnect tasks from the CPU scheduling queue (CPUQ) and places them into the wait queue.

- When PAGER determines that the system is not being fully utilized, it notifies LOAD to reconnect tasks in the wait queue to the CPUQ, resume system-suspended tasks, and/or submit new batch jobs to the CPU scheduler.

Estimates of the memory requirements of any task in the system are based on the size of the task´s working set, which is a function of the number of blocks of memory referenced or altered by the task in a given period of time. Before a task begins execution, it is assigned an initial and a maximum working set. The maximum working set is determined by the RESOURCE statement as explained in volume 1 of this reference manual. The CPU scheduler sets the initial working set based on the task´s drop file size. PAGER then monitors the memory usage of the task and adjusts its working set accordingly. PAGER will not evaluate a task´s working set to be higher than the maximum working set for the task. It will, however, keep track of the frequency with which a task is attempting to exceed its maximum working set. When the frequency becomes too high, PAGER flags the task as a candidate for suspension. When the number of candidates for suspension exceeds an installation-defined limit, PAGER notifies LOAD to suspend candidates for suspension in reverse priority order until the limit is no longer exceeded. (Observe that tasks which have a maximum working set equal to all of allocatable memory will not be considered candidates for suspension.) In addition to monitoring each task´s working set, PAGER keeps track of the sum of the working sets of all tasks in the CPUQ and wait queue (WQ). This variable is used to calculate a running average sum of working sets (known to the system as IQM_RWS).

IQM_RWS represents the load on the system after factoring out temporary dips in memory committed to tasks. For each job in the input queue, the input queue scheduler estimates the maximum working set that the job will require while in execution. One of the scheduling constraints on a job in the input queue is maximum memory overcommitment.

When a job fails this constraint, it is given a status of MXMO. IQM_RWS must be less than the lowest estimated working set among all jobs with a status of MXMO in order for any job with that status to be submitted without overcommitting memory. When IQM_RWS is evaluated by PAGER as being below this value, PAGER notifies LOAD to cause the input queue scheduler to reevaluate jobs in the input queue for submission. The input queue scheduler will attempt to resume system-suspended tasks before submitting new jobs to the CPU.

When the sum of the working sets of tasks in the CPUQ is low enough, PAGER notifies LOAD to remove tasks from the wait queue and reconnect them to the CPUQ within the limits of committable memory. A task qualifies for reconnection when it is the highest priority task in the wait queue whose working set plus memory for working set growth fits into uncommitted memory. When a task is evaluated as having a working set which will no longer fit into committable memory, it is disconnected from the CPUQ and placed into the wait queue.

PRIVILEGED SYSTEM TASKS

Any task can be run as a privileged task if it is running under a privileged user number or has the privilege flag set in the file index table entry of its source file. A privileged system task is any privileged task that is part of the central operating system. A privileged user task is any other privileged task, such as one that a system or installation-defined utility might use.

Privileged and virtual system tasks have similar characteristics in that they run in job mode, make resident calls, are pageable, and can access the files of other users. Unlike the virtual system tasks, privileged tasks (with the exception of IQM and ITFS) do not have direct access to system tables; through privileged calls to the virtual system, they are able to obtain indirect access to the tables.

Because they can make most resident system calls, privileged system tasks are able to perform some functions for the virtual system. This reduces virtual system overhead and frees the virtual system to process other functions. Tasks such as handling I/O files and operator communications are currently done by privileged system tasks.

IQM, OPERATOR, QTF, QTFS, PTFS, and ITFS are privileged system tasks that run under privileged user numbers. The system user number (for installation management), 999998, is also a privileged user number; EDITUD is run under this number.

System Task	Description
IQM	Input Queue Manager. Creates and routes error dayfiles to the user for batch jobs which could not be submitted to the CPU scheduler (user number 000003).
OPERATOR	Enables the operator to communicate with the system by issuing the EXECUTE OPERATOR COMMAND message (f=#0021), enabling the operator to display memory and task information. The operator is able to control the flow of jobs to be submitted to the CPU scheduler, the jobs which are running in the system, and the access to peripheral equipment and linked mainframes available to the system (user number 000098).
QTF	Queue File Transfer Facility. Queues input and output files from CYBER 200 to a Remote Host Facility (RHF) front end (user number 000006).
QTFS	Queue File Transfer Facility Servicer. Queues job files from an RHF front end to CYBER 200 (user number 000008).
PTFS	Permanent File Transfer Facility Servicer. Services the remote host connection for permanent file transfer (user number 000010). (Does not imply the direction of the transfer.)
ITFS	Interactive Transfer Facility Servicer. Allows interactive use of CYBER 200 (user number 000013).

Obtaining Privileged Status

A user is a privileged user if the privilege flag (udtrust field) in the user directory entry is set. The flag can be set by using EDITUD.

Privileged status for a running task is indicated in the descriptor block.

An executable file can run as a privileged task under a nonprivileged user number if the privilege flag in the source file's file index table entry is set.

If the task is not running under a privileged user number, but the privilege flag is set in its source file, controllees or a controllee chain started by the task must also have the privilege flag in their source files set to have privileged status.

The privilege flag in the user directory entry associated with each user is passed on to each task that the user executes.

Privileged Resident System Calls

Privileged resident system calls made to KERNEL are processed by KERNEL or by a peripheral device. Resident system calls available to privileged tasks are listed in table 1-1.

Before a C501, C502, or C503 call on a file can be issued by a privileged task, the file must be opened for explicit I/O, using the OPEN FILE (f=#0003) message. The file segment table (FST) ordinal (returned in the fsto Beta word field for a CREATE or OPEN message) must be supplied with the call; otherwise, the task aborts.

Privileged tasks can read and write segmented files with C500 and C501 calls, but files cannot be extended by tasks using these calls. Extensions are permitted, however, in other situations, such as when the privileged tasks perform implicit I/O or use the EXPLICIT I/O (f=#F500) message.

Table 1-1. Resident System Calls for Privileged Users (Sheet 1 of 2)

Call Number	Description
F002	Delete pages from the page table.
F003	Delete a virtual range with a given key from the page table.
F004	List a virtual range with a given key from the page table.
F005	Delete all pages in the page table under a given key.
F007	List the page table entry for the keyword and the virtual block identifier.
F008	Terminate the task.
F009	Complete outstanding boats.
F00C	Change key.
F00D	Get an input buffer.

Table 1-1. Resident System Calls for Privileged Users (Sheet 2 of 2)

Call Number	Description
F00E	Queue system/demand task.
F00F	Change system keys.
F010	Unlock a virtual range with a given key.
F015	Performance measurement call.
F016	Process checkpoint.
F017	Return to KERNEL from Virtual System Debug Tool (VSDT).
F018	List page table entry for large page. Fault for large page if not in memory.
F019	Checkpoint preprocessing.
C304	Teletype output message.
C305	Teletype input message.
C313	Full screen output message.
C320	Reserved for installation use.
C500	Read physical blocks.
C501	Write physical blocks.
C502	Read physical disk.
C503	Write physical disk.
C504	Write a disk pattern.
C510	Read logical blocks.
C511	Write logical blocks.
C512	Read logical disk.
C513	Write logical disk.
C514	Write logical pattern.
C700	Read Remote Host Facility (RHF)/loosely coupled network (LCN).
C701	Write RHF/LCN network.
C702	RHF NAD function.
C703	Receive RHF remote connection.
C704	Abort timed-out boat.

Virtual System Calls

Privileged tasks can make all nonprivileged calls to the virtual system. They can also make special virtual system calls and use options of nonprivileged calls that are restricted to privileged tasks. (All messages described in chapter 5 except EXPLICIT I/O, TAPE FUNCTION, ADVISE, PROCESS SYSTEM PARAMETER, and GIVE UP CPU ON OUTSTANDING RESIDENT I/O OR TIME are virtual system calls.)

Special virtual system calls available to privileged tasks are listed in table 1-2. The restricted capabilities of nonprivileged calls available to privileged tasks are listed in table 1-3.

Table 1-2. Virtual System Calls Available to Privileged User Tasks Only

Function Code	Message	Function
0021	EXECUTE OPERATOR COMMAND	Acts as the interface between the privileged task OPERATOR and the virtual system.
0022	EXECUTE PROGRAM FOR USER NUMBER	Starts the file transfer process for a user.
0023	UPDATE USER DIRECTORY	Modifies the user directory of an existing user or to create a new user.
002A	RHF_CALL	Reserved for RHF virtual system calls.
0030	EXECUTE IQM REQUEST	Deletes or inserts a job into the IQM.

Table 1-3. Virtual System Calls with Options Available to Privileged User Tasks Only

Function Code	Message	Function
0001	CREATE	Creates a public, pool, or private file for another user.
0002	DESTROY	Destroys a public, pool, or another user's private file.
0003	OPEN	Opens any file (private, public, or pool).
0005	CLOSE	Closes any public, private, or pool file. Destroys the file being closed.
0008	GIVE	Makes a file public; that is, gives to user 000000. Gives file to IQM (user 000003). Gives file specified by sector address and unit number. Gives file with trust bit set.
000B	CHANGE FILE ATTRIBUTES	Changes the account number of an input file.
000E	USER/ACCOUNTING COMMUNICATION	Makes accounting entry or dumps accounting temporary storage to permanent storage and terminates the accounting file.

JOB MANAGEMENT TABLES

2

Tables used by VSOS to control job processing within the system can be affected or altered indirectly by user programs. In all cases, access to the tables must be through system messages.

FILE INDEX TABLE (FILEI)

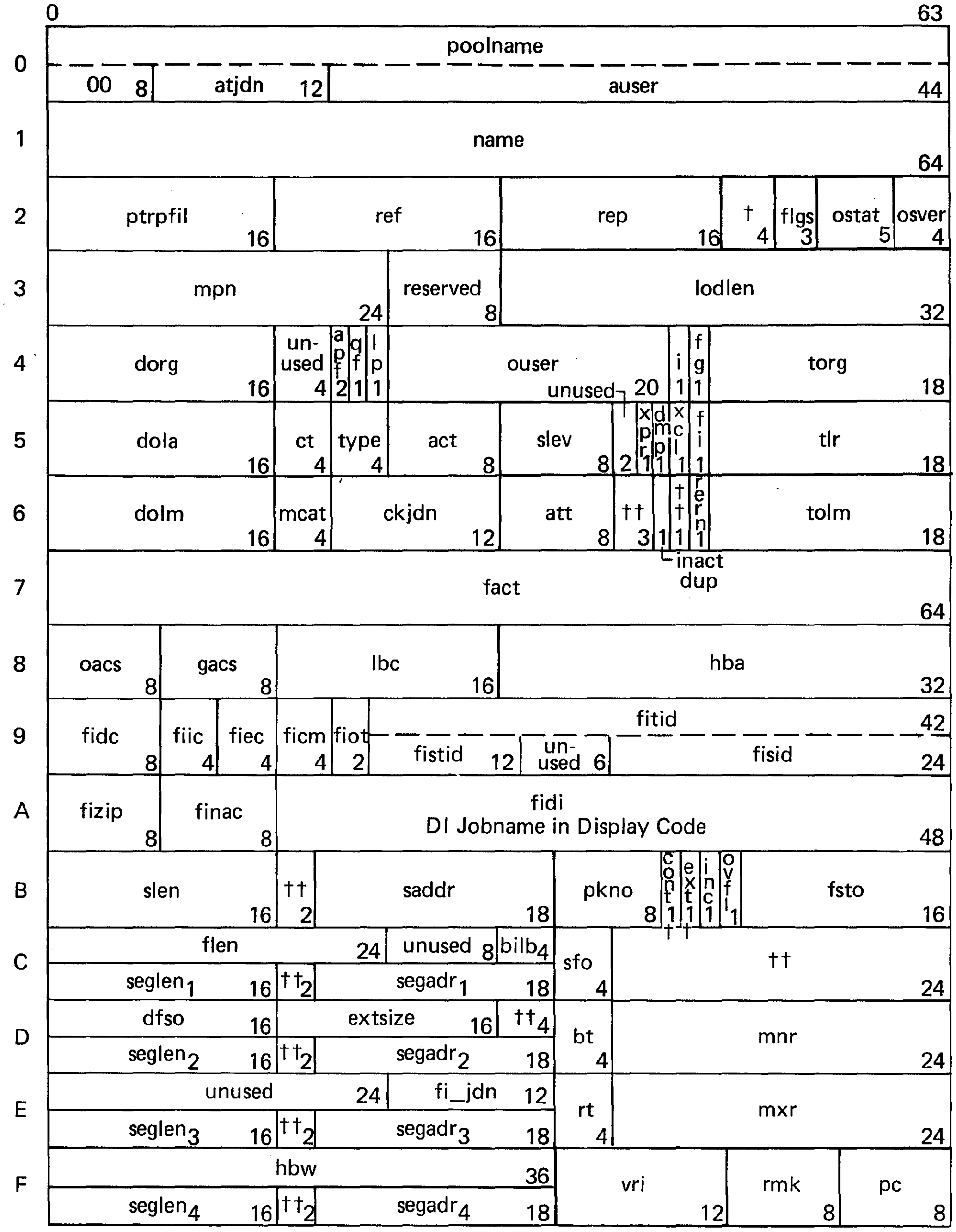
The file index table is a catalog of files connected to a terminal, mass storage files (public, private, and attached pool), and tape files for all active users in the system. The catalog entries of mass storage files for inactive users are maintained on a mass storage unit (in the inactive file index table). When a user becomes active, the catalog entries describing the user´s mass storage files are brought into the file index table in main memory.

Each entry in the file index table consists of at least one three-word top chapter and one or more 14-word bottom chapters. A top chapter exists for every attachment (or privileged open) of the file to a user. Every file has a bottom chapter containing basic file attributes. If individual user access permissions have been specified for the file, an additional bottom entry is present for the file access directory entries (FADE).

When the file index is returned, a 16-word format is returned to the caller. The third word of the top is not returned; relevant information is placed at the bottom as previously described. The format is shown in figure 2-1 and figure 2-2. All fields in an entry contain binary values, unless otherwise indicated.

The fidc, fiic, and fiec fields in the file index table entry contain specifications for the file´s disposition, internal characteristics, and external characteristics. The fidc field values are listed in table 2-1; the values of the fiec and fiic fields are listed in table 2-2.

RHF file disposition is controlled by the submitting host system and/or the user via the MFQUEUE statement. The ic field is the only field used by RHF. In the protocol exchange, the submitting RHF host includes disposition and routing information, which is saved as the last group of an input file. The batch processor copies the last group to a file Q5JRTHRF during job execution. If a file is MFQUEUEd during job execution, a copy of the Q5JRTHRF file is saved in a file named Q5Lxxxxx which is associated with the MFQUEUEd file which will be named Q5Oxxxxx, where xxxxx is a unique five-character string generated by the system microsecond clock. At the end of job execution, the Q5JRTHRF file is renamed PYYxxxxx and is given to QTF as the last member of the xxxxx family. The information in the PYYxxxxx file and the Q5Lxxxxx file are used in the RHF protocol exchange returning the output files. The front-end host examines the received protocol and disposes the file accordingly.



†Purge only (1 bit)
††Reserved.

Figure 2-1. File Index Table Format for Nontape Files (Sheet 1 of 8)

Word	Field	Description
0	poolname	Pool name in ASCII, left-justified with blank fill.
	atjdn	Job descriptor number to which this file is attached; it is assigned to a job for the life of the job in the system.
	auser	Owning user number in binary notation.
1	name	File name in ASCII, left-justified with blank fill. The information in this table refers to the file of this name. File names must be in the format described in chapter 3.
2	ptrpfil	Pointer into the proper block of the permanent file index (PFI) for this entry relative to the first block of the PFI. PFI = #FFFF for local files.
	ref	Number of times this file has been opened.
	rep	Retention period in days.
	flgs	1 (DFRSTRT) Drop file restartability flag. If 0, this drop file is restartable. If 1, this drop file is not restartable if IP_DFRSTRT is 1.
		2 (PRODTN) Production file flag. If 0, this file is not a production file. If 1, it is a production file.
		3 (PURGONLY) Purge-only flag. Indicates whether the file can be used or not because of PFI or directory of file segmentation (DFS) errors. Set by permanent file verification at autoload time. 0 File has not been flagged as purge-only. 1 File has been flagged as purge-only.
	ostat	Output file status (integer): 0 Normal status. 1 Destination LID disabled. 2 Destination not responding. 3 Destination rejecting file. 4 System interface language (SIL) error occurred during file transfer. 5 Diverted. 6 Hardware path to the logical identifier (LID) not available. 7 System error occurred during file transfer. 8-31 Reserved by CDC.
	osver	Version number of the operating system that created the file. Binary value defined as: 0 Release 2.0 or earlier. 1 Release 2.1, 2.1.5, or 2.1.6. 2 Release 2.2 or later.

Figure 2-1. File Index Table Format for Nontape Files (Sheet 2 of 8)

Word	Field	Description
3	mpn	Master project in ASCII, left-justified with blank fill.
	reserved	Reserved for installation.
	lodlen	Drop file length, in blocks, for the execute file; only valid if the file is a virtual code file (type = 2). For a file given to a privileged system task, this field contains the binary user number of the user who gave this file to the privileged system task.
4	dorg	Date this file was originated, in the format: Subfield — Description yy — Last two digits of the year. ddd — Number of days since the beginning of the year (1 through 366).
	apf	Access permission flags: Bit 1 = 1 if this entry is a file extension entry. Bit 2 = 1 if this file has an extension entry.
	qf	Queue flag: 0 IQM has read the file. 1 IQM has not read the file.
	lp	Local/permanent flag: 0 Permanent file. 1 Local file.
	ouser	Originating user number.
	i	Duplicate files flag: 0 No duplicate found. 1 Duplicate found at login.
	fg	File acquisition method: 0 User created this file. 1 File was given to user.
	torg	System clock time, in seconds after midnight, at which this file was originated.
5	dola	Date of last access to this file, in the same format as the dorg field. (To access a file means to open it.)
	ct	Communication type: 0 Non-RHF file. 1 Non-RHF file. 2 RHF file.

Figure 2-1. File Index Table Format for Nontape Files (Sheet 3 of 8)

Word	Field	Description
5	type	File type: 0 Physical file. 1 Virtual code file.
	act	Number of active I/O connectors for this file.
	slev	Security classification level of this file (1 through 8).
	xpr	If set, indicates that this file is currently being accessed by a privileged utility.
	dmp	Dump flag: 0 The file is a candidate for dumping by a privileged utility. 1 Indicates the file has been dumped by a privileged utility since creation or modification.
	xcl	Exclusive access flag: 0 File can be shared. 1 File cannot be shared.
	fi	Task privilege designator: 0 Not a privileged task. 1 Privileged task.
	tlr	System clock time, in seconds since midnight, at which the file was last opened.
6	dolm	Date the file was last opened with write access, in the same format as the dorg field.
	mcat	Present file management category: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file. 4 Magnetic tape file. 6 System-generated drop file. 7 Batch file. 9 Connected file. #A Checkpointed output file. #F Checkpointed input file.
	ckjdn	Checkpoint job descriptor number. This field is set when the file belongs to a checkpointed job.
	att	Count of the attaches and privileged opens for private files; the number of opens for public and pool files.

Figure 2-1. File Index Table Format for Nontape Files (Sheet 4 of 8)

Word	Field	Description
	inactdup	Inactive duplicate file.
	rern	Rerun indicator; valid for batch input files only (mcat = 7).
	tolm	Time, in seconds since midnight, when this file was last opened with write access.
7	fact	Account number of the file, in ASCII, left-justified with blank fill.
8	oacs	Owner´s access permission for private files and pool boss´ access permission for pool files. For public files, oacs is set equal to gacs: B1 B2 B3 B4 B5 B6 B7 B8 Bit Access Permitted 1-3 Reserved (ignored by the system). 4 Execute. 5 Modify. 6 Append. 7 Read. 8 Write.
	gacs	General access permissions. The format of this field is the same as oacs.
	lbc	Last used bit count in the last block to which something was written in the file. Used only for files with osver less than 2.
	hba	Bit address plus 1 of the highest page accessed. Used only for files with osver less than 2.
9	fidc	Numeric value indicating disposition of this file (refer to table 2-1).
	fiic	Numeric value indicating the internal format characteristics of the file (refer to table 2-2).
	fiec	Numeric value indicating the external punch representation characteristics of the file (refer to table 2-2).
	ficm	Numeric value indicating the conversion mode of the file. The values are: 0 Display code (64-character set). 1 Extended display code (128-character set). 2 Binary.
	fiot	Numeric value indicating the origin type of the file. Values are: 0 Local batch. 1 Remote batch. 2 Interactive.

Figure 2-1. File Index Table Format for Nontape Files (Sheet 5 of 8)

Word	Field	Description
	fitid	A seven-character terminal identifier stored as seven 6-bit display code characters.
9	fistid	A two-character terminal identifier stored as two 6-bit display code characters.
	fisid	A three-character (ASCII) mainframe identifier indicating the source or destination mainframe.
A	fizip	Numeric value indicating the destination processor zip number for this file. The mainframe table associates this zip with a three-character mainframe identifier.
	finac	Access station area code.
	fidi	Job name (eight 6-bit display code characters).
B	slen	Length of the file, in blocks. Not used for files with osver = 2 and cont = 0.
	saddr	Starting sector address on disk for the first segment of the file. Not valid for files with osver = 2 and cont = 1. (For these files, saddr is set to #3FFFF.)
	pkno	Pack number of the disk device on which the first segment of the file is allocated.
	cont	File contiguity flag: 0 Segmented file or more than #FFFF blocks. 1 Unsegmented file and less than #10000 blocks.
	ext	File extendability flag (set when file is created or opened): 0 File is extendable. 1 File is not extendable.
	inc	File completeness flag (set when file is attached and promoted to FILEI): 0 File does not span a downed device. 1 File spans a downed device.
	ovfl	File overflow flag: 0 File is contained on one device. 1 File spans more than one device.
	fsto	File segment table ordinal; when file is attached and promoted to FILEI, this field contains the ordinal of the first FST entry for the file.
C	flen	For files created on VSOS 2.2 or later systems, this field contains the total allocated space for the file in 512-word blocks.

Figure 2-1. File Index Table Format for Nontape Files (Sheet 6 of 8)

Word	Field	Description
	bilb	Bits used in last byte: 0 All of last byte is used. 1-7 Only from 1 to 7 bits of the last byte are used. Last byte is the NBWth byte.
C	seglen1	For files created on a system before VSOS 2.2, this field contains the length in blocks of the first segment.
	segadr1	For files created on a system before VSOS 2.2, this field contains the starting sector disk address for the first segment.
	sfo	File organization: 0 Sequential. 1 Direct.
D	dfso	For files created on a VSOS 2.2 or later systems, this field contains the ordinal of the first DFS entry associated with this file.
	extsize	Extension size, in 512-word blocks. This is the last extension size used when the file was extended; or, the user-specified allocation unit value adjusted to the next multiple of DAUs. This value is used to compute the next extension size.
	seglen2	For files created on a system before VSOS 2.2, this field contains the length, in blocks, of the second segment.
	segadr2	For files created on a system before VSOS 2.2, this field contains the starting sector disk address for the second segment.
	bt	Blocking type: 0 SIL assumes the file was created before SIL was added to the system. Therefore, it enters default values in the SIL fields of the file index entry. 1 C-type blocking.
	mnr	Minimum blocking length; 24-bit length, in number of bytes.
E	seglen3	For files created on a system before VSOS 2.2, this field contains the length, in blocks, of the third segment.
	segadr3	For files created on a system before VSOS 2.2, this field contains the starting sector disk address for the third segment.
	fiidn	FILEI jdn.
	rt	Record type: 0 Control word (W). 1 ANSI fixed length (F). 2 Record mark (R). 7 Undefined (U).

Figure 2-1. File Index Table Format for Nontape Files (Sheet 7 of 8)

Word	Field	Description
	mxr	Maximum record length; 24-bit maximum length, in number of bytes.
F	hbw	For files created on a VSOS 2.2 or later systems, this field contains a count of the total number of 8-bit bytes written to the file.
	seglen4	For files created on a system before VSOS 2.2, this field contains the length, in blocks, of the fourth segment.
	segadr4	For files created on a system before VSOS 2.2, this field contains the starting sector disk address for the fourth segment.
	vri	Variable rate index (VRI) transferred to the descriptor block to be used for variable rate accounting (VRA). Valid for virtual code files only.
	rmk	Record mark; 8-bit ASCII character (any character is valid).
	pc	Padding character; 8-bit ASCII character (any character is valid).

Figure 2-1. File Index Table Format for Nontape Files (Sheet 8 of 8)

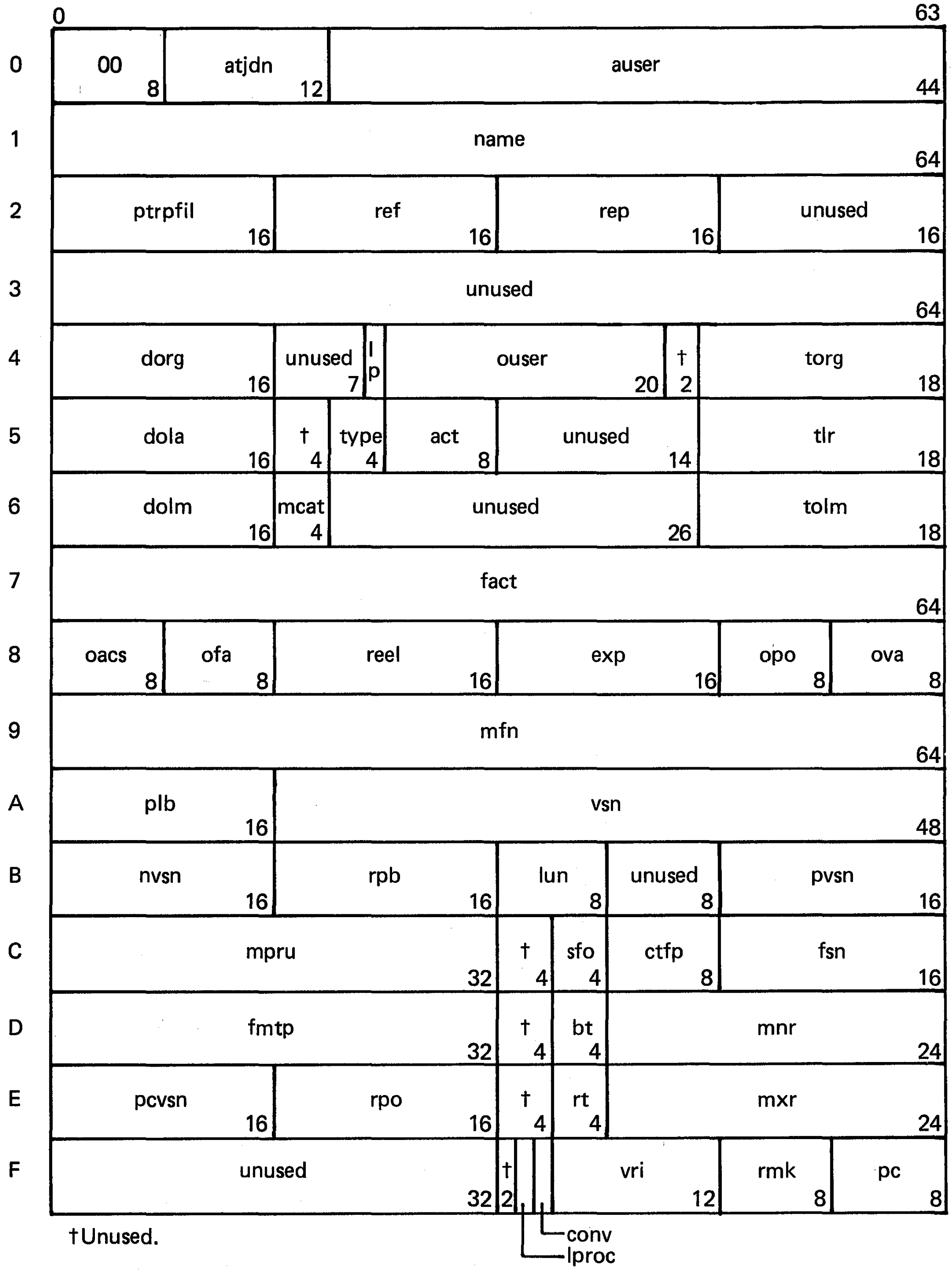
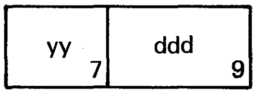


Figure 2-2. File Index Table Format for Tape Files (Sheet 1 of 6)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>0</td><td>atjdn</td><td>Job descriptor number to which this file is attached, assigned to the job for the life of the job in the system.</td></tr>
<tr><td></td><td>auser</td><td>Owning user number, in binary notation.</td></tr>
<tr><td>1</td><td>name</td><td>File name used to assign the tapes.</td></tr>
<tr><td>2</td><td>ptrpfil</td><td>Pointer into the proper block of the PFI for this entry relative to the first block of the PFI. PFI=#FFFF for local files.</td></tr>
<tr><td></td><td>ref</td><td>Number of times this file has been opened.</td></tr>
<tr><td></td><td>rep</td><td>Retention period, in days.</td></tr>
<tr><td>4</td><td>dorg</td><td>Date this file was organized, in the format:


<table><tr><th>Subfield</th><th>Description</th></tr><tr><td>yy</td><td>Last two digits of the year.</td></tr><tr><td>ddd</td><td>Number of days since the beginning of the year (1 through 366).</td></tr></table></td></tr>
<tr><td></td><td>lp</td><td>Local/permanent flag:
0 Permanent.
1 Local.</td></tr>
<tr><td></td><td>ouser</td><td>Originating user number.</td></tr>
<tr><td></td><td>torg</td><td>System clock time, in seconds since midnight, at which this file was originated.</td></tr>
<tr><td>5</td><td>dola</td><td>Date of the last access to this file, in the same format as the dorg field. (To access a file means to open it.)</td></tr>
<tr><td></td><td>type</td><td>File type:
0 Physical file.
2 Virtual code file.</td></tr>
<tr><td></td><td>act</td><td>Number of active I/O connectors for this file.</td></tr>
<tr><td></td><td>tlr</td><td>System clock time, in seconds since midnight, at which the file was last opened.</td></tr>
<tr><td>6</td><td>dolm</td><td>Date the file was last opened with write access, in the same format as the dorg field.</td></tr>
</table>

Figure 2-2. File Index Table Format for Tape Files (Sheet 2 of 6)

Word	Field	Description
6	mcat	Present file management category: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as write-temporary file). 4 Magnetic tape file. 5 User-generated drop file. 6 System-generated drop file. 7 Batch file. 8 Link file. 9 Connected file. #A Checkpointed output file. #F Checkpointed input file.
	tolm	Time, in seconds since midnight, when this file was opened with write access.
7	fact	Account number of the file in ASCII, left-justified with blank fill.
8	oacs	Owner´s access permission for private files and pool boss´ access permission for pool files. For public files, oacs is set equal to gacs: B1 B2 B3 B4 B5 B6 B7 B8 Bit — Access Permitted 1-3 Reserved (ignored by the system). 4 Execute. 5 Modify. 6 Append. 7 Read. 8 Write.
	ofa	Original file accessibility character.
	reel	Reel number of the current volume.
	exp	Julian expiration date obtained from the first header (HDR1) label.

Figure 2-2. File Index Table Format for Tape Files (Sheet 3 of 6)

Word	Field	Description
8	opo	Open processing options: B1 B2 B3 B4 B5 B6 B7 B8 Bit — Processing Option 1 — End-of-processing option. If 0, the system automatically switches volumes. If set, control is returned to the user at end of tape. 2 — Unused. 3 — User error processing. If 0, tape I/O errors are returned to the operator. If set, control is returned to the user if a tape I/O error occurs. 4-8 — Unused.
	ova	Original volume accessibility character.
9	mfn	Multifile set name.
A	plb	Index to the label buffer in a system table.
	vsn	Volume serial number (VSN) for the currently assigned volume.
B	nvsn	Number of VSNs assigned to this tape file.
	rpb	Records per block.
	lun	Logical unit number; the ordinal in the tapes table, or 0 if the unit is not assigned.
	pvsn	Index to the VSN list in a system table.
C	mpru	Maximum length of the physical record unit (PRU).
	sfo	File organization: 0 Sequential.
	ctfp	Current file position. Refer to the tapes table.
	fsn	File sequence number.
D	fmtp	Format parameter.
	bt	Blocking type: 0 SIL assumes the file was created before SIL was added to the system; therefore, it enters default values in the SIL fields of the file index entry. 2 Character type blocking (C).

Figure 2-2. File Index Table Format for Tape Files (Sheet 4 of 6)

Word	Field	Description
D	mnr	Minimum record length; 24-bit length in number of bytes.
E	pcvsn	Index to the current place in the VSN list in MFPP.
	rpo	Request processing options:

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16

Bit	Processing Option
1-10	Unused.
11	Error retry parameter. This field only applies when reading the tape: 0 Standard error recovery processing takes place when a hardware read/write error occurs. 1 Error inhibit; all hardware read/write errors are ignored and processing continues.
12	Unused.
13	Read unconditional processing option: 0 The user is not allowed to read past the end of information or the end of tape. 1 The user is allowed to read past the end of information or the end of tape. This could cause the tape to go off the reel.
14	Tape unload processing option (inhibit unload): 0 When the tape is released, the tape is rewound to the load point and unloaded from the drive. 1 When the tape is released, the tape is rewound to the load point, but it is not unloaded from the drive.
15-16	Unused.

Word	Field	Description
	rt	Record type: 0 Control word (W). 1 ANSI fixed length (F). 2 Record mark (R). 7 Undefined (U).
	mxr	Maximum record length; 24-bit maximum length in number of bytes.

Figure 2-2. File Index Table Format for Tape Files (Sheet 5 of 6)

Word	Field	Description
F	lproc	Label processing option: 0 Read labels. 1 Write labels.
	conv	Data conversion option: 0 There is no data conversion. 1 Convert data.
	vri	Variable rate index transferred to the descriptor block to be used for variable rate accounting. Valid for virtual files only.
	rmk	Record mark; 8-bit ASCII character (any character is valid).
	pc	Padding character; 8-bit ASCII character (any character is valid).

Figure 2-2. File Index Table Format for Tape Files (Sheet 6 of 6)

Table 2-1. File Disposition Specifications in the File Index Table

Destination	Value	Mnemonic	Disposition
Any	0	None	No disposition.
	1	SC	Scratch.
	#10	PU	Punch.
	#20	PR	Any available line printer.

Table 2-2. File Characteristic Specifications in the File Index Table

Type	Value	Mnemonic	Format
Internal characteristics (fiic field in word 9)	0	None	Use internal characteristics default (currently PA).
	1	PA	Eight-bit ASCII. If the fidc field indicates a print file, the file has free-form carriage control.
	2	BI	Binary.
	3	AS	Eight-bit ASCII. If the fidc field indicates a print file, the file has ANSI-defined carriage control.
External characteristics for punch files (fiec field in word 9)	0	None	Default 29.
	1	29	029 keypunch.
	2	26	026 keypunch.
	3	80	80-column binary.

MINUS PAGE

VSOS recognizes two types of files: virtual files, usually containing executable code, and physical files, always containing nonexecutable data only. A virtual file is prefaced by at least one 512-word block containing program execution and data access information to be used by the operating system. This preface is known as the minus page and is created by the operating system at load time. Physical files do not have minus pages.

The minus page of a virtual code file is created at load time to pass information such as the entry point address, the length of the drop file, the code origin, data base locations, and so forth. The operating system needs this information for starting program execution. Once execution begins, the operating system uses the minus page to store the invisible package, time-slicing data, I/O connection blocks to high-speed devices, maps of defined virtual space, time-sharing data, and statistics relating to resource usage. If the program execution terminates abnormally, the minus page is stored on the drop file and can be used for debugging purposes. The original minus page remains on the virtual code file as initialized at load time. In addition, drop files may contain a second minus page (immediately following the first), which is a logical extension of the first minus page.

An executing controllee cannot access its minus page except via system messages. A controllee can execute the SIL call Q5GETMPG to copy it´s minus page. Otherwise, the minus page is like any other part of a file, and can be accessed explicitly or implicitly.

The minus page has the format shown in table 2-3. Individual words are described in the table, with the contents of the invisible package shown in appendix E.

The working set pager information buffer in the minus page is used by PAGER to store information about a task´s working set. The buffer occupies words #29 through #31.

The leftmost 16 bits of word #88 or #136 of the first minus page contains the second minus page pointer. If this field contains a 0 or #FFFF, there is no second minus page. Otherwise, this field contains the physical page address of the second minus page.

Table 2-3. Minus Page Format (Sheet 1 of 2)

Words		Contents
Decimal	Hexadecimal †	
0 - 39	0 - 27	Executing program invisible package.
40	28	Unused.
41 - 49	29 - 31	Working set pager information buffer.
50 - 52	32 - 34	Program restart temporary buffers.
53, 54	35, 36	Time information required by the operating system for alternator and message management.
55, 56	37, 38	Same as words #32 to #34.
57, 58	39, 3A	Used by application accounting.
59 (bits 0 - 15)	3B	Error code saved during abnormal termination control processing.
59 (bits 16 - 23)	3B	Device number of the device causing the fatal PAGER I/O error.
59 (bits 24 - 31)	3B	Pack number of the device causing fatal PAGER I/O error.
59 (bits 32 - 63)	3B	Unused.
60	3C	Buffer flushing, ATC process, drop file and reload status information.
61	3D	Database address for buffer flushing.
64 - 123	40 - 7B (2 - D9)	I/O connectors for user disk or tape files.
124 - 127	7C - 7F	If controllee is dynamic, this is the I/O connector for the SHRLIB (#F).
128 - 131	80 - 83	I/O connector for the source file (#10).
132 - 135	84 - 87	I/O connector for the drop file (#11).

† Words shown in parentheses are second minus page values.

Table 2-3. Minus Page Format (Sheet 2 of 2)

Words		Contents
Decimal	Hexadecimal†	
136 (bits 0 - 15)	88	Second minus page pointer.
136 (bits 16 - 23)	88	Reserved.
136 (bits 24 - 31)	88	Unused.
136 (bits 32 - 63)	88	Directory for bound explicit map entries.
137 (bits 0 - 31)	89	Unused.
137 (bits 32 - 63)	89	Directory for bound implicit map entries.
138 (bits 0 - 15)	8A	Third minus page pointer.††
138 (bits 16 - 27)	8A	Unused.
138 (bits 28 - 63)	8A	Directory for drop file map entries.
139	8B	System error code.
140 - 150	8C - 96	Time usage and accounting entries.
151 - 152	97 - 98	Q5TERM information for buffer flushing.
153	99	Drop file size.
154 - 156	9A - 9C	Unused.
157 - 159	9D - 9F	Reserved for installation.
160 - 175	A0 - AF (DA - 10F)	Bound explicit maps (of file opened for explicit I/O).
176 - 255	B0 - FF (110 - 1FF)	Bound implicit maps (of file opened for implicit I/O).
256 - 511	100 - 1FF	Drop file map.

†Words shown in parentheses are second minus page values.

††The format of the third minus page is:
Word 0 to 340 contains drop file map full-word entries.
Word 341 to 511 contains drop file map half-word entries.

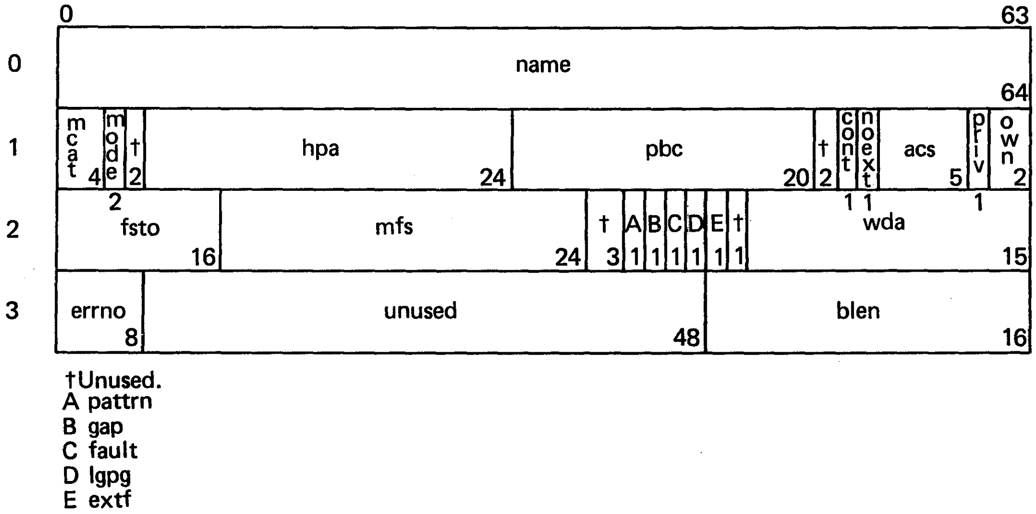
I/O CONNECTORS

Words #40 through #87 of the first minus page contain the first 18 I/O connectors. Words #2 through #D9 of the second minus page contain the remaining 54 I/O connectors. An I/O connector (IOC) is a four-word block used to establish a link between the program and an I/O device. The operating system also uses I/O connectors to keep track of the activity of a specific file and a program's use of that file. Each time a program issues an OPEN FILE request, an I/O connector is created and initialized by the system with information provided in the request and in the file index table.

Each program can have up to 70 connectors for user files, numbered 0 through #F and #12 through #47. The I/O connector for the program's source file is numbered #10, and the I/O connector for the program's drop file is #11. The I/O connector for the system shared library file, if used, is #0F. I/O connectors numbered 0 through #F and #12 through #47 can be allocated by the user or automatically allocated by the system.

Formats of the I/O connectors are illustrated in figures 2-3, 2-4, and 2-5. The connector for a mass storage file opened for explicit input and output is shown in figure 2-3; the connector for a mass storage file opened for implicit input and output is shown in figure 2-4. The connector for a tape file is shown in figure 2-5.

In figure 2-4, when the name field contains a drop file name, the fourth word of the I/O connector serves the same purpose as the second word of a bound implicit map entry for a source file.



Word	Field	Description
0	name	Name of the file in ASCII. File names must be in the format described in chapter 3, File Concepts.
1	mcat	File management category: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as write-temporary file). 4 Magnetic tape file. 5 User-generated drop file. 6 System-generated drop file. 7 Batch file.

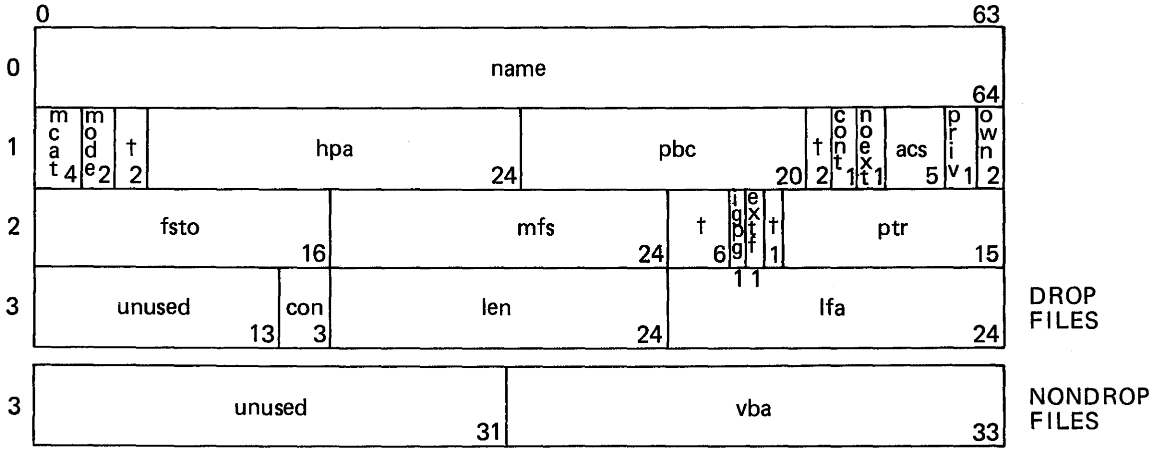
Figure 2-3. Format of I/O Connector for a Mass Storage File Opened for Explicit I/O (Sheet 1 of 3)

Word	Field	Description
1	mode	Mode of input/output: 0 Open for explicit I/O. 1 Open for implicit I/O.
	hpa	Page address of the highest page accessed.
	pbc	Page byte count; number of bytes written to user´s dayfile.
	cont	Contiguous flag. Set if file is contiguous.
	noext	No extension flag. Set if file is not extendable.
	acs	File access permissions. This 5-bit field is treated as five 1-bit fields with each bit specifying the associated permission: Bit / Hex. Value / Description x 10 Execute access permitted. m 8 Modify access permitted. a 4 Append access permitted. r 2 Read access permitted. w 1 Write access permitted.
	priv	Privileged open designator: 0 Regular open. 1 Privileged open.
	own	File ownership (refer to File Concepts, chapter 3). The values are: 0 Private. 1 Public. 2 Pool.
2	fsto	File segment table ordinal.
	mfs	Minimum file size to which this file needs to be extended. Set by XIOCALL when the extf flag is set.
	pattrn	Set if need to fault in the unused portion of the I/O buffer for patterning.
	gap	Set if GAP patterning needs to be done.
	fault	Flag bit. Set if FC=#f500 call will be reissued after PAGER I/O is complete.
	lgpg	Large page flag bit. Set only if the file is to be extended with large pages.

Figure 2-3. Format of I/O Connector for a Mass Storage File Opened for Explicit I/O (Sheet 2 of 3)

Word	Field	Description
2	extf	Extend flag bit. Set if the file needs to be extended.
	ptr	For a privileged open file, the owner´s user table (UT) entry or the pool list table (PLIST) entry number.
3	errno	Error number returned from EXTENDF call.
	blen	Length of I/O buffer.

Figure 2-3. Format of I/O Connector for a Mass Storage File Opened for Explicit I/O (Sheet 3 of 3)



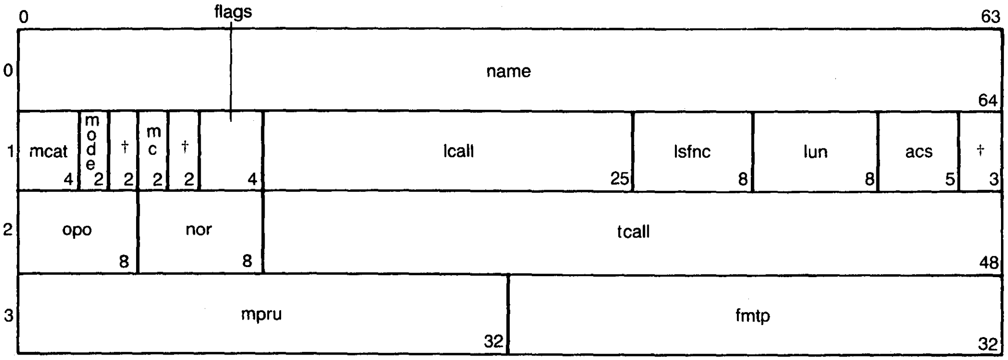
†Unused.

Word	Field	Description
0	name	Name of the file, in ASCII. File names must be in the format described in File Concepts (chapter 3).
1	mcat	File management category: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as write-temporary file). 4 Magnetic tape file. 5 User-generated drop file. 6 System-generated drop file. 7 Batch file.
	mode	Mode of input/output: 0 Open for explicit I/O. 1 Open for implicit I/O.
	hpa	Page address of the highest page accessed.
	pbc	Page byte count; number of bytes written to user´s dayfile.
	cont	Contiguous flag. Set if file is contiguous.
	noext	No extension flag. Set if file is not extendable.
	acs	File access permissions. This 5-bit field is treated as five 1-bit fields with each bit specifying the associated permission: Bit / Hex. Value / Description 1 10 Execute access permitted. 2 8 Modify access permitted. 3 4 Append access permitted. 4 2 Read access permitted. 5 1 Write access permitted.

Figure 2-4. Format of I/O Connector for a Mass Storage File Opened for Implicit I/O (Sheet 1 of 2)

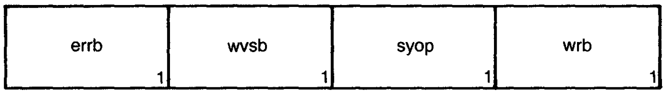
<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>1</td><td>priv</td><td>Privileged open designator:
0 Regular open.
1 Privileged open.</td></tr>
<tr><td></td><td>own</td><td>File ownership (refer to File Concepts, chapter 3). The values are:
0 Private.
1 Public.
2 Pool.</td></tr>
<tr><td>2</td><td>fsto</td><td>File segment table ordinal.</td></tr>
<tr><td></td><td>mfs</td><td>Minimum file size to which this file needs to be extended. This field is set by PAGER when the extf flag is set.</td></tr>
<tr><td></td><td>lgpg</td><td>Large page flag bit; set only if the file is to be extended with large pages.</td></tr>
<tr><td></td><td>extf</td><td>Set to 1 by the operating system if the file needs to be extended.</td></tr>
<tr><td></td><td>ptr</td><td>For a privileged open file, the owner's user table (UT) entry or the pool list table (PLIST) entry number.</td></tr>
<tr><td>3 (drop files)</td><td>con</td><td>For a drop file, a control field with the following format:
<table><tr><td>c1</td><td>c2</td><td>c3</td></tr></table>
<table>
<tr><th>Subfield</th><th>Description</th></tr>
<tr><td>c1=1</td><td>Write access is permitted.</td></tr>
<tr><td>c2=1</td><td>Read access is permitted.</td></tr>
<tr><td>c3=0</td><td>File is contained on small pages.</td></tr>
<tr><td>c3=1</td><td>File is contained on large pages.</td></tr>
</table>
Otherwise, this field is 0.</td></tr>
<tr><td></td><td>len</td><td>For a drop file, the length of the file, in blocks. Otherwise, this field is 0.</td></tr>
<tr><td></td><td>lfa</td><td>For a drop file, the logical mass storage sector address of this file's first page. Otherwise, this field is 0.</td></tr>
<tr><td>3 (non-drop files)</td><td>vba</td><td>Virtual block address of the start of the file. Zero if the file is not mapped in from the start of the file.</td></tr>
</table>

Figure 2-4. Format of I/O Connector for a Mass Storage File Opened for Implicit I/O (Sheet 2 of 2)



† Unused.

Word	Field	Description
0	name	Name of the file in ASCII. File names must be in the format described in File Concepts, Chapter 3.
1	mcat	File management category: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as write-temporary file). 4 Magnetic tape file. 5 User-generated drop file. 6 System-generated drop file. 7 Batch file.
	mode	Mode of input and output: 0 Open for explicit I/O. 1 Open for implicit I/O.
	mc	Reserved.
	flags	Status flags:



Bit	Name	Description
1	errb	An error was returned for at least one call.
2	wvsb	Set by RESIDENT if all I/O has completed, but virtual system completion routine has not run.
3	syop	Set if the system is processing OPEN.
4	wrb	Set if a write was issued for this file.

Figure 2-5. Format of I/O Connector for a Tape File (Sheet 1 of 2)

Word	Field	Description
1	lcall	Physical word address of the last outstanding request.
	lsfnc	Last subfunction issued for this tape file.
	lun	Logical unit number. This field contains the tapes table ordinal.
	acs	Access permission: 1 Write access only is permitted. 2 Read access only is permitted. 3 Read and write access are permitted.
2	opo	Open processing options. Refer to the OPEN FILE system message.
	nor	Number of outstanding requests.
	tcall	Virtual bit address of the first outstanding request or top call.
3	mpru	Maximum PRU size.
	fmtp	Format parameters as defined in the TAPE MANAGEMENT system message.

Figure 2-5. Format of I/O Connector for a Tape File (Sheet 2 of 2)

MAP DIRECTORIES

Words #88 through #8A of the first minus page contain map directories. Each map directory contains information relating to the location and length of its associated file map. Each directory occupies the second half-word of its location in the minus page. For the first minus page, the bound explicit map directory is at word #88; the bound implicit map directory is at word #89; and the drop file map directory is at word #8A. Each directory is formatted as shown in figure 2-6. For the second minus page, the bound explicit map directory is at word 0, and the bound implicit map directory is at word 1.

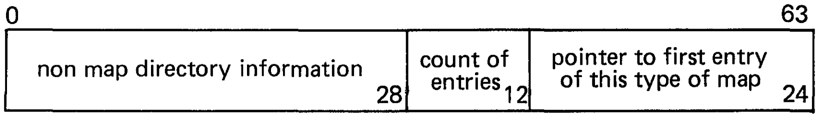


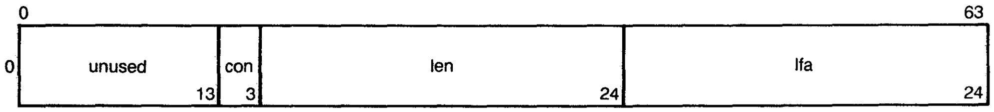
Figure 2-6. Map Directory Format

MINUS PAGE FILE MAPS

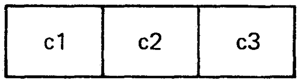
The file maps in the minus pages associate files with physical mass storage areas. For files opened for implicit I/O, the maps associate physical mass storage areas with virtual address areas. Each time a program opens a file for explicit I/O, one entry is made in a bound explicit map. The MAP message (f=#0004) places entries in a bound implicit map.

Bound Explicit Maps

Words #A0 through #AF of the first minus page and #DA through #10F in the second minus page contain bound explicit maps. These maps are related to files opened for explicit I/O (mode=0). Each file that has been opened for explicit I/O corresponds to one map entry; the files are identified by their I/O connector numbers. The format of the bound explicit map entry is shown in figure 2-6.1.



Word	Field	Description
0	con	Flag bits, in the following format:



Subfield	Description
c1=1	Write access is permitted.
c2=1	Read access is permitted.
c3=0	File consists of small blocks.
c3=1	File consists of large blocks.

Otherwise, the field is 0.

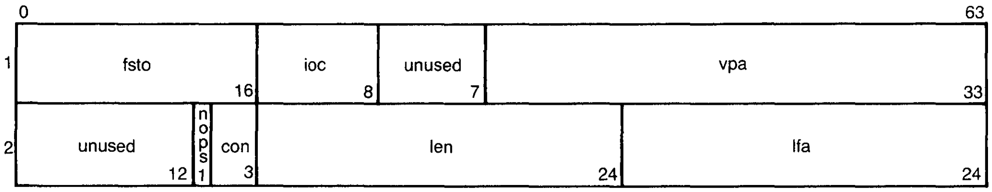
	len	Length of the file, in blocks.
	lfa	Logical file address of the first block of this segment of virtual space.

Figure 2-6.1. Bound Explicit Map Entry Format

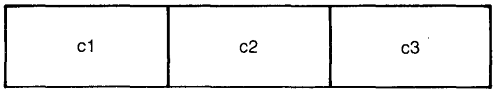
Bound Implicit Maps

Words #B0 through #FF of the first minus page and #110 through #1FF in the second minus page contain bound implicit maps. These maps are related to files opened for implicit I/O (mode=1); such files can consist of discontinuous virtual address ranges. Up to 160 virtual address space segments can be mapped simultaneously. All the segments can be associated with one I/O connector, or each segment can be so associated. The format of a bound implicit map entry is shown in figure 2-7.

In bound implicit map entries, all first words are in the first half of the map space, and all second words are in the second half. Entries are sorted by ascending virtual page address; blank entries are squeezed out. Observe that both minus pages have a first half of the map space so that no map entry is split between the two minus pages.



Word	Field	Description
1	fsto	File segment table ordinal.
	ioc	I/O connector number associated with this segment.
	vpa	Virtual page address of the first block of this segment.
2	nops	Flag bit indicating there is not yet physical space associated with this map entry.
	con	Flag bits in the following format:



Subfield	Description
C1=1	Write access permitted.
C2=1	Read access permitted.
C3=1	File consists of large pages.

	len	Length of this segment in blocks.
	lfa	Logical file address of the first block of this segment.

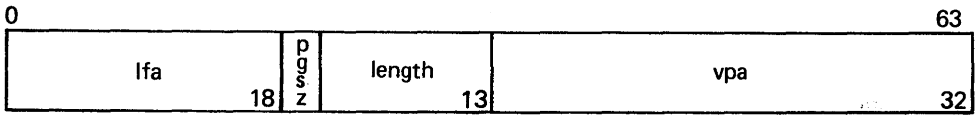
Figure 2-7. Bound Implicit Map Entry Format

DROP FILE MAP

Half of the first minus page and the whole third minus page are a drop file map. In the case of a free-space attachment to a file, the defined space is allocated a part of the drop file on which it can reside if it becomes necessary to swap the attachment out. Free-space attachments are cataloged in the drop file map in much the same way that other kinds of virtual space are cataloged in the bound implicit map.

Each drop file map entry consists of one full word and one half word. Up to 511 entries can be made in the drop file map, and each entry can have up to 32 associated pages. This allows for up to 511 noncontiguous address spaces to be part of the drop file.

The first and third minus page have 170 and 341 full word map entries respectively. The format of these entries is shown in figure 2-8. The 170 half-word entries that follow the full-word entries correspond as shown in figure 2-9. Each half-word entry consists of 32 bits, 1 bit per page. If the bit is 0, the page is either undefined or exists in main memory or on the paging device; if the bit is 1, the page has been written to mass storage on the drop file. Bit 0 or 32 corresponds to page 1 of a segment; bit 31 or 63 corresponds to page 32 of a segment.



Word	Field	Description
0	lfa	Logical file address of the first block of this segment of virtual space.
	pgsz	Size of the pages in this segment of virtual space: 0 Small pages. 1 Large pages.
	length	Length of this segment of virtual space in blocks.
	vpa	Virtual block address of the first page in this segment of virtual space.

Figure 2-8. Drop File Map Full-Word Entry Format

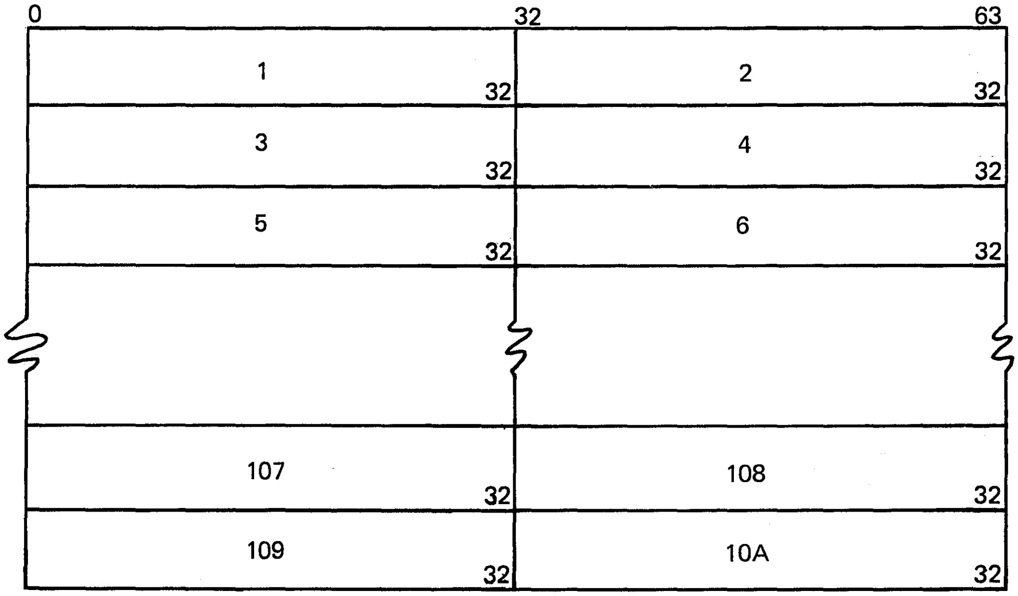


Figure 2-9. Drop File Map Half-Word Entry Format

TAPES TABLE

The tapes table holds pertinent information about the tape units and the volumes which are in use on them. There is one entry per unit. The tapes table entry is returned to the user in the TAPE FUNCTION system message after the completion of a request. The tapes table entry is also returned to the user in the OPEN, CLOSE, and TAPE SWITCH VOLUME system messages if the user supplied a buffer. Refer to chapter 4 for more detailed information on use of this table.

The format of the tapes table is shown in figure 2-10.

	0							63
1	† 4	uato 8	† 4	db 8	ioc 8	rpo 16	jdt 8	lbsn 8
2	pzip 8	bzip 8	tad1 8	tad2 8	pu 4	dev 12	cflgs 8	stun 8
3	lfn 64							
4	fsn 16	sn 16	fwe 8	fre 8	pepr 8	wcr 4	rcr 4	
5	time 64							
6	reel 16	vsn 48						
7	bid1 16	bid2 16	bid3 16	bid4 16				
8	bid5 16	bid6 16	bid7 16	bid8 16				
9	bid9 16	bid10 16	bid11 16	bid12 16				
A	fc 8	abc 24	ctfp 8	cbc 24				
B	twre 16	trre 16	stce 16	dtce 16				
C	elflg 8	pruct 24	tflgs 16	† 4	cml 4	denl 4	† 3	rdyb

†Unused.

Word	Field	Description
1	uato	User activity table ordinal.
	db	Descriptor block number; nonzero if opened.
	ioc	Input/output connector.

Figure 2-10. Tapes Table Format (Sheet 1 of 5)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>1</td><td>rpo</td><td>Request processing options:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>32-41</td><td>Unused.</td></tr>
<tr><td>42</td><td>Error retry parameter. This field only applies when reading the tape:
0 Standard error recovery processing takes place when a hardware read/write error occurs.
1 Error inhibit; all hardware read/write errors are ignored and processing continues.</td></tr>
<tr><td>43</td><td>Unused.</td></tr>
<tr><td>44</td><td>Read unconditional processing option:
0 The user is not allowed to read past the end of information or the end of tape.
1 The user is allowed to read past the end of information or the end of tape. This could cause the tape to go off the reel.</td></tr>
<tr><td>45</td><td>Tape unload processing option (inhibit unload):
0 When the tape is released, it is rewound to the load point and unloaded from the drive.
1 When the tape is released, it is rewound to the load point, but it is not unloaded from the drive.</td></tr>
<tr><td>46-47</td><td>Unused.</td></tr>
</table></td></tr>
<tr><td></td><td>jdt</td><td>Job descriptor table ordinal.</td></tr>
<tr><td></td><td>lbsn</td><td>Last boat sequence number; each boat is assigned a sequence number. lbsn is the sequence number for the last request on this unit.</td></tr>
<tr><td>2</td><td>pzip</td><td>Primary zip for this unit.</td></tr>
<tr><td></td><td>bzip</td><td>Backup zip for this unit.</td></tr>
<tr><td></td><td>tad1</td><td>First tape access driver NAD number.</td></tr>
<tr><td></td><td>tad2</td><td>Second tape access driver NAD number; 0 if single access.</td></tr>
<tr><td></td><td>pu</td><td>Physical unit (0 through #F).</td></tr>
<tr><td></td><td>dev</td><td>Device unit number (#100 through #1FF).</td></tr>
</table>

Figure 2-10. Tapes Table Format (Sheet 2 of 5)

Word	Field	Description
2	cflgs	Central flags:

Bit	Name	Description
48	PDWNB	Status of primary inboard NAD.
49-50	BDWNB	Status of backup inboard NAD.
51	T1DB	Status of tape access NAD 1.
52	T2DB	Status of tape access NAD 2.
53-54		Unused.
55	SACB	Single access bit.
56		Unused.

Word	Field	Description
	stun	Status of the unit:

Bit	Name	Description
56	RERB	Resident detected error.
57-59	STRES	Free.
60-62	ASNB	Assign bit.
61	ROB	Read-only bit.
62	OWNB	Down bit.
63	OFFB	Off bit.

Word	Field	Description
3	lfn	Logical file name; set if unit is assigned.
4	fsn	Current file sequence number for an ANSI labeled tape; 0 for unlabeled/nonstandard tape.
	sn	Current file chapter number for an ANSI labeled tape; 0 for unlabeled/nonstandard tape.
	fwe	Fatal write errors.
	fre	Fatal read errors.
	pepr	Positioning errors per reel.
	wcr	Number of consecutive reels in which write recoverable errors exceeded the threshold.
	rcr	Number of consecutive reels in which read recoverable errors exceed the threshold.
5	time	Length of time the unit was assigned, in microseconds.
6	reel	Reel number of the current volume.
	vsn	Volume serial number of the currently assigned reel.
7, 8, 9	bid1 through bid12	The block identifier of the eleventh through the last good PRU on tape.

Figure 2-10. Tapes Table Format (Sheet 3 of 5)

Word	Field	Description
A	fc	Failure code. This field is set when a unit exceeds an error threshold or because the unit got a nonfatal or fatal marginal drive indicator (MDI). The system automatically degrades unit status at unload time if fc is set. fc is cleared when the unit is brought up: 1 Nonfatal MDI (degraded to read only). 2 Erase/write errors exceeded the threshold on consecutive reels (degraded to read only). Write threshold and consecutive reel count are installation parameters. 3 Erase/read errors exceeded the threshold on consecutive reels (degraded to down). Read threshold and consecutive reel count are installation parameters. 4 Positioning errors exceed the threshold on one reel (degraded to down). Positioning threshold is an installation parameter.
	abc	Absolute physical record unit (PRU) count including tape marks from the beginning of the volume. abc is a count of the number of interblock gaps encountered on the tape. If abc=0, the tape is at load point.
	ctfp	Current tape file position flags. If ctfp=0, the tape is positioned in the middle of a logical record unit (LRU). The only legitimate combination of bits is end of group (EOG) and end of information (EOI). B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 **Bit** — **Tape File Position** B1-B4 Unused. B5 Beginning of information. B6 End of LRU. B7 End of group. B8 End of information.
	cbc	Current PRU count from the beginning of information. This PRU count does not include label PRUs if the tape is labeled.
B	twre	Total accumulation of recoverable write errors in the use of this volume. This count is put in the dayfile at unload time and cleared at the next reel mount time.
	trre	Total accumulation of recoverable read errors in the user of this volume. This count is put in the dayfile at unload time and cleared at the next reel mount time.
	stce	Total accumulation of single-track, hardware-corrected errors. This count is put in the dayfile at unload time and cleared at the next reel mount time.
	dtce	Total accumulation of double-track, hardware-corrected errors. This count is put in the dayfile at unload time and cleared at the next reel mount time.

Figure 2-10. Tapes Table Format (Sheet 4 of 5)

Word	Field	Description
C	elflg	Error log flags.
	pruct	PRU count.
	tflgs	Tape flags.
	cml	Conversion code for the label: 0 Unknown. 1 ASCII. 2 EBCDIC.
	denl	Density of the label: 1 6250 bpi. 2 1600 bpi.
	rdyb	Ready bit. This field is set/cleared by the scan when the tape is ready/not ready.

Figure 2-10. Tapes Table Format (Sheet 5 of 5)

FILE CONCEPTS 3

The file concepts used in VSOS are described in volume 1 of this reference manual. This chapter describes additional file specifications used in the system messages described in chapter 5.

FILE NAMES

A file has only one name associated with it; it is both its permanent file name and its logical file name. The file name can be one to eight letters and digits long, and must be left-justified and blank-filled within the field. User-created file names cannot contain special characters and must begin with a letter. System-created file names can contain any character; system-created drop files must begin with one digit. The conventions used for naming drop files are described in volume 1 of this reference manual.

FILE OWNERSHIP

Privileged users have ownership rights over all files except local files. A nonprivileged user has ownership rights determined by the values in file index table fields as shown in table 3-1. The file index table is described in chapter 2. The GIVE FILE system message can change file ownership.

The file owner specifies the file attributes. To permanently change file attributes, the appropriate fields in the file index table must be changed (refer to the CHANGE FILE ATTRIBUTES system message in chapter 5).

The three file ownership categories are private, pool, and public. Each private file cataloged in the file index table belongs to a particular user number and account identifier. When a private file is given by one user to another, the user number associated with the file changes immediately. However, the account identifier does not change until the new owner references the file. The system accounting tables indicate the total amount of time that the original account owned the file.

Calls to the POOL FILE MANAGER system message perform the same functions as the pool file utilities and SIL calls described in volume 1. The GIVE FILE system message can give files to a pool; the DESTROY FILE system message can destroy pool files.

Public files are owned by user number 000000, signifying system ownership. The list of public files is controlled by the installation administrator or by privileged tasks.

Table 3-1. File Index Table Fields that Affect File Ownership

File Category	File Index Table Field	Nonprivileged User	Privileged User
Public	buser = 0	All users	All users
Pool	POOLNAME = poolname	Pool boss	Users according to pool access list of user numbers
Private	gacs = 0 and no access directory	Originating user†	Originating user†
	gacs ≠ 0	Originating user†	All users
	Access directory exists	Originating user†	Users in access directory and originating user

†Originating user number determined by the user field of the file index table.

FILE ACCESS

File access is controlled by the file ownership category, file access permission fields, and security level in the file index table entry. All file references require the task to be at an equal or higher security level than the file which the user is trying to access. Assuming this requirement is met, all users can access public files; users given access to a pool can access files in the pool; the file owner can access private files. The file owner can also give other users access to a private file. Privileged users can access any permanent file on the system, regardless of the access permissions.

Read, write, append, modify, and execute access are controlled by the permission parameter, which is set either when the file is created or by subsequent permission calls. A file with write access can be written into by a user program or by the operating system.

An attempt to write into a read-only file produces a fatal error. The only exception is that the OPEN FILE message (f=#0003) can be used to indicate that during subsequent execution in the task, pages of read-only files can be mapped with MODDROP (write-temporary) access.

When the user opens a file, he requests read, write, modify, execute, or modify access. The system checks if the requested access is allowed for the file by checking the appropriate access permissions. For example, if write permission has not been specified for the file, write access cannot be granted to the file. If no access is explicitly requested, the default access is as many permissions of read and/or execute as the user is granted.

Each private file access defaults to wait for a file, if it is already attached in such a way that file access cannot be shared.

PRODUCTION FILES

A site may use additional security measures which include designating executable files as production or nonproduction files and users as production or nonproduction users. Production users can only execute production files. Refer to the Installation Handbook for further discussion.

FILE MANAGEMENT CATEGORIES

The management category field in the file index is a combination of device type, disposition information, and file origin information. This file index table entry determines the system management of the file. The possible category designations are:

- Mass storage file
- Scratch file
- Output file
- Drop file
- MODDROP file
- File connected to a terminal
- Tape file

MASS STORAGE FILES

The originating user controls the creation and disposition of mass storage files. VSOS protects mass storage files from access or destruction by other nonprivileged users.

SCRATCH FILES

Only a task can create scratch files. Scratch files exist during the originating task's activity. When the task terminates normally, all scratch files are automatically destroyed. When the operating system terminates the task and saves its drop file, scratch files are saved. A CLOSE FILE system message specifying a scratch file destroys the file. All scratch files have read and write access.

Scratch files are a subset of local files. Local files exist for the duration of the user job; scratch files exist only for the duration of the task. Scratch files can be created only on mass storage.

OUTPUT FILES

Output files contain information suitable for processing by an output device, such as a printer, card punch, or microfilm device. Only a user task or utility can create an output file. When one of the following occurs for batch jobs, VSOS gives all output files with valid disposition codes to privileged system tasks for output processing.

- The task terminates normally.
- The task issues a CLOSE FILE (f=#0005) message.
- The task issues a TERMINATE (f=#0006) message.

After output files are processed, they are destroyed.

DROP FILES

VSOS creates a drop file for each task called into execution. If a local file already exists which has the same name as the target file name, the system destroys the existing local file and creates a new drop file. The executing task is called the source file. Its drop file contains modified pages of the source file, free space, and write-temporary files. Modified pages for other files are written directly to the respective file. Drop files may exist on mass storage or tape, but must exist on mass storage when used in the execution of a controllee. Volume 1 gives further description of drop files and their naming conventions.

MODDROP (WRITE-TEMPORARY) FILES

A MODDROP file is a read-only file that has been modified while paged in to central memory. The modified pages cannot be paged back to the read-only file, and so are paged to the drop file. Subsequent references to those pages access the modified version from the drop file. To reference the original read-only version, the modified pages must be removed from the drop file. Only files being used implicitly can be MODDROP files. This form of access is selected when the file is opened.

FILES CONNECTED TO A TERMINAL

Files connected to a terminal are useful for small amounts of interactively entered I/Os. The task may create, open, and destroy such files through virtual system calls, but must perform using SIL subroutines. These files may be used only by a level-2 or lower controllee execution.

TAPE FILES

A tape file is a file that has been stored on tape rather than in mass storage. The system treats tape files as a separate file management category. (Refer to chapter 4, Tape Management.)

FILE I/O

As described in volume 1, VSOS performs two types of input/output, implicit and explicit. The type of input/output is specified in the OPEN call. The type of I/O that may be done to a file is dependent on the device type.

The EXPLICIT I/O (f=#F500) and TAPE FUNCTION (f=#F406) system messages perform explicit I/O. With a single system request, these messages can transfer one or more blocks between the specified buffer and a storage device. The system locks down the buffer in memory while the peripheral request is active; it cannot be paged out while I/O is going on. More system action is required to prepare for explicit than for implicit I/O.

The mass storage EXPLICIT I/O message (f=#F500), a single I/O request, may transfer up to 24 small or large pages.

The TAPE FUNCTION message (f=#F406) performs explicit tape I/O. The buffer cannot span more than 48 small or large pages. Small and large pages cannot both exist in the buffer at the same time.

Implicit I/O is performed only with mass storage files. With implicit I/O, information transfers directly between a storage device and its current location in central memory. The transfer occurs when the user causes an access interrupt by referencing a page of data or code not in memory. If the virtual page has been previously associated with physical space via the MAP function, the system transfers the data between memory and the physical device. If a virtual-to-physical relationship has not been previously established, the system defines the virtual page in free space so that it becomes an extension of program space.

Files connected to terminals do not use explicit or implicit I/O messages. This type of I/O is a SIL feature. SIL translates Q5GETN and Q5PUTN calls into Q5GETMCR and Q5SNDMCR calls and uses the message processing facilities in the system.

VSOS recognizes two types of file addressing, physical and virtual.

PHYSICAL FILES

A physical file is accessed by physical addresses. It is, by definition, a data file. It cannot be executed. File I/O can be implicit or explicit. A physical file never has a minus page.

VIRTUAL FILES

A virtual code file is a controllee file produced by the LOAD utility. Its first block is its minus page.

TAPE MANAGEMENT 4

TAPE ASSIGNMENT

Whenever a tape is mounted, the system checks for labels. If the tape is labeled, the system records the VSN from the VOL1 label in the system tapes table. If a requested VSN matches a VSN in the tapes table, the system automatically assigns the tape to the requesting job. If there is no match, the system suspends the job until the tape with the requested VSN is mounted. If a tape is unlabeled, the operator must type in a VSN for the tape. If the job did not specify a VSN when it requested the tape file, the sytem requests that the operator specify a VSN for the job. When the job and the tape unit have a matching VSN, the tape unit is assigned to the job. Observe that assignment of a NOS tape can occur only if the VSN is six characters long.

RECOVERY

VSOS handles tape recovery for the physical record unit (PRU) and for user errors in the following manner.

PRU RECOVERY

The system PRU recovery of bad reads and writes of tapes is done at the driver level. The advanced tape system (ATS) features such as controlled backspace, selectable clipping levels, and block ID identification are employed. For group-encoded (GCR) tapes, single-track write correction and dual-track read correction are used (single-track write correction can be disabled by the user).

A block ID is a hardware-generated identifier for use in positive positioning of tape during error recovery. There is 1 identifier per PRU and the last 12 identifiers are kept in the tapes table. The absolute block count is the count of PRUs, including tape marks since load point. The current block count is the count of PRUs since the previous label group. The block IDs and block counts are kept current on a volume-by-volume basis and are cleared on a rewind or unload. The block IDs are discarded one PRU at a time for a backspace.

USER ERROR RECOVERY

If the user selects user error processing (UEP) at open time, the system returns control to the user after a tape I/O error (#100 through #1FF) or tape subsystem error (#200 through #2FF) occurs. Observe that PRU recovery at the driver level has not been able to recover this error. The user can choose to skip the failing data.

SYSTEM LABEL PROCESSING

VSOS processes both nonstandard and ANSI standard labeled tapes.

NONSTANDARD LABELS

The system permits the user to process nonstandard labels if the installation parameter IP TPNSL equals 1. The user must request the tape with a label type of nonstandard. Then it is possible for the user to supply labels in the OPEN FILE system message. A nonstandard label consists of 80-character PRUs delimited by tape marks as in ANSI standard label. The only difference in system processing of standard/nonstandard labels is the system omits any verification of fields for nonstandard labels. Also, the system does not position to nonstandard labels. The system does not inhibit ANSI standard labels from being processed as nonstandard if the installation parameter IP_TPNSL equals 1.

ANSI LABELS

ANSI labels conform to the American National Standard Magnetic Tape Labels for Information Interchange X3.27-1978.

VSOS processes labels at level 2. All labels are 80 characters long. The first three characters of an ANSI label identify the label type. The fourth character indicates a number within a label type. Table 4-1 shows a summary of each label type, name, function, and whether or not it is required.

Required Labels

The VOL1, HDR1, and EOF1 labels are required on all ANSI-labeled tapes. In addition, an EOV1 label is required if the physical end-of-tape reflector is encountered before an EOF1 label is written or if a multifile set is continued on another volume. In the descriptions of the contents of these labels, n is any numeric digit and a is any letter, digit, or any of the special characters of the center four columns of the code table in ANSI X3.4-1977 except position 5/15. Refer to appendix A for this code table.

Some fields are optional. An optional field which does not contain the designated information must contain blanks. Fields which are not described as optional are required and written as specified. All n-type fields are right-justified and zero-filled, and a-type fields are left-justified and blank-filled.

For reading labels, nonzero fields in the user HDR1 label buffer are compared with the tape HDR1 label until a match occurs.

For writing labels, the fields in the user label buffer are verified for a-type or n-type as required; however, only the file sequence number field in the label buffer is used to position to the tape HDR1 label.

Table 4-1. Tape Label Format

Label Identifier	Number	Label Group Name	Label Set Name	Required/ Optional
VOL	1	Beginning-of-volume or beginning-of-file chapter	Volume header	Required
UVL	1-9		User volumes	Optional
HDR	1		File header	Required
HDR	2-9		File header	Optional
UHL			User header	Optional
HDR	1	Beginning-of-file	Beginning-of-file	Required
HDR	2-9		Beginning-of-file	Optional
UHL			User header	Optional
EOF	1	End-of-file	End-of-file	Required
EOF	2-9		End-of-file	Optional
UTL	†		User trailer	Optional
EOV	1	End-of-file chapter	End-of-volume	Required when a file crosses tape volume
EOV	2-9		End-of-volume	Optional
UTL	†		User trailer	Optional

† An a-type character defined in the Required Labels section.

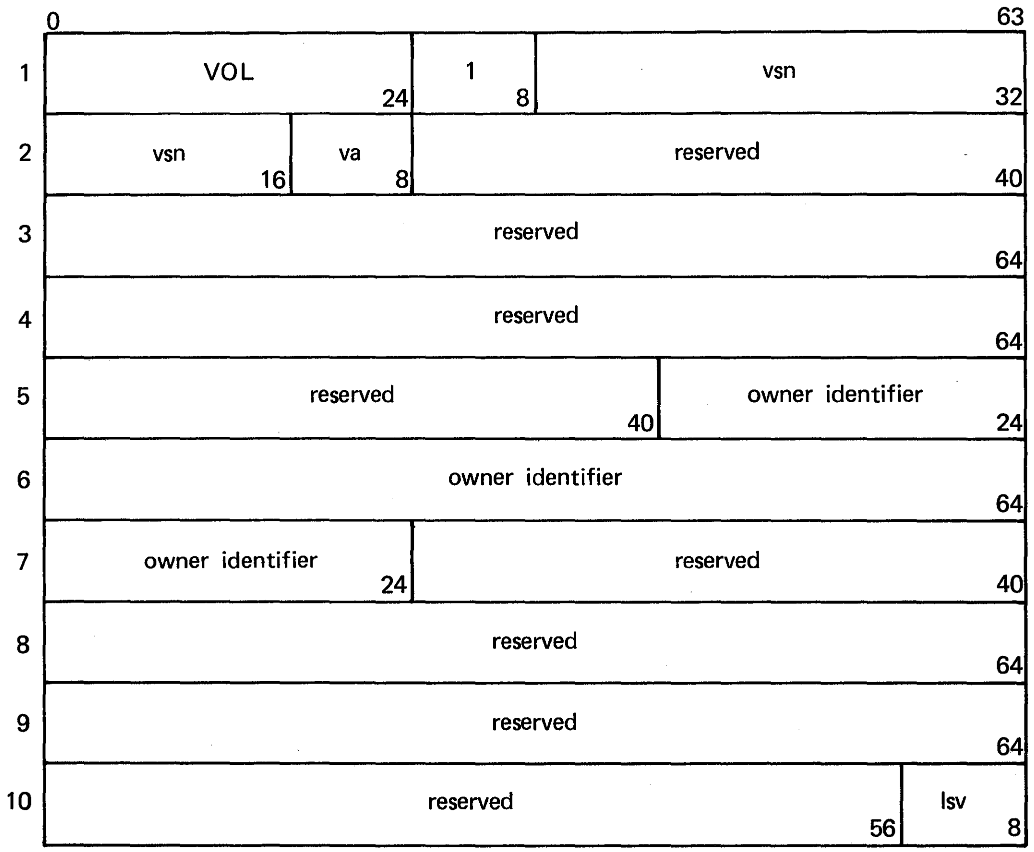
Volume Header Label (VOL1)

The volume header label must be the first label on a labeled tape. All reels begin with a VOL1 label. The user can use the existing VOL1 label or write a new VOL1 label. In either case, the volume accessibility character in the tape VOL1 label must match the original volume accessibility (ova) in the TAPE MANAGEMENT Beta. If the user is writing a new VOL1 label, UVL labels can also be written.

The system processes the following fields in the VOL1 label.

- Label identifier
- Label number
- Volume identifier
- Accessibility
- Label-standard version

The format of the volume header label is shown in figure 4-1.



Word	Field	Description
1-2	vsn	Volume serial number assigned by the owner to identify this physical reel of tape.
2	va	Accessibility. An a-type character which indicates the restrictions, if any, on who may have access to the information on the tape. A blank means unlimited access.
5-7	owner identifier	Any a-type characters identifying the owner of the physical volume.
10	lsv	Label-standard version: 3 Labels and data formats on this volume conform to the requirements of the ANSI X3.27-1978 standard. blank Labels and data formats on this volume require the agreement of the interchange parties.

Figure 4-1. VOL1 Format

First File Header Label (HDR1)

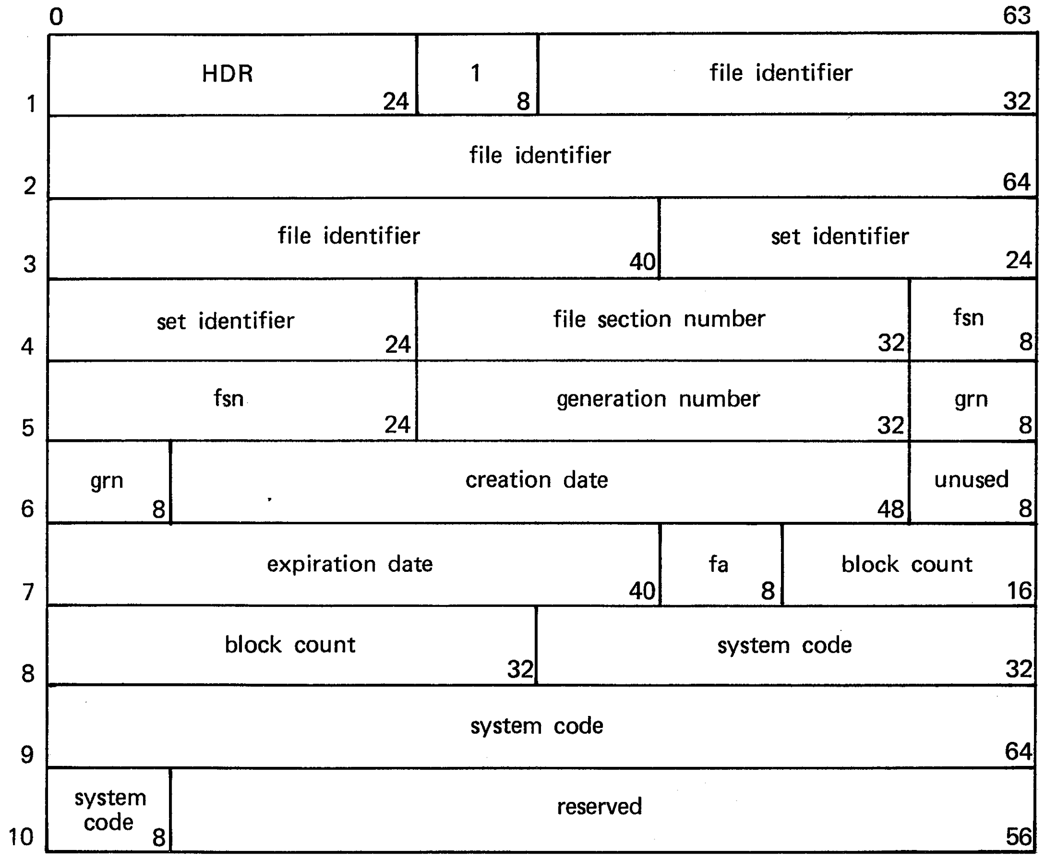
The first file header label must appear before each file. When a file is continued on more than one volume, the file header is repeated after the volume header label on each new volume for that file. If two or more files are grouped in a multifile set, each HDR1 label indicates the relative position of its associated file within the multifile set.

If writing labels, the system first positions the tape using only the file sequence number. If the file sequence number is 0, it defaults to the current tape file sequence number plus one (next file). In order to extend a multifile set, the file sequence number must be set to 9999. For this case, the system positions the tape after the last file in the multifile set and sets the file sequence number to the last member sequence number plus one.

The system processes the following fields in the HDR1 label.

- Label identifier
- Label number
- File identifier
- File set identifier
- File chapter number
- Expiration date
- Accessibility

The format of the first file header label is shown in figure 4-2.



Word	Field	Description
1-3	file identifier	Up to 17 a-type characters used to identify the file.
3-4	set identifier	Up to six a-type characters used to identify the file set. To conform to the ANSI tape standard, this value is the same for all files of a multifile set.
4	file section number	Four n-type characters identifying the file section number. The file section number of the first HDR1 label of a file is 0001. If the file extends to more than one volume, this number is incremented by one for each subsequent volume.
4-5	fsn	File sequence number. Four n-type characters to specify the position of a file within a file set. This value is 0001 for the first file, 0002 for the second, and so on. In all the labels for a given file, this field contains the same number.

Figure 4-2. HDR1 Format (Sheet 1 of 2)

Word	Field	Description
5	generation number	Four n-type characters specifying the generation number of a file. This value is 0001 for the first generation of a file, 0002 for the second, and so on.
5-6	grn	Generation version number. Two n-type characters used to distinguish successive iterations of the same generation. The generation version number of the first attempt to create a file is 00. This field is not checked for privilege jobs.
6	creation date	Date the file was created; it is recorded as a space followed by two n-type characters for the year followed by three n-type characters for the day within the year.
7	expiration date	The file is considered expired when today´s date is the same as or later than the date given in this field. When this condition is satisfied, the remainder of the volume may be overwritten. Thus, to be effective on multifile volumes, the expiration date of a file must be earlier than or the same as the expiration date of all preceding files on the volume. The expiration date is written in the same format as the creation date.
	fa	File accessibility. An a-type character which indicates the restriction, if any, on who may have access to the information in this file. A blank means unlimited access. An A means the owner identification field in the VOL1 label must contain the owner´s user number. If any other character, all future accesses to the tape must specify this character as the fa.
7-8	block count	This field must be zero-filled.
8-10	system code	Thirteen a-type characters identifying the operating system that recorded this file. The tape is considered to have been written under VSOS if the first 10 characters match the default.

Figure 4-2. HDR1 Format (Sheet 2 of 2)

First End-of-File Label (EOF1)

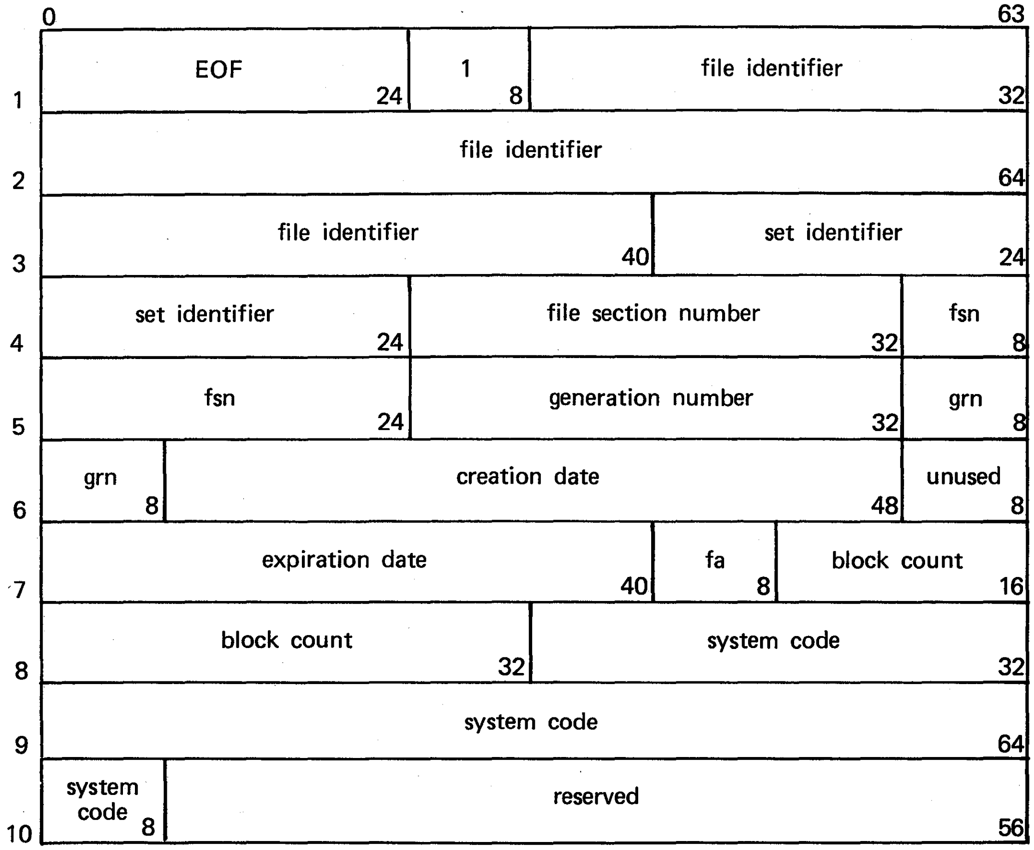
The end-of-file label is the last block of every file. It is the system end of information for the file. A single tape mark precedes EOF1. A double tape mark written after the EOF1 label marks the end of a multifile set.

When writing labels, the system uses the fields from the HDR1 label to write the corresponding fields in the EOF1 label.

The system processes the following fields in the EOF1 label.

- Label identifier
- Label number
- Block count

The format for the first end-of-file label is shown in figure 4-3.



Word	Field	Description
1-3	file identifier	Up to 17 a-type characters used to identify the file.
3-4	set identifier	Up to six a-type characters used to identify the file set. To conform to the ANSI tape standard, this value is the same for all files of a multifile set.
4	file section number	Four n-type characters identifying the file section number. The file section number of the first HDR1 label of a file is 0001. If the file extends to more than one volume, this number is incremented by one for each subsequent volume.

Figure 4-3. EOF1 Format (Sheet 1 of 2)

Word	Field	Description
4-5	fsn	File sequence number. Four n-type characters to specify the position of a file within a file set. This value is 0001 for the first file, 0002 for the second, and so on. In all the labels for a given file, this field contains the same number.
5	generation number	Four n-type characters specifying the generation number of a file. This value is 0001 for the first generation of a file, 0002 for the second, and so on.
5-6	grn	Generation version number. Two n-type characters used to distinguish successive iterations of the same generation. The generation version number of the first attempt to create a file is 00. This field is not checked for privileged jobs.
6	creation date	Date the file was created; it is recorded as a space followed by two n-type characters for the year followed by three n-type characters for the day within the year.
7	expiration date	The file is considered expired when today's date is the same as or later than the date given in this field. When this condition is satisfied, the remainder of the volume may be overwritten. Thus, to be effective on multifile volumes, the expiration date of a file must be earlier than or the same as the expiration date of all preceding files on the volume. The expiration date is written in the same format as the creation date.
	fa	File accessibility. An a-type character which indicates the restriction, if any, on who may have access to the information in this file. A blank means unlimited access. An A means the owner identification field in the VOL1 label must contain the owner's user number. If any other character, all future accesses to the tape must specify this character as the fa.
7-8	block count	Six n-type characters specifying the number of PRUs between this label and the preceding HDR label group. This total does not include labels or tape marks.
8-10	system code	Thirteen a-type characters identifying the operating system that recorded this file. The tape is considered to have been written under VSOS if the first 10 characters match the default.

Figure 4-3. EOF1 Format (Sheet 2 of 2)

First End-of-Volume Label (EOV1)

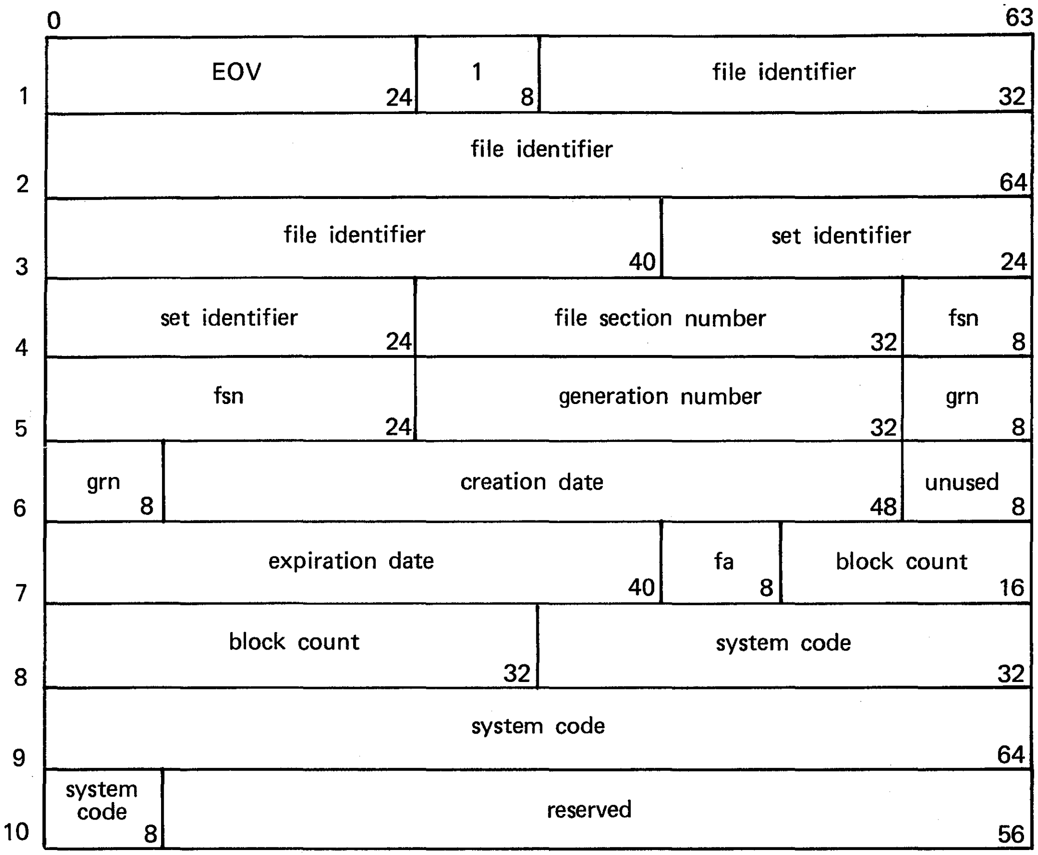
The end-of-volume label is required only if the physical end-of-tape reflector is encountered before an EOF1 label is written or if a multifile set is continued on another volume. EOV1 is preceded by a single tape mark and followed by a double tape mark.

When writing labels, the system uses the fields in the HDR1 label to write the corresponding fields in the EOV1 label.

The system processes the following fields in the EOV1 label.

- Label identifier
- Label number
- Block count

The format for the first end-of-volume label is shown in figure 4-4.



Word	Field	Description
1-3	file identifier	Up to 17 a-type characters used to identify the file.
3-4	set identifer	Up to six a-type characters used to identify the file set. To conform to the ANSI tape standard, this value is the same for all files of a multifile set.
4	file section number	Four n-type characters identifying the file section number. The file section number of the first HDR1 label of a file is 0001. If the file extends to more than one volume, this number is incremented by one for each subsequent volume.
4-5	fsn	File sequence number. Four n-type characters to specify the position of a file within a file set. This value is 0001 for the first file, 0002 for the second, and so on. In all the labels for a given file, this field contains the same number.
5	generation number	Four n-type characters specifying the generation number of a file. This value is 0001 for the first generation of a file, 0002 for the second, and so on.

Figure 4-4. EOV1 Format (Sheet 1 of 2)

Word	Field	Description
5-6	grn	Generation version number. Two n-type characters used to distinguish successive iterations of the same generation. The generation version number of the first attempt to create a file is 00. This field is not checked for privilege jobs.
6	creation date	Date the file was created; it is recorded as a space followed by two n-type characters for the year followed by three n-type characters for the day within the year.
7	expiration date	The file is considered expired when today's date is the same as or later than the date given in this field. When this condition is satisfied, the remainder of the volume may be overwritten. Thus, to be effective on multifile volumes, the expiration date of a file must be earlier than or the same as the expiration date of all preceding files on the volume. The expiration date is written in the same format as the creation date.
	fa	File accessibility. An a-type character which indicates the restriction, if any, on who may have access to the information in this file. A blank means unlimited access. An A means the owner identification field in the VOL1 label must contain the owner's user number. If any other character, all future accesses to the tape must specify this character as the fa.
7-8	block count	Six n-type characters specifying the number of PRUs between this label and the preceding HDR label group. This total does not include labels or tape marks.
8-10	system code	Thirteen a-type characters identifying the operating system that recorded this file. The tape is considered to have been written under VSOS if the first 10 characters match the default.

Figure 4-4. EOV1 Format (Sheet 2 of 2)

Optional Labels

Six types of optional labels are processed. They are additional file header (HDR2 through 9), end of file (EOF2 through 9), end of volume (EOV2 through EOV9), user volume (UVLa), header (UHLa), and trailer (UTLa) labels. These labels are written to tape if supplied in a label buffer or returned to the user if a label buffer is supplied.

Additional File Header Labels (HDR2 through HDR9)

HDR2 through HDR9 labels may immediately follow HDR1. Their format is as follows:

Character Position	Field Name	Contents
1-3	Label identifier	HDR
4	Label number	2 through 9
5-80		Any a-type character

Only the label identifier and the label number are checked when writing label. The label number must be in ascending order, beginning with 2.

Additional End-of-File Labels (EOF2 through EOF9)

EOF2 through EOF9 labels may immediately follow EOF1. Their format is as follows:

Character Position	Field Name	Contents
1-3	Label identifier	EOF
4	Label number	2 through 9
5-80		Any a-type character

Only the label identifier and the label number are checked when writing labels. The label number must be in ascending order, beginning with 2.

Additional End-of-Volume Labels (EOV2 through EOV9)

EOV2 through EOV9 labels may immediately follow EOV1. Their format is as follows:

Character Position	Field Name	Contents
1-3	Label identifier	EOV
4	Label number	2 through 9
5-80		Any a-type character

Only the label identifier and label number are checked when writing labels. The label number must be in ascending order, beginning with 2.

User Labels

User labels may immediately follow their associated system labels. Thus, user volume labels (UVLa) may follow VOL1, user header labels (UHLa) may follow the last HDRn label, and user trailer labels (UTLa) may follow the last EOVn or EOFn label. Their format is as follows:

Character Position	Field Name	Contents
1-3	Label identifier	UVL, UHL, or UTL
4	Label number	Must be 1, 2, 3, 4, and so on, consecutively for UVL labels. For other labels, any a-type character
5-80		Any a-type character

Only the label identifier and the label number are checked when writing labels. The system checks the number of user labels of a label type; a maximum of 32 is allowed.

SYSTEM MESSAGES 5

Programs use system messages to request VSOS processing. With five exceptions, the system messages described in this chapter are calls to the virtual system. (The ADVISE, EXPLICIT I/O, GIVE UP, PROCESS SYSTEM PARAMETER, and TAPE FUNCTION messages are calls to the resident system.)

SYSTEM MESSAGE EXECUTION

A program can use either of two methods to issue a system message. It can call an SIL subroutine which, in turn, issues a system message, or it can issue the system message directly. SIL subroutines are described in volume 1. The SIL method is recommended because it is recognized as the supported user interface and will remain unchanged even though the system messages may change.

To issue a system message directly, the user presets one or two blocks of words known as the Alpha and Beta of the message and then issues an exit force instruction. The Alpha and Beta formats for each message are referenced in the individual message descriptions. The exit force instruction is described in the CYBER 200 Hardware Reference Manual.

A 32-bit indirect or 64-bit direct pointer immediately follows the exit force instruction within the instruction stream. It points to the system message Alpha. When the exit force instruction is executed, system operation changes to monitor mode and the system message is executed.

The hexadecimal format of an indirect message pointer is:

```
00EE00rr
```

rr is the number of the register containing the virtual bit address of the message. The hexadecimal format of a direct message pointer is:

```
00FFaddress
```

address is the virtual bit address of the first full word of the message (12 hexadecimal digits).

When a message is processed without error, the operating system returns control to the half word or full word immediately following the message pointer.

ALPHA AND BETA WORD CONVENTIONS

System messages have a two-part standard format. The first part, called the Alpha portion, specifies the function to be performed, the length of the Beta portion, and where to proceed for error processing. The Alpha portion has the same general format for all messages, and is always either two or three words in length.

The second part, called the Beta portion, contains parameters and varies greatly in length from one message to the next. The format of the Beta portion depends on the function, as described later in this chapter for each function code. The user specifies in the Alpha portion what the length of the Beta portion is and, in some cases, where it is located. The message descriptions in this chapter specify what the minimum size of any particular Beta must be. The user can, however, specify a larger Beta, in which case the extra space is left unchanged.

Alpha and Beta portions must start on full-word boundaries. They must exist in virtual space and have read/write or write-temporary access. Alpha and Beta portions must not cross large page boundaries.

NOTE

Options/control field values of #E0 through #FF, response code (r field) values of #7000 through #7FFF, and error response (ss or cerr field) values of #E0 through #FF are reserved for installation use. The options/control field is the 8 bits to the left of the function code field in Alpha. The response code field is bits 0 through 15 in Alpha (1), where bits are numbered from 0.

Values returned in the r, ss, serr, and cerr fields are in hexadecimal notation.

In the figures in this chapter, some of the Alpha and Beta words are drawn with dashed lines. These words are optional.

OVERVIEW

The following are the available system messages listed according to functional areas. (The comments in parentheses are meant to clarify the purpose of the message.)

File Management

ACCESS CONTROL (f=#002B)
ATTACH FILE (f=#0010)
CHANGE FILE ATTRIBUTES (f=#000B)
CREATE FILE (f=#0001)
DESTROY FILE (f=#0002)
GIVE FILE (changes file ownership) (f=#0008)
POOL FILE MANAGER (f=#0026)
FILE DISPOSITION (f=#000D)

Tape Management

LABEL (f=#002E)
TAPE MANAGEMENT (f=#002C)
TAPE SWITCH VOLUME (f=#002D)

Input/Output Operation

CLOSE FILE (f=#0005)
EXPLICIT I/O (f=#F500)
GIVE UP CPU ON OUTSTANDING RESIDENT I/O OR TIME (f=#FF02)
MAP (into virtual space) (f=#0004)
OPEN FILE (f=#0003)
TAPE FUNCTION (f=#F406)

Interrupt Processing

ABNORMAL TERMINATION CONTROL (f=#0020)
PROGRAM INTERRUPT CONTROL (f=#001C)
RETURN FROM INTERRUPT (f=#0051)

Starting and Ending Program Execution

EXECUTE IQM REQUEST (f=#0030)
EXECUTE PROGRAM FOR USER NUMBER (f=#0022)
RECALL (suspends program execution) (f=#0025)
TERMINATE (ends program execution) (f=#0006)
USER REPRIEVE (f=#002F)

Controllee Chain Processing

INITIALIZE CONTROLLEE CHAIN (f=#001D)
INITIALIZE OR DISCONNECT CONTROLLEE (f=#001B)
LIST CONTROLLEE CHAIN (f=#0013)
REMOVE CONTROLLEE FROM MAIN MEMORY (f=#0019)

Message Communication

GET MESSAGE FROM CONTROLLEE (f=#0017)
GET MESSAGE FROM CONTOLLER OR OPERATOR (f=#0016)
SEND MESSAGE TO CONTROLLEE (f=#0015)
SEND MESSAGE TO CONTROLLER (f=#0014)
SEND MESSAGE TO OPERATOR (f=#001A)
SEND MESSAGE TO DAYFILE (f=#0029)
SEND MESSAGE TO JOB SESSION (f=#0033)

File Space Allocation

ADVISE (on virtual space requirements) (f=#FF00)
PROCESS SYSTEM PARAMETER (sets memory limits) (f=#FF01)

Information Retrieval

GET PACK LABEL AND PFI (f=#0011)
LIST FILE INDEX TABLE (f=#0007)
LIST SYSTEM TABLE (f=#0009)
MISCELLANEOUS (f=#0024)

Accounting

UPDATE USER DIRECTORY (f=#0023)
USER/ACCOUNTING COMMUNICATION (f=#000E)
VARIABLE RATE ACCOUNTING (f=#0028)

Special Functions

EXECUTE OPERATOR COMMAND (for operator user number) (f=#0021)
RHF_CALL (RHF functions) (f=#002A)
SHRLIB ALTER or RESTORE (f=#0053)

MESSAGES

The message descriptions in this chapter are in function code order. Table 5-1 lists the messages in alphabetical order.

Table 5-1. Message Function Codes (Sheet 1 of 3)

Message	Hexadecimal† Function Code
ABNORMAL TERMINATION CONTROL	0020
ACCESS CONTROL	002B
ADVISE	FF00
ATTACH	0010
CHANGE FILE ATTRIBUTES	000B
CLOSE FILE	0005
CREATE FILE	0001
DESTROY FILE	0002
EXECUTE IQM REQUEST††	0030
EXECUTE OPERATOR COMMAND††	0021
EXECUTE PROGRAM FOR USER NUMBER††	0022
EXPLICIT I/O	F500
GET MESSAGE FROM CONTROLLEE	0017
GET MESSAGE FROM CONTROLLER OR OPERATOR	0016
GET PACK LABEL AND PFI	0011
GIVE FILE	0008
GIVE UP CPU ON OUTSTANDING RESIDENT I/O OR TIME	FF02
INITIALIZE CONTROLLEE CHAIN	001D
INITIALIZE OR DISCONNECT CONTROLLEE	001B

† #1E, #1F, and #E0 through #FF are reserved for installation use. Rightmost field in Alpha(1). Abbreviated as f.

†† Available to a privileged system task.

Table 5-1. Message Function Codes (Sheet 2 of 3)

Message	Hexadecimal† Function Code
LABEL	002E
LIST CONTROLLEE CHAIN	0013
LIST FILE INDEX TABLE	0007
LIST SYSTEM TABLE	0009
MAP	0004
MESSAGE CONTROL	0018
MISCELLANEOUS	0024
OPEN FILE	0003
POOL FILE MANAGER	0026
PROCESS SYSTEM PARAMETER	FF01
PROGRAM INTERRUPT CONTROL	001C
RECALL	0025
REMOVE CONTROLLEE FROM MAIN MEMORY	0019
RETURN FROM INTERRUPT	0051
RHF_CALL ††	002A
ROUTE AND FILE DISPOSITION	000D
SEND MESSAGE TO CONTROLLEE	0015
SEND MESSAGE TO CONTROLLER	0014
SEND MESSAGE TO DAYFILE	0029
SEND MESSAGE TO JOB SESSION	0033
SEND MESSAGE TO OPERATOR	001A
SHRLIB ALTER OR RESTORE	0053
TAPE FUNCTION	F406
TAPE MANAGEMENT	002C

†#1E, #1F, and #E0 through #FF are reserved for installation use. Rightmost field in Alpha(1). Abbreviated as f.

††Available to a privileged system task.

Table 5-1. Message Function Codes (Sheet 3 of 3)

Message	Hexadecimal† Function Code
TAPE SWITCH VOLUME	002D
TERMINATE	0006
UPDATE USER DIRECTORY	0023
USER REPRIEVE	002F
USER/ACCOUNTING COMMUNICATION	000E
VARIABLE RATE ACCOUNTING	0028

†#1E, #1F, and #E0 through #FF are reserved for installation use. Rightmost field in Alpha(1). Abbreviated as f.

CREATE FILE (f=#0001)

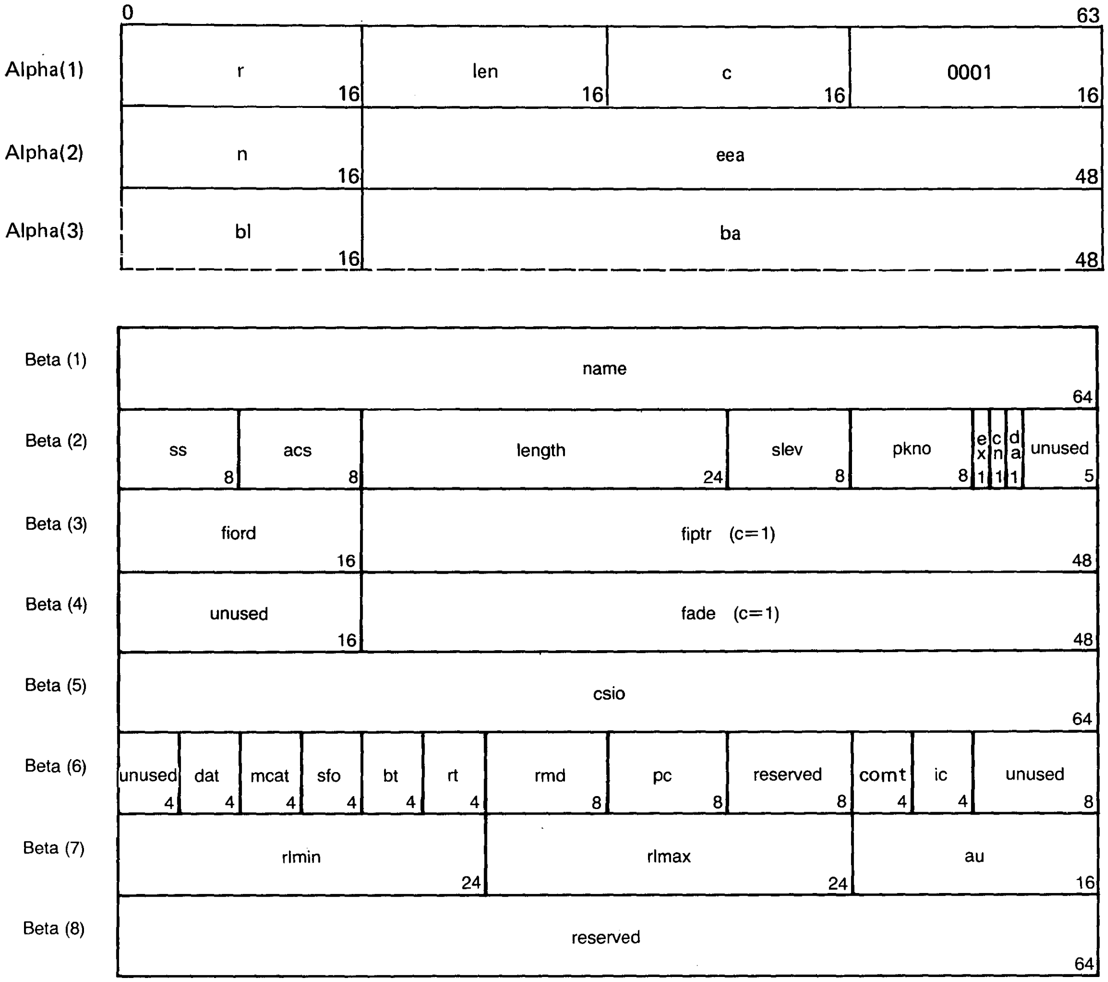
The CREATE FILE message defines parameters for files. Except for files connected to a terminal, this message also assigns space, usually on a mass storage device, names it, and gives that space to a user. The operating system makes an entry in the file index table and PFI for this named space (file), and initializes fields in the entry using information given in the message. The format of the CREATE FILE message is shown in figure 5-1.

A privileged user can set some of the values in the new file index table entry the operating system creates every time a file is created. Eight Beta words are required for a privileged create. Only one Beta is processed per Alpha issued. In Beta(4) the user can provide a file´s access directory entry.

Beta(3) contains the virtual bit address of a file index table entry copy as shown for the file index table in chapter 2. The user sets the following fields of the copy, which the system uses to initialize the created file´s file index table entry.

For the file´s access directory entry option, Beta(4) contains the virtual bit address of a file index table extension entry. The format of this entry is as described for the Beta portion (message option #10) of the LIST SYSTEM TABLE message. The first two words are filled in by the system before storing to ensure that there is no mismatch between the file being created and its associated file access directory entry. For files that do not have a file access directory defined, Beta(4) must contain 0.

The operating system sets the mcat and acs fields for a privileged create (c=1); otherwise, values of the message fields are provided by the user.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Error code was returned in an ss field of Beta(3). 2 User is not privileged (privileged creates only). #214 Beta buffer length error; the buffer length must be greater than or equal to 8.

Figure 5-1. CREATE FILE (f=#0001) Message Format (Sheet 1 of 5)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. All requests must provide at least four Beta words and for privileged creates, len must be a multiple of 5.
	c	Create mode: 0 Request a local file. 1 Define an unattached permanent file (privileged only). 2 Define a permanent file (make local file permanent or create a permanent file).
Alpha(2)	n	Number of creates in this message; maximum is 16.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.
Beta(1)	name	File name, in ASCII, left-justified with blank fill. File names must be in the format described in chapter 3.
Beta(2)	ss	Error response field. The values are: 0 No error. 1 File already exists. 2 No available mass storage for this file. 3 Invalid mcat specified. 4 Invalid C option specified. 5 The file index table, user table, or File Segment table is full. 6 Invalid file name. 7 Invalid data type. 8 Unable to find the requested pack identifier. #A If c=1 or c=2, error in attempt to make file permanent. #B If c=1, cannot locate user or pool. #C Requested file size is greater than installation parameter LDSK. #D Number of user files exceeds installation limit. #E If c=2, attempt to define a tape file. #F Attempt to create a file at a higher security than allowed. #10 If c=2, attempt to define a file connected to a terminal. #11 Illegal value in the comt (communications type) field. #12 Invalid access. #13 Illegal value in the sfo (file organization) field. #14 Illegal value in the bt (blocking type) field. #15 Illegal value in the rt (record type) field. #16 Invalid sfo/rt combination. #17 Illegal value for ostat (bits 59 through 63 of Beta(6) must be zero). #18 Caller not the file owner. #19 Production status lost on the file. Warning only, the file is created. Privileged create only.

Figure 5-1. CREATE FILE (f=#0001) Message Format (Sheet 2 of 5)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>Beta(2)</td><td>acs</td><td>Initial access permissions. This 8-bit field is treated as eight, 1-bit fields with each bit specifying the associated permission:
<table>
<tr><th>Bit</th><th>Hexadecimal Value</th><th>Description</th></tr>
<tr><td>1-3</td><td>-</td><td>Unused.</td></tr>
<tr><td>4</td><td>10</td><td>Execute access permitted.</td></tr>
<tr><td>5</td><td>8</td><td>Modify access permitted.</td></tr>
<tr><td>6</td><td>4</td><td>Append access permitted.</td></tr>
<tr><td>7</td><td>2</td><td>Read access permitted.</td></tr>
<tr><td>8</td><td>1</td><td>Write access permitted.</td></tr>
</table></td></tr>
<tr><td></td><td>length</td><td>Length of the file to be created in 512-word blocks. The actual file length is rounded up to a disk allocation unit boundary and returned to the called.</td></tr>
<tr><td></td><td>slev</td><td>Security level (1 through 8) to be given to the file if this field is not zero and is not greater than that of the interactive job issuing this message. If the field is zero, use the security level belonging to the interactive job issuing this message.</td></tr>
<tr><td></td><td>pkno</td><td>Indicates pack number. If a calling parameter, this field contains the number of the disk pack in the device set on which the file is to be created. VSOS returns the number of the disk pack on which the initial segment of the file was created. Valid pack number entries are all binary numbers from #1 through #80 for which a disk pack exists. Specifying a 0 allows the operating system to choose the disk pack on which to allocate space.</td></tr>
<tr><td></td><td>ex</td><td>File extensions:
0 File may be extended.
1 File may not be extended.
If cn is set to 1 but ex is set to 0, a contiguous, extendable file is created. Therefore, a file that was contiguous when created may become noncontiguous when later extended.</td></tr>
<tr><td></td><td>cn</td><td>File contiguity requirements:
0 File may be created as a noncontiguous (segmented) file.
1 File must be created as a contiguous (nonsegmented) file.</td></tr>
<tr><td></td><td>da</td><td>If c=2, action statement returned by the system:
0 New file created.
1 Existing local file made permanent.</td></tr>
</table>

Figure 5-1. CREATE FILE (f=#0001) Message Format (Sheet 3 of 5)

Word	Field	Description
Beta(3)	fiord	File position (local file identifier); returned to the caller.
	fiptr	For privileged creates (c=1), this field contains the virtual bit address (furnished by the user) of a 16-word copy of the file index table extension entry that is used to define the characteristics of the file being created.
Beta(4)	fade	For privileged creates (c=1), this field contains the virtual bit address (furnished by the user) of a 16-word copy of the file index table extension entry. The system uses the file access directory portion of this entry to initialize the file´s access directory entry in the file index table. The format of the file index table extension entry copy is the same as for the Beta portion of the LIST SYSTEM TABLE message (f=#0009), option #10.
Beta(5)	csio	Field reserved for the operating system. The contents are not defined on return to the caller.
Beta(6)	dat	Data type: 0 Physical data file. 1 Virtual code file.
	mcat	File management category: 0 Mass storage file. 1 Scratch file (valid only if c=0). 2 Output file. 5 User-created drop file. 9 File connected to a terminal (valid only if c=0). The operating system sets the mcat field to 0 for a privileged create. For categories 0 through 2 of this field, standard file name conventions apply.
	sfo	File organization: 0 Sequential file. 1 Direct file.
	bt	Blocking type field. This field is ignored by the system on entry, and is set to 2 on return.
	rt	Record type: 0 Control word (W). 1 ANSI field length (F). 2 Record mark (R). 7 Undefined.
	rmd	The record mark delimiter may be any 8-bit ASCII character.
	pc	A padding character is used only with F-type records. It may be any 8-bit ASCII character.

Figure 5-1. CREATE FILE (f=#0001) Message Format (Sheet 4 of 5)

Word	Field	Description
Beta (6)	reserved	Reserved for the operating system.
	comt	Communication type: 0 Non-RHF. 1 RHF.
	ic	Not used.
Beta(7)	rlmin	Contains the minimum record length in bytes.
	rlmax	Contains the maximum record length in bytes.
	au	Allocation unit size is used by the operating system as a guideline when extending a file. The value in this field is given as the number of 512-word blocks.
Beta(8)	reserved	Reserved for the operating system.

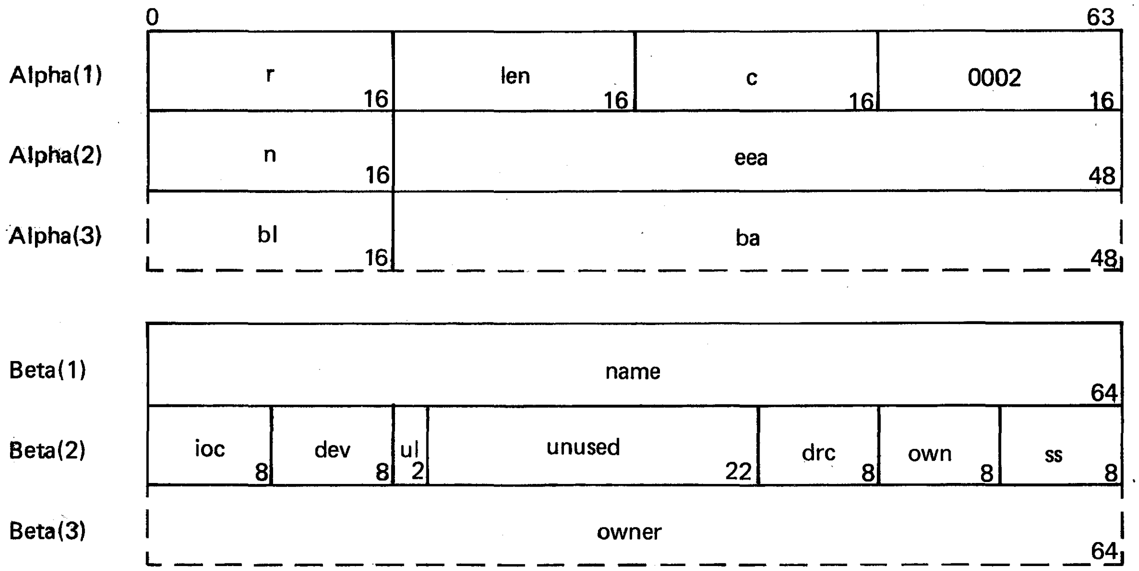
Figure 5-1. CREATE FILE (f=#0001) Message Format (Sheet 5 of 5)

DESTROY FILE (f=#0002)

The DESTROY FILE message can be issued to sever the program's connection with a file and/or release the mass storage space. At the conclusion of DESTROY FILE message processing, any mass storage file referenced by the message has ceased to exist, as have any modified pages of the file. Virtual address definitions pertaining to this file are no longer defined, and the I/O connection and map entries are erased. The format of the message is shown in figure 5-2. (Only one Beta is processed for each Alpha.)

If a mass storage file is at a sufficiently high security level, it is overwritten with a pattern when it is destroyed. Some installations can choose to overwrite all files when they are destroyed. A privileged destroy is not a close and destroy, as is the nonprivileged destroy; the privileged destroy must be preceded by a privileged close.

If the name refers to a tape file, the system rewinds and unloads the current volume. If the name is a multifile set, all logical files belonging to the multifile set are returned.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Error code was returned in an ss field of Beta(2). 2 User is not privileged (privileged destroys only). #211 Number of destroys in this message is illegal (the n field is 0 or greater than 16). #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-2. DESTROY FILE (f=#0002) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in word of the Beta protion. The value of len must be a multiple of 2 (for regular destroys) or 3 (for privileged destroys).
	c	Destroy mode: 0 Return local and attached permanent files. 1 Privileged purge of a permanent file. 2 Purge of a permanent file (makes file local if attached). 3 Purge of a pool file in pool in Beta(3).
Alpha(2)	n	Number of requests in this message; maximum is 16.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.
Beta(1)	name	File name, in ASCII, of the file to be destroyed. File names must be in the format described in chapter 3.
Beta(2)	ioc	Input/output connector number. If the file is connected to a terminal, this field is #FE. If a mass storage file is being destroyed, the operating system returns, in this field, the inclusive OR of all I/O connector numbers connected to this file.
	dev	Device type: 0 Mass storage device or magnetic tape device. 8 Reserved.
	ul	Unload Tape. This field is significant only for returning files (c=0) and is only applicable to tape files. 0 When the tape is released, the tape is rewound to the load point and is then unloaded in accordance with the iu option specified in the TAPE MANAGEMENT system message (f=#002C). 1 When the tape is released, the tape is rewound to the load point, but is not unloaded from the drive. 2 When the tape is released, the tape is rewound to the load point and unloaded from the drive.
	drc	Decrement resource count if this field is nonzero. If drc=0, do not decrement resource count. This field applies only to tape files.
	own	Ownership of the file to be destroyed. This field is significant only for nonprivileged users (c=1). The values are: 0 Private ownership. 1 Public ownership; valid only for privileged users. 2 Pool ownership; valid only for the pool boss.

Figure 5-2. DESTROY FILE (f=#0002) Message Format (Sheet 2 of 3)

Word	Field	Description
Beta(2)	ss	Error response field. The values are: 0 Normal completion. 1 File name does not exist. 2 File name given is in conflict with that in the I/O connector. 3 Another active program has the file open, or the file has been privileged opened. 4 Attempt to purge a permanent file attached to another job. 5 Nonprivileged task tried to destroy a public file. 6 User other than the pool boss tried to destroy a pool file. 7 Illegal I/O connector number specified. 8 Drop file map is full. 9 Error trying to remove the PFI entry. #A Disk is logically off. #B Caller is not the file owner. #C No room in FILE1 for privileged destroy pseudologon. #D Attempt to destroy an open tape file. #E Illegal ul option specified. #F Cannot destroy a public file unless privileged. #10 Attempt to purge a tape file. #11 Pool not attached or does not exist.
Beta(3)	owner	For privileged destroys, a user number or pool name to which the file being destroyed belongs. The binary user number must be right-justified with zero fill or, if this is the pool name, it must be left-justified with blank fill.

Figure 5-2. DESTROY FILE (f=#0002) Message Format (Sheet 3 of 3)

OPEN FILE (f=#0003)

The format of the OPEN FILE message is shown in figure 5-3. (The Beta portion of the message can actually consist of more than one of the five- or six-word sets shown in the figure.)

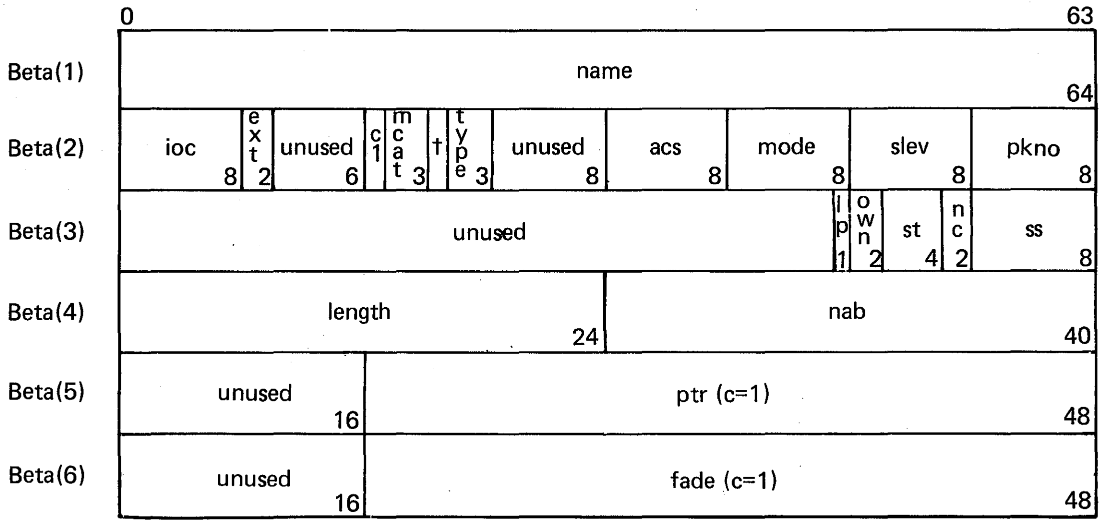
	0			63
Alpha(1)	r 16	len 16	c 16	0003 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Error code was returned in an ss field of Beta(3). 2 User is not privileged (privileged opens only). 3 Input/output error code was returned. #4 Illegal message option (c field). #211 Number of opens in this message is illegal (the n field is 0 or greater than 16). #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified. #216 Descriptor is out of bounds.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion.
	c	Open mode: 0 Regular open; file must be attached. 1 Privileged open; fields dola, tlr, dolm, and tolm in the file index table are not updated for the file being opened.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 1 of 9)

Word	Field	Description
Alpha(1)	c	This field determines the settings of several other fields in Beta(2) of this message. When this field is 0, the cl option enables the user to modify fields in the file index table. Permission to modify these fields is granted by the system if the file ownership is: private; pool, and the user is the pool boss; or public, and the user is privileged. When this field is 1, the cl option enables the privileged user to specify who can access the file for the duration of this open.
Alpha(2)	n	Number of files to be opened at this time; maximum is 16. At times, it might be more efficient to open more than one file at a time. When this is to be done, the Alpha portion for the OPEN FILE message is used once, with n equaling the number of files to be opened; this is followed by groups of Beta words, one group per file.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full words of the Beta portion.

For nontape files:



†Unused.

Word	Field	Description
Beta(1)	name	File name, in ASCII. File names must be in the format described in chapter 3. If the format is not proper, error response #21 is returned in the ss field.
Beta(2)	ioc	The file's input/output connector number (0 to #F and #12 to #47), #FE, or #FF. #FE indicates that a file connected to a terminal is to be opened.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 2 of 9)

Word	Field	Description
Beta(2)	ioc	#FF causes the operating system to allocate an input/output connector and then to return the number in this field. If no input/output connector is available when the system attempts to allocate one, the system returns an error code of #37 in the error response field in this message.
	ext	File extendability; set by the user. The values are: 0 Extensions allowed if extensions were not prohibited on creation of the file (same as ext=2). 1 No extensions allowed (same as ext=3). Value that was set at creation time is returned by the operating system after a successful open. This field is 0 if extensions were allowed or 1 if not allowed. If the file was created with no extensions allowed, it would not have been opened with extensions allowed; however, if the file was created with extensions allowed, it can be opened with either extensions allowed or not allowed.
	cl	Open the file as specified in the mode field. For regular opens, the values are: 0 Do not change the file type. 1 Change the file type to the one in the type field. For privileged opens, the values are: 0 Other privileged and nonprivileged opens are allowed, but without write access. 1 No other opens are allowed until the privileged open is complete; the privileged open cannot occur if any other opens or attaches currently exist.
	mcat	File management category to be associated with the file. For privileged opens, the operating system sets this field to 0. For regular opens, this field is copied into the mcat field of the I/O connector. A file connected to a terminal is indicated by ioc=#FE instead of in the mcat field: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as a write-temporary file). 4 Tape file.
	type	File type. If the cl option is 0, the operating system returns the file type to this field. If the cl option is 1, the file type is to be changed to the type specified by this field, which can be one of the following: 0 Physical data. 1 Virtual data. 2 Virtual code. The operating system sets this field to 0 for privileged opens.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 3 of 9)

Word	Field	Description
Beta(2)	acs	File access desired. Only the indicated access combinations are allowed. The values are: Hex. Value — Description 00 — Open file for R, W, or RW access as determined by access permissions. 01 — Write access requested. 02 — Read access requested. 03 — Read, write access requested. 04 — Append access requested. 06 — Read, append access requested. 08 — Modify access requested. 0A — Read, modify access requested. Observe that if acs is 0, the system will attempt to open the file for read and write access. If the caller has read, write, or read and write permissions, the file is opened accordingly. The actual access obtained is returned in acs. If the caller has neither read nor write access, an access violation error is returned.
	mode	Input/output mode. This field is 0 if the file is to be opened for explicit I/O, or set to 1 if the file is to be opened for implicit I/O.
	slev	Security level of this file, 1 through 8; set by the operating system.
	pkno	Pack number of the disk pack on which the initial segment of the file resides; returned by the operating system.
Beta(3)	lp	Field returned by the operating system. This field is 0 for a permanent file, and 1 for a local file.
	own	File ownership; set by the operating system. The values are: 0 Private. 1 Public. 2 Pool.
	st	Management category of the file; set by the operating system. The values are: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 Write-temporary file. 4 Magnetic tape file. 5 Drop file created by the user. 6 Drop file created by the operating system. 7 Batch file. 9 File connected to a terminal.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 4 of 9)

Word	Field	Description
Beta(3)	nc	File extendability and contiguity; returned by the operating system if the open is successful. The values are: 0 Contiguous create was not requested but extensions are allowed. 1 Contiguous create was not requested and extensions are allowed. 2 Contiguous create was requested and extensions are allowed. 3 Contiguous create was requested but extensions are allowed.
	ss	Error response field. The values are: 0 Normal completion. #21 Either no name was given or the file is not attached. #22 Illegal value in the mcat field. #24 I/O connector is already in use or not #0 through #45. #25 Illegal value in the acs or type field. #2A File spans downed device, and open access is not read only (privileged opens only). #2B User directory was not found or the pool was not found (privileged opens only). #2C Read or write open is not allowed; the file has been privileged opened by another user. #2D Nonprivileged user. #2F No more write opens permitted. #31 No more room for the user table (privileged opens only). #32 Cannot open an attached file (privileged opens only). #33 No FST space available. #34 File access violation. #35 Implicit mode required with write temporary. #37 No I/O connector available. #3A Attempt to implicitly open a file with write-only access. #3B Cannot privilege open tape file. #3C Cannot locate tape volume. #3D Cannot open tape file implicitly. #3E File does not exist. #3F Cannot privilege open local disk file. #40 Calling task is not a level-2 controllee. #41 Warning; file is open but may be only partially available. #50 Error in modifying the PFI entry for this file. #61 Attempt to open a purge only file. #62 File is currently privileged open.
Beta(4)	length	The length of this file in blocks, set by the operating system. When ss=#41, this is the number of blocks available.
	nab	Relative byte address, returned by the system, of the next byte to be written in the file.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 5 of 9)

Word	Field	Description
Beta(5)	ptr	For privileged opens (c=1), this field contains the virtual bit address (furnished by the user) of the first word of a 16-word area in which the operating system is to return a formatted copy of the file index table entry for the opened file. The format of the file index table entry copy is the same as for the Beta portion of the LIST FILE INDEX OR SYSTEM TABLE message (f=#0009), option 1, or unformatted as described in chapter 2, depending on the setting of fmt. The first word of the file index table entry copy must be prefilled by the user with the user number or the pool name of the file to be opened; the second word contains the file name; and the remaining words contain the file index table, as supplied by the operating system.
Beta(6)	fade	For privileged opens (c=1), this field contains the virtual bit address (furnished by the user) of the first word of the 16-word area in which the operating system is to return a copy of the file index table extension entry for the opened file. The entry copy contains the file access directory for the file and is the same format as for the Beta portion of the LIST FILE INDEX OR SYSTEM TABLE message (f=#0009), option #10. This first two words of the copy are set to 0 by the operating system if the file does not have an extension entry.

For tape files:

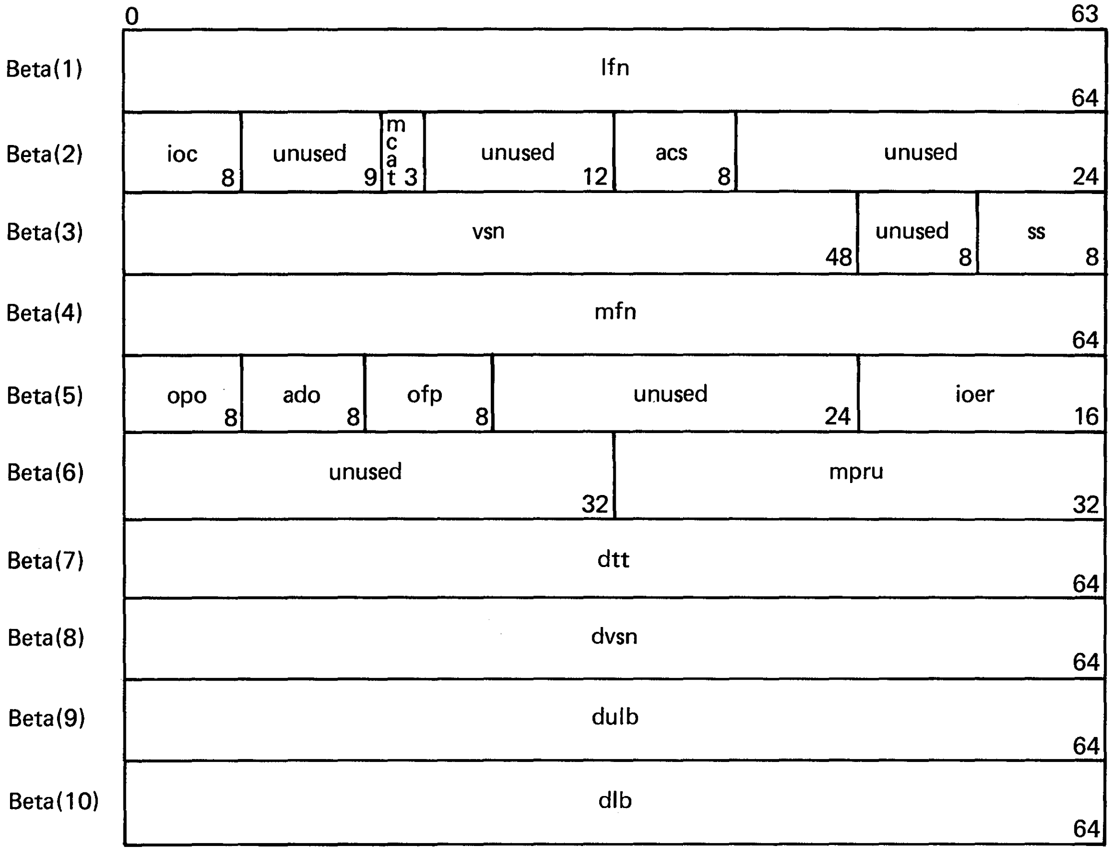


Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 6 of 9)

Word	Field	Description
Beta(1)	lfn	Logical tape file name, in ASCII.
Beta(2)	ioc	The file´s input/output connector number.
	mcat	Management category, returned by the system. This field is set to 4 for a logical tape file.
	acs	Access permissions. If acs=1 and unexpired HDR1 label date is found, an error is returned. It is possible for the installation to allow the operator to override this condition and allow writing on an unexpired tape: 0 acs is set from the file index. 1 Write-only permission only. 2 Read-only permission only. 3 Read/write permission.
Beta(3)	vsn	Volume serial number of the currently assigned tapes. This field is returned by the system.
	ss	Error response field: 0 Normal completion. #21 Either no name was given or the file is not attached. #22 Illegal value in the mcat field. #24 I/O connector is already in use or not #0 through #45. #25 Illegal value in the acs or type field. #2A Disk is logically off. #2B User directory was not found or the pool was not found (privileged opens only). #2C Read or write open is not allowed; the file has been privileged opened by another user. #2D Nonprivileged user. #2F No more write opens permitted. #31 No more room for the user table (privileged opens only). #32 Cannot open an attached file (privileged opens only). #33 No FST space available. #34 File access violation. #35 Implicit mode required with write temporary. #37 No I/O connector available. #38 Need six Beta words for option c=2. #39 Illegal user number for option c=2. #3A Attempt to implicitly open a file with write-only access. #3B Cannot privilege open tape file. #3C Cannot locate tape volume. #3D Cannot open tape file implicitly. #3E File does not exist. #3F Cannot privilege open local disk file. #40 Calling task is not a level-2 controllee. #41 Not all entries were returned in the tapes table array; OPEN was completed. #42 Not all VSNs were returned in the VSN array; OPEN was completed. #43 Label buffer was too short; OPEN was completed. #44 File identifier does not match.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 7 of 9)

Word	Field	Description
Beta(3)	ss	#45 File set identifier does not match.
		#46 File chapter number does not match.
		#47 File sequence number does not match.
		#48 Generation number does not match.
		#49 Generation version number does not match.
		#4A File accessibility character does not match.
		#4B File accessibility character is A and user number does not match.
		#4C Illegal labels.
		#4D Volume not available.
		#4E Header 1 not found.
		#50 Error in modifying the PFI entry for this file.
		#51 No unit was assigned.
		#52 Illegal assembly/disassembly.
		#53 Illegal access.
		#54 Only one tape open per Alpha allowed.
		#55 Logical tape file already opened.
		#56 Label unexpired and IP_TPEXP=0.
		#58 Tape coded mode with ado=3.
		#59 Attempted to write expiration data greater than the multifile set expiration date.
		#5B Illegal tape position option.
Beta(4)	mfn	Multifile set name, returned by the system for the currently assigned tape. This field equals 0 if the logical file name does not belong to a multifile set.
Beta(5)	opo	Open tape file processing options. These processing options are in effect for as long as the tape file is opened:

Bit	Name	Description
0	ETP	End-of-tape processing option: 0 The system automatically switches volumes. 1 Control is returned to the user at end of tape.
1	3-7	Unused.
2	UEP	User error processing option: 0 Tape I/O errors encountered when reading or writing a tape are returned to the operator. The operator makes a decision whether to repeat or ignore the error, drop or rerun the job, and so forth. Refer to appendix B for more information on tape I/O errors. 1 Control is returned to the user when a tape I/O error occurs. Refer to appendix B.
3-7	Unused.	

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 8 of 9)

Word	Field	Description
Beta(5)	ado	Bit string assembly/disassembly option. This field specifies what type of assembly or disassembly is to be done on the data: 0 No assembly/disassembly is done. 3 Bits 60 to 64; 60 bits on tape, 64 bits in memory with the upper 4 bits equal to 0. The buffer address must be on a 64-bit word boundary for the TAPE FUNCTION call.
	ofp	Open file positioning option. If ofp=0, the file positioning selected is returned: 0 No rewind. 1 Rewind to the beginning of information of the current file.
	ioer	Error number. The r field in Alpha is set to 3 if ioer is nonzero. Refer to appendix B for a complete description of the ioer error numbers.
Beta(6)	mpru	Maximum PRU size in bytes. This field is valid only for tape formats V and NV. If mpru is 0, the mpru from the REQUEST is used. If mpru=0 and no mpru was specified at request time, the system default is 32,768 bytes.
Beta(7)	dtt	Tapes table descriptor. If nonzero, the system returns the tapes table entry. For ofp=3, the user supplies the tapes table and on completion, the updated tapes table entry is returned: 0-15 ltt Length of the tapes table, in words. This field must be 12 words long. 16-63 att Virtual bit address of the tapes table buffer. The buffer must be on a word boundary.
Beta(8)	dvsn	Descriptor for the VSN list. If nonzero, the system returns the VSN list: 0-15 lvsn Length of the VSN list, in words (0 < lvsn < 256). 16-63 avsn Virtual bit address of the VSN list. The buffer must be on a word boundary.
Beta(9)	dulb	Descriptor for the user header labels. If dulb is nonzero, the user header labels are supplied by the user. This field only applies when writing labels: 0-15 lulb Length of the user label buffer, in words (0 < lulb ≤ 512). 16-63 aulb Virtual bit address of the user label buffer. The buffer must begin on a word boundary.
Beta(10)	dlb	Descriptor for the label buffer. If dlb is nonzero, the system returns all labels here: 0-15 llb Length of the label buffer, in words (0 < llb ≤ 512). 16-63 alb Virtual bit address of the label buffer. The label buffer must be on a word boundary.

Figure 5-3. OPEN FILE (f=#0003) Message Format (Sheet 9 of 9)

Mass Storage Files

The OPEN FILE message connects the user's program to a preexisting file for performing input and output on the file. In opening a file, the user can accept the parameters given to the file when it was created; otherwise, if the file owner has given permission, the user can alter the parameters. Both physical and virtual files can be opened for either explicit or implicit I/O. Once opened for explicit I/O, however, a file cannot be accessed implicitly, and vice versa. Nevertheless, a file can be opened in several I/O connectors at the same time; some for implicit I/O, and others for explicit I/O.

When a program opens a physical file in explicit mode, the specified I/O connector in the program's minus page is filled in as required and an entry is made in the explicit file map area of the minus page. This allows initiation of explicit I/O. In this mode, the file is accessed by explicit requests to transfer data into buffer areas. The EXPLICIT I/O message (f=#F500), or its SIL counterpart, must be used to define the buffers and initiate data transfers.

When a program opens a physical file in implicit mode, the specified I/O connector in the program's minus page is completed. No entry is made in the bound explicit map. Explicit input/output cannot be accomplished on a physical file that is opened in implicit mode.

When a program opens a virtual file in explicit mode, all input/output must be done explicitly through the program's buffers in the same manner as for physical files opened in explicit mode. The I/O connector number specified in the program's minus page is filled in, and one entry is made in the explicit map. When a file is opened in explicit mode, no implicit access is possible to any of the virtual space usually represented by the file.

When a program opens a virtual file in implicit mode, the I/O connector number in the program's minus page is filled in.

For privileged opens to occur, the file must not be open with write access by anyone; while the file is privileged open, all attempts to open with write access are barred. If the cl field in Beta(2) is 1, these rules are extended to exclude an open of any sort to assure that the privileged open is successful.

A privileged user can get a copy of the opened file's file index table entry by specifying a virtual bit address in Beta(5). The copy is returned beginning at the specified address. This copy is not used in the same way that the copy can be used on a privileged create; initializing fields in the copy associated with an OPEN FILE message does not alter the values in the file index table entry. If this is used, the fmt=1 option to return the unformatted file index should be used.

A privileged user can also get a copy of the opened file's file index table extension entry, which contains the file access directory, by specifying a virtual bit address in Beta(6). The copy is returned, beginning at the specified address in the same format as the Beta portion of the LIST SYSTEM TABLE message (f=#0009), option #10. If no file index table extension entry exists, the first two words of the area, starting at the specified address, are set to 0 by the system.

If the file was created with no extensions allowed, it cannot be opened with extensions allowed; however, if the file was created with extensions allowed, it can be opened with either extensions not allowed or extensions allowed.

Magnetic Tape Files

The OPEN FILE message can be issued only for a logical tape file requested in the TAPE MANAGEMENT message or for a logical tape file requested in the LABEL message. If the logical tape file belongs to a multifile set, only one of the logical files can be opened at one time. There can be only one tape file specified in the Beta for each OPEN FILE message. After a successful open, the ioc is built, and the user can issue input/output and positioning functions to the tape file.

The file position at the time of the open is determined by the ofp field. Label processing is not required for a file that is being reopened after previous use in which label processing was done and the tape was left positioned within this file.

Observe that only explicit I/O is allowed for tape files. Implicit use may not be specified on the open.

Files Connected to a Terminal

The OPEN FILE message is also used to connect the user´s program to a file connected to a terminal. A connected file can be opened only if this is done by a level-2 or lower level controllee of an interactive processor. It cannot be opened implicitly.

A connected file does not use an I/O connector.

Since a file connected to a terminal is a SIL feature, no explicit or implicit I/O is done to this file. SIL traps all I/O requests and converts them to either GET MESSAGE FROM CONTROLLER or SEND MESSAGE TO CONTROLLER requests. This is why a file connected to a terminal needs no ioc. It does not use any buffers as explicit I/O does. Instead, it uses the numbered common block 99434642.

Observe that the following Beta fields are not valid for a file connected to a terminal: ext, nc, saddr, unit, fsto, length, mlength, and packid.

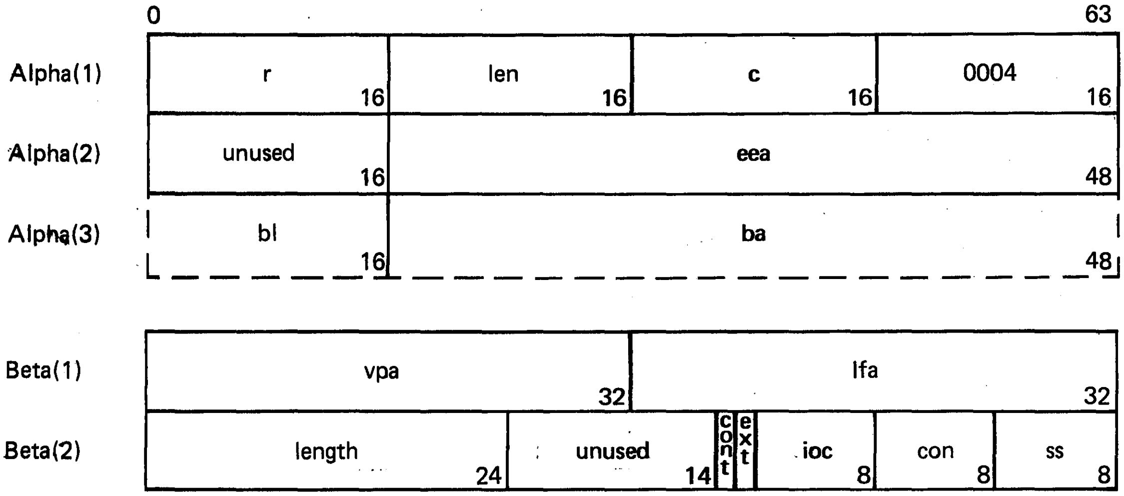
MAP (f=#0004)

The MAP message gives a program access to a virtual region by defining a correspondence of virtual addresses to physical mass storage addresses. The process of defining the virtual region associated with a file is called mapping-in the file. Once a program maps in a file, the program can perform implicit reads and writes on the file. The message might also be used to release (map out) a virtual region by erasing the correspondence of the virtual addresses with mass storage. The map-out operation can also be performed by using the CLOSE FILE message. The mass storage space that is being mapped could contain a file already defined and opened, or it could be space that is not associated with any file (free space).

Before virtual space can be accessed implicitly, the definition of that space must be cataloged in the implicit map area of the program's minus page. The definition can be made using MAP with the map-in option. Up to 40 noncontiguous address regions can be cataloged. The user associates a virtual starting address and length with the mass storage address of an open file or free space and indicates the access rights pertaining to that virtual region. The operating system makes the necessary entries in the bound implicit map (for an open file) or drop file map (for free space) of the program. Overlaps of space are signaled as an error. If all entries of a map are full, an error is signaled and no further map-in calls are permitted until some space is released with a map-out.

The map-out option allows for release of virtual address space. Virtual address space that has been mapped out is no longer accessible to the program, but the mass storage file itself is not closed (the I/O connector for the file remains intact). The mass storage region can, after the map out, be mapped in again to the same or other virtual space. Mapping out free space causes the corresponding drop file map entries to be deleted and frees the mass storage space for reassignment. If the mass storage file represented by a virtual region has write access and is mapped out, all modified pages of that space are written on that mass storage file before the map-out process is complete. If the file itself did not have write access, all modified pages are lost through the map-out process.

The MAP call must not be used with files opened for explicit I/O. Also, source files cannot be mapped in. The format of the message is shown in figure 5-4.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Error has been returned to the ss field of Beta(2). #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. This must be zero when WPA is zero and IOC is #OF to MAPIN SHRLIB. Used by debug.
	c	Map mode: 0 Map-in the file specified in the ioc field. 1 Map-out the file specified in the ioc field (both the bound implicit map and drop file map entries are altered). 2 Map-out only this program's drop file (a c field value of 0 must be used to map-in a drop file).
Alpha(2)	eea	Virtual bit address at which control transfers if an error occurs during processing of this message ($r \neq 0$). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.

Figure 5-4. MAP (f=#0004) Message Format (Sheet 1 of 3)

Word	Field	Description
Beta(1)	vpa	Virtual page address of the first small page of the space defined. Must be zero when len is zero and ioc is #0F to MAPIN SHRLIB.
	lfa	Logical file address associated with the virtual page address. If this field is #FFFFFFFF, free space is appended as defined by the virtual page address and length fields.
Beta(2)	length	Length of the virtual region, in blocks. If this call is not for free space, the space on the mass storage file must be contiguous. When returned to the caller by the operating system, this field is adjusted to the next page multiple.
	cont	File contiguity; set by the operating system after a successful map-in (value is set at creation time). This field is 0 if the file was not created contiguously (in two segments), or set to 1 if the file was created.
	ext	File extendability; set by the user. The values are: 0 Extensions allowed if extensions were not prohibited on creation of the file (same as ext=2). 1 No extensions allowed (same as ext=3). Value that was set at creation time by the operating system after a successful map-in. This field is 0 if extensions were allowed or set to 1 if not allowed. If the file was created with no extensions allowed, it could not have mapped-in with extensions allowed; however, if the file was created with extensions allowed, it can be mapped-in with either extensions allowed or not allowed.
	ioc	Input/output connector number for the mass storage file being mapped (a source file cannot be mapped in): #0-#E, #12-#47 Mass storage file map-in or map-out. #10 Source file map-out. #11 Drop file map-in or map-out. #0F SHRLIB MAPIN: used by debug. VAP and LEN must be zero.
	con	A set of 8 bits providing control information as follows:

c1	c2	c3	c4	c5	wa	ac
1	1	1	1	1	1	2

Subfield	Description
c3	Page map request. 0 Small. 1 Large.

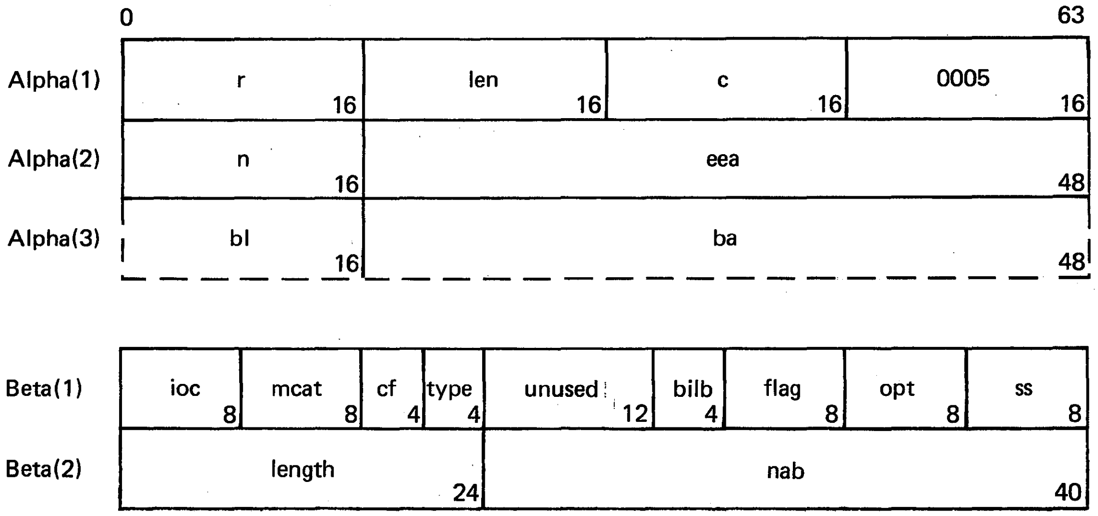
Figure 5-4. MAP (f=#0004) Message Format (Sheet 2 of 3)

Word	Field	Description		
		Subfield		Description
Beta(2)	con	wa		Access:
			0	Get access rights (determined when the file was opened) from the I/O connector.
			1	Get access rights from the ac field if allowed by the ioc access field.
		ac		Access:
			0	No read or write access.
			1	Read access.
			2	Write access.
			34	Both read and write access.
		Bits c1, c2, c4, and c5 are not used. The wa and ac fields are examined by the system when mapping in files associated with I/O connectors 0 through #E.		
	ss	Error response field. The values are:		
			0	Normal completion.
			1	Virtual address overlap of file space.
			2	Cannot map-in file in virtual page 0.
			3	Length field in a map message is 0 or greater than the length in the map.
			4	Length in the request is not modulo page size.
			5	I/O connector does not exist or the mode specified in the I/O connector is not implicit.
			6	Virtual address is the same as that of an existing ADVISE call.
			7	Bound implicit map was full at map-in.
			8	Logical mass storage address plus length exceeds the file length.
			9	Page requested for map-out is locked in.
			#A	Space is undefined at map-out.
			#B	Map entry virtual address is not on a page boundary.
			#C	Bound implicit map is full at map-out.
			#D	I/O connector is not proper for a free space request.
			#E	Drop file map is full at map-out.
			#F	Drop file map is full at map-in.
			#10	Mass storage file index table entry cannot be found.
			#11	Virtual address overlap of free space.
			#12	For a map-in request, no read access was specified; map-in has not been performed.
			#13	File is privileged opened by the user and cannot be mapped.
			#14	Logical file address overlap.
			#15	Logical file address plus length exceeds user or pool maximum.
			#16	Error in extending file.
			#17	No more disk space available when extending the file.

Figure 5-4. MAP (f=#0004) Message Format (Sheet 3 of 3)

CLOSE FILE (f=#0005)

This message terminates immediate access to the data. System operation varies slightly, depending on the medium. Only one Beta is processed for each Alpha. The format of the CLOSE FILE message is shown in figure 5-5.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Error code was returned in an ss field of Beta(1). 2 User is not privileged (privileged closes only). #211 Number of files specified in this call is illegal (the n field is 0 or greater than 16). #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified. #270 Beta portion is in read-only space.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. This field must be a multiple of 3 for a regular close issued by a privileged system task (c=2); this field must be a multiple of 2 for other regular closes (c=0). For a privileged close (c=1), Beta must have a length of at least two words per file closed.
	c	Close mode: 0 Regular close. 1 Privileged close. 2 Regular close by a privileged task.

Figure 5-5. CLOSE FILE (f=#0005) Message Format (Sheet 1 of 6)

Word	Field	Description
Alpha(2)	n	Number of files closed by this message; maximum is one.
	eea	Virtual bit address to receive control if an error occurs during processing of this message ($r \neq 0$). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.
Beta(1)	ioc	Input/output connector number of the file being closed.
	mcat	File management category of the file being closed; stored in the file index table if the flag field is 2. The categories are: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as write-temporary file). 5 User-created drop file. 7 Batch file.
	cf	A set of four control bits, as follows:

c1	c2	c3	c4
1	1	1	1

Bit	Description
c1	File type: 0 Do not change the file type in the file index table. 1 Change the type in the file index table to the value given in the type field.
c2	Unused.
c3	Drop file size: 0 Do not change the drop file size. 1 Change the drop file size in the file index table to that given in the length field.
c4	Drop file length: 0 Do not remove the drop file length from the file index table. 1 Remove the drop file length from the file index table.

These flags can cause changes to be made in the file index table if the file ownership is private; pool, and the user is the pool boss; or public, and the user is privileged. This field must be all zeros for privileged closes (the ss field is A otherwise).

Figure 5-5. CLOSE FILE (f=#0005) Message Format (Sheet 2 of 6)

Word	Field	Description
Beta(1)	type	File type: 0 Physical data file. 1 Virtual data file. 2 Virtual code file.
	bilb	Bits used in last byte: 0 All bits in last byte are used. 1-7 From 1 to 7 bits in last byte are used.
	flag	Flag for special action, as follows:

f1	f2	f3	f4	f5	f6	f7	f8

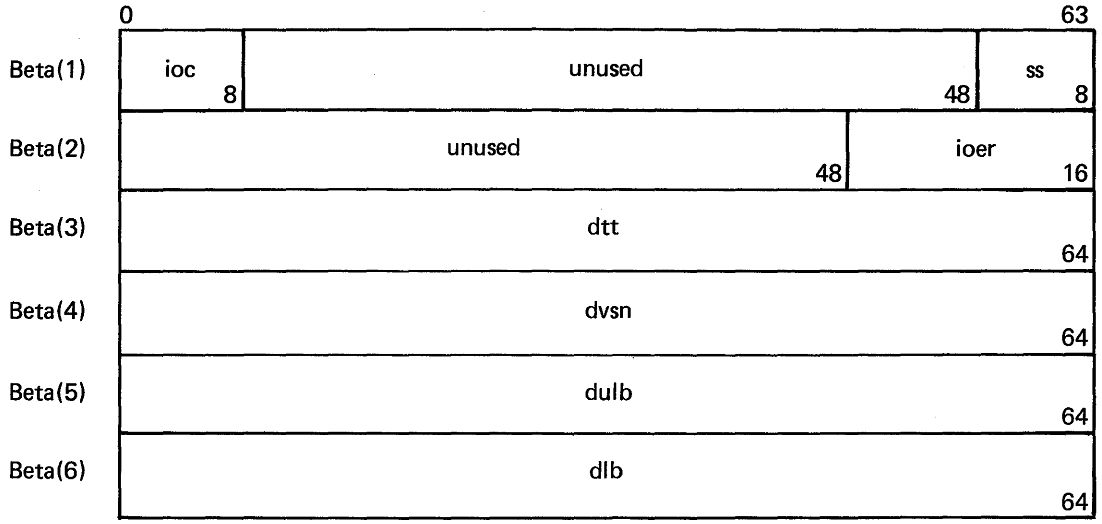
Bit	Description
f1-f3	Reserved.
f4	bilb field: 0 Do not set the bilb field in the file index table. 1 Set the bilb field in the file index table to the value given in bilb in Beta(1).
f5	dmp flag: 0 Do not change the dmp bit in the file index table. 1 Set the dmp bit in the file index table.
f6	nab fields: 0 Do not change the nab fields in the file index table. 1 Change the nab fields in the file index table according to the nab value supplied in Beta(2).
f7	mcat fields: 0 Do not change the mcat field in the file index table. 1 Change the file index table´s field management category to that given in the mcat Beta field. If mcat is changed to drop file and the caller is a production user number, the drop file is given production status and all write access permissions are removed from the file.
f8	Reserved.

Figure 5-5. CLOSE FILE (f=#0005) Message Format (Sheet 3 of 6)

Word	Field	Description
	opt	Option field; for a privileged close only. If this field is 0, the file is not to be destroyed with this call; or, if it is 1, the file is to be destroyed with this call.
Beta(1)	ss	Error response code. The values are: 0 Normal completion. 1 I/O connector was not for a mass storage file. 2 I/O connector number was out of range. 3 Attempt to alter a public file index table entry; the I/O connector is cleared, but no information is altered in the file index table (file is closed). 4 File type, access right, or lockout specified by this request is illegal (file is closed). 5 Close was requested for a file, one of whose pages is still locked in. 6 A scratch or output file is open to another program of this user (file is closed to this problem program but not destroyed or given). 7 Invalid name for a file with a management category of output (file is closed). 8 Specified I/O connector was not open. 9 Drop file map is full. #A Format error (privileged close only). #B Nonprivileged user (privileged close only). #C File was not found or not attached. #E Supplied nab value is inconsistent with the system-maintained value. (File is closed.) #F One file close per Alpha. #10 Not all VSNs were returned in the VSN array; CLOSE completed. #11 Label buffer too short; CLOSE completed. #12 Not all entries are returned in the tapes table array; CLOSE completed. #13 Illegal label in the buffer label. #14 lun field in FILEI or IOC is invalid. #15 db in tapes table does not match db of task.
Beta(2)	length	Length of the drop file in blocks; set in the file index table if the c3 field is 1.
	nab	Relative bit address, supplied by the caller, of the next byte to be written in the file. If the supplied value does not correspond to that maintained by the system at the block level, this value is ignored, the lbc/hbw field in the FILEI is set to indicate that the last block is full, and a warning error is returned.

Figure 5-5. CLOSE FILE (f=#0005) Message Format (Sheet 4 of 6)

For tape files:



Word	Field	Description
Beta(1)	ioc	The files's input/output connector number.
	ss	Error response field:

0	No errors.
#F	Only one file close per Alpha allowed.
#10	Not all VSNs were returned in the VSN array; CLOSE completed.
#11	Label buffer too short; CLOSE completed.
#12	Not all entries are returned in the tapes table array; CLOSE completed.
#13	Illegal label in buffer label.

Beta(2)	ioer	Error number. The r field in Alpha is set to 3 if ioer is nonzero. Refer to appendix B for a complete description of the ioer error numbers.
Beta(3)	dtt	Tapes table descriptor. If nonzero, the tapes table entry is returned by the system:

Bit	Name	Description
0-15	ltt	Length of the tapes table buffer, in words. The buffer must be 12 words long.
16-63	att	Virtual bit address of the tapes table buffer. The buffer must be on a word boundary.

Beta(4)	dvsn	Descriptor for the VSN list. If nonzero, the VSN list is returned by the system:

Bit	Name	Description
0-15	lvsn	Length of the VSN list, in words (0 < lvsn < 256).
16-63	avsn	Virtual bit address of the VSN list. The buffer must be on a word boundary.

Figure 5-5. CLOSE FILE (f=#0005) Message Format (Sheet 5 of 6)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>Beta(5)</td><td>dulb</td><td>Descriptor for the user trailer label buffer. If dulb is nonzero, the user trailer labels are supplied by the user. This field applies only when writing labels:
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0-15</td><td>lulb</td><td>Length of the user label buffer, in words.</td></tr>
<tr><td>16-63</td><td>aulb</td><td>Virtual bit address of the user label buffer. The buffer must begin on a word boundary.</td></tr>
</table></td></tr>
<tr><td>Beta(6)</td><td>dlb</td><td>Label buffer descriptor. If dlb is nonzero, the system returns the end-of-file labels here:
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0-15</td><td>llb</td><td>Length of the label buffer, in words.</td></tr>
<tr><td>16-63</td><td>alb</td><td>Virtual bit address of the label buffer. The label buffer must be on a word boundary.</td></tr>
</table></td></tr>
</table>

Figure 5-5. CLOSE FILE (f=#0005) Message Format (Sheet 6 of 6)

Mass Storage Files

A program can issue the CLOSE FILE message to sever its connection to a file. After the file has been closed, the program no longer has access to the file through the severed connection, although other unsevered I/O connections might remain. Existence of the mass storage file is not affected by a close, but some file attributes in the file index table entry for the file are modified, and virtual address space associated with an implicit file is no longer defined. A file that has been privileged created or privileged opened can be closed only with a privileged close. The user must do a privileged close before doing a privileged destroy.

When a file is closed, the operating system gives the file to an output processor if the activity count (the count of programs accessing the file, that is, of I/O connectors for the file) is 0 and the management category is output. Other ways of outputting a file are to use the FILE DISPOSITION message (f=#000D) or the GIVE FILE message (f=#0008).

When a file opened for implicit I/O and with write access is closed, modified pages of the file are rewritten in mass storage before the close function has completed. If the file does not have write access, modified pages are lost at the time the close function completes.

All outstanding input/output requests are completed before any file index table changes are made. The file index table entry will exist in its new state only at the completion of CLOSE FILE message processing.

Magnetic Tape Files

The CLOSE FILE message can be issued only for a logical tape file that is open. There can be only one tape file specified in the Beta for each CLOSE message. After the successful completion of the CLOSE, the ioc is cleared and no input/output or positioning functions can be issued until a subsequent OPEN is issued. The CLOSE does not return the logical tape file.

Files Connected to a Terminal

The CLOSE FILE message is used to relinquish access to a file connected to a terminal. Existence or contents of the FILEI entry is not affected by a close.

TERMINATE (f=#0006)

A user program can issue a TERMINATE message to signal the operating system that it has completed execution. All lower level controllees are also terminated. The message consists of an Alpha portion only, as shown in figure 5-6.

	0			63
Alpha(1)	unused 16	0000 16	c 16	0006 16
Alpha(2)	rc 8	unused 8	resume 48	

Word	Field	Description
Alpha(1)	c	Indicates disposition of the drop file when the program is removed from main memory. The values are: 0 Drop file, scratch files, and output-type files are preserved (but not made permanent) so the program can be restarted. All modified pages belonging to write access files overwrite their current disk images and all other modified pages are written to the drop file. The resume address is stored in the drop file´s minus page. 1 Drop file and scratch files are destroyed. Give output-type files to the output processor. 2 Same as option 0 except the terminate state is set to report an abort (#3D).
Alpha(2)	rc	Return code, set by the user. The values are: 0 Successful completion. 4 Nonfatal error. 8 Fatal error. This return code value is also referred to as the termination value.
	resume	Virtual bit address at which the program is to be resumed when it is restarted.

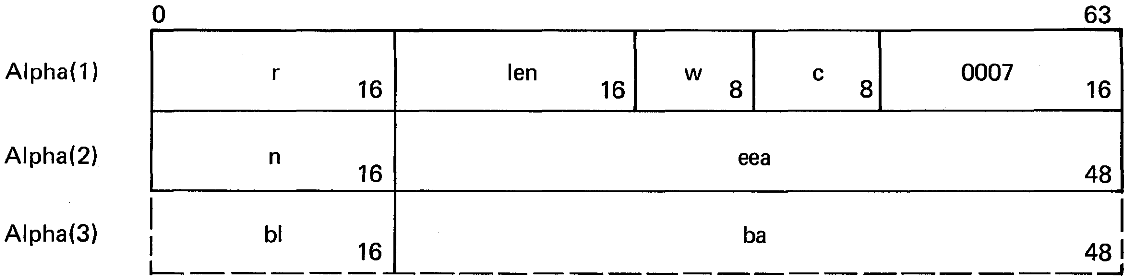
Figure 5-6. TERMINATE (f=#0006) Message Format

LIST FILE INDEX TABLE (f=#0007)

The LIST FILE INDEX TABLE system message can retrieve copies of one or more file index table entries. The message issuer can specify the file ownership category and file attributes of the entries to be returned.

The file index entries are returned in the Beta portion of the message. The Beta length must be a multiple of the file index entry length (refer to figure 2-1). Qualifiers for the file index table search are specified only in the first entry length of the Beta.

The file index table entry format is shown in figure 2-1. The message format is shown in figure 5-7.



Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing completes: 0 No errors. 1 No files exist which match the options and qualifiers specified. 2 More file entries exist than can fit in the Beta portion. At least one more file entry exists for the owner, but the file index entry has not been checked for matching qualifying fields. 3 Caller is not a member of a pool or the pool is not attached. 4 Illegal message option (c field). 5 Incorrect file index entry length (w field). 6 User table is full. 7 FILEI is full. 8 Incorrect user number. 9 Invalid account number. #214 Beta length error. Either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	Total Beta length. If len is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta buffer; otherwise, the Beta buffer is assumed to begin at Alpha(3). The length must be a multiple of the file index entry length.

Figure 5-7. LIST FILE INDEX TABLE (f=#0007) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	w	Current file index entry length (refer to figure 2-1). If an incorrect value is specified, a response code of 5 is returned and the system returns the correct value in this field.
	c	Message option specifying the file index information to be returned: 0 Public file entries. 1 Pool file entries. 2 Private file entries. 3 Entries for private files attached to this job. 4 Entries found according to the file search hierarchy. The first search is for a private file attached to this job. If no file is found, the second search is for a pool file. If no file is found again, the third search is for a public file. 5 File owner's access permissions, whether the file is attached or not. 6 This option is the same as option 4, except that an additional word containing the third word of the file index entry is returned for each file. The format of the entries returned will parallel the actual format of the FILEI with the 3 word top entry returned first, followed by the 14 word bottom entry.
Alpha(2)	n	Maximum number of file index entries to return. If fewer than n qualifying file entries are found, this field is reset to the number of the file entries found.
	eea	Error exit address; virtual bit address to receive control if an error occurs during message processing (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the len value is #FFFF indicating the Alpha and Beta portions of the message are not contiguous, these fields give the length and virual bit address of the first full word of the Beta portion.
Beta		The Beta format is the same as the file index table format, except that the third word of the top is not returned (refer to figure 2-1). The Beta buffer length must be at least the file index table entry length multiplied by the number of entries to be returned (n). Values specified in the first entry of the Beta buffer are used as qualifiers in the file index table search. A pool name must be specified for option c=1; all other qualifiers are optional.

Figure 5-7. LIST FILE INDEX TABLE (f=#0007) Message Format (Sheet 2 of 3)

The following qualifiers can be used with any option (c=0, 1, 2, 3, or 4). Refer to figure 2-1 for the field format.

Qualifier	Description
name	File name.
qf	Queue flag, indicating whether the file has been read by the IQM.
mcat	Management category.
fidc	Disposition code.
fiic	Internal format characteristics.
fiec	External format characteristics.
fisid	Source or destination processor mainframe identifier.
fizip	Destination processor zip number.
mpn	Master project number.
acct	Account number.

The following qualifier can be used with options c=2 or 3.

Qualifier	Description
user	Owner´s user number. When specified, the owner´s file index entries are returned for files to which the caller has access. The oacs field in the file index table entry is set to the largest set of access permissions as determined from the general access permissions and the caller´s individual access permissions.

If within the Beta portion, a field in the first file index entry length is nonzero, the specified value is used as a qualifier. Zero cannot be used as a qualifier. Only those file index table entries with field values matching the specified qualifiers are returned.

Figure 5-7. LIST FILE INDEX TABLE (f=#0007) Message Format (Sheet 3 of 3)

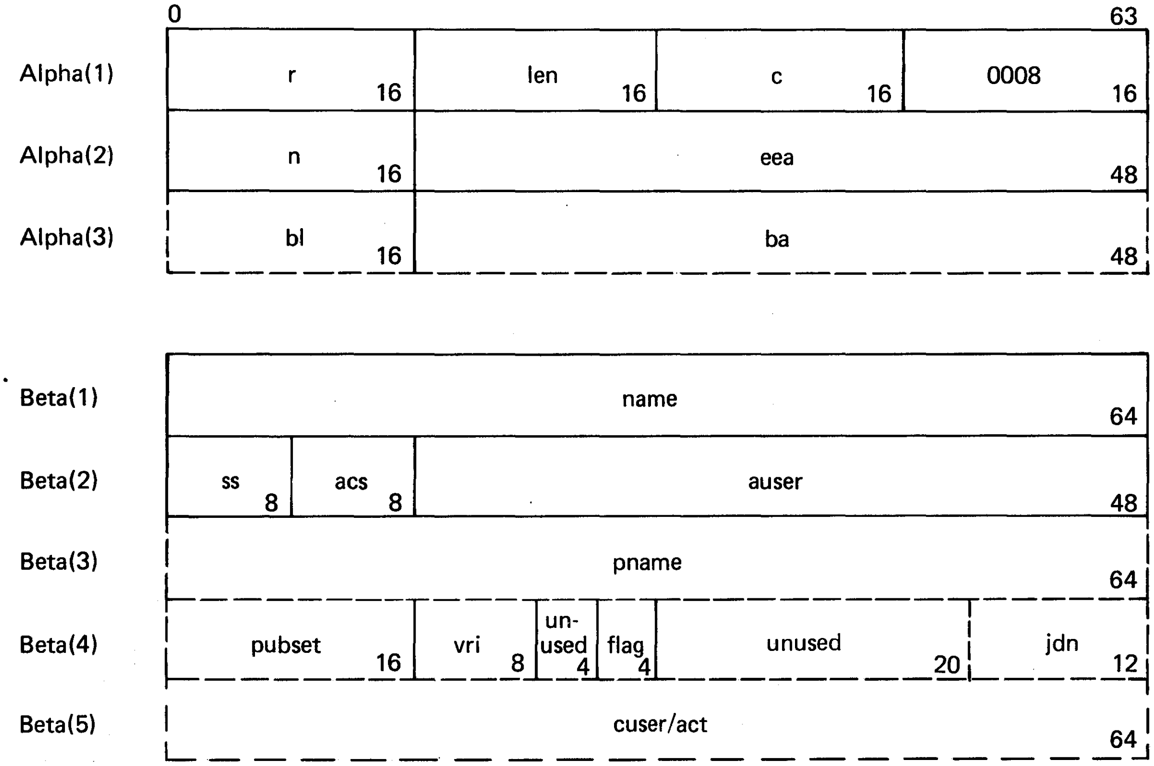
GIVE FILE (f=#0008)

The GIVE FILE system message transfers file ownership. If the GIVE mode is 0 or 1, the file must be an attached permanent file, a local file, or an attached private pool file. (Only the pool boss can give pool file.) If the give mode is 2, the file must be a private, unattached permanent file. (Only privileged users can use GIVE mode 2.)

A nonprivileged user can give a file to another user or to a pool. A privileged user can also give a file to the public file list. (The file must not have the same name as an existing public file.) The privileged system task can give a file to the IQM and the IQM can give a file to a user.

Whenever a file is given to another user, the dmp flag in the file index table is cleared to indicate that the attribute of the file has been changed.

The message format is shown in figure 5-8. More than one 2- to 4-word Beta portions can be specified. Only one Beta is processed for each Alpha.



Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 Error code returned as ss field of Beta(2). #211 Number of files given was 0 or greater than 16. #214 Beta buffer length error. Either the first word address of Beta plus its length is greater than the maximum virtual user address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-8. GIVE FILE (f=#0008) Message Format (Sheet 1 of 4)

Word	Field	Description

Alpha(1)	len	Total Beta length. If len is #FFFF, Alpha(3) contains the length and virtual bit address of Beta; otherwise, Beta is assumed to begin at Alpha(3). A Beta portion must be specified for each file to be given. The length of each portion depends on the give mode as follows:
		c=0 Two words to give a file to a user; four words to give a file to the public file list.
		c=1 Four words.
		c=2 Five words.
		c=3 Four words.

	c	Give mode: 0 The file is given to the private user number specified in Beta(2) or, if the specified user number is 0 and the issuer is privileged, the file is given to the public file list. 1 The file is given to the pool specified in Beta(3). 2 The file whose user number is cuser is given to the user number specified by auser. The caller must be a privileged user to use this mode. 3 The local file that is the last group file of an output file family is given to the output queue and a JDN is associated with the last group file.
Alpha(2)	n	Number of files to be given; maximum is 16.
	eea	Error exit address; virtual bit address to receive control if an error occurs during message processing (r≠0). If this field is 0 when an error occurs, the task is aborted.
Alpha(3)	bl, ba	If the len value is #FFFF (indicating the Alpha and Beta portions of the message are not contiguous) these fields give the length of Beta and the virtual bit address of its first full word.
Beta(1)	name	File name (eight ASCII characters, left-justified, blank-filled).
Beta(2)	ss	Error response code: 0 No errors; normal completion. 1 File recipient already has a permanent file with this name. 3 The specified file is not attached. 4 The specified user number does not exist. 5 Output file is incorrectly named. 6 File to be given is still active. 7 User is not privileged. 8 Failure in modifying PFI. 9 File recipient has a security classification less than that of the file. #A Either the specified pool does not exist, or the giver does not have access to the pool. #B cuser is not a valid user number. #C Not used. #D Permanent file space limit exceeded for new power.

Figure 5-8. GIVE FILE (f=#0008) Message Format (Sheet 2 of 4)

<table>
<tr><th>Word</th><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Beta(2)</td><td>ss</td><td>#E</td><td>No space available in system tables FILEI or UDMINI.</td></tr>
<tr><td></td><td></td><td>#F</td><td>Issuer specified nonzero vri field, but the site does not use variable rate accounting.</td></tr>
<tr><td></td><td></td><td>#10</td><td>Issuer attempted to set a variable rate index (vri field) for a data file being given to the public file list. Only a code file can have a variable rate index set.</td></tr>
<tr><td></td><td></td><td>#11</td><td>Issuer attempted to give a file belonging to a pool for which she/he is not the pool boss.</td></tr>
<tr><td></td><td></td><td>#12</td><td>File is a magnetic tape file.</td></tr>
<tr><td></td><td></td><td>#13</td><td>File recipient already has the maximum number of files she/he can own.</td></tr>
<tr><td></td><td></td><td>#14</td><td>IQM user number specified.</td></tr>
<tr><td></td><td></td><td>#16</td><td>Unable to give attached permanent file (if cuser is used).</td></tr>
<tr><td></td><td></td><td>#17</td><td>Cannot give connected file.</td></tr>
<tr><td></td><td></td><td>#18</td><td>Caller is not the owner of the file.</td></tr>
<tr><td></td><td></td><td>#19</td><td>File size exceeds limits for user.</td></tr>
<tr><td></td><td></td><td>#1A</td><td>No JDNs available to assign to file (c=3 and jdn=0).</td></tr>
<tr><td></td><td></td><td>#1B</td><td>Task is not privileged (c=3 and jdn=0).</td></tr>
<tr><td></td><td></td><td>#1C</td><td>JDN specified is not caller´s (c=3 and jdn .ne. 0).</td></tr>
<tr><td></td><td></td><td>#1D</td><td>Destination user number is not an output spooler (c=3).</td></tr>
<tr><td></td><td>acs</td><td colspan="2">File access permissions. This 8-bit field is treated as eight 1-bit fields, with each bit specifying the associated permission:
<table>
<tr><th>Bit</th><th>Hexadecimal Value</th><th>Description</th></tr>
<tr><td>1-3</td><td>-</td><td>Unused.</td></tr>
<tr><td>4</td><td>10</td><td>Execute access permitted.</td></tr>
<tr><td>5</td><td>8</td><td>Modify access permitted.</td></tr>
<tr><td>6</td><td>4</td><td>Append access permitted.</td></tr>
<tr><td>7</td><td>2</td><td>Read access permitted.</td></tr>
<tr><td>8</td><td>1</td><td>Write access permitted.</td></tr>
</table></td></tr>
<tr><td></td><td>auser</td><td colspan="2">User number (six ASCII characters, left-justified, blank-filled). For c=0 or 1, auser is the user number of the file recipient. If the issuer is privileged and the auser and c fields are 0 and the len field is 4, the file is given to the public file list. When auser is nonzero, the len field must be 2 or 3.</td></tr>
<tr><td>Beta(3)</td><td>pname</td><td colspan="2">Pool name (eight ASCII characters, left-justified, blank-filled). This field specifies the pool to which the file is given when c=1.</td></tr>
<tr><td>Beta(4)</td><td>pubset</td><td colspan="2">Reserved for public file sets.</td></tr>
<tr><td></td><td>vri</td><td colspan="2">Variable rate index set in the descriptor block (refer to Variable Rate Accounting in chapter 8). This field is only specified for virtual code files.</td></tr>
</table>

Figure 5-8. GIVE FILE (f=#0008) Message Format (Sheet 3 of 4)

Word	Field	Description
Beta(4)	flag	Other operations to be performed when giving the file to the public file list. This field is used only when the auser and c fields are both 0: 1 Clear the originating user field in the file index table. 2 Clear the originating user field in the file index table. 4 The file is given to the specified pool; if c=1, this flag also causes the file to become privileged. 5 Do both 1 and 4. 6 The file is given to the specified pool; if c=1, this flag also causes the file to become privileged. 7 Do both 1 and 4.
	jdn	Job descriptor number (binary, 1 through 2047). Used only if c=3.
Beta(5)	cuser/acct	The user number to which this file currently belongs. If c=2, indicates that a file is to be given to the user number specified by auser.

Figure 5-8. GIVE FILE (f=#0008) Message Format (Sheet 4 of 4)

The effect of the access parameter is determined by the current ownership and the resulting ownership of the file.

If the current ownership is private and the resulting ownership is private, acs establishes the new owner´s access permissions. If acs=0, the new owner´s permissions will be the same as the previous owner had prior to the give.

If the ownership goes from private to pool, acs establishes the access permissions that all pool members will have, including the pool boss. The default is the access permission the owner had prior to the give.

If the ownership goes from private to public, acs establishes the access permissions all users will have to the public file. The default is read and execute access permissions.

If the ownership goes from pool to private, acs establishes the access permissions the new owner will have. The default is the access permissions the pool boss had prior to the give.

If the ownership goes from pool to pool, acs establishes the access permissions that all pool members of the receiving pool will have, including its pool boss. The default is that the new pool boss will have the access permissions the old pool boss had and the general access permissions are retained.

If the ownership goes from pool to public, acs establishes the access permissions all users will have to the public file. The default is read and execute access permissions.

LIST SYSTEM TABLE (f=#0009)

With this message, a user can retrieve a formatted copy of part or all of certain system tables.

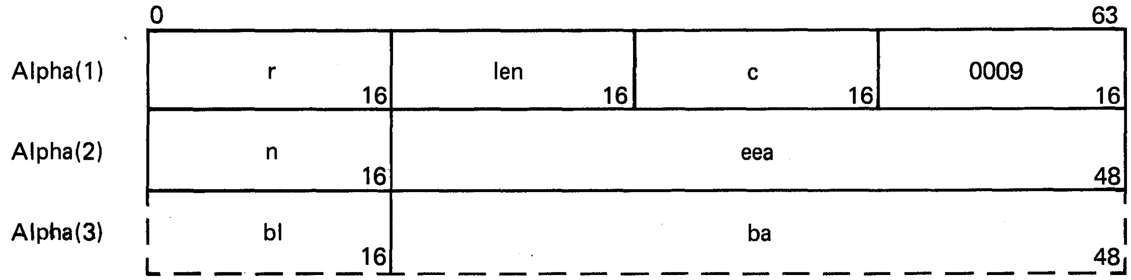
For option 9, two word entries (one from the top and one from the bottom) are listed sequentially in the Beta area. The operating system moves entries from the disk status table to the Beta area until either the table or the Beta area is exhausted. The number of entries transferred is returned in the n field of the Alpha portion of the message.

The Beta format for option #10 of this message is also used by the privileged options of the CREATE FILE and OPEN FILE messages.

For c field value of #10, the number of Beta words returned for each file entry is specified by the quantity len divided by n. For example, if 4 files (n=4) were to be listed and the len field is 16, only the first 4 Beta words of information for each file would be returned. To get all 16 words of information for each file would require that n=4 and the len field be at least 64.

The format of the LIST SYSTEM TABLE message is shown in figure 5-9. (For some of the message options, the Beta portion of the message can consist of more than one of the multiword sets shown in the figure.)

For a c field value of #10, the user can set up more than one Beta word group, and the operating system returns the same number of groups as the user set up. If the file was not found, the file name field of that group is zeroed. If the file name field of the first group is 0, the system returns all file index extension entries for those files which have extensions.



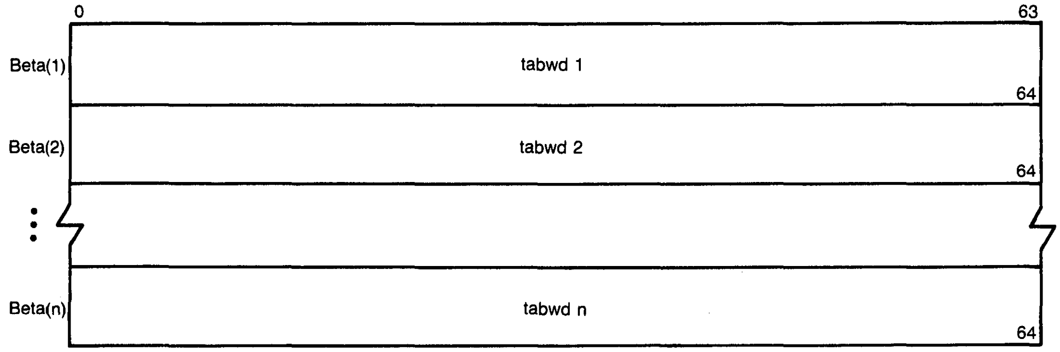
Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Illegal message option. 2 Nonfatal; more files exists than fit in the Beta buffer. At least one more file exists for the owner but this file has not been checked for valid qualifiers (for option #10 only). #214 Beta buffer length error. Either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-9. LIST SYSTEM TABLE (f=#0009) Message Format (Sheet 1 of 6)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha (3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the total length in words of the Beta portion. For option #10, the len field should be at least 16n. For options 2 through 5 and option 9, this field specifies the number of Beta words available for the requested system table (refer to the c field).
	c	Message option field, specifying the system table information that the operating system is to return in the Beta portion. The values are: 2 Timecard buffer; a Beta length of at least 512 is required. 3 Statistics buffer; a Beta length of at least 100 is required. 4 Bank update table; a Beta length of at least 32 is required. 5 Miscellaneous table; a Beta length of at least 104 is required. 9 Disk status table; a Beta length of at least 32 is required. If more than the required number of words are specified to the bl field, the operating system resets the bl field to 32. (The system does not reset the bl field of option 2, 3, 4, or 5.) #F Job category table entries; a Beta length of 198 is required. #10 File index table extension entries for all private files with extensions, or for individually specified private files.
Alpha(2)	n	If the c field value is #10, n is the number of files to be listed; the quantity len divided by n specifies the number of words returned per file index table entry. For the other options, n is the size of the table to be listed (in words), and the Beta area should be at least n words long. The operating system moves words from the table into the Beta area until either the table or the Beta area is exhausted. The value of n must always be greater than 0. For a c field value of #10, if the operating system finds fewer than n files to list, it resets n to the number of files found.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion. For a c field value of 9, the operating system sets the bl field to the length of the table.

Figure 5-9. LIST SYSTEM TABLE (f=#0009) Message Format (Sheet 2 of 6)

For message options 2, 3, 4, 5, and #F:



Word	Field	Description
Beta(1) through Beta(n)	tabwd i	One word of an n-word table; returned by the operating system when the c field is 2, 3, 4, 5, or #F.

For message option #9:

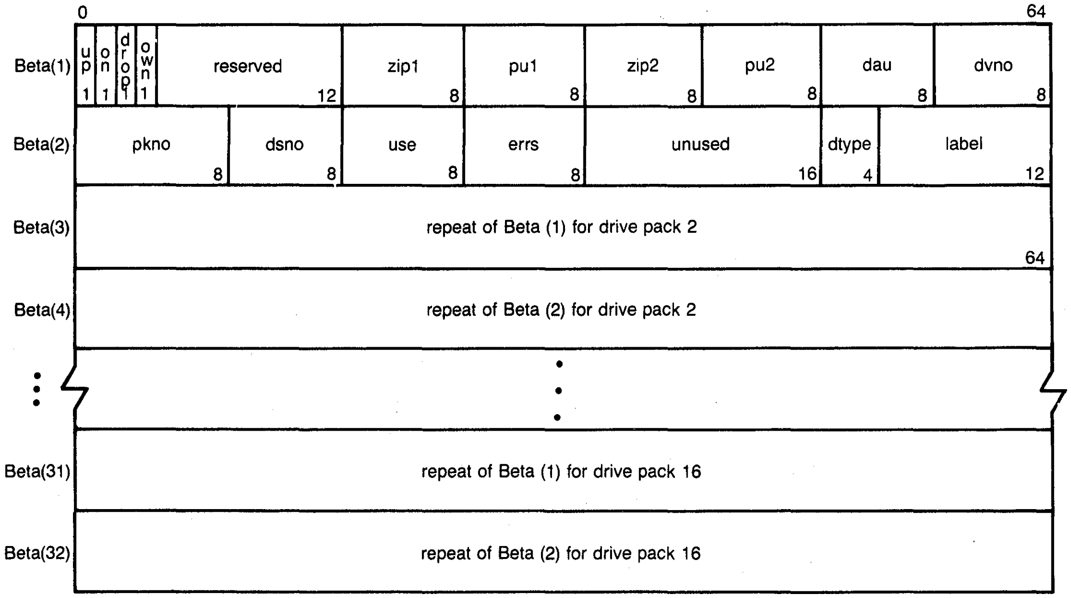
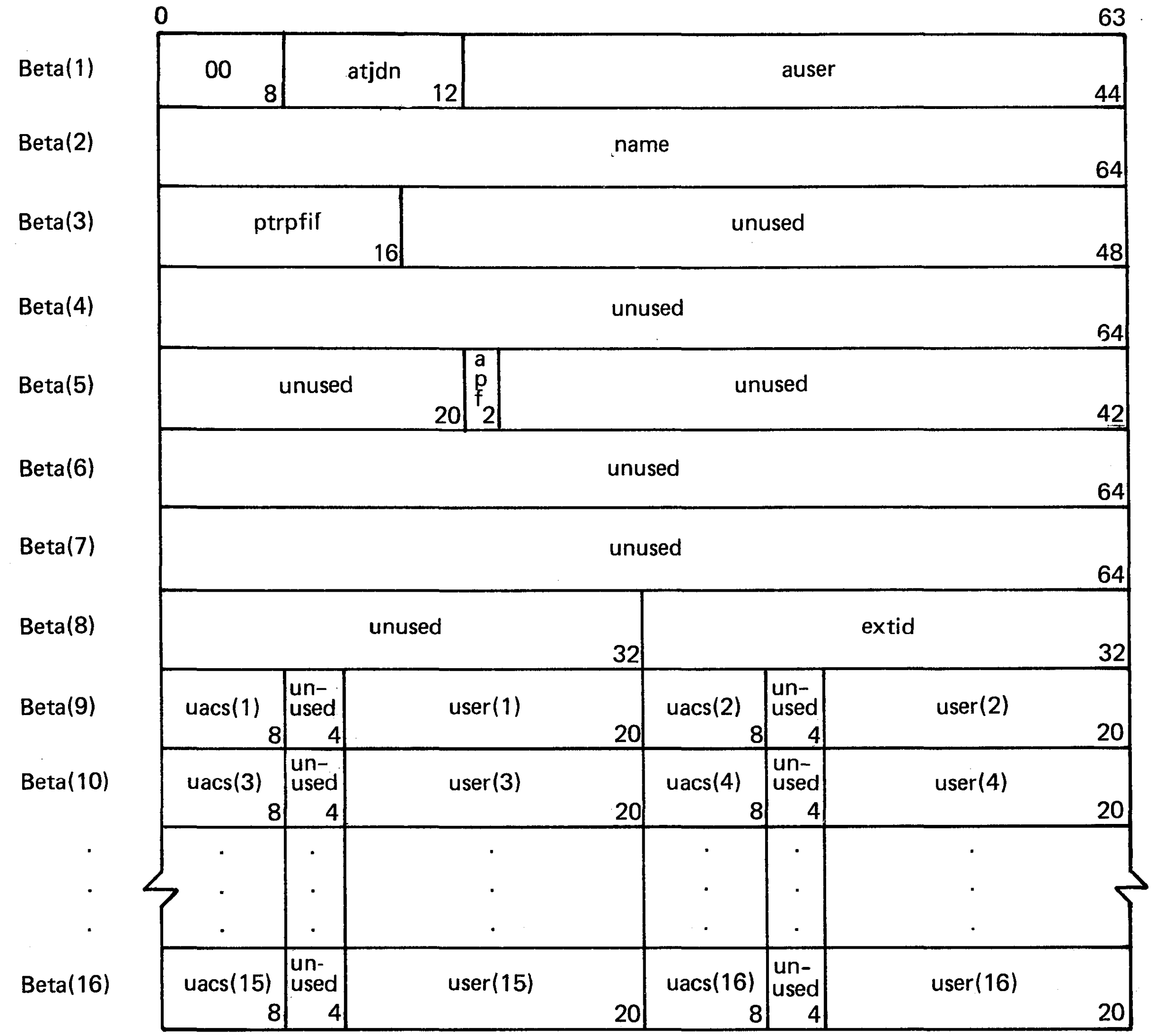


Figure 5-9. LIST SYSTEM TABLE (f=#0009) Message Format (Sheet 3 of 6)

Word	Field	Description
Beta(1)	up	Status. The operating system sets this field to one of the following values: 0 Down. 1 Up.
	on	Usage. The operating system sets this field to one of the following values: 0 Off. 1 On.
	drop	Drop file. The operating system sets this field to one of the following values: 0 Drop files disallowed. 1 Drop files allowed.
	own	Ownership. The operating system sets this field to one of the following values: 0 System ownership. 1 Private ownership.
	zip1	Primary zip.
	pu1	Primary physical unit number.
	zip2	Secondary zip.
	pu2	Secondary physical unit number.
	dau	Disk allocation unit.
	dvno	Device number associated with this disc device.
Beta(2)	pkno	Pack number written in pack label.
	dsno	Device set number of which this pack is a member.
	use	Percent of disk space in use (0-100).
	errs	Fatal disk errors since autoload.
	dtype	Device type of this pack.
	label	Address of label of this pack.

Figure 5-9. LIST SYSTEM TABLE (f=#0009) Message Format (Sheet 4 of 6)

For message option #10:



Word	Field	Description
Beta(1)	atjdn	The job descriptor number (1 through 2047): =0 File is not attached. ≠0 File is attached; atjdn contains the job descriptor to which the file is attached.
	auser	The binary user number issuing the request; returned by the operating system.

Figure 5-9. LIST SYSTEM TABLE (f=#0009) Message Format (Sheet 5 of 6)

Word	Field	Description
Beta(2)	name	The name field of each Beta entry supplied can contain the ASCII name of a file for which information is to be returned. The operating system places a zero here if the name given is not found or does not have a file index extension entry.
Beta(3)	ptrpfil	Pointer to the proper block of the PFI for this entry, relative to the first block of the PFI.
Beta(5)	apf	Access permission flags: Bit 20=1 File index table extension. Bit 21=0 Used in file index table entry.
Beta(8)	extid	Extension identifier used to match file index table entries when duplicate files occur.
Beta(9) through Beta(16)	uacs(i)	Individual user access permission for the user specified in the user(i) field: B1 | B2 | B3 | B4 | B5 | B6 | B7 | B8 Bit — Description 1-3 Reserved (ignored by the system). 4 Execute access permitted. 5 Modify access permitted. 6 Append access permitted. 7 Read access permitted. 8 Write access permitted.
	user(1)	Binary user number of a user whose access permission is defined by uacs(i). An entry of 0 indicates the end of the list.

Figure 5-9. LIST SYSTEM TABLE (f=#0009) Message Format (Sheet 6 of 6)

CHANGE FILE ATTRIBUTES (f=#000B)

This message allows a user program to change various attributes of an existing local file, an attached permanent file, or a tape file. Nonprivileged users may change the file name, account, master project number, or retention period of their own files. Privileged users may also change the account number of another user's file, and they can change ostat, the output file status. The site security administrator user number can clear the drop file restart flag so that the drop file may be restarted. For tape files, only SIL file attributes may be changed. Whenever a change in file attributes occurs, the dump flag in the file index table is cleared to indicate the file has been modified. The format of the message is shown in figure 5-10.

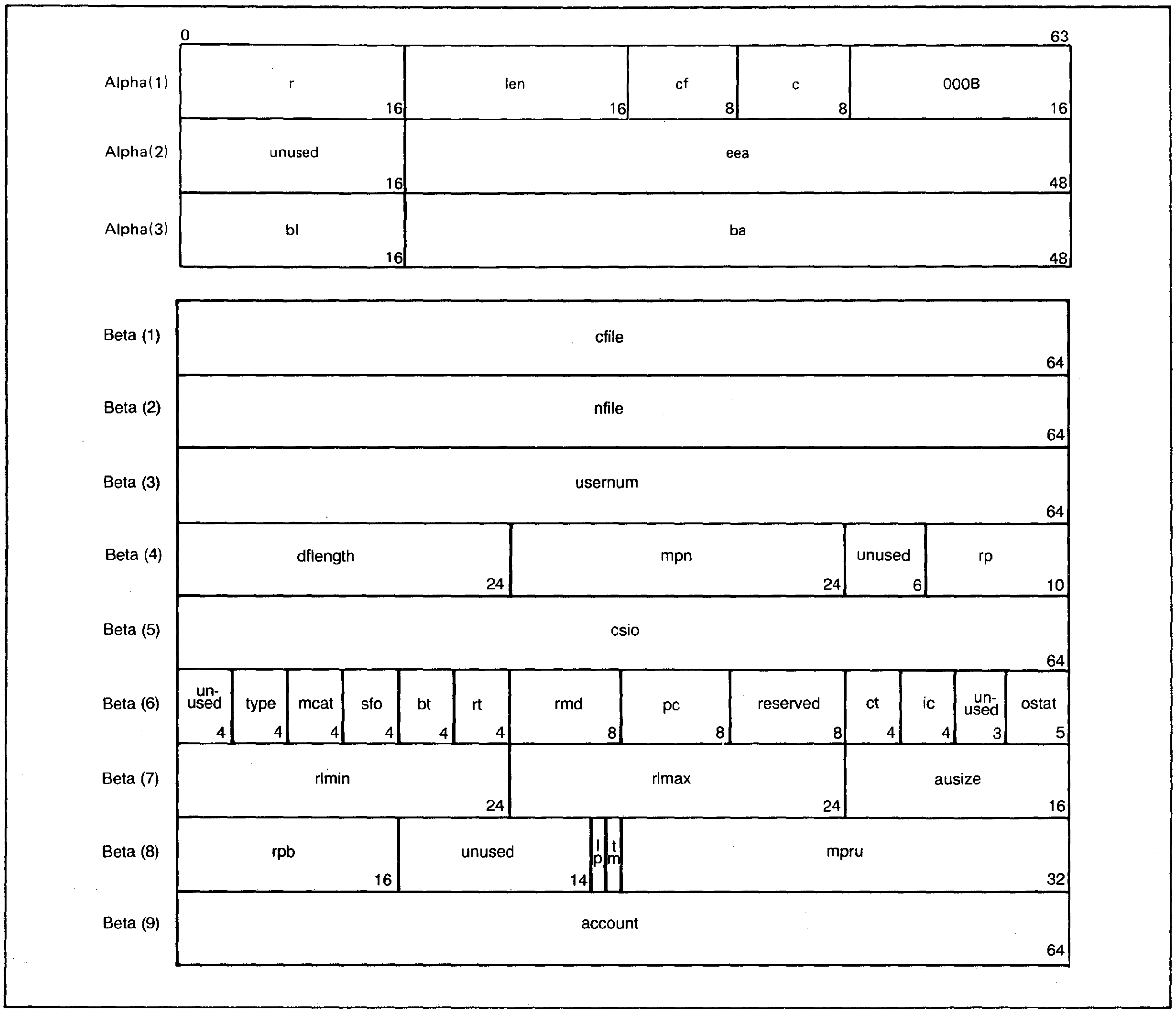


Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 1 of 8)

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Current file name is still active. 2 Current file name does not exist or is not attached. 3 New file already exists. 4 New account number is not valid. 5 New file name is invalid. 6 User is not privileged, although the len field is 3. 7 Invalid c field value. 8 Duplicate permanent file name. 9 Not a level-1 task (a level-1 task is one with no controller). #A Caller is not owner of the file. #C Illegal type code. #D Unable to change characteristics of a connected file. #E For virtual code file only. #F Illegal management category. #10 No match found in ioc for the file name given in cfile. #11 Illegal class code. #12 Illegal file organization code. #13 Illegal blocking type code. #14 Illegal record type code. #15 Must have write access to cfile in order to change hba. #16 Illegal attribute for a tape file. #17 Invalid ct value. #18 Invalid ic value. #19 Illegal record type change for a direct access file. #20 Illegal ostat field. #21 Unable to change ostat field in the FILEI. #23 Invalid account. #24 Illegal master project number. #27 Caller is not the site security administrator user number. #28 File is not a drop file. #29 No user table entry available or the FILEI is full. #30 Undefined user number. #31 Nonprivileged caller. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the message; otherwise, Beta is assumed to begin at Alpha(3), and len must be 9.

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 2 of 8)

Word	Field	Description
Alpha(1)	cf	File attribute change options for mass storage files only. Control bits are represented as follows:

cf8	cf7	un-used	cf5	cf4	cf3	cf2	cf1
1	1	1	1	1	1	1	1

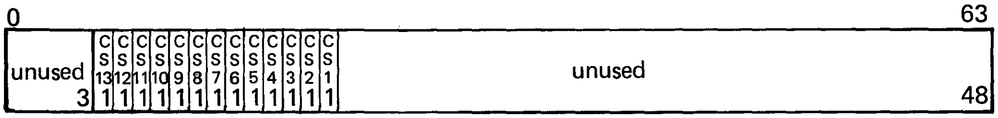
Subfield	Description
cf8	Internal characteristics: 0 Do not change internal characteristics. 1 Change internal characteristics.
cf7	Communication type: 0 Do not change communication type. 1 Change communication type.
cf5	Byte address (hba): 0 Do not reset hba. 1 Reset hba to 1 in the file index table and to 0 in the ioc. Source file (ioc number 16) and drop file (ioc number 17) cannot be changed.
cf4	Management category: 0 Do not change the mcat field in the file index table. 1 Change the mcat field in the file index table to the value given in the mcat field in Beta(5).
cf3	Drop file size: 0 Do not change the drop file size. 1 Change the drop file size in the file index table to the value given in the dflength field in Beta(4).
cf2	Retention period: 0 Do not change retention period. 1 Change the retention peroid.
cf1	File type: 0 Do not change the file type in the file index table. 1 Change the type field in the file index table to the value given in the type field in Beta(5).

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 3 of 8)

Word	Field	Description
Alpha(1)	c	Type of change. The values are: 0 Change mpn, account number, and/or file attributes. Only c=0 is valid for tape files. 1 Enable the restart of the drop file identified by cfile and usernum. This option is valid for the site security administrator user number only. (No other file attributes are changed.) 2 Change the account number of this executing level-1 task (such as a batch processor); cfile is the drop file name of the task.
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion. There is only one Beta per Alpha.
Beta(1)	cfile	For c=0, this field must contain the current file name in ASCII, left-justified with blank fill. If changing only the filename, the c field must be 0 and nfile must contain the new filename (refer to the following description of nfile). The cf and csio fields must also be 0. To change file attributes (cf, csio) without changing the filename, the c field and the new filename field must both be 0. The current filename (cfile) must also be given. File names must be in the format described in File Concepts. For c=2, this field must contain the task´s drop file name.
Beta(2)	nfile	When the c field is 0, this field must contain the new file name in ASCII, left-justified with blank fill. The file name cannot be changed for tape files.
Beta(3)	usernum	Binary user number under which the new account is valid (a privileged user issued a call with the c field set to 0 or the site security administrator issued the call with c set to 1).
Beta(4)	dflength	Length of the drop file in blocks when cf3 is set to 1. (The drop file may be no larger than #3FFFF due to drop file map limitations.)
	mpn	Master project number in ASCII, left-justified with blank fill. To change the master project number, c must be set to 0. The mpn cannot be changed for tape files.
	rp	Retention period in days; in binary notation when cf2 is set to 1.

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 4 of 8)

Word	Field	Description
Beta(5)	csio	SIO file attribute change options for local, permanent, or tape. Control bits are respresented as follows:



Subfield	Description
cs13	Output file status: 0 Do not change output file status. 1 Change the output file status to value specified in ostat field.
cs12	Allocation unit 0 Do not change allocation unit. 1 Change allocation unit to that specified in the ausize field.
cs11	Maximum PRU size: 0 Do not change the maximum PRU size. 1 Change the maximum PRU size to that specified in the mpru field.
cs10	Tape mode: 0 Do not change the tape mode. 1 Change the tape mode to that specified in the tm field.
cs9	Label processing: 0 Do not change the label processing. 1 Change the label processing to that specified in the lp field.
cs8	Records per block: 0 Do not change the records per block. 1 Change the records per block to that specified in the rpb field.
cs7	Record mark: 0 Do not change the record mark. 1 Change the record mark to that specified in the rmd field.

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 5 of 8)

Word	Field	Description	
		Subfield	Description
Beta(5)	csio	cs6	Padding character: 0 Do not change the padding character. 1 Change the padding character to that specified in the pc field.
		cs5	Record type: 0 Do not change the record type. 1 Change the record type to that specified in the rt field.
		cs4	Maximum record length: 0 Do not change the maximum record length. 1 Change the maximum record length to that specified in the rlmax field.
		cs3	Minimum record length: 0 Do not change the minimum record length. 1 Change the minimum record length to that specified in the rlmin field.
		cs2	Blocking type: 0 Do not change the blocking type. 1 Change the blocking type to that specified in the bt field.
		cs1	File organization: 0 Do not change the file organization. 1 Change the file organization to that specified in the sfo field.
Beta(6)	type	File type code when cf1 is set to 1: 0 Physical data file. 2 Virtual code file.	
	mcat	File management category when cf4 is set to 1: 0 Mass storage file. 1 Scratch file. 2 Output file. 3 MODDROP file (formerly known as write-temporary file). 7 Batch file.	
	sfo	File organization when cs1 is set to 1: 0 Sequential. 1 Direct.	

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 6 of 8)

Word	Field	Description
Beta(6)	bt	Blocking type when cs2 is set to 1: 0 SIL assumes the file was created before SIL was added to the system; therefore it enters default values in the SIL fields of the file index entry. 1 Internal blocking (I). 2 Character type blocking (C). 4 Exact records blocking (K).
	rt	Record type when cs5 is set to 1: 0 Control word (W). 1 ANSI fixed length (F). 2 Record mark(R). 4 Lower CYBER controlword (L). 5 System block (B). 7 Undefined (U).
	rmd	Record mark when cs7 is set to 1 (any 8-bit ASCII character).
	pc	Padding character when cs6 is set to 1 (any 8-bit ASCII character).
	ct	Communication type: 0 Reserved. 1 Access station. 2 Remote Host Facility.
	ic	Internal characteristics, indicating the format of the file: 0 Default; currently 1. 1 Eight-bit ASCII. If dc=SC (refer to figure 5-9), file has free form carriage control. 2 Binary notation. 3 Eight-bit ASCII. If dc=SC (refer to figure 5-9), file has ANSI carriage control.
	ostat	Output file status: 0 Normal status. 1 Destination LID disabled. 2 Destination not responding. 3 Destination rejecting file. 4 SIL error occurred during file transfer. 5 DIVERTED. 6 Hardware path to LID not available. 7 SYS error occured during file transfer. 8-32 Reserved by CDC.

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 7 of 8)

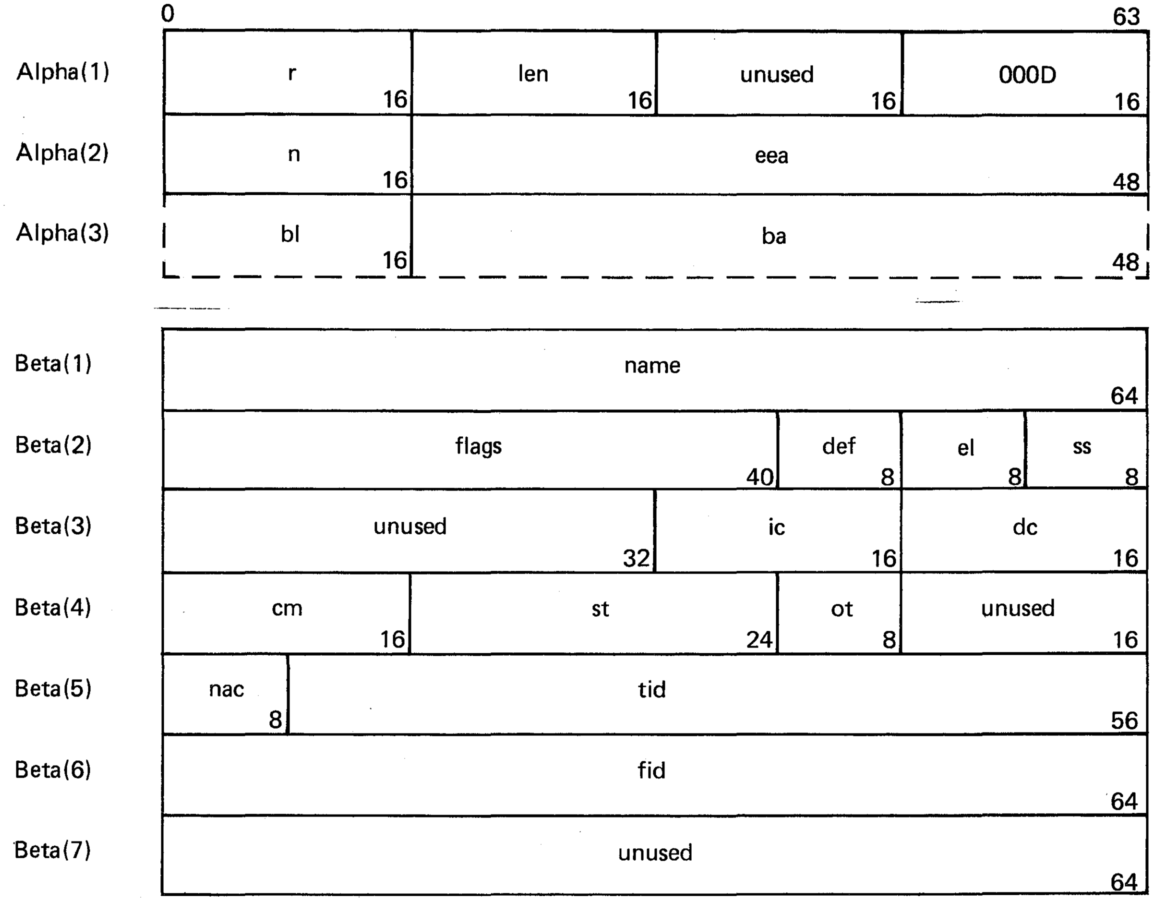
Word	Field	Description
Beta(7)	rlmin	Minimum record length when cs3 is set to 1 (24-bit, user-supplied length in number of bytes).
	rlmax	Maximum record length when cs4 is set to 1 (24-bit, user-supplied length in number of bytes).
	ausize	Allocation unit size in blocks when cf2 is set to 1. User specified guideline for operating system to follow when extending file.
Beta(8)	rpb	Records per block (used only for bt=k) when cs8 is set to 1.
	lp	Label processing when cs9 is set to 1: 0 Read and verify the existing labels. 1 Write new labels.
	tm	Tape mode when cs10 is set to 1: 0 Binary. 1 Coded.
	mpru	Maximum PRU size (used only for V-format tapes) when cs11 is set to 1.
Beta(9)	account	Account number, in ASCII, left-justified with blank fill. This field must be valid under the user number of the task issuing the message, or if the user is a privileged user, this field must be a valid account number under the user number specified in Beta(3).

Figure 5-10. CHANGE FILE ATTRIBUTES (f=#000B) Message Format (Sheet 8 of 8)

FILE DISPOSITION (f=#000D)

A user program can issue this message to specify the disposition of a file, freeing the user from the burden of using naming conventions to accomplish disposition of a file. The file must be either a local mass storage file or an attached permanent file.

The format of the FILE DISPOSITION message is shown in figure 5-11. (The Beta portion of this message can consist of more than one of the seven-word entries shown in the figure.)



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Error code was returned in an ss field of Beta(2). #211 Number of files routed by this message was illegal (the n field is 0). #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-11. FILE DISPOSITION (f=#000D) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len in the length in words of the Beta portion (a multiple of 7).
Alpha(2)	n	Number of files to be routed by this message. Maximum is #FFFF files.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.
Beta(1)	name	File name, in ASCII. File names must be in the format described in File Concepts, Chapter 3.
Beta(2)	flags	Flag bits. Each bit must be set to 0 if the specified parameter is to be ignored, or set to 1 if the specified parameter is to be processed (the value is to be placed in the appropriate file index table entry field). Beta(2) bits and corresponding parameters in this field, from left to right starting from 0, are: 0 Unused 1 Unused 2 ic 3 dc 4 cm 5 st 6 ot 7 Unused 8 tid 9 fid 10 Unused 11 nac
	def	If set to 1, indicates that file disposition is to be deferred. The operating system stores the information about the file into the file index table but does not dispose of the file.
	el	Beta entry length; must be at least 2 and no more than 7.
	ss	Error response field. The values are: 1 Immediate release (def=0) of an active file. 2 Immediate release (def=0) of a nonallocated file. 3 Beta entry length (el) error. 4 File must be attached before the route message is executed. 5 Immediate release (def=0) with no disposition set. 6 Could not write a PFI entry. 7 Illegal disposition code. 8 Illegal site identifier, or the mainframe site identifier is not logged in. 9 Illegal file name. #B Attempt to route a magnetic tape file. #C RHF files may not be routed. #D Files connected to a terminal may not be routed. #E Caller is not the private file owner.

Figure 5-11. FILE DISPOSITION (f=#000D) Message Format (Sheet 2 of 3)

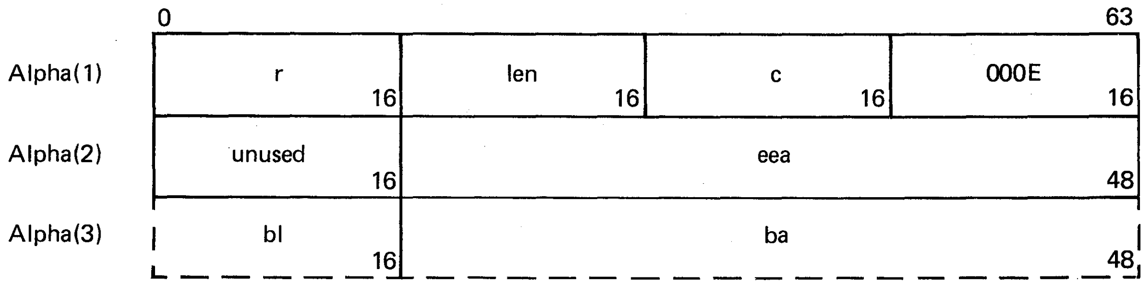
Word	Field	Description
Beta(3)	ic	Internal characteristic, indicating the format of the file. Refer to figure 5-9 for the possible values in this field.
	dc	Disposition code, indicating how the file is to be disposed. Refer to figure 5-9 for the possible values in this field.
Beta(4)	cm	Conversion mode, indicating the type of conversion to be performed on the file when it reaches the access station (st=AST): DI Display code (64-character set). EC Extended display code (128-character set). BI Binary.
	st	Site identifier, identifying the processor responsible for processing the file. If the disposition code is IN (input for batch processing), this field identifies the processor on which the file is to be executed. If the disposition code specifies an output queue, this field identifies the processor on which the file is output. For possible values for this field, contact a site analyst.
	ot	Origin type of a file. B Local batch. E Remote batch. I Interactive.
Beta(5)	nac	Access station area code.
	tid	Terminal identifier. The central site is indicated by tid=0. (Not meaningful for files destined for the CYBER 205.)
Beta(6)	fid	The first five characters of the file name that is to designate the file while it is in the output queue. Any combination of one to five letters and numbers can be specified, with the first character a letter. Two unique job sequence characters added by the system to the job name are used as the sixth and seventh characters of the file name. The eighth character (CYBER 200 only) is a blank.

Figure 5-11. FILE DISPOSITION (f=#000D) Message Format (Sheet 3 of 3)

USER/ACCOUNTING COMMUNICATION (f=#000E)

A user program can issue the USER/ACCOUNTING COMMUNICATION message to retrieve accounting statistics from the cumulative accounting buffer. This call is used by the batch processor to communicate with the accounting system. Only the accounting statistics for the program issuing the message are available to the program; the statistics are available via this message only for the duration of the job.

The format of this message is shown in figure 5-12. The Beta portion of the message is described under the c field definition. The value in each of the Beta words, except for the leftmost 16 bits of Beta(15), is a sum over all accounting periods for the job up until issuance of the USER/ACCOUNTING COMMUNICATION message.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 2 Illegal message option. 3 Nonprivileged task attempted to use a privileged option; or, option 1 or 2 was issued by a user other than a level-1 task. 4 System time unit (STU)/system billing unit (SBU) value is less than 0. 5 Too many entries are written to the accounting file. 7 Error in call to user validation routine. 8 Account number is not valid. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. For message option 1, len must be 1; for option 3, len must be 22; for option 4, len must be 0; and for option 9, len must be 5.

Figure 5-12. USER/ACCOUNTING COMMUNICATION (f=#000E) Message Format (Sheet 1 of 4)

Word	Field	Description
Alpha(1)	c	Message options: 0 Reserved. 1 Start of this batch job; can be issued only from a level-1 task. 2 End of this batch job; can be issued only from a level-1 task. 3 Retrieve accounting information for this task or for all level controllees executed since the start of this batch job. 4 Dump accounting temporary storage to permanent storage and terminate the accounting file; can be issued only by a privileged task. No Beta portion is used for this option. 5 Close out current system dayfile and start a new one; can be issued only by a privileged task. No Beta is used for this option. 6 Allows operating system file transfer utilities to make accounting record entries. Beta(1) through Beta(6) will contain accounting record information. 7 Adds accounting information pertinent for bill usage to the account file. 8 Adds STUs/SBUs to user´s accounting statistics. 9 Adds project accounting information to the account file.
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Message options:		For option 1, the Beta portion of the message consists of one word, a job name of up to eight ASCII characters starting with a letter, left-justified with blank fill. For option 2, Beta words 2 through 6 are optional; therefore, the Alpha length must be set accordingly (len is either 1 or 6).

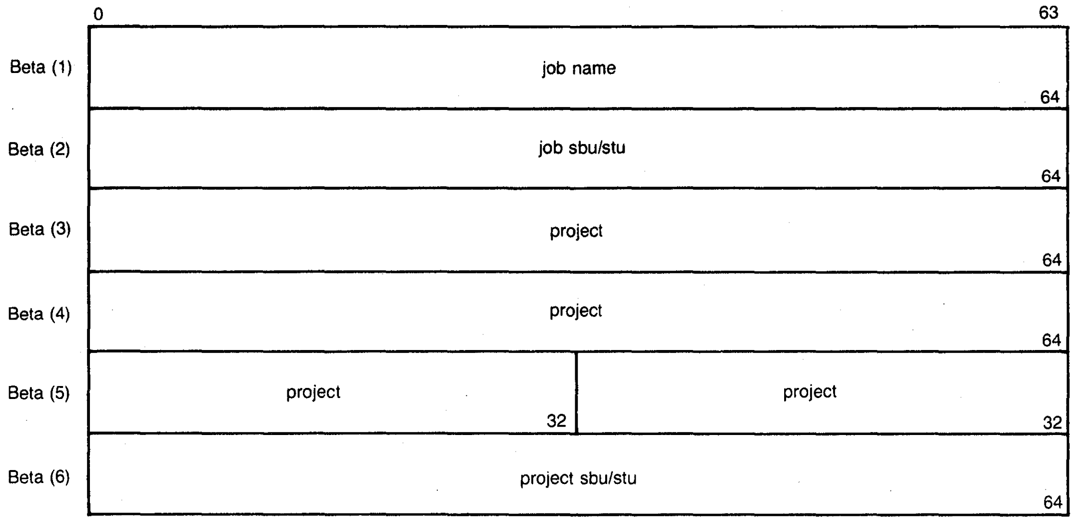


Figure 5-12. USER/ACCOUNTING COMMUNICATION (f=#000E) Message Format (Sheet 2 of 4)

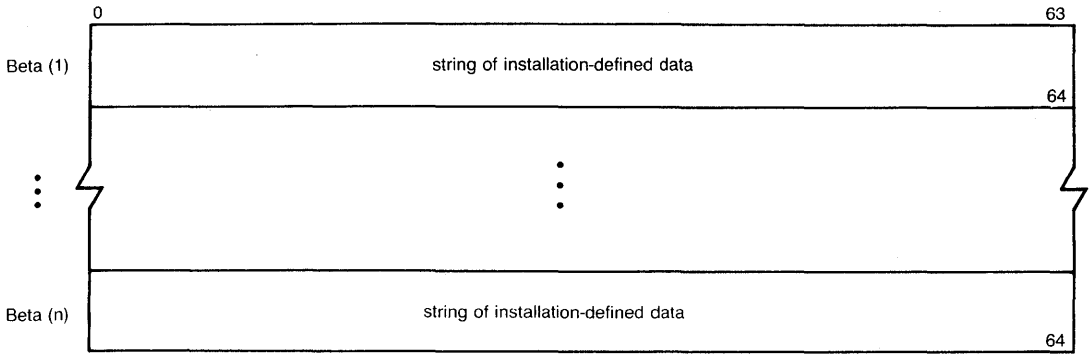
Word	Field	Description
Beta(1)	jobname	Job name in ASCII, left-justified with blank fill.
Beta(2)	job sbu/stu	Total SBU/STU amount accumulated by the job.
Beta(3) through Beta(5)	project	1- to 20-character project number, in ASCII, left-justified with blank fill.
Beta(6)	project sbu/stu	Total SBU/STU amount accumulated by the project number.

For option 3, VSOS returns the following Beta words:

Beta(1)	User execution CPU time, in microseconds.
Beta(2)	Memory usage; at the end of each accounting period, (current working set size)*(user CPU time for current accounting period) is computed and added to a running total kept in this field.
Beta(3)	Number of 16-bit bytes transferred to or from tape files.
Beta(4)	Number of tape accesses (input/output requests issued) for reads and writes.
Beta(5)	Number of nonread and nonwrite tape functions, such as read hardware status.
Beta(6)	Virtual and resident CPU time, in microseconds, for user program execution.
Beta(7)	Number of disk accesses (input/output requests issued) for large page explicit reads and writes.
Beta(8)	Number of disk accesses (output requests issued) for large page implicit writes.
Beta(9)	Number of disk accesses (input/output requests issued) for small page explicit reads and writes.
Beta(10)	Number of disk accesses (output requests issued) for small page implicit writes.
Beta(11)	Number of disk sectors transferred for explicit reads and writes.
Beta(12)	Number of disk sectors transferred for implicit writes.
Beta(13)	Number of disk accesses (input requests issued) that resulted from large page faults (large page implicit reads).
Beta(14)	Number of disk accesses (input requests issued) that resulted from small page faults (small page implicit reads).
Beta(15)	Current working set size (leftmost 16 bits), and the number of virtual system user calls made (rightmost 48 bits).
Beta(16)	STUs (cumulative TCHARGE calculations, integer).
Beta(17)	SBUs (cumulative MCHARGE calculations, real).
Beta(18)	Number of large pages lost.
Beta(19)	Number of small pages lost.
Beta(20)	Cumulative amount of CPU time for which this task´s working set size limit appeared to be too small.
Beta(21)	Account block STU value (integer).
Beta(22)	Account block SBU value (real).

Figure 5-12. USER/ACCOUNTING COMMUNICATION (f=#000E) Message Format (Sheet 3 of 4)

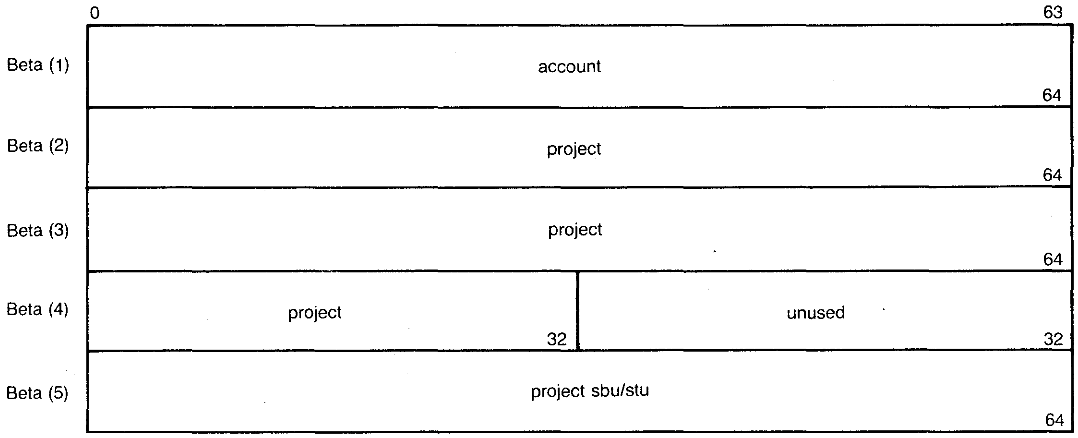
For options 4 through 6, there is no Beta portion. For option 7, the Beta portion of the message is as follows, with up to 10 words of Beta:



Maximum length of 80 characters containing billing information.

For option 8, the Beta portion of the message consists of one word, a positive floating-point STU/SBU amount.

For option 9, the account set in Beta(1) is checked for validity and the accumulated account block SBUs or STUs are returned to the caller.

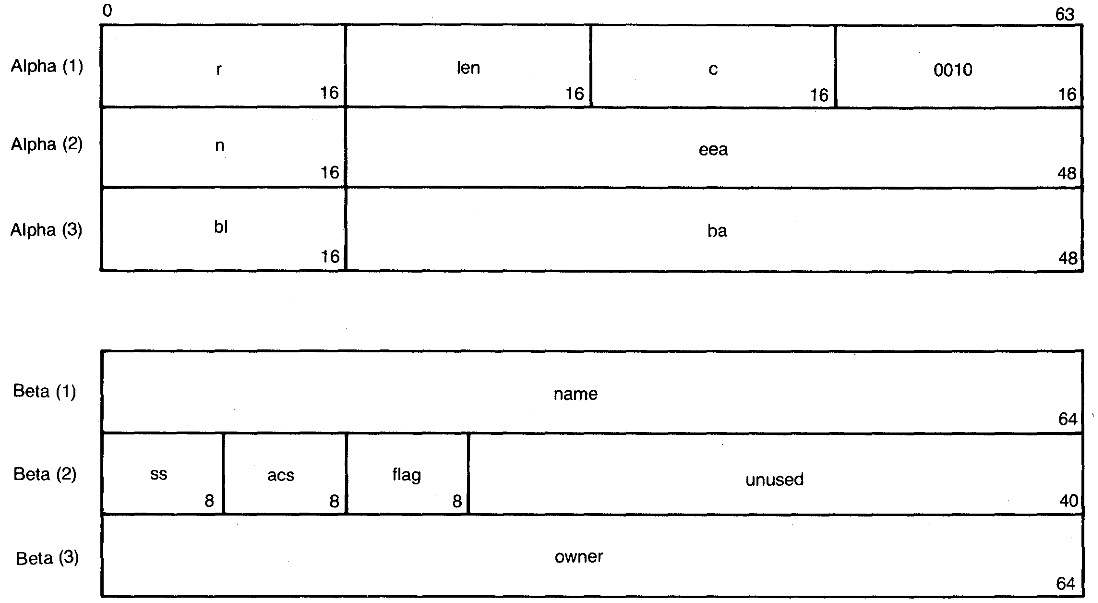


Word	Field	Description
Beta(1)	account	Account number in ASCII, left-justified with blank fill.
Beta(2) through Beta(4)	project	1 to 20 character project number in ASCII, left-justified with blank fill.
Beta(5)	project sbu/stu	Total SBU/STU amount accumulated by the project number.

Figure 5-12. USER/ACCOUNTING COMMUNICATION (f=#000E) Message Format (Sheet 4 of 4)

ATTACH PERMANENT FILE (f=#0010)

A program issues this message to attach an existing permanent file. Only one Beta is processed for each Alpha used. The format of this message is shown in figure 5-13.



Word	Field	Description
Alpha(1)	r	Response code returned by the operating system when this message has been processed. If no error occurs, the code is 0; otherwise: 1 Refer to Beta field ss for the specific error. All Beta entries are processed. 2 More than one of the following response codes 4, 5, and 6. 3 Illegal option. Control field contains an illegal value (not 0 or 1). 4 Access conflict with another job of this user. 5 Filename is in use as a local file. 6 Access conflict with another user. #211 Number of files given was 0 or greater than 16. #214 Beta buffer length error. Either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) exists and contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to be located beginning at Alpha(3), and len is the length of the Beta portion of the message.
	c	Control field. If this field is 0, attach files given in the Beta field. If this field is 1, attach all files (no Beta required).
Alpha(2)	n	Number of requests in this message (maximum is 16). Not used for c=1.

Figure 5-13. ATTACH PERMANENT FILE (f=#0010) Message Format (Sheet 1 of 2)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>Alpha(2)</td><td>eea</td><td>Virtual bit address to receive control if an error occurs while this message is processed (r not equal to 0); if eea is 0 when an error occurs, the error is considered fatal.</td></tr>
<tr><td>Alpha(3)</td><td>bl, ba</td><td>If the Beta portion of the message is not contiguous to the Alpha portion (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.</td></tr>
<tr><td>Beta(1)</td><td>name</td><td>Name of the file in ASCII; file names (left-justified with blank fill) must be in the format described in chapter 3. Drop file can also be attached.</td></tr>
<tr><td>Beta(2)</td><td>ss</td><td>Error responses:
<table>
<tr><td>0</td><td>Normal completion.</td></tr>
<tr><td>1</td><td>Permanent file name not found.</td></tr>
<tr><td>2</td><td>File already attached to this job.</td></tr>
<tr><td>3</td><td>Access conflict with another of user's jobs.</td></tr>
<tr><td>4</td><td>File already attached as a local file or attached permanent file.</td></tr>
<tr><td>5</td><td>No user table entry available.</td></tr>
<tr><td>6</td><td>Access violation; user does not have requested access permissions.</td></tr>
<tr><td>7</td><td>Access conflict with another user.</td></tr>
<tr><td>8</td><td>Specified user number does not exist.</td></tr>
<tr><td>9</td><td>Not enough space in the FILEI system table.</td></tr>
<tr><td>#A</td><td>File spans a downed device.</td></tr>
<tr><td>#B</td><td>Read-only access required for partial attach.</td></tr>
<tr><td>#C</td><td>Attempted to attach a purge-only file.</td></tr>
<tr><td>#D</td><td>User attempted to attach a file with write, modify, or append access when the field is privileged open.</td></tr>
</table></td></tr>
<tr><td></td><td>acs</td><td>Desired file access. This field is treated as eight 1-bit fields. Each bit set requests the associated access. Combinations are allowed. The values are:
<table>
<tr><th>Bit</th><th>Hexadecimal Value</th><th>Description</th></tr>
<tr><td>1-3</td><td>-</td><td>Unused.</td></tr>
<tr><td>4</td><td>10</td><td>Execute access.</td></tr>
<tr><td>5</td><td>8</td><td>Modify access.</td></tr>
<tr><td>6</td><td>4</td><td>Append access.</td></tr>
<tr><td>7</td><td>2</td><td>Read access.</td></tr>
<tr><td>8</td><td>1</td><td>Write access.</td></tr>
</table>
If acs is binary 0, the default is all access types permitted to the caller.</td></tr>
<tr><td></td><td>flag</td><td>A set of 8 bits (Fl through F8) indicating a special action. Values are:
<table>
<tr><td>f1=0</td><td>Do not attach file spanning a downed device.</td></tr>
<tr><td>f1=1</td><td>Attach file spanning a downed device.</td></tr>
<tr><td>f1=2-8</td><td>Reserved.</td></tr>
</table></td></tr>
<tr><td>Beta(3)</td><td>owner</td><td>ASCII user number of file owner (six ASCII characters, right-justified, zero-filled). If this field is 0, the caller's user number is used.</td></tr>
</table>

Figure 5-13. ATTACH PERMANENT FILE (f=#0010) Message Format (Sheet 2 of 2)

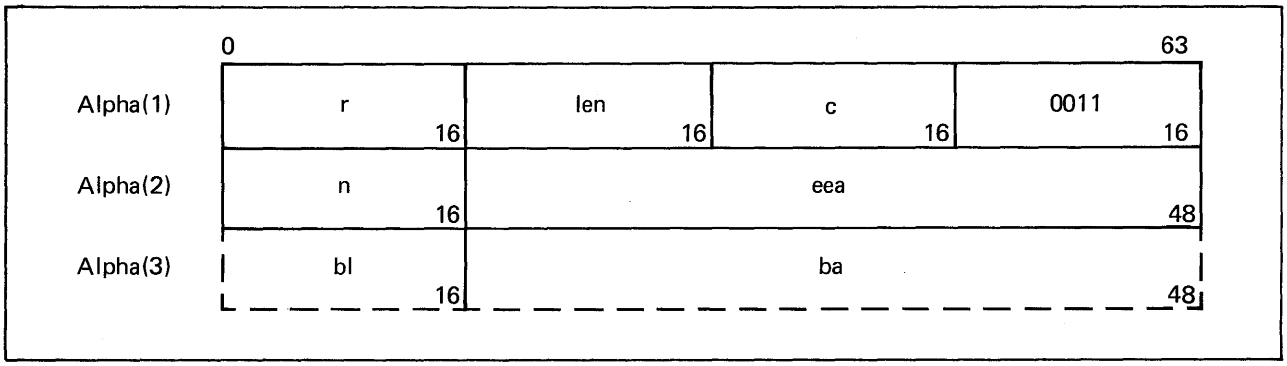
GET PACK LABEL AND PFI (f=#0011)

A privileged user or master user issues this message to retrieve the pack label and pack file index for a specified pack. The preferred option is to return unformatted entries (c=1).

If the user is a privileged user, all the PFI entries are returned for the specified pack. If the user is a master user, only those PFI entries (for which the user is a master user of the account) are returned for the specified pack.

In the first call, the user initializes the Alpha words (n set to 0) and the packid field in Beta(1). The length of the Beta portion must be at least 528 (one block plus 16 words for the pack label). The pack label is returned in Beta(1) through Beta(11) and the PFI entries are returned starting at Beta(17). A count is returned to the n field indicating the number of entries plus one that have been returned in the current call (or series of calls, if more than one call is issued). If a 1 is returned in the r field of Alpha(1), more PFI entries exist and the call must be reissued to get the rest of the entries. In the second and any subsequent calls, the n field must contain the count returned in the previous call. PFI entries are then returned starting at Beta(1).

The label format, as set by the system routine NAMEPACK, is shown in figure 5-14. The format of the entries is the same as for the LIST FILE INDEX OR SYSTEM TABLE message option 1 (shown in figure 5-9), except that the user/ref field is always ref and oacs always contains the oacs value.



	0			63
Alpha(1)	r 16	len 16	c 16	0011 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		

Figure 5-14. GET PACK LABEL AND PFI (f=#0011) Message Format (Sheet 1 of 4)

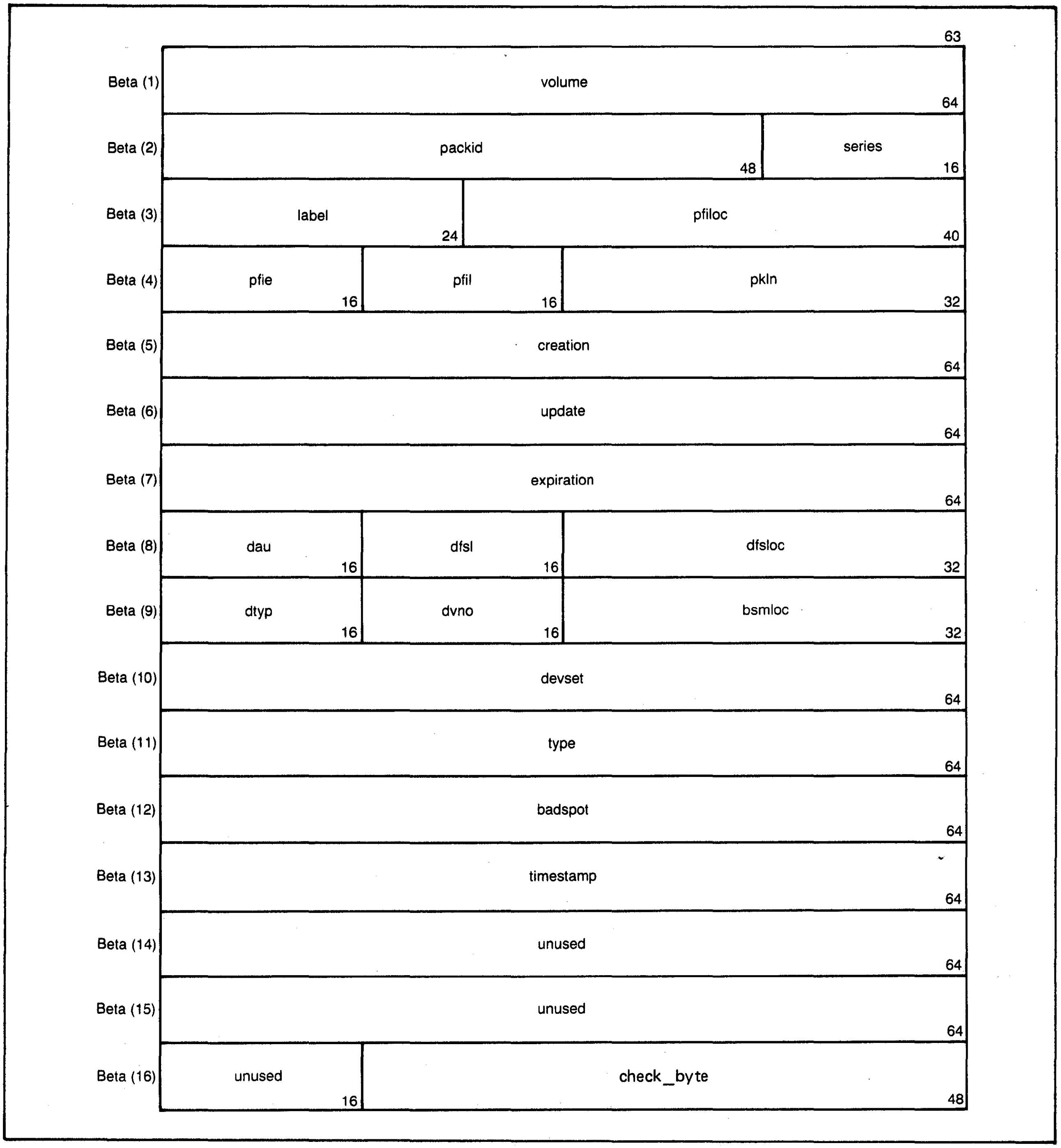


Figure 5-14. GET PACK LABEL AND PFI (f=#0011) Message Format (Sheet 2 of 4)

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Nonfatal error; more files exist than the Beta portion could hold; reissue the call to get the rest. 2 User is not privileged or is not a master user. 3 Disk I/O error. 4 Pack identifier was not found. 5 Illegal option. 6 User directory not found.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. This field must be multiple of 16 and a minimum of 528.
	c	Message options: 0 Return pack label (Beta is at least 16 words). 1 Return PFI entries (Beta is a multiple of 16, at least 512 words). 2 Return bad spot map (Beta is at least 512 words).
Alpha(2)	n	This field must be set to 0 by the user for the first call, causing the pack label alone to be returned. For a reissued call, this field must be set to the value that the operating system returned to the user in this field for the previous call. If C=1, n indicates the starting entry.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	volume	When the call is issued, Beta(1) contains the pack number. Contains the characters VOL 3 to distinguish the label from the earlier versions that contained VOL 2.
Beta(2)	packid	Pack identifier of the pack sought, in ASCII, left-justified with blank fill. For a first call (n=0), the pack identifier in Beta(1) is overwritten with the returned label. Sixteen words of the pack label are returned (last three words are not used); all 16 words of each used pack file index entry are returned. Pack identifiers are obtained using option 9 of the LIST SYSTEM TABLE (f=#0009) message.
	series	Value of 2031, in hexadecimal notation.

Figure 5-14. GET PACK LABEL AND PFI (f=#0011) Message Format (Sheet 3 of 4)

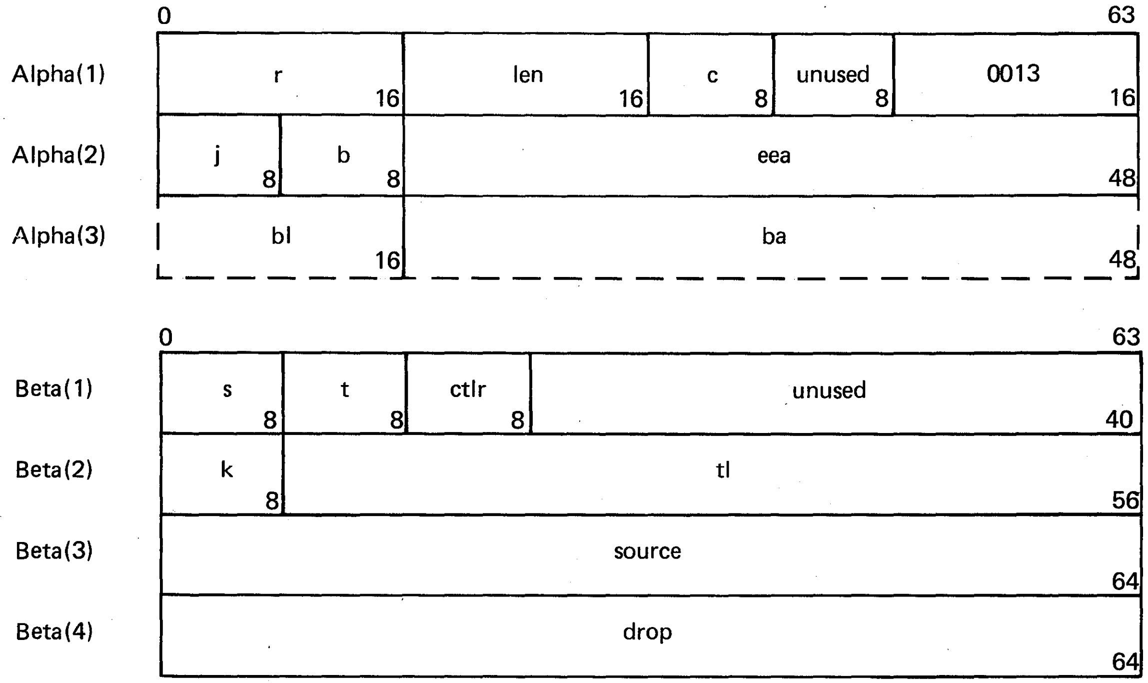
Word	Field	Description
Beta(3)	label	Disk block address of this label.
	pfiloc	Disk block address of the first block of the Pack File Index (PFI).
Beta(4)	pfie	Entry number of this entry within the PFI, counting from 0.
	pfil	Length of the PFI in blocks.
	pkln	Pack length that is the number of 512-word blocks that can be allocated on this disk pack.
Beta(5)	creation	ASCII date, in the format mm.dd.yy, of the creation of this label.
Beta(6)	update	ASCII date, in the format mm.dd.yy, of the last update of the disk.
Beta(7)	expiration	ASCII date, in the format mm.dd.yy, of the expiration of the disk.
Beta(8)	dau	The disk allocation unit contains the binary number of 512-word blocks in an allocation unit. It is the minimum allocation unit for this disk pack.
	dfsl	Length of the directory of file segmentation (DFS).
	dfsloc	Starting disk block address of the DFS.
Beta(9)	dtyp	Device type indicator: 1 Reserved. 2 81912 disk pack (single density, 18 sector). 3 81922 disk pack (double density, 18 sector).
	dvno	Device number associated with this disk pack.
	bsmloc	Starting disk block address of the bad spot map (BSM).
Beta(10)	devset	Device set name in the format DVSTxx, where xx is the device set number. The field is left-justified and blank-filled.
Beta(11)	type	Type of disk pack: 81912 or 81922 in hexadecimal notation; used by the operating system to determine the length of the disk pack. (Returned for release version 2.1.5 compatibility only.)
Beta(12)	badspot	Name of pseudo file converting the bad spot map. (Retained for release version 2.1.5 capability only.)
Beta(13)	timestamp	Time of last autoload.
Beta(16)	check_byte	Check sum of selected fields of the pack label. Words 1 through 4 and 8 through 12 are used to generate the check_byte.

Figure 5-14. GET PACK LABEL AND PFI (f=#0011) Message Format (Sheet 4 of 4)

LIST CONTROLLEE CHAIN (f=#0013)

A user program can obtain a list of the controllee chain, including the program level and descriptor block number, the executable source file name, drop file name, and so forth, of each task in the chain, by using the LIST CONTROLLEE CHAIN message shown in figure 5-15. The issuing program can determine its own position in the chain by comparing fields j and b in Alpha(2) with fields s and t in Beta(1).

A total of nine levels is the maximum; that is, eight controllees plus the level-1 batch processor or virtual system interactive processor. The descriptor block number is unique and is associated with the program until it terminates. Observe that level 1 will be listed for a batch job. Level 2 will be the highest level for which information is returned if the task is running interactively.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, control proceeds normally and this field contains the number of words returned in Beta; otherwise, the response code is:

1	The length specified for Beta is 0.
2	An illegal option was selected.
#214	Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-15. LIST CONTROLLEE CHAIN (f=#0013) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. For message option 0, len must be a multiple of 4, up to a limit of 36. For message options 1, 2, and 3, len must be 4.
	c	Message options: 0 List all controllees in the chain; controllees are listed in ascending order, starting with the job control processor. 1 List only the program that issued the message. 2 List only the controller of the program that issued the message. 3 List only the controllee of the program that issued the message.
Alpha(2)	j	Level in the controllee chain of the program that issued the message. Level numbers in this field range from 1 to 9.
	b	Descriptor block number of the program that issued the message.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	s	The level in the controllee chain of the program whose name is in Beta(3). Level numbers in this field range from 1 to 9.
	t	Descriptor block number of the program whose name is in Beta(3).
	ctlr	Descriptor block number of the controller of the program whose name is in Beta(3): #FF Controller is interactive processor.
Beta(2)	k	Descriptor block number of the controllee of the program whose name is in Beta(3). This field can be 0.
	tl	Time limit of the program whose name is in Beta(3).
Beta(3)	source	Name of the executable source file, in ASCII.
Beta(4)	drop	Name of the drop file, in ASCII.

Figure 5-15. LIST CONTROLLEE CHAIN (f=#0013) Message Format (Sheet 2 of 2)

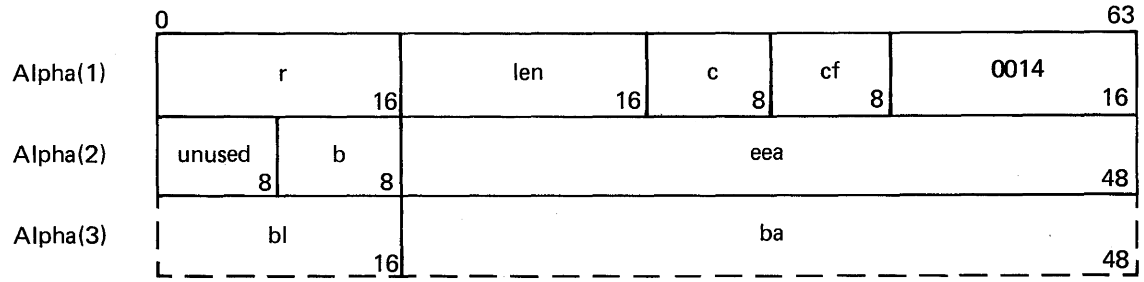
SEND A MESSAGE TO CONTROLLER (f=#0014)

This message is used by a program to send a string of binary or ASCII data to a program controller or the job control processor (the batch processor or virtual system interactive processor). When this message is issued, the operating system copies the data string from the Beta portion of the message into a system buffer.

When output requests are being sent to a virtual system interactive processor (for example, a user at a terminal) from a task (its controllee) and the wait or replace option (m=0) has been selected, the system message buffer can hold up to 5 data strings or 4096 character bytes, whichever limit is reached first. For a logged-out user, only one data string can be held in the buffer. The data is grouped in blocks of 151 character bytes and sent, 1 block at a time, from the virtual system interactive processor to the output device. If the last block is fewer than 151 character bytes, an end-of-message character is added after the last character byte. The issuer of the message is responsible for formatting any multiline strings to be sent to a terminal by inserting line feed and carriage return characters at the appropriate places in the string.

If a data string from a controller has been sent but not requested by the controllee when the controllee issues this message, the data string from the controller to the controllee is lost. The controllee should check, therefore, to see if any data strings are waiting to be received before it issues this message.

If the controller is the batch processor, the message is put in the job dayfile. The format of this message is shown in figure 5-16. The Beta portion contains the string of binary or ASCII data sent to the program controller or job control processor. The maximum length of the Beta portion, when present, is 4096 character bytes. When a data string is sent in this way to the virtual system interactive processor, the processor sends it to an output device (a terminal).



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Length of Beta in character bytes is either 0 or greater than 4096. 2 Illegal option was selected for this message. 3 No controllee matches the value of b=0, or this task is a level-1 task; therefore, sending this particular message was pointless.

Figure 5-16. SEND A MESSAGE TO CONTROLLER (f=#0014) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	r	4 If the notify option was selected (c=1), the controller designated was a job control processor for a logged-out user. 6 If the notify option was selected, the system output buffer is full. 7 Error in sending message to job dayfile. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length in character bytes and the virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in character bytes of the Beta portion.
	c	Message options: 0 If the controller to whom the data string was sent is a logged-out user, replace any existing string waiting in the buffer with the new string. If the job control processor buffer is full, stop running this program until the buffer is free. 1 If the data string cannot be sent to the controller, return control to the error exit address. 2 If the data string cannot be sent to the controller, stop running this program until the message can be sent.
	cf	Control field. The values are: 0 Send the data string to the controller. If the controller is a virtual system interactive processor or batch processor, continue running this controllee program (the program issuing this message); otherwise, start running the controller and stop running this controllee program. 2 Send the data string to the level-1 task. Continue running this controllee program.
Alpha(2)	b	Descriptor block number of the controller. If the data string is to be sent directly to a level-1 task (c=2), or if this program's controller is a level-1 task, this field is ignored. If this field is 0, the data string is sent to the next higher controller.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in character bytes and virtual bit address of the first full word of the Beta portion.

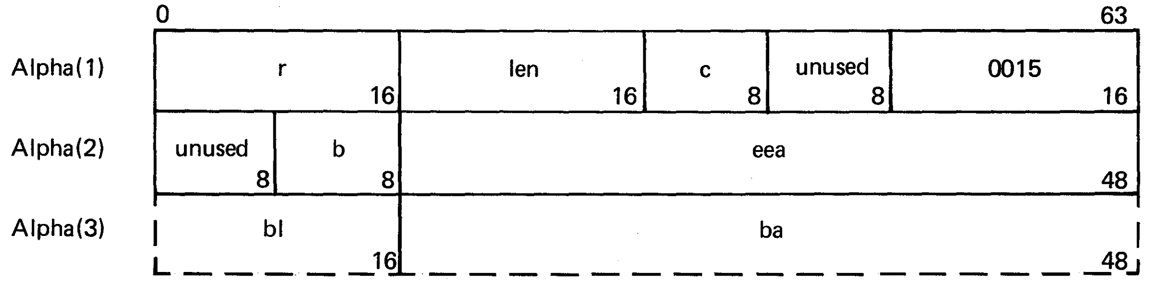
Figure 5-16. SEND A MESSAGE TO CONTROLLER (f=#0014) Message Format (Sheet 2 of 2)

SEND A MESSAGE TO CONTROLLEE (f=#0015)

A program starts a controllee running by issuing a SEND A MESSAGE TO CONTROLLEE message. The optional Beta portion of this message contains a string of binary or ASCII data for the controllee to receive as soon as the controllee has been started running. If the Beta portion is present when the operating system processes the SEND A MESSAGE TO CONTROLLEE message, the operating system copies the data string from the Beta portion into a system buffer before it starts the controllee. The controllee will have to issue a GET MESSAGE FROM CONTROLLER OR OPERATOR message to retrieve the data string from the system buffer.

A special situation arises if any controllee (except for the immediate controllee of a level-1 task) issues a GET MESSAGE FROM CONTROLLER OR OPERATOR message when no data string is waiting in the system buffer. In this case, the controllee stops running and waits until a message is sent to it, and the next higher controller in the controllee chain is started running.

The format of this message is shown in figure 5-17. The maximum length of the Beta portion, when present, is 4096 character bytes.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise:

1	Length of Beta in character bytes is 0 or greater than 4096.
2	Illegal option was selected for this message.
3	No controllee matches the value of the b field (for b=0).
4	For b=0, no controllee exists.
7	Error exit has been taken, since the controllee already has a text string from the controller.
#214	Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-17. SEND A MESSAGE TO CONTROLLEE (f=#0015) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length in character bytes and the virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in character bytes of the Beta portion.
	c	Message options: 0 This message has a Beta portion containing a data string for the controllee. If the controllee already has a data string waiting from the controller, replace it with the new data string. 1 This message has a Beta portion containing a data string for the controllee. If the controllee already has a data string waiting from the controller, return control to the error exit address. 2 This message does not have a Beta portion.
Alpha(2)	b	Descriptor block number of the controllee; if 0, the data string is sent to the next lower controllee in the controllee chain.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.

Figure 5-17. SEND A MESSAGE TO CONTROLLEE (f=#0015) Message Format (Sheet 2 of 2)

GET MESSAGE FROM CONTROLLER OR OPERATOR (f=#0016)

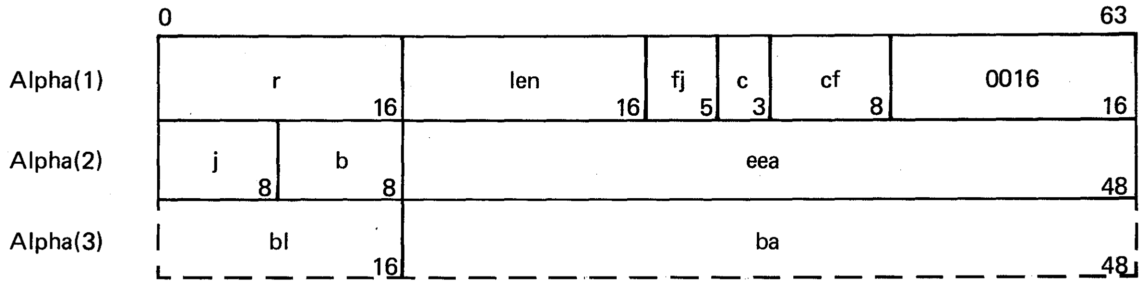
A string of binary or ASCII data sent by a controller program or the operator and waiting in a system buffer can be retrieved by this controllee program using a GET MESSAGE FROM CONTROLLER OR OPERATOR message. Depending on the message option selected, the data string being retrieved might be copied into Beta, or it could be processed into a set of symbols before it is stored into Beta. In any case, the data string being retrieved must not exceed 512 words (4096 character bytes).

Multiword symbols are permitted and processed without any special treatment. If the number of symbols exceeds the number requested, only the number requested are stored in Beta. If fewer symbols are returned than are requested, all symbols are stored in Beta. The operating system in this case never appends an end-of-message character.

Delimiters are always returned right-justified with blank fill. Blanks are never treated as a special case (if a space is a delimiter, all occurrences of blank result in a delimiter being returned; if space is not a delimiter, spaces are processed the same as any other character).

A special situation occurs if there is no data string waiting in the system buffer when a controllee (except for the immediate controllee of a level-1 task) issues this message. The controllee will stop running and wait for a data string from its controller. The next higher controller in the controllee chain will start running.

The format of this message is shown in figure 5-18. The Beta portion is discussed under the c field description.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, control proceeds normally and this field contains the number of words (if an unprocessed data string was obtained) or character bytes (if a data string processed into symbols was obtained) returned in Beta. If control returns to the error exit address, nothing was returned in Beta and the values are:

1	Count of bytes returned was either 0 or greater than 4096.
2	Illegal option was specified for this message.
3	No controller or operator message existed.
5	$EOF received for file connected to a terminal.

Figure 5-18. GET MESSAGE FROM CONTROLLER OR OPERATOR (f=#0016) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	r	7 $EOG received for file connected to a terminal. 8 $EOR received for file connected to a terminal. 7 More than 200 delimiters are defined by this program. 9 This message was issued from a level-1 task. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length in words (if the data string is to be translated into symbols) or character bytes (if an untranslated data string is to be obtained) and the virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length of the Beta portion in words or character bytes. Maximum is 512 words (4096 bytes).
	fj	Fill and justification for the message. For all options except c=0, the values are: 0 Left-justification, blank fill to the right. 1 Left-justification, zero fill to the right. 4 Right-justification, blank fill to the left. 5 Right-justification, zero fill to the left.
	c	Message options: 0 The data string is to be copied from the system buffer to Beta, beginning at Beta(1). If the number of words in the data string exceeds the number specified by the len field, only the first len words are copied to Beta. If there are fewer than the number requested, the last word of the data string is left-justified with binary zero fill. 1 The data string is to be translated into symbols. Delimiters must be defined by the program issuing this message, and their number must not exceed 200. Symbols are stored in Beta, one symbol per word, starting with Beta(2). Beta(1) contains the number of delimiters (leftmost 16 bits), and the virtual bit address of the delimiter buffer (rightmost 48 bits). Delimiters are stored left to right, character byte by character byte, in the buffer. 2 The data string is to be translated into symbols. Delimiters are blank, period, comma, slash, equals, plus, minus, and left and right parentheses. Symbols are stored in Beta starting with Beta(1). 3 The data string is to be translated into symbols. Delimiters are defined as installation parameter options. Symbols are stored in Beta starting with Beta(1).

Figure 5-18. GET MESSAGE FROM CONTROLLER OR OPERATOR (f=#0016)
Message Format (Sheet 2 of 3)

Word	Field	Description
Alpha(1)	cf	Control field: 0 If no data string from this program´s controller is waiting in the system buffer, stop running this program until a data string arrives. Process and return the data string to Beta, and release the system buffer space occupied by the data string. 1 If no data string from this program´s controller is waiting in the system buffer, return control to the error exit address. If there is a data string waiting, process and return it to Beta, and release the system buffer space occupied by the data string. 2 If no data string from this program´s controller is waiting in the system buffer, stop running this program until a data string arrives. Process and return the data string to Beta, but do not release the system buffer space occupied by the data string. 3 If no data string from this program´s controller is waiting in the system buffer, return control to the error exit address. Process and return the data string to Beta, but do not release the system buffer space occupied by the data string. 4 If no data string from the operator is waiting in the system buffer, stop running the program until a data string arrives. Process and return the data string to Beta, and release the system buffer space occupied by the data string. 5 If no data string from the operator is waiting in the system buffer, return control to the error exit address. Process and return the data string to Beta, and release the system buffer space occupied by the data string.
Alpha(2)	j	Level of the controller that sent the data string being retrieved; supplied by the operating system. If the data string came from the operator, this field is 0. If no data string was found in the system buffer, the operating system returns in j the level of the task that issued this message.
	b	Descriptor block number of the controller that sent the data string being retrieved; supplied by the operating system. (If the interactive processor was the conroller, b is FF.) If the control field is 4 or 5, this field is the descriptor block number of the operator. If no data string was found in the system buffer, the operating system returns in b the descriptor block number of the task that issued this message.
	eea	Virtual bit address to receive control if an error occurs during processing of this message. If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), this parameter indicates the length in words (if an untranslated data string is to be obtained) or character bytes (if the data string is to be translated into symbols) and virtual bit address of the first full word of the Beta portion.

Figure 5-18. GET MESSAGE FROM CONTROLLER OR OPERATOR (f=#0016)
Message Format (Sheet 3 of 3)

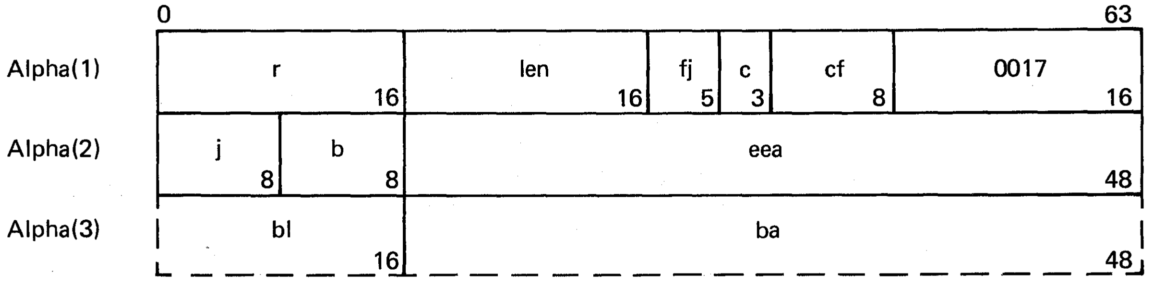
GET MESSAGE FROM CONTROLLEE (f=#0017)

A string of binary or ASCII data sent by a controllee program and waiting in a system buffer can be retrieved by this controller program using a GET MESSAGE FROM CONTROLLEE message. Depending on the message option selected, the data string being retrieved might be simply copied into Beta, or it could be processed into a set of symbols before it is stored in Beta. In any case, the data string being retrieved must not exceed 512 words (4096 character bytes).

Multiword symbols are permitted and processed without any special treatment. If the number of symbols exceeds the number requested, only the number requested are stored in Beta. If fewer symbols are returned than are requested, all symbols are stored in Beta. The operating system in this case never appends an end-of-message character.

Delimiters are always returned right-justified with null fill. Blanks are never treated as a special case (if a space is a delimiter, all occurrences of blank result in a delimiter being returned; if space is not a delimiter, spaces are processed the same as any other character).

The format of this message is shown in figure 5-19. The Beta portion is discussed under the c field description.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, control proceeds normally and this field contains the number of character bytes (if a message was obtained) or words (if symbols were obtained) returned in Beta. If control returns to the error exit address, nothing is returned in Beta and the values are: 1 Count of bytes returned was 0 or greater than 4096. 2 Illegal option was specified for this message. 3 No controllee message existed. 4 Message from the controller (not the controllee) was waiting. 6 This program started because the controllee, whose level and descriptor number are stored in fields j and b, is waiting for a message from the controller. 7 More than 200 delimiters are defined by this program. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.

Figure 5-19. GET MESSAGE FROM CONTROLLEE (f=#0017) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length in character bytes (if an untranslated data string is to be obtained) or words (if the data string is to be translated into symbols) and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length of the Beta portion in words or character bytes. Maximum is 512 words (4096 bytes).
	fj	Fill and justification for the message. For all options except c=0, the values are: 0 Left-justification, blank fill to the right. 1 Left-justification, zero fill to the right. 4 Right-justification, blank fill to the left. 5 Right-justification, zero fill to the left.
	c	Message format options: 0 The data string is to be copied from the system buffer to Beta, beginning at Beta(1). If the number of words in the data string exceeds the number specified by the len field, only the first len words are copied to Beta. If there are fewer than the number requested, the last word of the data string is left-justified with binary zero fill. 1 The data string is to be translated into symbols. Delimiters must be defined by the program issuing this message, and their number must not exceed 200. Symbols are stored in Beta, one symbol per word, starting with Beta(2). Beta(1) contains the number of delimiters (leftmost 16 bits), and the virtual bit address of the delimiter buffer (rightmost 48 bits). Delimiters are stored left to right, character byte by character byte, in the buffer. 2 The data string is to be translated into symbols. Delimiters are blank, period, comma, slash, equals, plus, minus, and left and right parentheses. Symbols are stored in Beta starting with Beta(1). 3 The data string is to be translated into symbols. Delimiters are defined as installation parameter options. Symbols are stored in Beta starting with Beta(1).
	cf	Control field: 0 After data string has been retrieved, release system buffer space occupied by string. 2 After data string has been retrieved, do not release system buffer space occupied by string.

Figure 5-19. GET MESSAGE FROM CONTROLLEE (f=#0017) Message Format (Sheet 2 of 3)

Word	Field	Description
Alpha(2)	j	Level of the controllee that sent the data string; supplied by the operating system. The value in this field has no meaning if the controllee that sent the data string being retrieved has been disconnected.
	b	Descriptor block number of the controller that sent the data string; supplied by the operating system. The value in this field has no meaning if the controllee that sent the data string being retrieved has been disconnected.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in words (if an untranslated data string is to be obtained) or character bytes (if the data string is to be translated into symbols) and virtual bit address of the first full word of the Beta portion.

Figure 5-19. GET MESSAGE FROM CONTROLLEE (f=#0017) Message Format (Sheet 3 of 3)

REMOVE CONTROLLEE FROM MAIN MEMORY (f=#0019)

A controller (the user program) can swap a controllee program or itself from main memory to mass storage. The controller program stops running until all controllee pages are written to mass storage. The format of this message is shown in figure 5-20. For option 0, the program issuing this message must have only one controllee.

	0			63
Alpha(1)	r 16	len 16	unused 16	0019 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		
Beta(1)	b 8	unused 56		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 No controllee exists with this descriptor block number. 3 More than one controllee for an n field value of 0. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the len field is 0 (n=1 only).
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. This field is 0 or 1.
Alpha(2)	n	Specifies the controllee to be removed. The values are: 0 Remove the controllee program to mass storage; no Beta portion is required for this option. 1 Remove the controllee specified in Beta(1) to mass storage. 2 Remove this program to mass storage; no Beta portion is required for this option.

Figure 5-20. REMOVE CONTROLLEE FROM MAIN MEMORY (f=#0019) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	b	Descriptor block number of the controllee to be removed to mass storage.

Figure 5-20. REMOVE CONTROLLEE FROM MAIN MEMORY (f=#0019) Message Format (Sheet 2 of 2)

SEND A MESSAGE TO OPERATOR (f=#001A)

A program uses this message to send a string of binary or ASCII data to the operator. The system copies the data string from the Beta portion of the system message to a system buffer.

If the system buffer is full, the string cannot be sent. If the buffer is full, the system continues task execution at the error exit address.

Because the operator of a busy system could miss a string sent to him, the system provides a string save table. If the operator is logged in and n=2 or 3, the string is kept in the string save table.

Only one string per task is kept. If the task sends another string, only the most recent string requiring a response is kept.

The operator can access the string save table to see the most recent strings sent by executing tasks. The operator clears a string from the save table with the command CFO. (Strings are kept by descriptor block number.)

The format of this message is shown in figure 5-21. The Beta portion contains the string being sent; maximum length of the string is 80 character bytes (10 words).

	0			63
Alpha(1)	r 16	len 16	unused 16	001A 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Specified length of the Beta portion in character bytes was out of range (either 0 or greater than 80). 2 String in Beta could not be sent to the operator because no system buffer was available; error processing is indicated by the value of the n field. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the len field is 0.

Figure 5-21. SEND A MESSAGE TO OPERATOR (f=#001A) Message Format (Sheet 1 of 2)

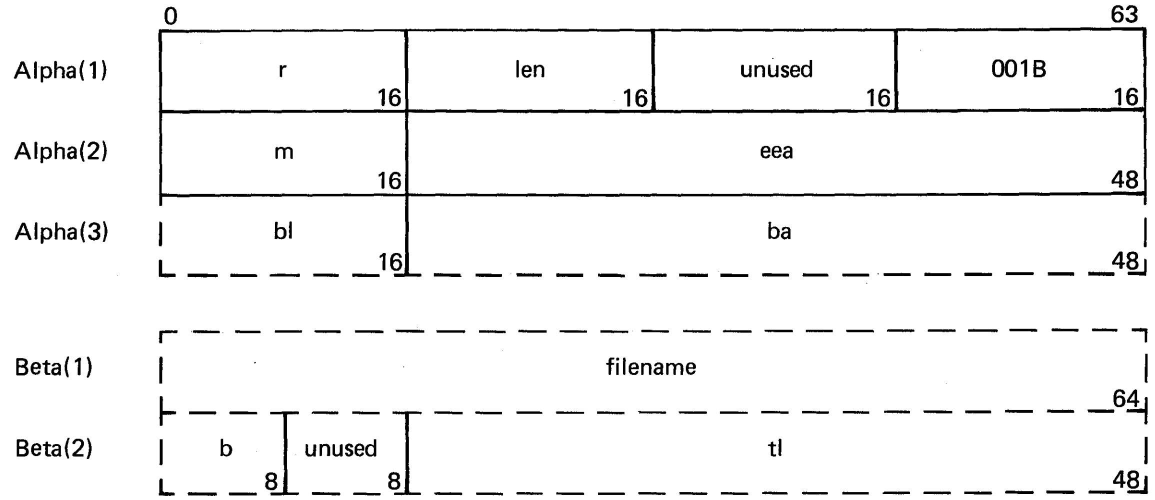
Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length in character bytes and the virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in character bytes of the Beta portion. The len field should be greater than 0 and less than or equal to 80.
Alpha(2)	n	Indicates action to be taken if the string cannot be sent; also indicates whether the string should be kept in the save table: 0,1 If the system buffer is full, continue execution at the error exit address. Do not enter the string in the save table. 2,3 If the operator is not logged in or the system buffer is full, continue execution at the error exit address. Enter the string in the save table. 4 Send the message to the remote operator. Return r=2 if the remote operator is not logged in.
	eea	Virtual bit address to receive control if an error occurs during processing of this message ($r \neq 0$). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.

Figure 5-21. SEND A MESSAGE TO OPERATOR (f=#001A) Message Format (Sheet 2 of 2)

INITIALIZE OR DISCONNECT CONTROLLEE (f=#001B)

A user program can make another program a controllee, and optionally start the controllee running, by using the INITIALIZE OR DISCONNECT CONTROLLEE message.

It can also be used to disconnect a previously connected controllee. Up to eight levels of program controllees are permitted in a controllee chain, making a possible total of up to nine levels. The format of this message is shown in figure 5-22.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Controllee program is already present. 2 Illegal message option. 3 Controllee program file was not found or not attached. 4 Insufficient time to run the controllee program. 5 Illegal priority. 6 Error in creating the drop file. 7 Controllee program file is not executable. 8 Mass storage device error. 9 Full system tables inhibit initialization of the controllee program at this time. #A Abnormality in the controllee program file or drop file I/O connector entry. #B Nine task levels are present already. #C No controllee is present (for disconnect only); if a controllee program is being disconnected, this response code indicates no controllee program is present. #D Controllee program drop file is too small. #E Unable to destroy the existing drop file. #F Unable to restart the controllee because the interrupt register table is full.

Figure 5-22. INITIALIZE OR DISCONNECT CONTROLLEE (f=#001B) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	r	#10 Drop file cannot be verified.
		#11 C500 request error.
		#12 Bad minus page in the controllee file.
		#13 Undefined error in the drop file verification.
		#14 Controllee program file is privileged open.
		#15 No FST space.
		#17 IOC for file not found.
		#18 User does not have execute access.
		#19 Execute file has wrong small page size.
		#1A Drop file has wrong small page size.
		#1B File is incomplete.
		#1C Charge statement must be supplied.
		#1D SHRLIB is not active.
		#1E Controllee must be reloaded.
		#1F Controllee using wrong libraries.
		#21 Controllee is purge-only.
		#22 Nonproduction program not permitted (production users only).
		#214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. The length of the Beta portion is at least 2.
Alpha(2)	m	Message options:
		0 Initialize the controllee program and restart this program.
		1 Initialize the controllee program and immediately begin running it; stop running this program.
		10 Disconnect the controllee program (the Beta portion of the message is required for this option, but is not used).
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	filename	Name of the controllee program (must be a virtual code file), left-justified with blank fill.
Beta(2)	b	Controllee program's descriptor block number; returned by the operating system when the message is issued. If the controllee program is disconnected and reconnected, this number could change.
	tl	Time limit for the controllee program, in microseconds. When this field is 0, the controller's time limit is used.

Figure 5-22. INITIALIZE OR DISCONNECT CONTROLLEE (f=#001B) Message Format (Sheet 2 of 2)

PROGRAM INTERRUPT CONTROL (f=#001C)

The operating system supports one level of software interrupt for any task. With the PROGRAM INTERRUPT CONTROL message, a user program can tell the operating system whether or not the task can be interrupted. If interrupts are enabled, an ASCII character string must be waiting at the interrupt address specified in Beta(1) of the PROGRAM INTERRUPT CONTROL message at the time control passes to the interrupt address. For control to return to the calling routine, the interrupt routine must issue a RETURN FROM INTERRUPT CONTROL message when it has finished performing its tasks.

When a program is interrupted, the program's minus page is altered before control is passed to the virtual address specified by the user in a PROGRAM INTERRUPT CONTROL message. The minus page has space for the current invisible package (level 0). The interrupt register table has space for the interrupted routine invisible package (level 1). (These level designations are not to be confused with the level of a task in the controllee chain, nor with the security level of a task.) At the time of an interrupt, the level-1 invisible package becomes the current execution invisible package (level 0), and the level-0 invisible package is saved in the interrupt register table. The operating system saves the register file image for the old level 0, and places in register 3 a pointer to the Alpha portion of, and an index to the Beta portion of, the message that caused the interrupt. The operating system also puts into register 1E the length and address of the data base to be used by the interrupt routine. Initializing the rest of the register file is the responsibility of the interrupt routine.

When message option 1 is specified, any ASCII character string preceded by (sc)I that is received from a terminal interrupts the user program (the currently executing program). The symbol (sc) is a special character defined by the installation (refer to volume 1). When the string has been received, the (sc)I preceding the string is stripped and the string is realigned at the beginning of the word. An (sc)I interrupt causes any outstanding output message to be released to the output device. When (sc)I precedes a string, the message interrupts the highest level controller that issued a PROGRAM INTERRUPT CONTROL message with message option 1 (highest level refers to level in the controllee chain).

The format of the PROGRAM INTERRUPT CONTROL message is shown in figure 5-23.

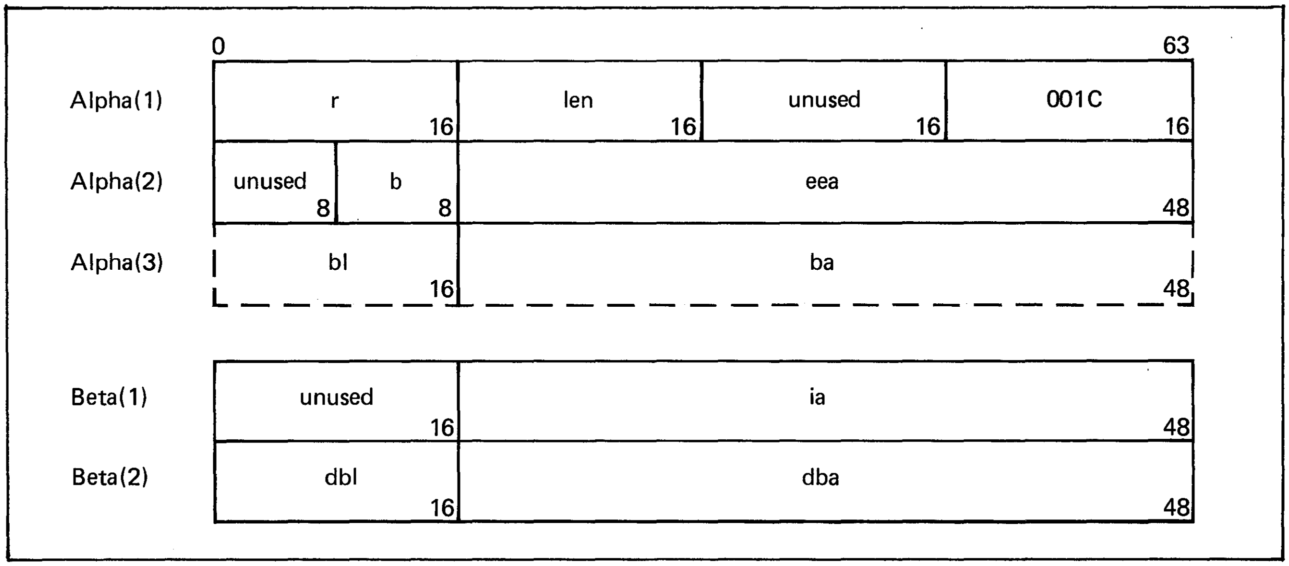


Figure 5-23. PROGRAM INTERRUPT CONTROL (f=#001C) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0, otherwise: 1 Value of the interrupt address is greater than the upper limit of the virtual bit address range. 2 Program selected an illegal interrupt option. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion.
Alpha(2)	b	Interrupt options: 0 This program can be interrupted by any program. 1 This program can be interrupted by a terminal if the data at the interrupt address begins with the two characters (sc)I. 2 This program must not be interrupted. When this message is issued for options 0 or 1, the program issuing this message can be interrupted by all subsequent messages and interrupts coming from a terminal until this program either issues this message with option 2 or terminates.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.
Beta(1)	ia	Interrupt address, the virtual bit address of the word to which control transfers upon occurrence of an interrupt.
Beta(2)	dbl	Length of the data base to be established if an interrupt occurs.
	dba	Address of the data base to be established if an interrupt occurs. If this field is 0, the data base of the interrupted program (the program issuing this message) is used.

Figure 5-23. PROGRAM INTERRUPT CONTROL (f=#001C) Message Format (Sheet 2 of 2)

INITIALIZE CONTROLLEE CHAIN (f=#001D)

A user program can issue this message to make itself the controller of a chain of controllees. Up to nine levels of controllee programs are permitted in any controllee chain (for example, if the program issuing this message is the controllee of a level-1 task, a maximum of seven controllees can be specified in this message). Control is always returned to the user program after the call has been processed; unlike the INITIALIZE OR DISCONNECT CONTROLLEE message (f=#001B), this message cannot be used to start a controllee running.

Any error in the request causes the entire chain to be ignored, and none of the controllees are initialized. The format of this message is shown in figure 5-24.

	0					63
Alpha(1)	r 16	len 16	unused 16	001D 16		
Alpha(2)	n 16	eea 48				
Alpha(3)	bl 16	ba 48				
Beta(1)	s 8	t 8	db 8	unused 16	ss 8	unused 16
Beta(2)	k 8	unused 8	tl 48			
Beta(3)	source 64					
Beta(4)	drop 64					

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Controllee program is already present. 2 Full system tables inhibit initialization of the controllee program at this time. 3 Controllee chain levels exceed nine. 4 Error in an attempt to initialize the controllee; refer to an ss field in Beta. 5 Controllee file is privileged open. #214 Beta buffer length error. Either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and lengths specified.

Figure 5-24. INITIALIZE CONTROLLEE CHAIN (f=#001D) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion (a multiple of 4, and always less than or equal to 14).
Alpha(2)	n	Number of tasks to be initialized in the chain; n is always less than or equal to 9.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	s	Controllee chain level (2 through 9) of the program named in Beta(3); returned by the operating system. Beta words appear in the order in which the controllees are to be initialized, highest to lowest in the chain.
	t	Descriptor block number of the program named in Beta(3); returned by the operating system.
	db	Descriptor block number of the controller of the program named in Beta(3); returned by the operating system.
	ss	File initialization error. The values are: 2 Full system tables inhibit initialization of the controllee program. 3 Controllee program file was not found or was not attached. 4 Insufficient time to run the controllee program. 6 Error in creating the drop file. 7 Controllee program file is not executable. 8 Mass storage error. 9 Abnormality in the controllee program file or drop file I/O connector entry. #A Controllee program file is privileged open. #B No FST space. #D IOC for file not found. #E Bad minus page. #F User does not have execute access for controllee program file. #10 Drop file is too small. #11 Unable to destroy existing drop file. #13 Shared library needed. #14 Controllee must be reloaded for new bound implicit map (BIM). #15 Source file bad small page size. #16 Drop file bad small page size. #17 Drop file is too long. #18 File is incomplete. #19 File has purge-only status.

Figure 5-24. INITIALIZE CONTROLLEE CHAIN (f=#001D) Message Format (Sheet 2 of 3)

Word	Field	Description
Beta(1)	ss	#1A Controllee using wrong libraries. #1C Charge statement must be supplied. #30 Operating system version mismatch. #31 Nonproduction program not permitted (production users only). #32 Drop file cannot be restarted (see site security administrator).
Beta(2)	k	Descriptor block number of the controllee (in this chain) of the program named in Beta(3). This field is 0 if there is no controllee.
	tl	Amount that remains, in microseconds, of the time allowed for running the controllee program. When the controllee has exhausted the time, this field is 0.
Beta(3)	source	ASCII name of the executable source file to be initialized if the drop field is 0. The name is left-justified with blank fill. Returned by the operating system if the drop field is not 0.
Beta(4)	drop	ASCII name of the drop file, left-justified with blank fill. If the user provides a nonzero value in this field, the program is started from this drop file (the drop file contains the I/O connector containing the source file name associated with the drop file).

Figure 5-24. INITIALIZE CONTROLLEE CHAIN (f=#001D) Message Format (Sheet 3 of 3)

ENABLE/DISABLE ATC (f=#0020)

The ENABLE/DISABLE ATC message allows the user to process what is normally a fatal error. The user does this by setting/zeroing the interrupt subroutine address and data base descriptor in the minus page. If abnormal termination control (ATC) is ready, the user may reissue this message to change the interrupt routine information. However, all of the fields must be supplied as if this message had not previously been issued. The format of the message is shown in figure 5-25.

	0			63
Alpha (1)	r 16	len 16	unused 16	0020 16
Alpha (2)	unused 16	eea 48		
Alpha (3)	bl 16	ba 48		
Beta (1)	ito 16	ira 48		
Beta (2)	dbl 16	dba 48		

Word	Field	Description
Alpha(1)	r	Response code returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: #61 Entry point or data base address out of virtual address range for user program or hexadecimal bit address less than 8000. #62 Interrupt type option is illegal. #63 Data base length out of range. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If len=#FFFF, Alpha(3) contains the length(bl) and virtual bit address(ba) of the Beta portion. Otherwise, Beta is assumed to begin at Alpha(3) and len is the length of the Beta portion (two words).
Alpha(2)	eea	Virtual bit error exit address to receive control if an error occurs during message processing (r≠0). If eea=0, the error is fatal.

Figure 5-25. ENABLE/DISABLE ATC (f=#0020) Message Format (Sheet 1 of 2)

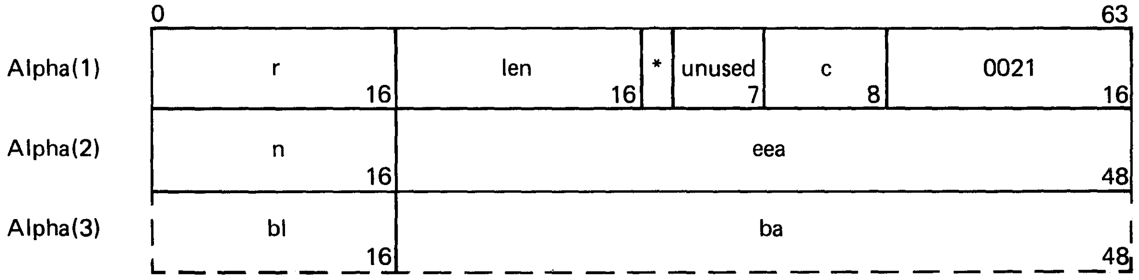
Word	Field	Description
Alpha(3)	bl	Beta length, if len=#FFFF.
	ba	Virtual bit address of the Beta portion´s first full word, if len=#FFFF.
Beta(1)	ito	Interrupt type option: 0 Enable ATC. 1 Disable ATC.
	ira	User interrupt subroutine address (virtual bit address) to be entered if predefined system errors occur.
Beta(2)	dbl	Data base length (in words) of user´s interrupt subroutine.
	dba	Data base bit address of user´s interrupt subroutine.

Figure 5-25. ENABLE/DISABLE ATC (f=#0020) Message Format (Sheet 2 of 2)

EXECUTE OPERATOR COMMAND (f=#0021)

This message can be issued only by a privileged user and executes exactly one of the operator commands that are listed as possible values for the c field in Alpha(1). The user number of the issuer must be the primary, the remote operator, or the site security administrator user number. User numbers running the Q utility are allowed to use options #10, #2c, and #42 only. A nonprivileged user may execute option #2F only for the user number executing the system message.

The format of the Alpha portion of this message is shown in figure 5-26. The Beta word formats depend on the message option (c field) in Alpha(1), and are shown in figure 5-27. Each option is a correspondence between the Beta format and the operator command. (Refer to the VSOS 2 Operator's Guide.)



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed:
		0 No errors.
		1 Illegal option (c field value is out of bounds, or there is a bad parameter).
		2 User is not privileged and is not running under the operator's user number.
		3 Error in a resident system call issued by the virtual system routine OPCOM.
		4 Error in a resident system call issued by the virtual system routine OPCOM.
		5 A user directory editor is already active. Retry later.
		#F4 Beta overflow in message option #F.
	len	This field must always be #FFFF, indicating that Alpha(3) contains the length and virtual bit address of the Beta portion of the message.
	* f_sor	Outstanding messages:
		1 There are outstanding messages awaiting operator response.
		0 There are no outstanding messages.

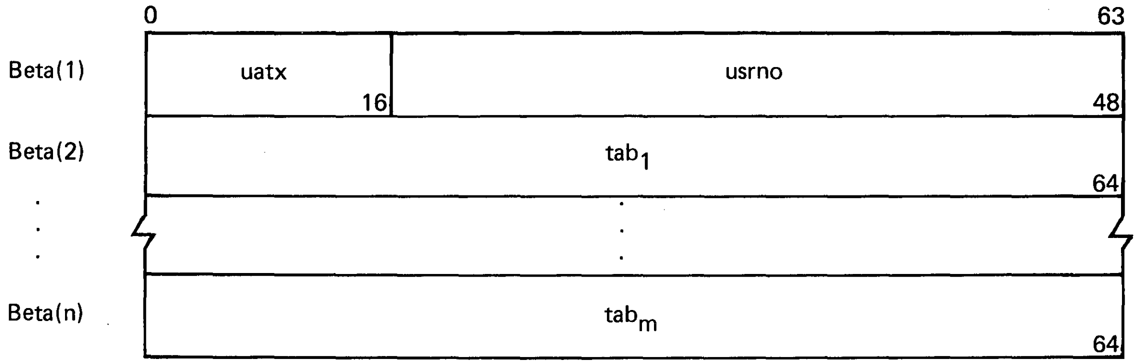
Figure 5-26. EXECUTE OPERATOR COMMAND (f=#0021) (Alpha) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	c	Message options; each of the options has a different Beta format. The Beta formats are described following the Alpha field definitions: 1 Display user. 5 Send a message to the users. 7 Drop (terminate the task and log out the user). This is used only by system checkpoint. #A Return data from virtual system in Beta area for specified virtual address. #B Modify virtual system address with data specified in Beta parameters. #D Date. #E Time. #F Return copy of system configuration table. #10 Return B or Q(E) display information. #11 Display all tasks. #12 Return information for S display. #13 Return information for J display. #16 Suspend or resume the task. #17 List the account. #18 Return the default project number. #19 Turn output processing on or off. #1C Checkpoint jdn. #1D Turn on or off the no login flag. #1E Turn off F_RESTART bit in MISCTAB. #1F Set job category priority. #21 Test and set user directory editor serialization flag. #22 Clear user directory editor serialization flag. #24 Turn on or off job submission to the CPU scheduler. #25 Set maximum large page limit for job category. #26 Set maximum memory overcommitment percentage. #27 Set maximum combined time limit for all executing jobs. #28 Set maximum job time limit for job category. #29 Set maximum working set size limit for job category. #2A Set maximum executing jobs for job category. #2C Retrieve H and Q(I) display information. #2D Retrieve V display information. #2F Retrieve validated job categories for user number. #31 Adjust SHRLIB working set. #32 User drop support functions. #38 Checkpoint functions. #42 Return a set of file index entries for H and Q(O) displays.
Alpha(2)	n	The number of words returned in the Beta portion of the message, the value of which depends on the message option (c field). (Refer to the specific c option description.)
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.

Figure 5-26. EXECUTE OPERATOR COMMAND (f=#0021) (Alpha) Message Format (Sheet 2 of 2)

Message option:

c=1 Return the 16-word UAT entry for one user or the binary user number of all users who are currently on the system. The value of the n field in Alpha(2) is returned by the operating system. The value of n is #FFFF if m is 5, 18 if m is 16, and m otherwise.

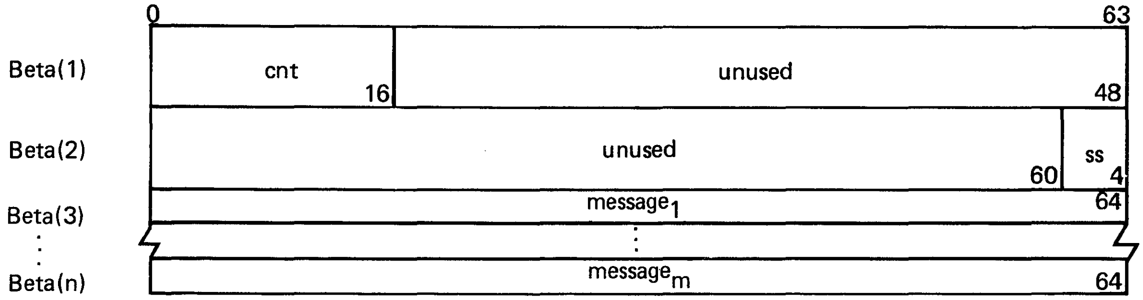


Word	Field	Description
Beta(1)	uatx	Type of information in the tab field; returned by the operating system. If this field is 0, the information is from the user directory. If this field is nonzero, it is from the user activity table; the uatx field contains the user activity table index of the table entry.
	usrno	Binary user number of the user table or user directory entry in the tab fields; or, the character string ALL, left-justified with blank fill, indicating that user numbers and teletypewriter (TTY) numbers of all logged-in users are to be placed in the tab field. Supplied by the user.
Beta(2) through Beta(n)	tab$_i$	The tab$_i$ field contains binary zeros if no user identified by the usrno field was found. Otherwise tab$_i$ contains the first word of a 16-word copy of the user activity table entry (uatx=0) for an active user; a 5-word copy of the first part of the user directory entry (uatx=0) for an inactive user; or m words, one for each currently logged-on (active) user, each of which contains the binary user number in the rightmost 48 bits and a TTY number in the leftmost 16 bits. For an active user, the user activity table is followed by a list of all active accounts, with one account per word.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 1 of 27)

Message option:

c=5 Transmit a specified character string to the output buffer or all interactive users (WARN command):



Word	Field	Description
Beta(1)	cnt	Number of characters in the message.
Beta(2)	ss	Error response field: 0 No error. 1 No users are logged on the system.
Beta(3) through Beta(n)	message	A word of a message in ASCII.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 2 of 27)

Message option:

c=7 Terminate all tasks and log off the user(s) [interactive] or terminate the job(s) [batch]:

	0			63
Beta(1)	dbn 8	sf 8	ss 16	usrno 32

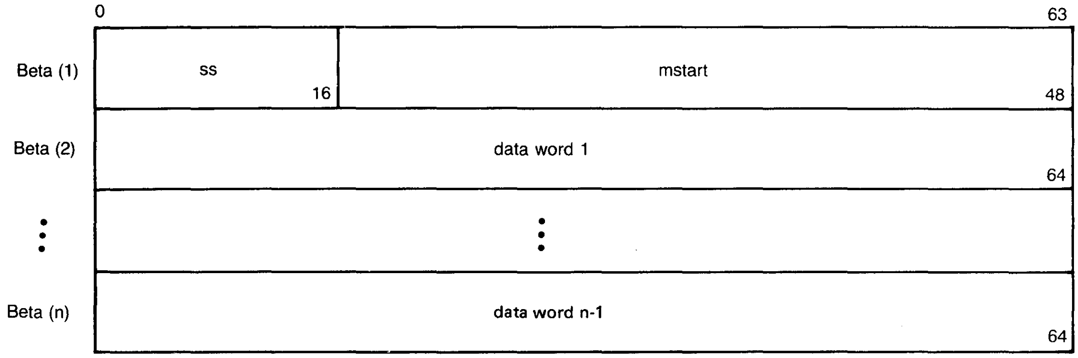
Field	Description
dbn	DB number of task to terminate.
sf	Subfunction, set by the user to one of the following values: 0 Drop job, but allow BATCHPRO to clean up. 1 Rerun job. 2 Drop job and do not allow BATCHPRO to clean up.
ss	Error response field. The operating system sets this field to one of the following values: 0 No error. 1 No user table entry. 2 Attempted to drop operator. 3 Attempted to drop system user.
usrno	Binary user number for the task to be terminated.

If Beta(1) is #FFFFFFFFFFFF, terminate all interactive tasks and log off the interactive users.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 3 of 27)

Message option:

c=#A Return data from virtual system in Beta(2) through Beta(n) for the specified virtual address in Beta(1) word.

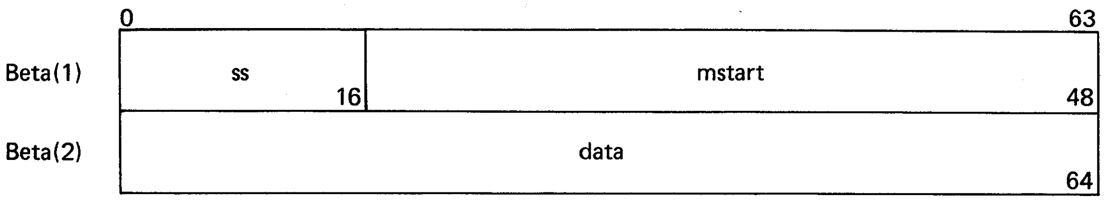


Word	Field	Description
Beta(1)	ss	Error response field. The operating system sets this field to one of the following values: 0 No error. 1 Beta length error from Alpha parameter n (n≠2). 2 Address specified is not in the virtual system. 3 Address specified is not on a word boundary.
	mstart	User-specified virtual system memory starting address on a word boundary.
Beta(2) through Beta(n)	data word 1 through data word n-1	Word of data returned from the virtual system.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 4 of 27)

Message option:

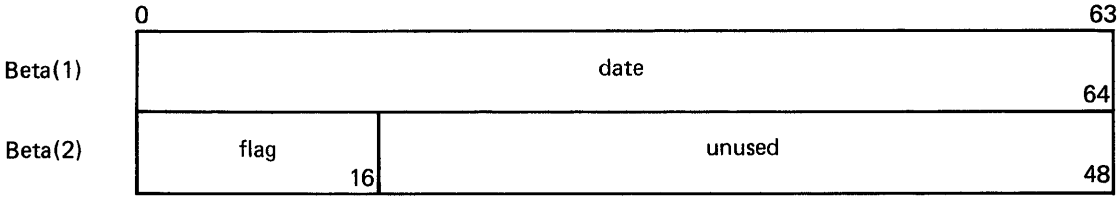
c=#B Modify specified virtual system address with data from BETA parameter.



Word	Field	Description
Beta(1)	ss	Error response field. The operating system sets this field to one of the following values. 0 No error. 1 Beta length error from Alpha parameter n (n≠2). 2 Address specified is not in the virtual system. 3 Address specified is not on a word boundary.
	mstart	User specified virtual memory starting address on a word boundary.
Beta(2)	data	Hexadecimal data to be entered at virtual bit address specified by mstart.

Message option:

c=#D Set or request the date (DATE command):

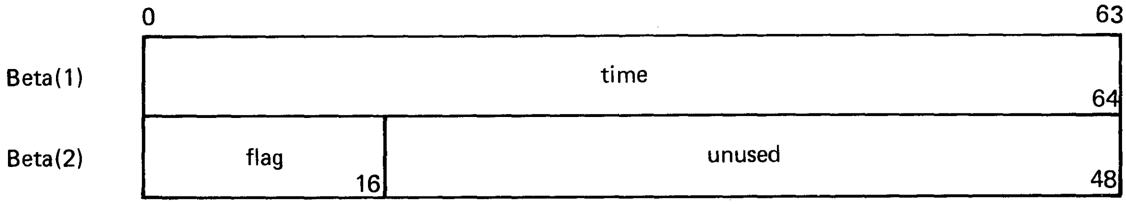


Word	Field	Description
Beta(1)	date	ASCII date, in the format: mm/dd/yy (month/day/year); set by the user if the flag field is 1, or set by the operating system if the flag field is 0.
Beta(2)	flag	This field must be 0 for the data to be returned, or 1 to set the data.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 5 of 27)

Message option:

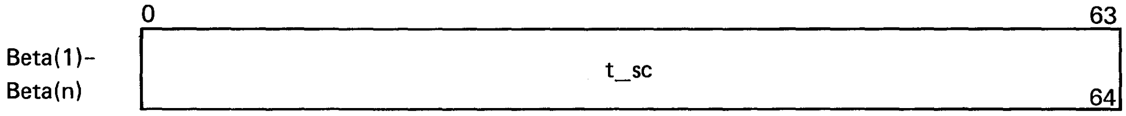
c=#E Set or request the time (TIME command):



Word	Field	Description
Beta(1)	time	ASCII clock time, in the format hh.mm.ss (hour.minutes.seconds); set by the user if the flag field is 1, or set by the operating system if the flag field is 0.
Beta(2)	flag	This field must be 0 for the time to be returned, or 1 for the time to be set.

Message option:

c=#F Return a copy of the system configuration table (T_SC). The length of the Beta area must be supplied by the caller in the Alpha field n. If the supplied length is less than the actual table length, the Beta is filled with as much data as will fit, and an error code of #F4 is returned in the Alpha field r. If the supplied length is greater than or equal to the actual table length, the Alpha field n is set to the table length by the system.



Word	Field	Description
Beta (1) through Beta(n)	t_sc	The system configuration table (SCT).

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 6 of 27)

Message option:

c=#10 Return job and task information for remote operator for the B display and user Q(E) display. A return Beta entry consists of six words each. The number of entries returned is indicated by n in Alpha(2):

	0					63
Beta(1)	unused 8	status 8	pmt 8	pr 8	susflg 8	tl 24
Beta(2)	jdn 12	.wset 16	cbc 16	lp 12	nt 8	
Beta(3)	userno 64					
Beta(4)	job name 64					
Beta(5)	task name 64					
Beta(6)	jce 8	db 8	priv 4	unused 20	lid 24	

Word	Field	Description
Beta(1)	status	Program state number.
	pmt	Priority map table index (0 through 15).
	pr	Subpriority (1 through 255).
	susflg	If the task is in a suspended state, the operating system sets this field to one of the following values: 0 Task was suspended by the operator. 1 Task was suspended by the operating system.
	tl	Time left for this task, in seconds.
Beta(2)	jdn	Job descriptor for this job or session (1 through 2047).
	wset	Number of 512-word blocks in the task´s working set.
	cbc	Number of 512-word blocks currently assigned to the job.
	lp	Number of large pages of memory assigned.
	nt	Number of tape drives reserved for this job or session.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 7 of 27)

Word	Field	Description
Beta(3)	userno	ASCII user number; six characters, left-justified, blank-filled.
Beta(4)	job name	Name of the batch job file, interactive session name, or SYSTEM.
Beta(5)	task name	Name of the file currently executing.
Beta(6)	jce	System job category table index.
	db	Top DE index (BATCHPRO).
	priv	Privileged flag.
	lid	Logical identifier of the front-end system from which the job originated. If the user is not privileged, this option will return only Beta entries that have a userno equal to the user.

Message option:

c=#11 Return information about all active tasks in the system. Three Beta words are returned for each task; the operating system sets the n field in Alpha(2) to the number of tasks in the system. Used during SYSTEM checkpoint/restart only.

	0					63
Beta(1)	unused 8	lev 8	state 8	db 8	jdn 12	unused 20
Beta(2)	fname 64					
Beta(3)	userno 64					

Word	Field	Description
Beta(1)	lev	Level of the task in the controllee chain.
	state	Program state of the task (refer to appendix F).
	db	Descriptor block number for the task.
	jdn	Job descriptor number for the task (1 through 2047).
Beta(2)	fname	Source file name of the task, in ASCII.
Beta(3)	userno	Binary user number under which the task is running.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 8 of 27)

Message option:

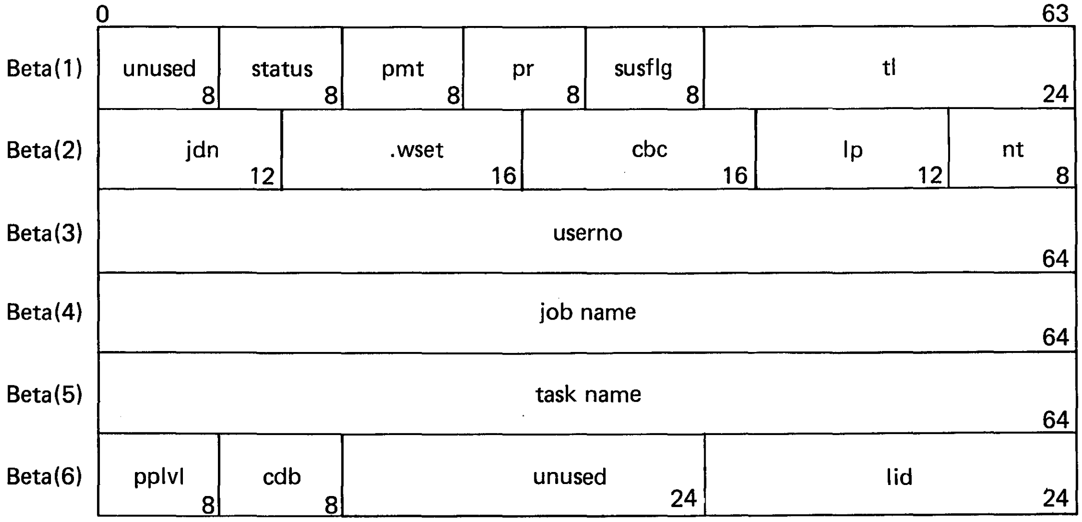
c=#12 Return information for the S display.

	0	63
Beta (1)	number of active users	64
Beta (2)	number of active tasks	64
Beta (3)	total system up time (seconds)	64
Beta (4)	total loads	64
Beta (5)	total cpu time (seconds)	64
Beta (6)	user page faults	64
Beta (7)	system page faults	64
Beta (8)	page faults per cpu second	64
Beta (9)	total kernel time (seconds)	64
Beta (10)	total pager time (seconds)	64
Beta (11)	total virtual time (seconds)	64
Beta (12)	total user time (seconds)	64
Beta (13)	total wait time (seconds)	64
Beta (14)	total idle time (seconds)	64
Beta (15)	percent kernel time	64
Beta (16)	percent pager time	64
Beta (17)	percent virtual time	64
Beta (18)	percent user time	64
Beta (19)	percent wait time	64
Beta (20)	percent idle time	64
Beta (21)	SHRLIB working set	64
Beta (22)	SHRLIB unused pages	64

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 9 of 27)

Message option:

c=#13 Return task EXECUTE OPERATOR COMMAND (f=#0021) information for J display. A six-word entry is returned for each descriptor block in every active task chain for the user. The first 6-word entry is empty. The first word of the Beta field contains the user for whom information is returned.



Word	Field	Description
Beta(1)	status	Program state number.
	pmt	Priority map table index (0 through 15).
	pr	Subpriority (1 through 255).
	susflg	If the task is in a suspend state, the operating system sets this field to one of the following values: 0 Task was suspended by the operator. 1 Task was suspended by the operating system.
	tl	Time left for this task, in seconds.
Beta(2)	jdn	Job descriptor for this job or session (1 through 2047).
	wset	Number of 512-word blocks in the task's working set.
	cbc	Number of 512-word blocks currently assigned to the job.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 10 of 27)

Word	Field	Description
Beta(2)	lp	Number of large pages of memory assigned.
	nt	Number of tape drives reserved for this job or session.
Beta(3)	userno	ASCII user number, six characters, left-justified, blank-filled.
Beta(4)	job name	Name of the batch job file, interactive session name, or SYSTEM.
Beta(5)	task name	Name of the file currently executing.
Beta(6)	pplvl	Level of DB in chain.
	cdb	Ordinal of current DB.
		If the user is not privileged, this option will return only Beta entries that have a userno equal to the user.
	lid	Logical identifier of the front-end mainframe from which the job originated.

Message option:

c=#16 Suspend or resume execution of the task specified. Used during SYSTEM checkpoint/restart only.

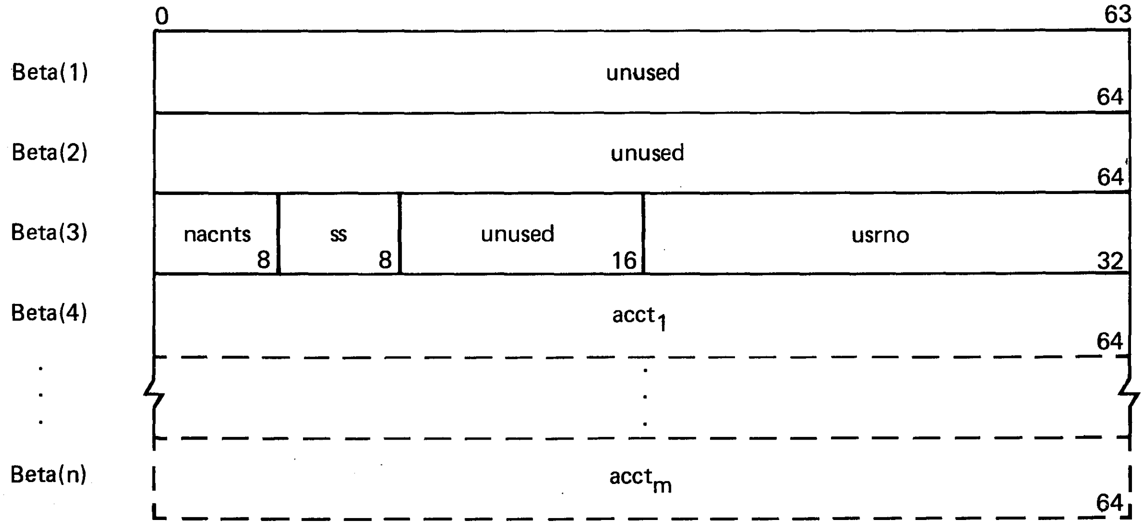
	0			63
Beta(1)	ss	db	sf	unused
	8	8	8	40

Word	Field	Description
Beta(1)	ss	Error response field. The values are: 0 No error. 1 Descriptor block is not assigned. 2 Invalid descriptor block number. 3 Descriptor block is not in a suspended state. 4 Descriptor block is already suspended. 5 Cannot suspend the operator task or system user. 6 Descriptor block is in a terminate or initiating state. 7 Not enough central memory (CM) space to resume task execution.
	db	Descriptor block number of the task to be resumed or suspended.
	sf	Subfunction. This field must be set to 0 to suspend execution, or 1 to resume execution.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 11 of 27)

Message option:

c=#17 List all accounts for a user. The operating system returns the length of the Beta portion in the n field of Alpha(2):

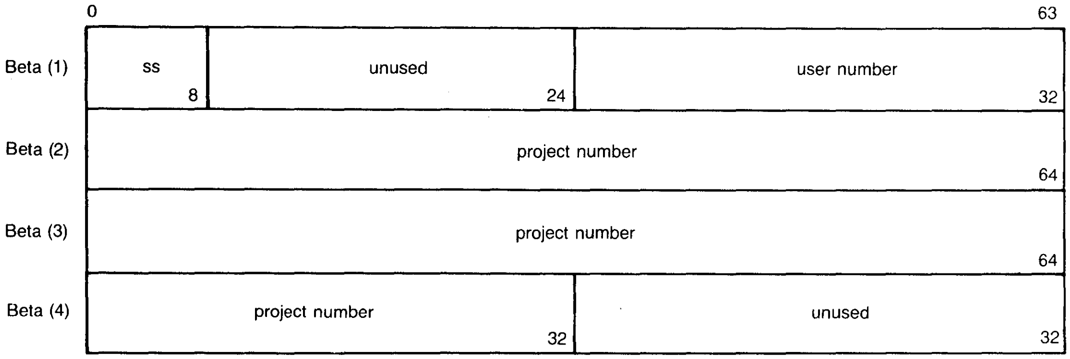


Word	Field	Description
Beta(3)	nacnts	This field is set to the number of accounts listed.
	ss	Error response field. This field is set to 0 for a successfully completed function.
	usrno	User number for which the account is to be modified.
Beta(4) through Beta(n)	$acct_i$	Account number of an account in the system; eight ASCII characters, left-justified with blank fill. Returned by the operating system.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 12 of 27)

Message option:

c=#18 Return the default project number for a user.



Word	Field	Description
Beta(1)	ss	Error response field: 0 No error. 1 Default project number does not exist.
	user number	Binary user number for which the default project number is to to be returned.
Beta(2) through Beta(4)	project number	1- to 20-character default project number, in ASCII, left-justified with blank fill.

Message option:

c=#19 Turn output file processing on or off for all users (OUTP command). If output is turned off, no output files are transferred until output is turned on. The Beta portion consists of one word in which bits 1 through 56 are unused and bits 57 through 64 are the a field:

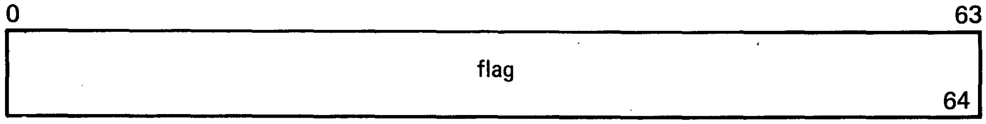
a=0	Turn output on.
a=1	Turn output off.
a=2	Turn output off, save current state.
a=3	Restore output to former state.

Options 2 and 3 are used during checkpoint/restart only.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 13 of 27)

Message option:

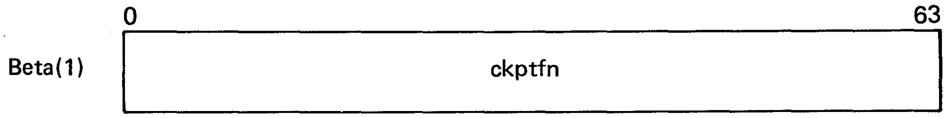
c=#1C	Checkpoint option. Set the checkpointed jdn for all the files belonging to the batch job in the file index. Write file index entries for all local files belonging to user jobs. Switch batch input files to checkpointed input files. Switch output files to checkpointed output files and give back to originating user. Save pool names.
c=#1D	Turn terminal login processing on or off for all users except the operator (LOGO command). If terminal login processing is turned off, no login lines are accepted until terminal login processing is turned on:



Field	Login Terminal Processing Flag	Description
flag	0	Turn on.
	1	Turn off.

Message option:

c=#1E	Restart option. Turns off F_RESTART flag and MIS_CKPT in MISCTAB to indicate that system restart is complete. No Beta is required. Used during system checkpoint/restart only.



Word	Field	Description
Beta(1)	ckptfn	System checkpoint file.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 14 of 27)

Message option:

c=#1F Set a new maximum or default priority for the specified job category (CHPR operator command):

	0			63
Beta(1)	ss 8	sf 8	unused 32	prior 16
Beta(2)	jcat 64			

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Job category not found in system table. 2 Invalid parameter value.
	sf	Subfunction code indicating the priority type to be set: 0 Maximum priority. 1 Default priority.
	prior	Priority: 1 (lowest) through 15 (highest).
Beta(2)	jcat	Job category (eight ASCII characters, left-justified, blank-filled).

Message option:

c=#21 Test and set the user directory editor serialization flag. If not already set, return r=0. If already set, return r=5. No Beta is required.

c=#22 Clear the user directory editor serialization flag. No Beta is required.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 15 of 27)

Message option:

c=#24 Turn on or off the submission of jobs for the specified job category (INPT operator command):

	0		63
Beta(1)	ss 8	unused 48	sf 8
Beta(2)	jcat 64		

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Job category does not exist. 2 Invalid subfunction (sf field value).
	sf	Subfunction code: 0 Turn input on for the specified job category. 1 Turn input off for the specified job category.
Beta(2)	jcat	Job category (eight ASCII characters, left-justified, blank-filled). If jcat is all blanks, the command applies to all job categories.

Message option:

c=#25 Set the maximum large page limit for any job within the specified job category (MXLP operator command):

	0		63
Beta(1)	ss 8	unused 24	n 32
Beta(2)	jcat 64		

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Job category does not exist. 2 Invalid parameter value.
	n	Maximum number of large pages allowed (0 through maximum large pages in machine, MAXWS/128).
Beta(2)	jcat	Job category (eight ASCII characters, left-justified, blank-filled).

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 16 of 27)

Message option:

c=#26 Set the percentage of memory overcommitment allowed (MXMO operator command):

	0		63
Beta(1)	ss 8	unused 24	n 32

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Invalid parameter value.
	n	Integer (0 through 10000) indicating allowed memory overcommitment. The integer is the percentage value.

Message option:

c=#27 Set the maximum combined time limit for all jobs currently executing in the system (MXRR operator command):

	0		63
Beta(1)	ss 8	unused 24	n 32

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Invalid parameter value.
	n	Number of system minutes (1 through 9999).

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 17 of 27)

Message option:

c=#28 Set the maximum time limit for any job within the specified job category (MXTL operator command):

	0		63
Beta(1)	ss 8	unused 24	n 32
Beta(2)	jcat 64		

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Job category does not exist. 2 Invalid parameter value.
	n	Time limit in system seconds (1 through 599,940).
Beta(2)	jcat	Job category (eight ASCII characters, left-justified, blank-filled).

Message option:

c=#29 Set the maximum working set size for any job within the specified job category (MXWS operator command):

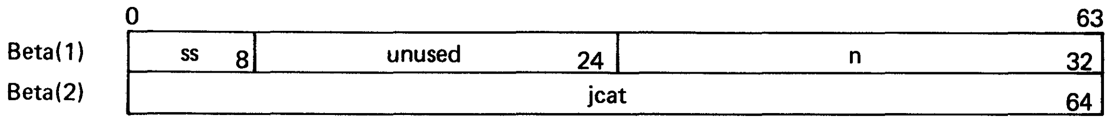
	0		63
Beta(1)	ss 8	unused 24	n 32
Beta(2)	jcat 64		

Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Job category does not exist. 2 Invalid parameter value.
	n	Working set size limit in 512-word blocks (0 through the maximum blocks in machine MAXWS).
Beta(2)	jcat	Job category (eight ASCII characters, left-justified, blank-filled).

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 18 of 27)

Message option:

c=#2A Set the maximum number of executing jobs for the specified job category (SJCT operator command):



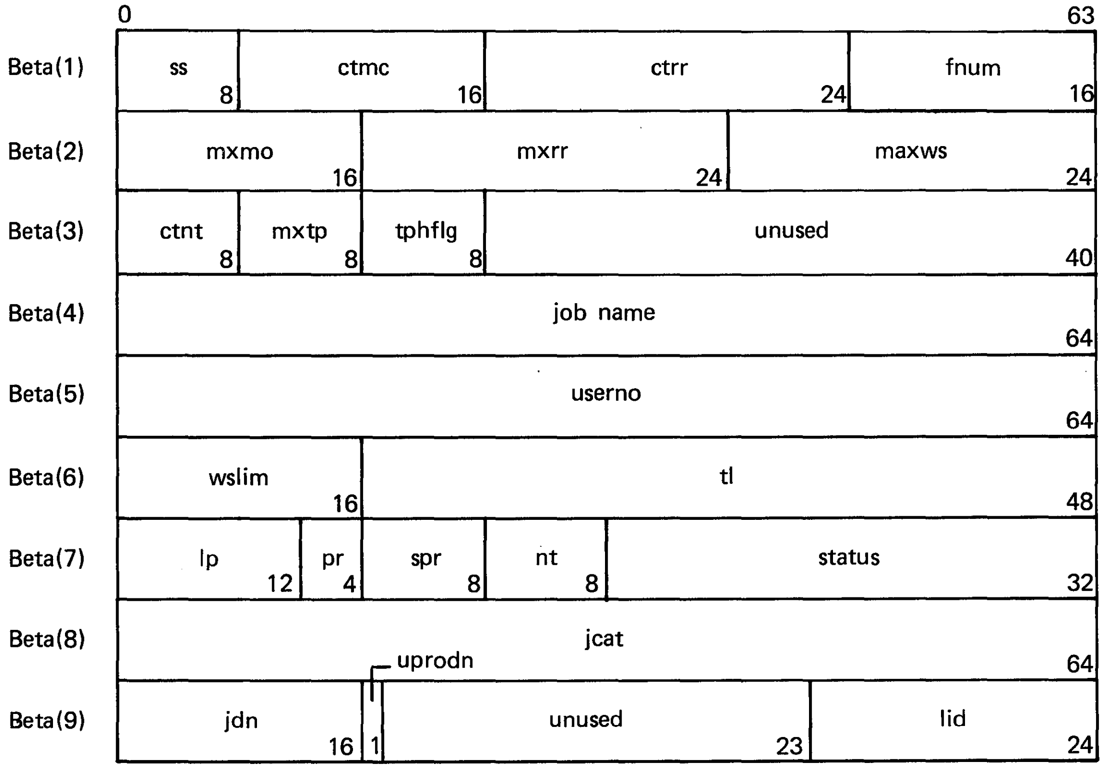
Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 Job category does not exist. 2 Invalid parameter value.
	n	Maximum number of jobs that can concurrently execute (1 to 50).
Beta(2)	jcat	Job category (eight ASCII characters, left-justified, blank-filled).

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 19 of 27)

Message option:

c=#2C Retrieve H,I or Q,I display information for all input queue entries (refer to the VSOS 2 Operator's Guide for a description of the H display). The message returns values in all Beta fields.

For this option, the user must set the value of the bl field in Alpha(3). If the caller is not privileged, any entries belonging to the caller will be returned:



Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 More input queue entries exist than can fit in the reserved Beta area.
	ctmc	Current percentage of memory overcommitted.
	ctrr	Current number of system minutes required to rerun all executing jobs.
	fnum	Number of input queue entries returned.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 20 of 27)

Word	Field	Description
Beta(2)	mxmo	Maximum percentage of memory overcommitment allowed.
	mxrr	Maximum rerun time (in system minutes) allowed for all executing jobs.
	maxws	Maximum working set size.
Beta(3)	ctnt	Number of tape drives not in use.
	mxtp	Maximum number of tape units available.
	tphflg	Tape holding flag. The operating system sets this field to one of the following values: 0 Jobs requiring tape units are not held in the input queue. 1 Jobs requiring tape units are held in the input queue.
Beta(4)	job name	Name of the batch job file.
Beta(5)	userno	User number; six ASCII characters, left-justified, blank-filled.
Beta(6)	wslim	Working set limit, in blocks.
	tl	Time limit, in system seconds.
Beta(7)	lp	Large pages required.
	pr	Priority (1 through 15).
	spr	Subpriority (0 through 255).
	nt	Number of tape drives required by the job.
	status	Input queue status (four ASCII characters, left-justified, blank-filled).
Beta(8)	jcat	Job category (eight ASCII characters, left-justified, blank-filled).
Beta(9)	jdn	Job descriptor number (1 through 2047).
	uprodn	1 is returned if user is a production user. 0 is returned if user is not a production user.
	lid	Logical identifier of the front-end mainframe from which the job originated.

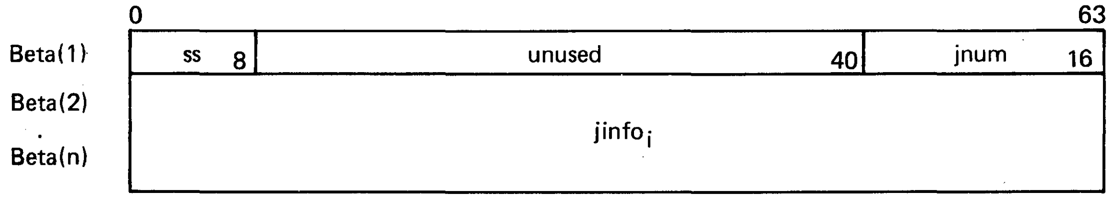
The words Beta(4) through Beta(9) are repeated for every input queue entry returned.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 21 of 27)

Message option:

c=#2D Retrieve V display information for all job categories (refer to the VSOS 2 Operator's Guide for a description of the V display). The message returns values in all Beta fields.

For this option, the user sets the value of the bl field in Alpha(3). The system returns the information in the Beta:



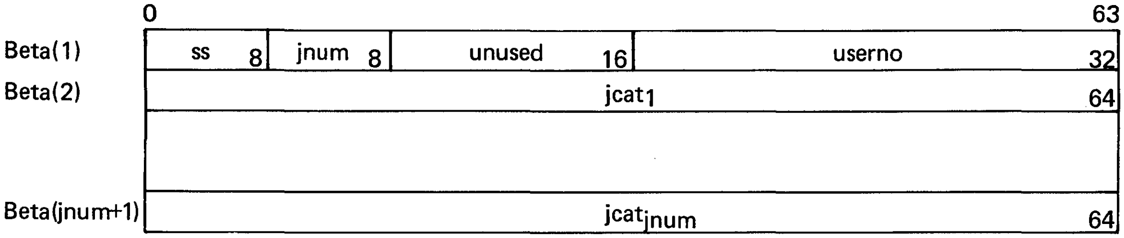
Word	Field	Description
Beta(1)	ss	Error response code: 0 No error. 1 More entries exist than can fit in the Beta.
	jnum	Length of job category table returned in Beta. (The entire table is returned.)
Beta(2) through Beta(n)	$jinfo_i$	Copy of the T_JCAT system table (four words per job category). The format of the T_JCAT table is shown in figure 5-28.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 22 of 27)

Message option:

c=#2F Retrieves job categories for which the specified user number is validated.

For this option, the user must set the bl field of the Alpha to the Beta length. The length must not be less than the number of valid job categories plus 1:



Word	Field	Description
Beta(1)	ss	Error response code: 0 No errors. 1 Beta length specified in the bl field of the Alpha portion is insufficient for return of all valid job categories. 2 User number not found.
	jnum	In this field, the system returns the number of job categories it returned.
	userno	Binary user number whose valid job categories are returned. For a nonprivileged user executing this option, the userno must be the executing user number.
Beta(2) through Beta(n)	jcat_i	Job category mnemonic (one to eight ASCII characters, left-justified, blank-filled).

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 23 of 27)

Message option:

c=#31 Alters the working set for the system shared library. Virtual system routine OPCOM uses mis_slmax to mis_slmin for a legal range when adjusting the SHRLIB working set.

	0		63
Beta(1)	unused 32	minws 16	maxws 16

Word	Field	Description
Beta(1)	minws	Minimum working set size for SHRLIB to be placed in mis_slmin.
	maxws	Maximum working set size for SHRLIB to be placed in mis_slmax.

Message option:

c=#32 Drops jobs for a user. Only one of the jdn or userno fields must be specified in the Beta. The format of the Beta is as follows:

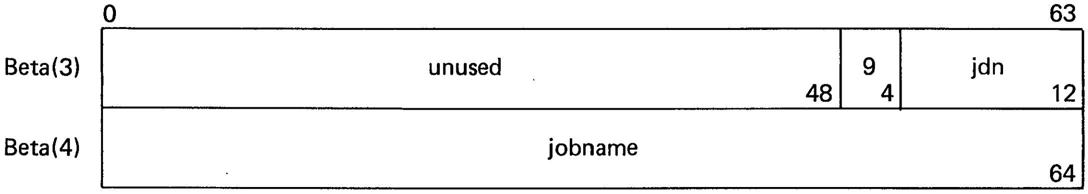
	0			63
Beta(1)	ss 16	sf 16	jdn 12	userno 20
Beta(2)	jobname 64			

Word	Field	Description
Beta(1)	ss	Error response field. Set by the operating system to one of the following values: 0 No error. 1 Nonprivileged user cannot specify alternate user. 2 No jobs found for specified user number. 3 Cannot drop interactive task. 4 Cannot drop task issuing request. 5 Both jdn and jobname specified. 6 Beta is not large enough. 7 Beta length greater than 512 words.
	sf	Subfunction (set by the user). If set, bits 0 through 15 (left to right) have the following meaning: 0 Kill the job. 1-12 Not used. 13 Search the input queue. 14 Search the execute queue. 15 Search the output queue.

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 24 of 27)

Word	Field	Description
	jdn	Job descriptor number (binary) of the job to drop (set by the user). If 0, then jdn is not used as a qualifier to search for jobs and jobname must be specified.
	userno	User number (binary) of the job(s) that are to be dropped (set by the user). If userno is set to 0, then the user number of the task that issued the request is used. If the user is not privileged, then userno must be set to 0 or set to the user number of the task that issued the request.
Beta(2)	jobname	Job name (ASCII, left-justified, blank-filled) of the job(s) to be dropped (set by the user). If 0, then jobname is not used as a qualifier to search for jobs and jdn must be specified. If jobname is * (left-justified, blank-filled), all jobs belonging to userno are dropped.

For each job that is dropped, a 2-word entry is returned in the Beta. The returned entries begin at word 3 of the Beta. The format of each entry is as follows:



Word	Field	Description
Beta(3)	q	Queue flags. When set, bits 0 through 3 (left to right) indicate in which queue the specified job was found: 0 Not used. 1 Input queue. 2 Execute queue. 3 Output queue.
	jdn	Job descriptor number (binary) of the job that was dropped.
Beta(4)	jobname	Job name (ASCII, left-justified, blank-filled) of the job that was dropped.

Message option:

c=#38 Perform support functions for system checkpoint processing.

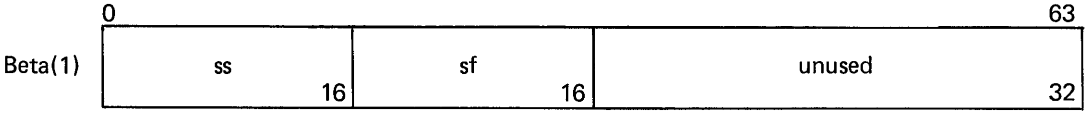


Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 25 of 27)

Word	Field	Description
Beta(1)	ss	Returned error: 1 Called when checkpoint not in progress. 2 Caller is not operator. 3 Invalid subfunction. #10 I/O will not complete interactive task during subfunction #10 processing. #20 Not all batch jobs were suspended during subfunction #20 processing.
	sf	Subfunction code indicating the type of support processing: #10 Remove interactive users from the system. #20 Suspend all batch jobs that are in a running state. #30 Drop ITFS if running.

Message option:

c=#42	Return a set of file index entries for a specified group of files. The value of the n field in Alpha(2) is set by the user. This option is used for the H, H(0), and H(P) displays. For the Q(0) display where sf2=1, only files created by a caller that is not privileged will be returned:

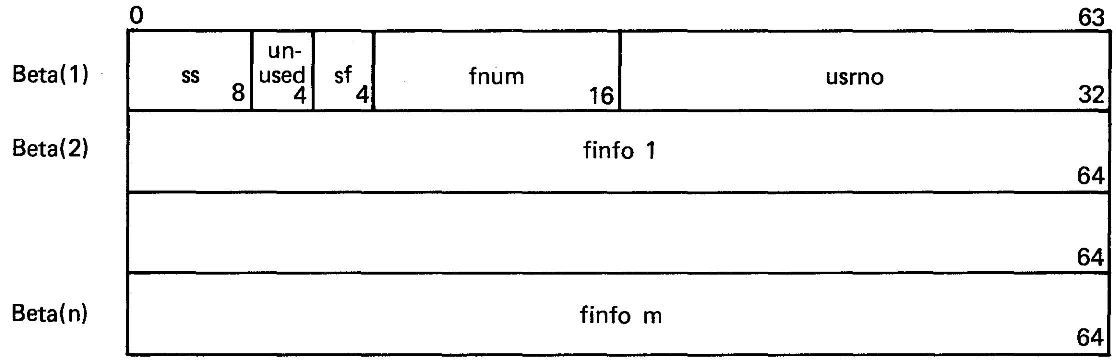


Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 26 of 27)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>Beta(1)</td><td>ss</td><td>Returned error:<table><tr><td>0</td><td>No error.</td></tr><tr><td>1</td><td>User is inactive.</td></tr><tr><td>2</td><td>Invalid subfunction.</td></tr></table></td></tr>
<tr><td></td><td>sf</td><td>Set of flags for the file set desired. Only one flag must be set:<table><tr><td>sf1</td><td>sf2</td><td>sf3</td><td>sf4</td></tr></table><table><tr><th>Flag</th><th>Description</th></tr><tr><td>sf1=1</td><td>Public files.</td></tr><tr><td>sf2=1</td><td>Print files.</td></tr><tr><td>sf3=1</td><td>Punch files.</td></tr><tr><td>sf4=1</td><td>Private files for the user specified in the usrno field.</td></tr></table></td></tr>
<tr><td></td><td>fnum</td><td>Number of files in this file set. This number is returned by the operating system.</td></tr>
<tr><td></td><td>usrno</td><td>Binary user number if sf4=1.</td></tr>
<tr><td>Beta(2) through Beta(n)</td><td>finfo i</td><td>A 16-word copy of the file index table entry for each file in the set specified is returned:<table><tr><td>Public files</td><td>Return the files for user 0.</td></tr><tr><td>Print files</td><td>Search system processor table (SPT) for zip codes with disposition code=#20 (print). Return all mcat=output files for the associated user number. Files with names PYYxxxxx and Q5Lxxxxx are assumed to have a disposition code of #20.</td></tr><tr><td>Punch files</td><td>Search SPT for zip codes with disposition code=#10 (punch). Return all mcat=output files stored under the associated user numbers.</td></tr><tr><td>Private files</td><td>Search FILEI for all files under the specified user number.</td></tr></table></td></tr>
</table>

Figure 5-27. EXECUTE OPERATOR COMMAND (f=#0021) (Beta) Message Format (Sheet 27 of 27)

	0						63
0	jcat$_1$ 64						
	⋮						
65	jcat$_{66}$ 64						
66	mxws 16	reserved 16	mxtl 32				
	⋮						
131	mxws 16	reserved 16	mxtl 32				
132	mxlp 16	mp 8	dp 8	ioo 8	mjc 8	cjc 8	unused 8
	⋮						
197	mxlp 16	mp 8	dp 8	ioo 8	mjc 8	cjc 8	unused 8
198	class	df+1					
	⋮						
263	class	df+1					

Figure 5-28. T_JCAT System Table Format (Sheet 1 of 2)

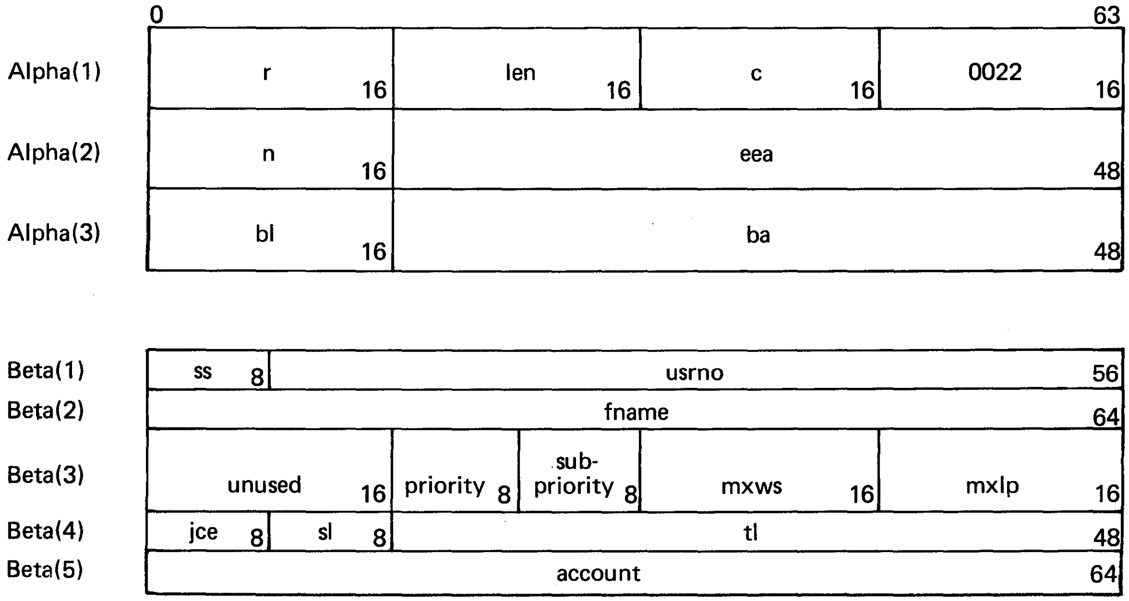
Word	Field	Description
0 - 65	jcat	Job category mnemonic (ASCII left-justified, blank-filled).
66 - 131	mxws	Maximum working set size limit, in blocks.
	mxtl	Maximum time limit, in system seconds.
132 - 197	mxlp	Maximum large page limit.
	mp	Maximum priority.
	dp	Default priority.
	ioo	Input on or off status: 0 Input on. 1 Input off.
	mjc	Maximum job count; the maximum number of jobs from this category that can execute concurrently.
	cjc	Current job count.
198 - 263	class	Job category class mnemonic (four ASCII characters, left-justified, blank-filled).
	df+l	Default category time limit.

Figure 5-28. T_JCAT System Table Format (Sheet 2 of 2)

EXECUTE PROGRAM FOR USER NUMBER (f=#0022)

The EXECUTE PROGRAM FOR USER NUMBER system message initiates execution of a file for a specified user number. Only a privileged user can issue this message. The initiated file can be local, attached permanent, attached pool, or public.

The message format is shown in figure 5-29.



Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 User is not privileged. 2 File request error [refer to the ss field in Beta(1)]. 3 No table space available. 4 Invalid message option (c field). #214 Beta buffer length error. Either the first word address of Beta plus its length is greater than the maximum virtual user address, or the Beta buffer is too small for the number of requests and length specified.
	len	Beta location. If this field is #FFFF, Alpha(3) contains the length and virtual address of Beta. If this field is not #FFFF, the Beta buffer immediately follows word Alpha (2) and contains one word.
	c	Message option: 2 Switch task to user number. (This is the only option.)

Figure 5-29. EXECUTE PROGRAM FOR USER NUMBER (f=#0022) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	n	Number of files put into execution with this message.
	eea	Error exit address; virtual bit address to receive control if an error occurs during message processing (r≠0). If this field is 0 when an error occurs, the task is aborted.
Alpha(3)	bl, ba	If len=#FFFF, these fields contain the length in words and the virtual bit address of the first full word of the Beta portion.
Beta(1)	ss	Error response code: 0 No errors. 2 User directory is not on the disk. 3 File was not found or not attached. 4 Nonexecutable file. 5 Invalid priority value. 6 Invalid user number. 7 Not authorized to run at priority level. 8 Invalid security level specified. 9 File access conflict with another user. #A FILEI (file index table) is full. #B Requested time limit exceeds time available to the user. #C User does not have execute access for the file. #D Nonproduction program not permitted (production users only). #E No JDNs available to assign to user program.
	usrno	User number (six ASCII characters, left-justified, blank-filled).
Beta(2)	fname	File name (eight ASCII characters, left-justified, blank-filled).
Beta(3)	priority	Job priority 1 (lowest) through 15 (highest).
	subpriority	Subpriority of the job (1 through 255).
	mxws	Maximum working set limit in blocks (0 through the maximum blocks in machine, MAXWS in MISCTAB).
	mxlp	Maximum large page limit (0 through the maximum large pages in machine, MAXWS/128).
Beta(4)	jce	Job category entry number (0 through 65).
	sl	Security level (1 through 8) to be given to the task. Default is an installation-defined parameter.
	tl	Time limit in system seconds (0 through 599,940). The default (+1=0) is the maximum amount of time allocated for a user number.
Beta(5)	account	Account number (eight ASCII characters, left-justified, blank-filled).

Figure 5-29. EXECUTE PROGRAM FOR USER NUMBER (f=#0022) Message Format (Sheet 2 of 2)

UPDATE USER DIRECTORY (f=#0023)

This message can be issued only by a privileged user and allows the user to create, delete, or modify a user directory. One purpose for which an installation can use this message is to create a utility for managing batch job accounting (refer to Accounting, chapter 8). A nonprivileged user may execute option #4 only for the user number executing the call.

The format of the Alpha portion of this message is shown in figure 5-30. The Beta word formats depend on the message option (c field) in Alpha(1), and are shown in figure 5-31. Only one Beta will be processed per Alpha.

	0			63
Alpha(1)	r 16	l 16	c 16	0023 16
Alpha(2)	unused 16	eea 48		
Alpha(3)	bl 16	ba 48		

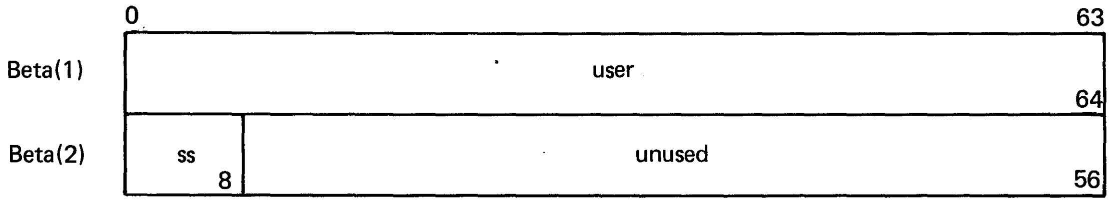
Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. #214 Beta buffer length error. Either the first word address of Beta plus its length is greater than the maximum virtual user address, or the Beta buffer is too small for the number of requests and length specified.
	l	If this field is #FFFF, bl in Alpha(3) contains the length of the remote Beta buffer. ba contains the location of the remote Beta buffer. If this field is not #FFFF, the Beta buffer immediately follows word Alpha(2) and contains one word.
	c	Message options, each of which has a different Beta format: 1 Delete user from user directory. 2 Replace password for user. 3 Write (modify/create) user directory entry and encode password. 4 Read user directory entry. 5 Write (modify/create) user directory entry, do not encode password. 6 Verify user directory permissions. 7 Read user accounts. 8 Initialize user directory (delete all but special users). 9 Read user directory file into caller´s buffer. #A Return default project number for a user. #B Return first eight words for user activity table entry.

Figure 5-30. UPDATE USER DIRECTORY (f=#0023) (Alpha) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message. If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (l=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.

Figure 5-30. UPDATE USER DIRECTORY (f=#0023) (Alpha) Message Format (Sheet 2 of 2)

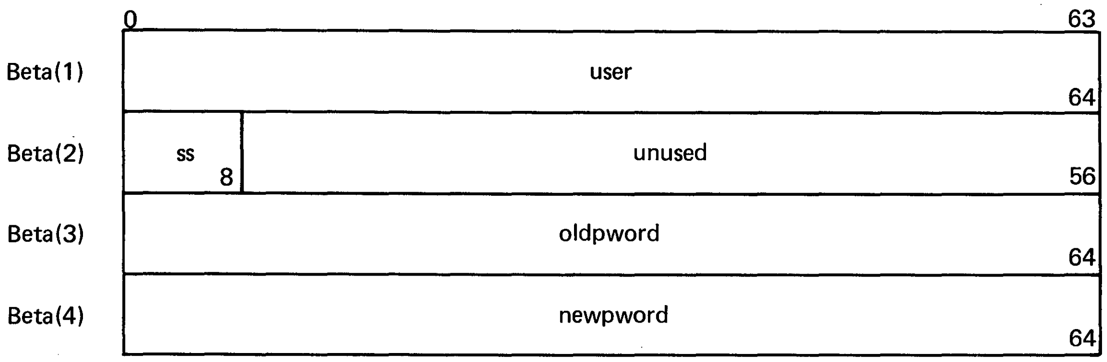
For message option 1:



Word	Field	Description
Beta(1)	user	Binary user number.
Beta(2)	ss	Error response code: 0 No error. 1 User is not privileged. 2 Illegal c option. 3 User is active (c=1, 3, 5). 4 User directory is full (c=3, 5, crflg≠0).

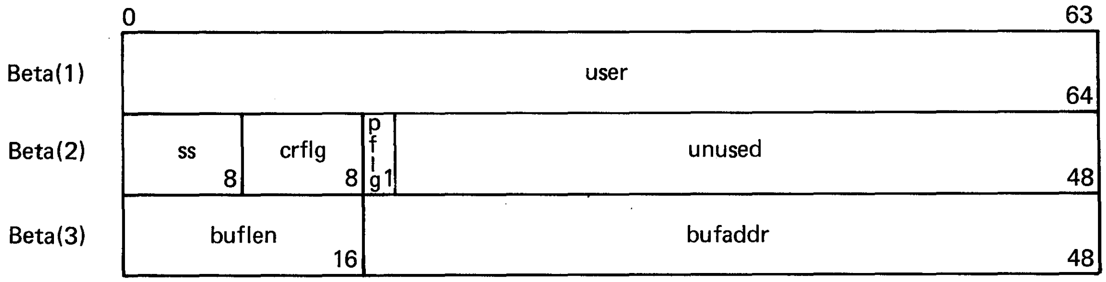
Figure 5-31. UPDATE USER DIRECTORY (f=#0023) (Beta) Message Format (Sheet 1 of 5)

For message option 2:



Word	Field	Description
Beta(1)	user	Binary user number.
Beta(2)	ss	Error response code: 0 No error. 1 User issuing the call is not privileged. 2 Illegal c option. 5 User does not exist. 6 Invalid password. 7 Passwords do not match.
Beta(3)	oldpword	User´s password in ASCII before the issuance of this call.
Beta(4)	newpword	User´s new password in ASCII after the successful completion of this call. This field is set to 0 on return.

For message options 3, 4, 5, 9, and #B:



Word	Field	Description
Beta(1)	user	Binary user number.

Figure 5-31. UPDATE USER DIRECTORY (f=#0023) (Beta) Message Format (Sheet 2 of 5)

Word	Field	Description
Beta(2)	ss	Error response code: 0 No error. 1 User issuing the call is not privileged. 2 Illegal c option. 3 User is active (c=3, 5). 4 User directory is full (c=3, 5, and crflg≠0). 5 User does not exist (c=4, #B; c=3, 5 and crflg=0). 6 Invalid password (c=2). 8 User already exists (c=3, 5, and crflg≠0). 9 User in Beta does not match the user in the buffer (c=3, 5). #B Buffer is not long enough to contain user directory (c=9). #E Caller is not site security administrator (only if production status is selected for the specified user).
	crflg	If 0, this request is to modify an existing user's user directory entry. If set to #7F, this request is to create a new user directory entry.
	pflg	If pflg=1, the production user flag is set in the user directory entry (c=3 or 5).
Beta(3)	buflen	Length of the buffer specified at bufaddr. For c=3, 4, 5, and 7, buflen is in words. For c=#B, buflen should be set to 8. For c=7, buflen is returned to the caller and is the number of accounts read. For c=9, buflen is in blocks.
	bufaddr	Virtual bit address of the buffer to or from which data is moved.

For message option 6:

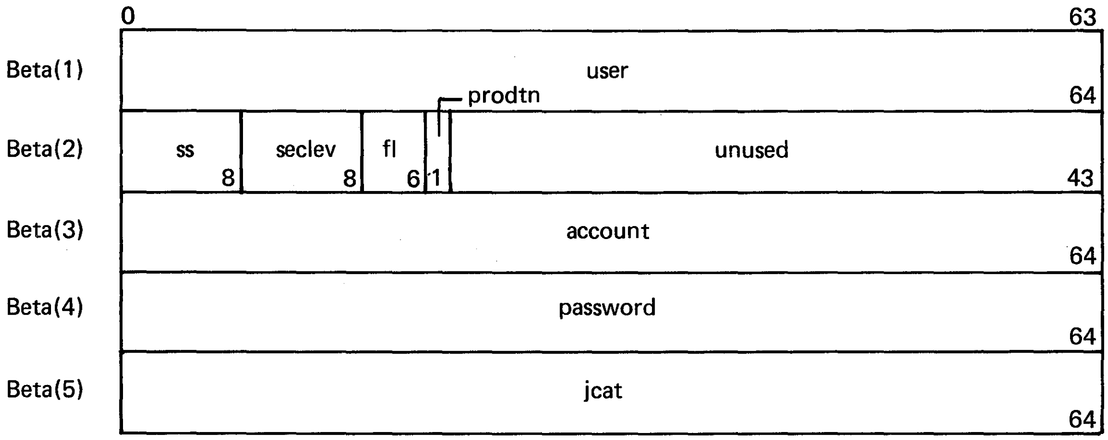
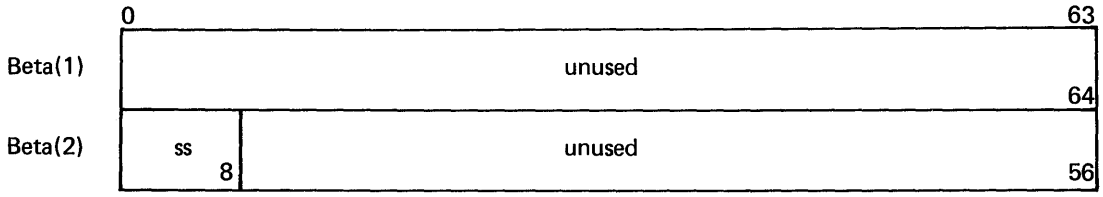


Figure 5-31. UPDATE USER DIRECTORY (f=#0023) (Beta) Message Format (Sheet 3 of 5)

Word	Field	Description
Beta(1)	user	Binary user number.
Beta(2)	ss	Error response code: 0 No error. 1 User issuing the call is not privileged. 2 Illegal c option. 5 User does not exist. #A Parameter in Beta does not match parameter in user directory. #C Specified job category does not exist.
	seclev	Security level to be verified.
	fl	This field is comprised of six subfields: Bit / Subfield / Description 0 acflg If set, verify account identifier. 1 pwflg If set, verify password. 2 slflg If set, verify security level. 3 jcflg If set, verify job category (jcat). 4 daflg If set, return default account. 5 tpflg If set, verify tape access. Account verification, if requested, occurs first. Password verification, if requested, precedes security level verification, which precedes job category verification, which precedes tape access verification. As a parameter is successfully verified, the associated flag in fl is cleared to 0. On detection of an unsuccessful verification, all verification processing stops. The system returns an ss code of #A, and the flag (and all following flags) associated with the unsuccessful verification in fl is still set. If both actflg and daflg are set, the account in Beta(3) will be verified. If the account doesn't verify, then the default account will be returned in Beta(3).
	prodtn	Returned value = 1 if a production user. Returned value = 0 if not a production user.
Beta(3)	account	Account identifier to be verified, in ASCII.
Beta(4)	password	Password to be verified, in ASCII. This field is set to 0 on return.
Beta(5)	jcat	Job category to be verified, in ASCII.

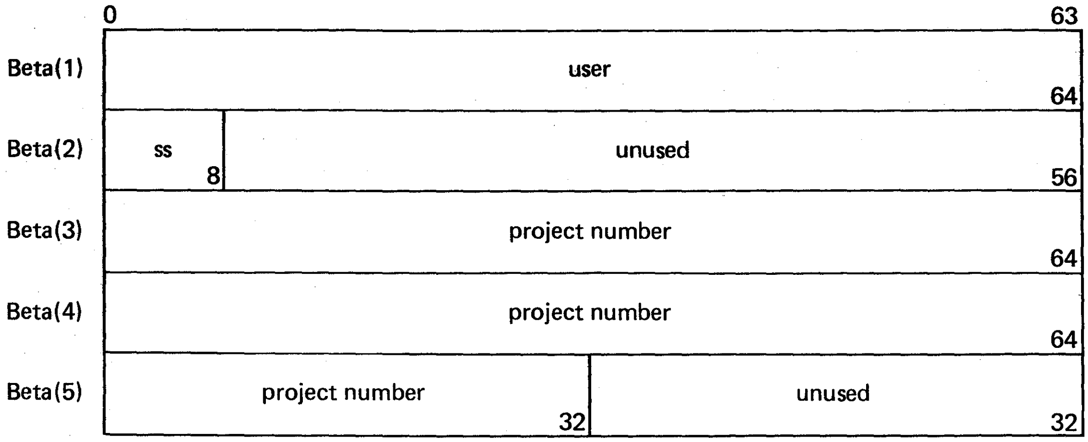
Figure 5-31. UPDATE USER DIRECTORY (f=#0023) (Beta) Message Format (Sheet 4 of 5)

For message option 8:



Word	Field	Description
Beta(2)	ss	Error response code: 0 No error. 1 User issuing the call is not privileged. 2 Illegal c option.

For message option #A:



Word	Field	Description
Beta(1)	user	Binary user number.
Beta(2)	ss	Error response code: 0 No error. 1 User issuing the call is not privileged. 2 Illegal c option. 5 User does not exist. #D Default project number does not exist.
Beta(3), (4), (5)	project number	A 1- to 20-character default project number in ASCII, left-justified with blank fill.

Figure 5-31. UPDATE USER DIRECTORY (f=#0023) (Beta) Message Format (Sheet 5 of 5)

MISCELLANEOUS (f=#0024)

With this message, a user program can determine a variety of information concerning itself, its controller, and its controllees. Also, raw accounting statistics can be retrieved with option 9 (c field). The format of this message is shown in figure 5-32. The Beta portion of the format is discussed under the c field description.

	0			63
Alpha(1)	r 16	len 16	c 16	0024 16
Alpha(2)	unused 16	eea 48		
Alpha(3)	bl 16	ba 48		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs the response code is 0; otherwise: 1 No controllee, illegal information option, or the error response field for Beta(1) of option value C, 12, or 13 is set. 2 Program not currently running in interrupt program state (c=#F or #11 only). 3 Error in system call to read the user's account numbers. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion. For option 8, if this field is 3, Beta(4) and Beta(5) are not present. For option 1, if this field is 3, Beta(4) through Beta(9) are not present.

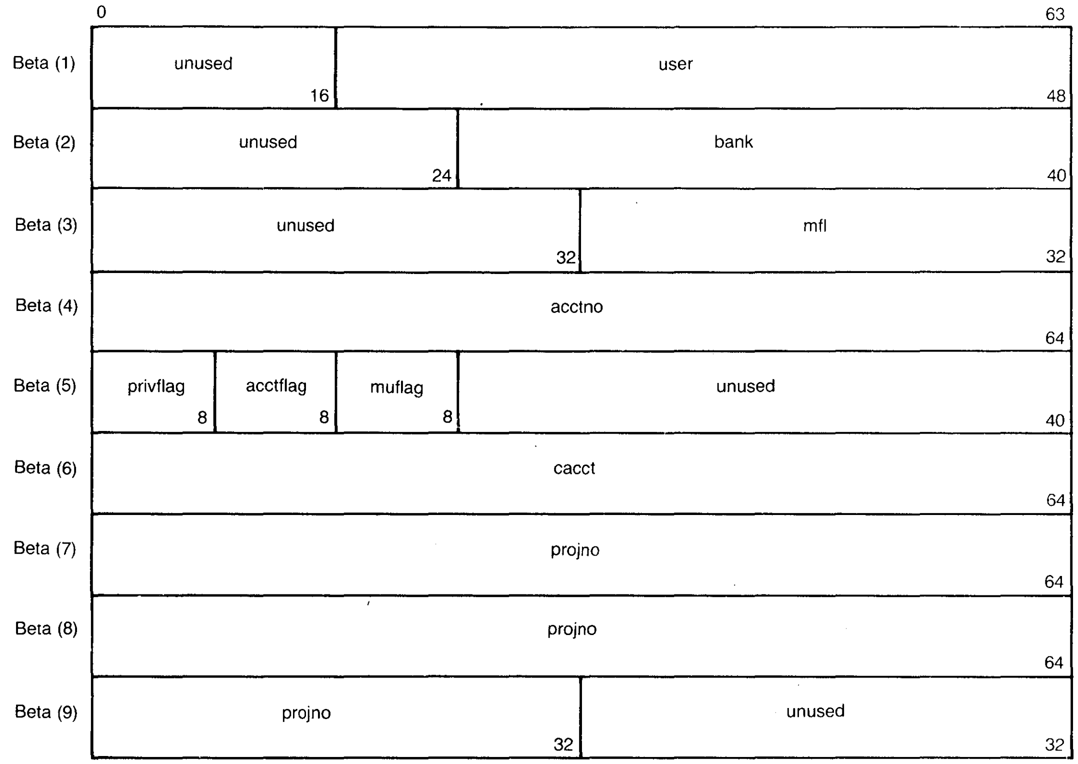
Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 1 of 8)

Word	Field	Description
Alpha(1)	c	Message options (c=4, c=#B, and c=#14 are reserved). All information is returned by the operating system unless otherwise specified. The values are: 1 Get the user number, bank account, and maximum file length. 2 Verify logged-in workstation user. 3 Get the time limit. 4 Reserved. 5 Get the controllee´s termination state. 6 Get the controllee´s name and place. 7 Get the controller´s name and place. 8 Get this program´s name and place. 9 Get the raw page fault counts, CPU times, and memory usage. If the batch processor issues this option, the statistics returned include the cumulative statistics for the batch processor and all its controllees. The Beta portion of the format is as follows: Beta(1) Small page fault count. Large page fault count. Beta(2) CPU time (microseconds). Beta(3) Memory usage. Beta(4) System CPU time. #A Get clock times as of message issuance. #B Reserved. #D Get the contents of minus pages. #E Get the version identifiers. #F Get the interrupt invisible package. #10 Get the task CPU time. #11 Get the interrupted register file. #12 Destroy batch job´s input file if system fails. #13 Rerun batch jobs input file if system fails. #14 Reserved for DEBUG.
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 2 of 8)

Message option:

c=1 Get the user number, bank account, and maximum file length. If Beta(4) is set to an account number, validate the account and return additional parameters in Beta(5) through Beta(9). (The len field must be set to either 3 or 9.)



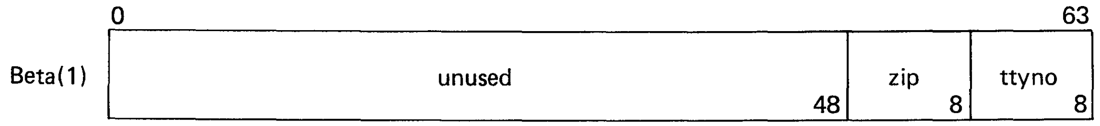
Word	Field	Description
Beta(1)	user	User number in ASCII, right-justified with blank fill.
Beta(2)	bank	Quantity of STU units in the user´s bank account.
Beta(3)	mfl	Maximum file length.
Beta(4)	acctno	Eight character account number to be verified.

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 3 of 8)

Word	Field	Description
Beta(5)	privflag	User privileged flag: 0 Nonprivileged. 1 Privileged.
	acctflag	Account number valid flag: 0 Invalid account. 1 Valid account.
	muflag	Master user flag: 0 User is not the master user of the account. 1 User is the master user of the account.
Beta(6)	cacct	Current account number in execution (in ASCII, left-justified with blank fill).
Beta(7) through Beta(9)	projno	1- to 20-character project number in execution (in ASCII, left-justified with blank fill).

Message option:

c=2	Verify that a user program executing a workstation utility is permitted to communicate with the correct workstation zip code. If the user is making the call from a workstation, the zip and ttyno are returned in Beta(1).

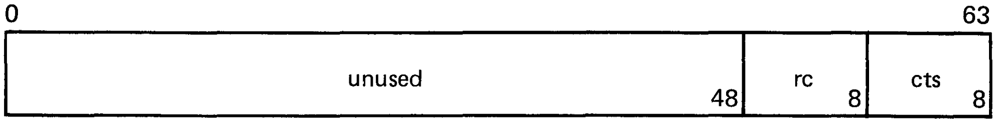


Word	Field	Description
Beta(1)	zip	Zip code of workstation.
	ttyno	Terminal number of logged-on user.

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 4 of 8)

Message option:

c=3	Get the time limit. Beta(1) contains the existing time limit, in microseconds, in the rightmost 48 bits; the leftmost 16 bits are unused.
c=5	Get the controllee's termination state. Beta(1) contains the following:

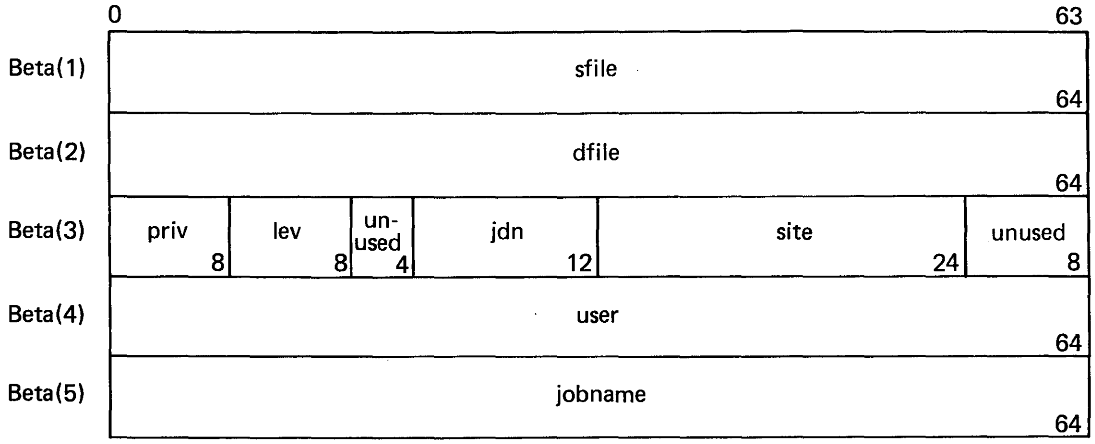


Field	Description
rc	Controllee's return code. rc=0 means successful completion; other values are: 4 Nonfatal error. 8 Fatal error.
cts	Controllee's termination state. The values are: 0 Still active. 1 User terminal break to the EXIT control statement. 2 Operator break to the EXIT control statement. 3 Operator break to the end-of-job (6/7/9) card. 4 Operator drop of the entire batch deck. #39 Normal termination. #3D Abort. #3E Normal termination. The drop, scratch, and output files are saved for #3D and #3E. For #39, the drop and scratch files are destroyed, while the output file is given to an output processor according to the disposition code for the file.

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 5 of 8)

Message option:

c=6	Get the controllee's name and place. Beta(1) contains the source file name for the controllee, and Beta(2) contains the drop file name for the controllee (see the following Beta format).
c=7	Get the controller's name and place. Beta(1) contains the source file name for the controller, and Beta(2) contains the drop file name for the controller (see the following Beta format).
c=8	Get this program's name and place. The Beta portion of the format is:

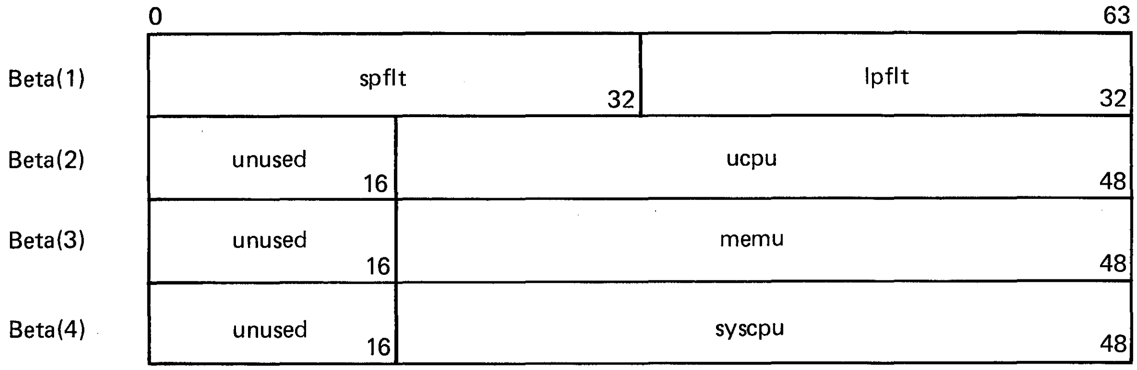


Word	Field	Description
Beta(1)	sfile	Source file name for this program.
Beta(2)	dfile	Drop file name for this program.
Beta(3)	priv	Identifies the privileged task. This field must be set to 0 for a nonprivileged task, or nonzero for a privileged task.
	lev	Level in the controllee chain; $1 \leq lev \leq 9$.
	jdn	Job descriptor number (integer; 1 through 2047) of the calling job.
	site	Origination site ID (ASCII).
Beta(4)	user	User number in ASCII, right-justified with blank fill.
Beta(5)	jobname	Job name of the calling task in ASCII, left-justified with blank fill.

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 6 of 8)

Message option:

c=9 Get the raw page fault counts, CPU times, and memory usage. If the batch processor issues this option, the statistics returned include the cumulative statistics for the batch processor and all its controllees. The Beta portion of the format is:



Word	Field	Description
Beta(1)	spflt	Number of disk accesses (input requests issued) that resulted from small page faults (small page implicit reads) up until the issuance of the MISCELLANEOUS message.
	lpflt	Number of disk accesses (input requests issued) that resulted from large page faults (large page implicit reads) up until the issuance of the MISCELLANEOUS message.
Beta(2)	ucpu	User execution CPU time, in microseconds, up until the issuance of the MISCELLANEOUS message.
Beta(3)	memu	Memory usage; the values (current working set size)*(user CPU time for the current accounting period) summed over all accounting periods for the task up until the issuance of the MISCELLANEOUS message.
Beta(4)	syscpu	Virtual and resident system CPU time, in microseconds, for user execution up until the issuance of the MISCELLANEOUS message.

Message option:

c=#A Get clock times as of message issuance. The Beta portion of the format is as follows:

Beta(1)	Master clock value, expressed as yymmddhhmmssppppp. pppp is fraction of a second.
Beta(2)	ASCII clock value, expressed as hh.mm.ss (hour.minutes.seconds).

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 7 of 8)

Beta(3)	Calendar value, expressed as mm/dd/yy (month/day/year).
Beta(4)	Value of the millisecond clock (0 at midnight).
Beta(5)	Value of the microsecond central processor clock (0 at power on).
Beta(6)	Current date, in the rightmost 16 bits. The leftmost 7 bits of the 16 bits contain the last 2 digits of the year, in binary; the remaining 9 bits contain the number of days since the beginning of the year (1 to 366), in binary. The leftmost 48 bits of Beta(6) are unused.

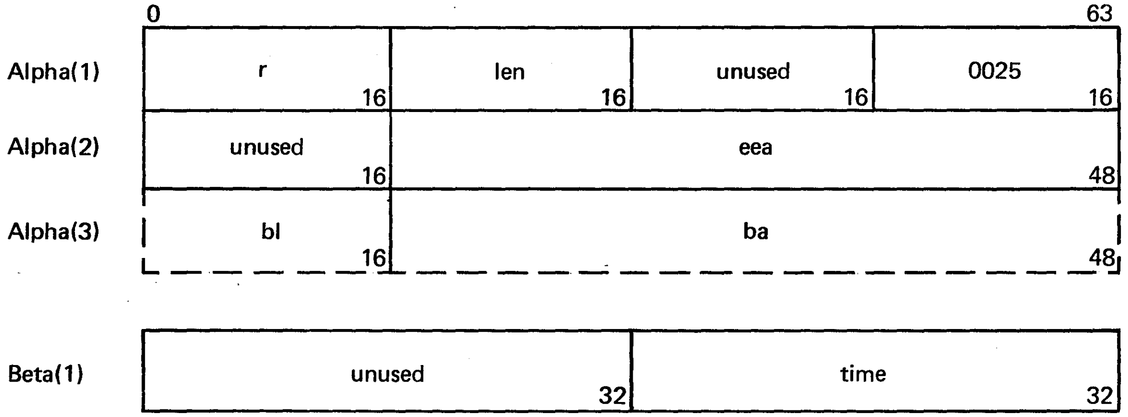
Message options:

c=#B	Reserved.
c=#D	Get the contents of the minus pages and return them in Beta(1) through Beta(1536). (The format of the minus pages is described in chapter 2.) If there is no second minus page, Beta(513) is #FFFF. If there is no third minus page, BETA(1025) is #FFFF. If Beta length is 512, only the first minus page is returned. If Beta length is 1024, only the first and second minus pages are returned.
c=#E	Get the version identifiers. Beta(1) contains the resident system version identifier, and Beta(2) contains the virtual system version identifier.
c=#F	Get the interrupt invisible package. If the program is running in interrupt state, Beta(1) through Beta(40) contain the contents of the invisible package saved when a program interrupt occurred. (Refer to appendix E, which describes the invisible package.) If the program is not currently running in interrupt state, a response of 2 is returned.
c=#10	Get the task CPU time. Beta(1) contains the task CPU time, in microseconds.
c=#11	Get the interrupted register file. Beta(1) through Beta(256) contain the contents of registers 0 through 255, when a program interrupt occurred. If the program is not currently running in the interrupt program state, a response code of 2 is returned.
c=#12	If the batch input file whose name is supplied by the user in Beta(1) fails to complete due to a system failure, destroy the batch job´s input file. The name of the batch input file must be left-justified with blank fill. Beta(2) contains the return code; the value is 0 if the batch job´s input file is successfully destroyed, or 1 if the batch file names does not exist.
c=#13	If the batch input file whose name is supplied by the user in Beta(1) fails to complete due to a system failure, rerun the batch input file. The name of the batch input file must be left-justified with blank fill. Beta(2) contains the return code; the value is 0 if the batch input file is successfully rerun, or 1 if the batch file name does not exist.
c=#14	Reserved for DEBUG.

Figure 5-32. MISCELLANEOUS (f=#0024) Message Format (Sheet 8 of 8)

RECALL (f=#0025)

The RECALL message allows a program to suspend its own execution for not fewer than 30 seconds nor more than 30 minutes. At the end of suspension, the program is recalled to an active status. The format of this message is shown in figure 5-33.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 This message is not allowed for this program. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion, that is, 1.
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	time	Period of suspension; specified as a hexadecimal integer number of microseconds. No fewer than 30 seconds and no more than 30 minutes must be specified by this field. Values outside this range are set to the nearest interval limit.

Figure 5-33. RECALL (f=#0025) Message Format

POOL FILE MANAGER (f=#0026)

This message offers a variety of options relating to pool file management, including attaching the user to a specified pool (thus giving the user access to the files in the pool) and detaching a pool (after which the files in the pool are no longer accessible to that user). A user who issues option 1 of this message to create a pool becomes that pool´s pool boss. Only the pool boss can issue options 2, 3, 6, and 7. Only the pool boss or a user authorized by the pool boss can issue options 4 and 5. Any user can issue option 8.

At the end of each batch job, when the batch processor issues the USER/ACCOUNTING COMMUNICATION message option 2 (end of job), any pools that were first attached by the job are detached.

Pools that have been attached interactively remain attached until the user detaches them or until that JDN is no longer active [that is, has done a (sc)BYE].

The format of the POOL FILE MANAGER message is shown in figure 5-34. The Beta formats are described under the c field definition.

	0			63
Alpha(1)	r 16	len 16	c 16	0026 16
Alpha(2)	unused 16	eea 48		
Alpha(3)	bl 16	ba 48		

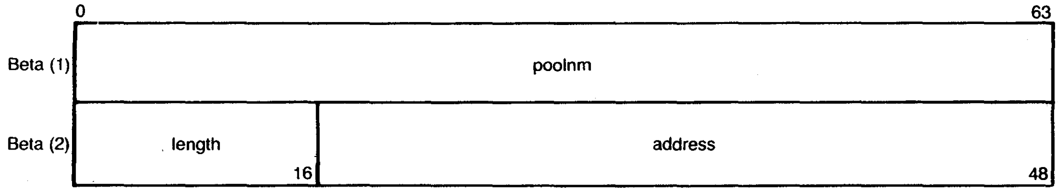
Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0. A nonzero response code does not necessarily mean an error has occurred (refer to message options 7 and 8). Other values are:

#11	Pool name was already attached by this job.
#12	Pool name is undefined.
#13	Four pools already attached.
#14	Pool is not attached.
#15	Cannot attach to the pool; user has no access to the pool.
#16	Undefined user number.
#17	Duplicate pool name.
#18	Unable to destroy the pool.
#19	Pool access directory (PAD) is full.
#20	One or more pool files span a downed device.

Figure 5-34. POOL FILE MANAGER (f=#0026) Message Format (Sheet 1 of 4)

Word	Field	Description
Alpha(1)	r	#1A Pool list (PLIST) table is full.
		#1B Invalid pool.
		#1C Invalid pool name.
		#1D Not a pool boss.
		#1E PAD or PLIST file was not found (refer to the VSOS 2 Installation Handbook).
		#1F File index table is full.
		#214 Beta buffer or user list buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer or user list buffer is too small for the number of results and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. If this field is 1 under message option 3, all users can access the pool.
	c	Message options. The pool name field in Beta(1) contains up to eight letters and numbers and must start with a letter; it is left-justified with blank fill. Binary user numbers can range from 1 to 999999; they are right-justified with zero fill. The options are:
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion. Under option 3, all users can access the pool if the bl field is 1.
		1 Create a pool. Adds the pool name to the pool list and clears (zeros) the pool access directory (invalid when universal access is set) for that pool. The creator is the pool boss. Beta(1) contains the pool name.
		2 Destroy the pool. If no users are attached and no files are in the pool, the pool name is deleted from the pool list. Beta(1) contains the pool name.
		3 Grant access to the pool. Places the specified user numbers into the pool access directory. If either the len field or bl field is 1, all users can access the pool. Beta(1) contains the pool name. Beta(2) contains the length and address of the user number list buffer:

Figure 5-34. POOL FILE MANAGER (f=#0026) Message Format (Sheet 2 of 4)



Word	Field	Description
Beta(1)	poolnm	Name of this pool, in ASCII.
Beta(2)	length	Length of user number list buffer, in words.
	address	Bit address of user number list buffer.

4 Attach to the pool. Attaches the requesting job to the named file pool. Beta(1) contains the pool name.

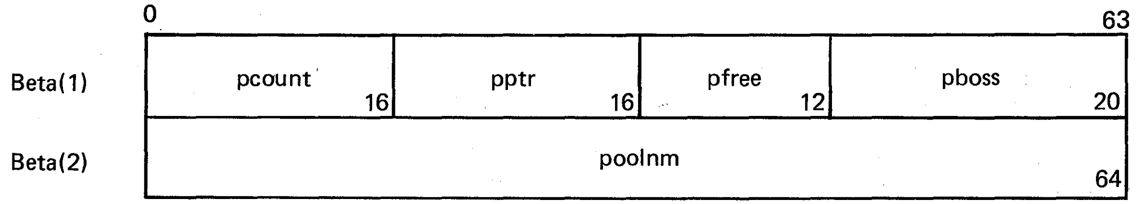
5 Detach the requestor from the pool. Beta(1) contains the pool name.

6 Remove the access privilege. Specified user numbers are removed from the pool access directory. Beta(1) contains the pool name. Beta(2) contains the length and address of the user number list buffer.

7 List the users having access to the pool. Beta(1) contains the pool name. Beta(2) contains the length and address of the user number list buffer.

8 List the pools and the pool boss. All nonzero entries in the pool list file are copied into a variable number of Beta words; the number of words copied is returned in the response code field. Each pool list entry returned by the operating system has the format shown next.

Figure 5-34. POOL FILE MANAGER (f=#0026) Message Format (Sheet 3 of 4)



Word	Field	Description
Beta(1)	pcount	Count of users attached to this pool.
	pptr	Pointer into the pool access directory that contains a list of user numbers which can attach to this pool (the value is used by the operating system only).
	pfree	Currently unused.
	pboss	User number of the boss of this pool.
Beta(2)	poolnm	Name of this pool, in ASCII.

Figure 5-34. POOL FILE MANAGER (f=#0026) Message Format (Sheet 4 of 4)

LINK (f=#0027)

With this message, a privileged user can process a CYBER interactive output message.

The Alpha portion of the LINK message is shown in figure 5-35. The Beta word formats depend on the message option (c field) in Alpha(1) and are shown in figure 5-36.

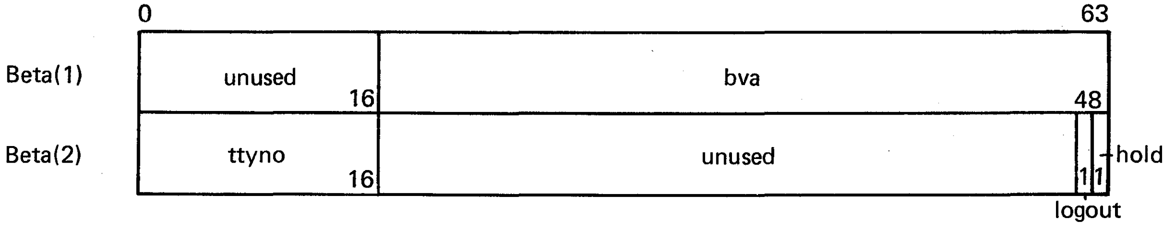
	0			63
Alpha(1)	r 16	len 16	c 16	0027 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Illegal option. 2 The message is for privileged use only. #214 Beta buffer length error; either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length of the Beta portion.
	c	Message option: #16 The system returns the identifiers of terminals that have outstanding output along with the output messages.
Alpha(2)	n	This field is the size of the Beta portion of the message, and the Beta area should be at least n words long. When the message has been processed, n is set equal to the word length of the information returned in the area pointed to by bva.
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.

Figure 5-35. LINK (f=#0027) (Alpha) Message Format

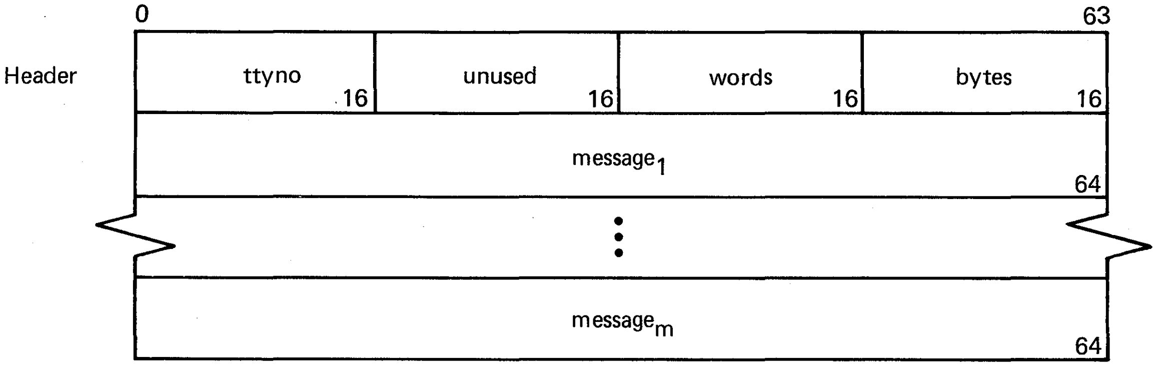
Message option:

c=#16 Poll. The caller puts the terminal number of all terminals logged into VSOS beginning in Beta(2). The number of terminals is put in the n field of Alpha(2) by the caller. The length of the output in words is returned in the n field of Alpha(2) by the operating system:



Word	Field	Description
Beta(1)	bva	The system returns terminal output messages at this bit address.
Beta(2)	ttyno	Terminal number.
	logout	Set by VSOS if user is to be logged out.
	hold	Set if caller cannot accept any output for this terminal at this time. The system does not return output (if any) for this terminal.

Format for output messages:



Word	Field	Description
Header	ttyno	Terminal number.
	words	Number of words, including header.
	bytes	Number of bytes in the message.
(1)-(m)	message	Message text.

Figure 5-36. LINK (f=#0027) (Beta) Message Format

VARIABLE RATE ACCOUNTING (f=#0028)

This message dynamically changes the variable rate during execution of a task. This call can be made only by a public controllee, a controllee which has the variable rate permit flag set in the user directory, or a nonpublic controllee with the proper password. Dynamic calls to change variable rates are made by applications programs rather than utilities. The rates to be indexed are in the Q5VRF file.

If IP_F_VR is 0, the call is illegal (r=2).

The change to the variable rate index is made in the descriptor block entry. At the time the change is made, BANKAC is called to compute the accumulated SBUs to be charged at the old rate and to decrement the available time remaining to complete the task.

Figure 5-37 shows the format of the Alpha portion of the VARIABLE RATE ACCOUNTING message.

0 63

Alpha(1)	r 16	len 16	unused 16	0028 16
Alpha(2)	vri 12	aaf	† 3	eea 48
Alpha(3)	bl 16	ba 48		

Beta(1)	password 64

† Unused.

Word	Field	Description
Alpha(1)	r	Response code returned by the operating system when this message has been processed: 0 Normal. 1 File is not public or privileged. 2 Call is not valid at this installation. 3 Accounting access denied. This could be caused by three events: the call has not been verified as being issued by a legitimate task, the password does not match the password in the variable rate table, or the variable rate table contains a password to that variable rate index. #214 Beta buffer length error.
	len	Length of the Beta; if len=#FFFF, then Alpha(3) contains the starting address and length of the Beta.
Alpha(2)	vri	Index into variable rate table (set by user).

Figure 5-37. VARIABLE RATE ACCOUNTING (f=#0028) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	aaf	Accounting flag. If set, accounting statistics will not be accumulated to the minus page or to BACCTG: =0 Not accounting (default). ≠0 Accounting.
	eea	Virtual bit address to receive control if an error occurs during message processing (if r is different from 0). If eea=0, the error is considered fatal.
Alpha(3)	bl	Beta length.
	ba	Beta address.
Beta(1)	password	Password (64-bit) to the variable rate table.

Figure 5-37. VARIABLE RATE ACCOUNTING (f=#0028) Message Format (Sheet 2 of 2)

SEND MESSAGE TO DAYFILE (f=#0029)

This message allows the user to send a string of ASCII data to the program dayfile.

The format of the message is shown in figure 5-38.

	0				63
Alpha(1)	r 16	len 16	cf 8	c 8	0029 16
Alpha(2)	unused 16	eea 48			
Alpha(3)	bl 16	ba 48			

Beta(1) ⋮ Beta(n)	string of ASCII data

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when the message has been processed. If no error occurs the error code is 0; otherwise:
		1 Length of Beta in character bytes is greater than 4096. 2 Illegal c field selected for this message. 3 Dayfile is full. The message in dayfile will be DAYFILE FULL. 4 Dayfile is not opened for implicit I/O. 5 Invalid base virtual bit address for Q5DAYFILE. 6 Unable to find Q5DAYFILE. 7 The call is currently blocked waiting for access to dayfile (caller will not see a response code 7 when control is returned). 8 User not authorized to make the call. #214 Beta buffer length error. Either the first word address of Beta plus length is greater than the maximum user virtual address, or the Beta buffer is too small for the message; length is 0.
	len	If len=#FFFF, Alpha(3) contains the length in character bytes and the virtual bit address of the Beta portion of the message. Otherwise, Beta is assumed to begin at Alpha(3), and len is the length in character bytes of the Beta portion.

Figure 5-38. SEND MESSAGE TO DAYFILE (f=#0029) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	cf	Type of entry (used only when c=2): 1 USER; job dayfile and terminal entries. 2 SYST; messages to and from the operator. 3 LABL; new system dayfile started. 4 DIAG; customer engineering diagnostics.
	c	Control field: 1 Send message to the system and job dayfile. 2 Send message to the system dayfile, for privileged/authorized users only (such as operator and privileged system tasks). 3 Send message to the job dayfile. The message does not go into the system dayfile.
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during message processing ($r \neq 0$). If eea is 0, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta portion of the message is not contiguous to the Alpha portion (len=#FFFF), these parameters indicate the length in character bytes and virtual bit address of the first full word of the Beta portion.
Beta(1) through (n)	string of ASCII data	Maximum length of 4096 characters. If there is no #1F at the end of the line, one will be added. Illegal characters #00 through #1E and #7F through #FF will be changed to blanks. The combination #0D0A will be changed to #201F.

Figure 5-38. SEND MESSAGE TO DAYFILE (f=#0029) Message Format (Sheet 2 of 2)

RHF_CALL (f=#002A)

This message controls the RHF-related tables. The RHF applications issue this message.

The Alpha format of the RHF_CALL system message is shown in figure 5-39.

The Beta format is shown in figure 5-40.

	0			63
Alpha(1)	r 16	len 16	c 16	002A 16
Alpha(2)	unused 16	eea 48		
Alpha(3)	bl 16	ba 48		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. 0 No errors. 1 Error code was returned in Beta(n). #214 Beta buffer length error.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3) and len is the length of the Beta portion. All requests must provide at least 15 Beta words.
	c	Function code: 1 Return Remote Host Facility table (T_RHFT) entry. 2 Create new currently active table (T_CAT) entry for servicer RHF application. 3 Set NETON flag in T_CAT for servicer application. 4 Status currently running table (T_CRT) for application name. 5 NETOFF and clear T_CAT entry. 6 Create new entry in T_CAT for requested initiator application. 7 Change maximum number of copies of an RHF application. 8 Change default output LID. 9 Save information about an error condition. #A Return currently active table (T_CAT). #B Return RHF mainframe table (T_RHMFT). #C Return application table (T_APPT). #D Return T_CRT table. #E Enable/disable physical identifier/logical identifier (PID/LID).

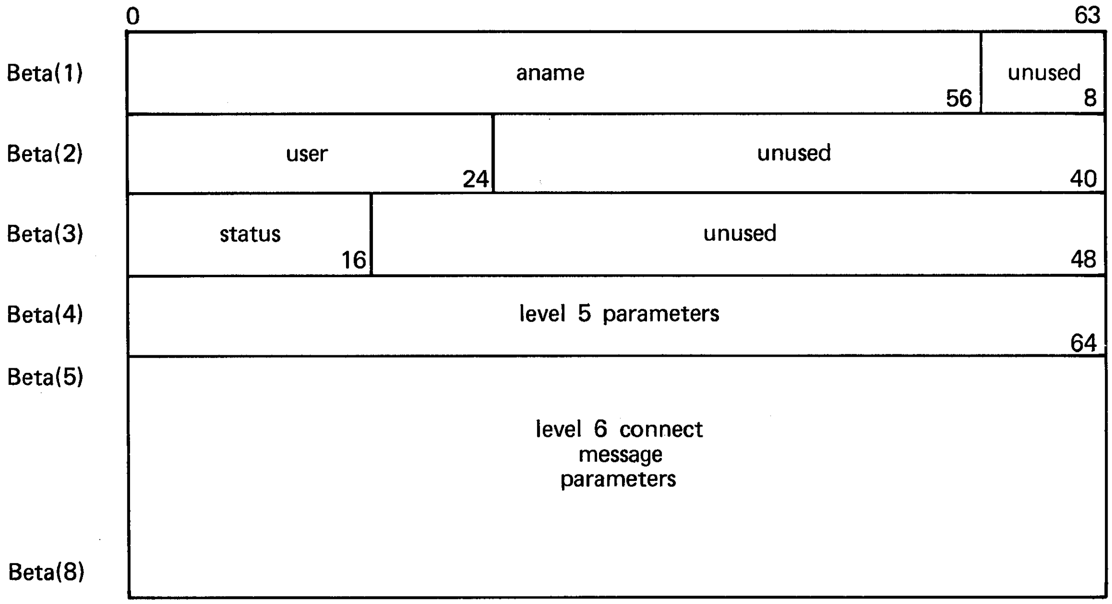
Figure 5-39. RHF_CALL (f=#002A) (Alpha) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Alpha and Beta portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in words of Beta and the virtual bit address of its first full word.

Figure 5-39. RHF_CALL (f=#002A) (Alpha) Message Format (Sheet 2 of 2)

Function code:

c=1 Return T_RHFT entry:



Word	Field	Description
Beta(1)	aname	Application name; used to search T_RHFT.
Beta(2)	user	User number of the application that issues the call.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 1 of 13)

Word	Field	Description
Beta(3)	status	Error message returned: #11 No matching application connect waiting. #12 Invalid user for this application. #13 No entry in T_CRT for this application.
Beta(4)	level 5	Level 5 connection parameters as received from the LCN parameters (returned by system).
Beta(5) through Beta(8)	level 6	Incoming application connection request as received from the LCN parameters (returned by system).

Function code:

c=2 Create new T_CAT entry for requested servicer RHF application:

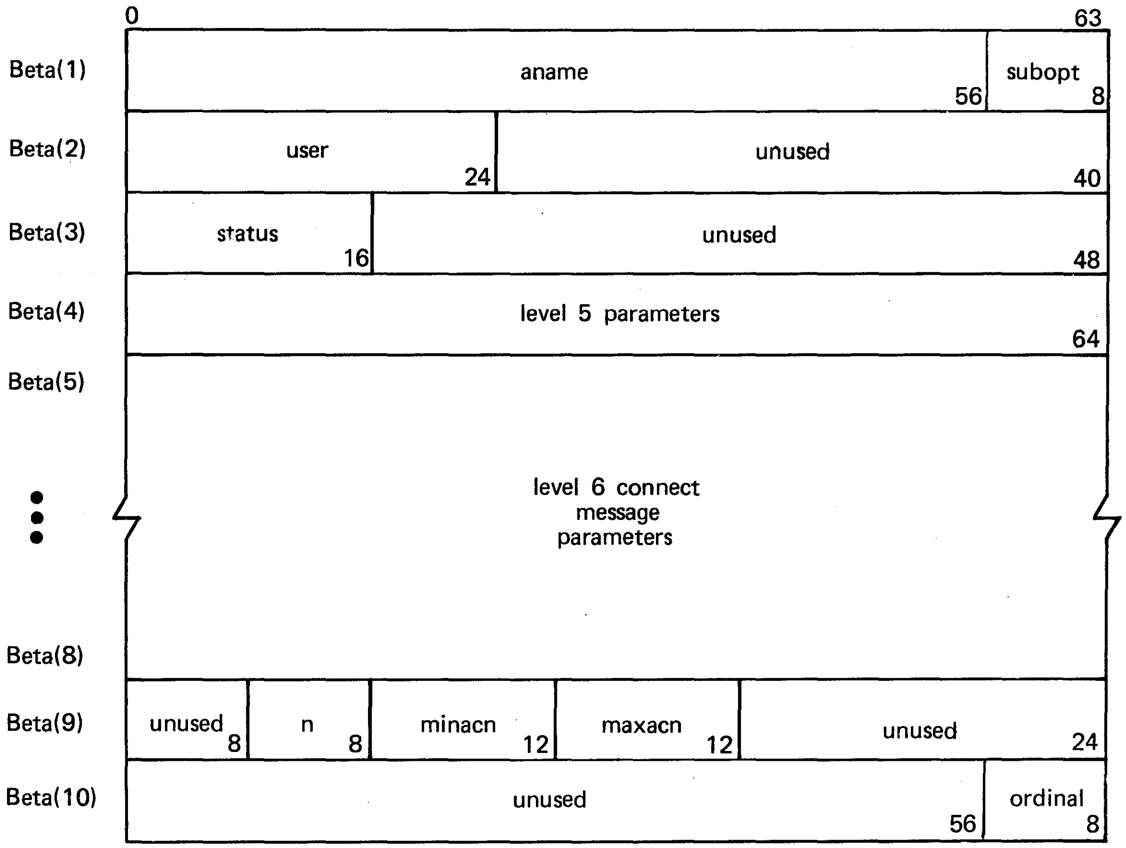


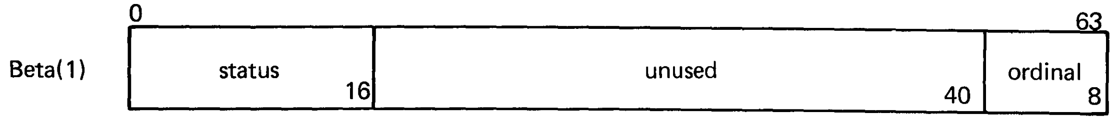
Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 2 of 13)

Word	Field	Description
Beta(1)	aname	Application name.
	subopt	Determines if LIDs should be validated for multiple file transfers on a single operation. 0 Do not validate the LID. 1 Validate the LID.
Beta(2)	user	User number of the application that issues the call.
Beta(3)	status	Error message returned: #21 Local LID specified by level 6 parameters not found. #22 Local LID specified by level 6 parameters disabled. #23 Currently running limit exceeded for this application. #24 Undefined RHF application name. #25 Invalid user number specified for this application. #26 Application not in currently running table (T_CRT). #27 No empty entries in connected application table (T_CAT). #28 No empty slot in connected application table (T_CAT).
Beta(4)	level 5 parameters	Level 5 connection parameters as received from the LCN.
Beta(5) through Beta(8)	level 6 parameters	Incoming application connection request as received from the LCN (provided by caller).
Beta(9)	n	Number of outstanding connections for this application.
	minacn	Minimum application connection number.
	maxacn	Maximum application connection number.
Beta(10)	ordinal	Index to entry in T_CAT.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 3 of 13)

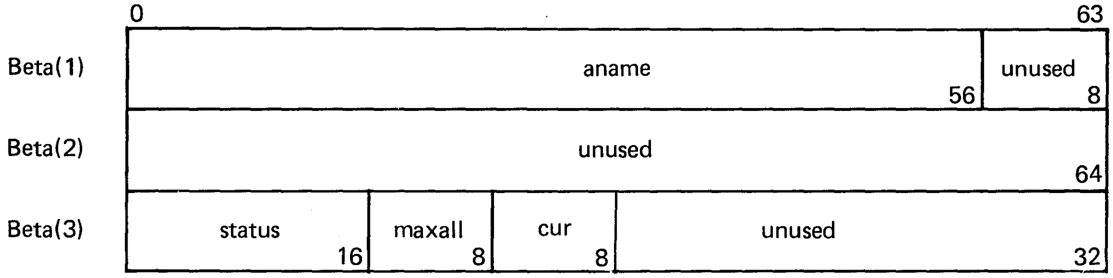
Function code:

c=3 Set NETON flag in T_CAT for servicer application:



Word	Field	Description
Beta(1)	status	Error message returned: #31 Invalid T_CAT ordinal. #32 No entry for this ordinal.
	ordinal	Index to entry in T_CAT. Valid range of the ordinal is 3 to the length of T_CAT.

c=4 Status CRT for application name:



Word	Field	Description
Beta(1)	aname	Application name.
Beta(3)	status	Error message returned: 0 Currently running limit not exceeded. 1 Currently running limit equals the maximum allowed. 2 Currently running limit exceeds the maximum (occurs if operator reduces maximum to a number lower than what is currently running). #41 Application name not found.
	maxall	Maximum number of currently running applications allowed (returned by system).
	cur	Number of applications currently running (returned by system).

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 4 of 13)

Function code:

c=5 NETOFF and clear T_CAT entry:

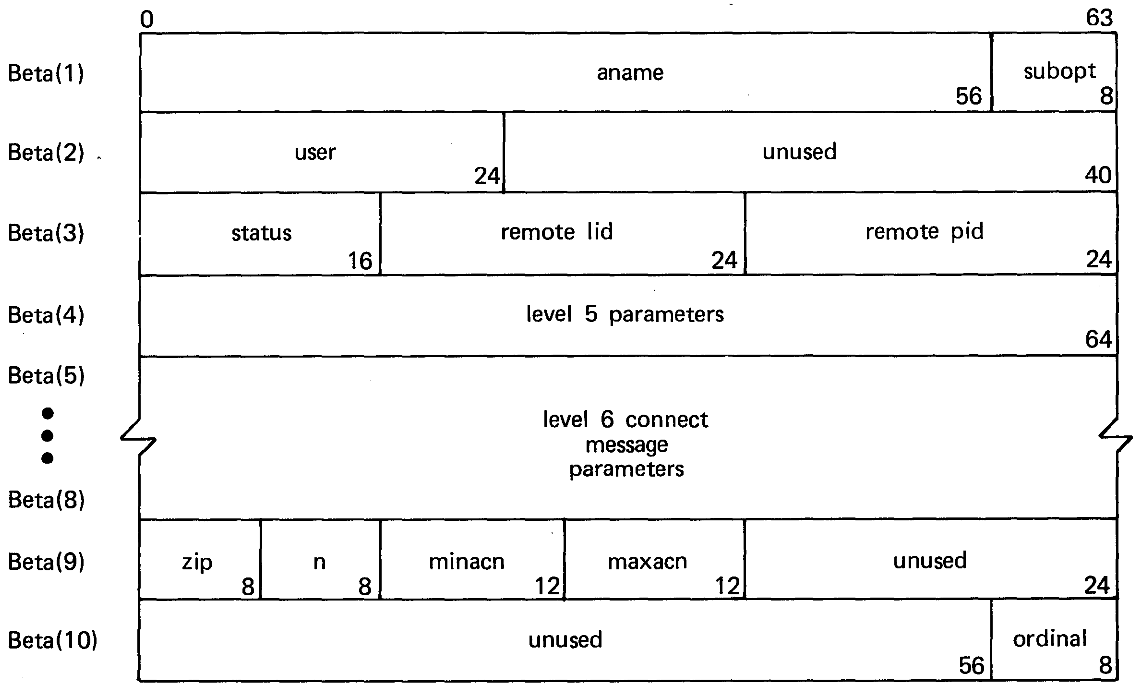
	0		63
Beta(1)	status 16	unused 40	ordinal 8

Word	Field	Description
Beta(1)	status	Error message returned: #51 Invalid T_CAT ordinal. #52 No entry for this ordinal.
	ordinal	Index to entry in T_CAT.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 5 of 13)

Function code:

c=6 Create new entry in T_CAT for requested initiator application and return initial connection request packet.



Word	Field	Description
Beta(1)	aname	Application name.
	subopt	Determines if LIDs should be validated for multiple file transfers on a single connection. 0 Do not validate the LID. 1 Validate the LID.
Beta(2)	user	User number of the application that issues the call.
Beta(3)	status	Error message returned: #61 LID not found. #62 LID disabled. #63 Currently running limit exceeded. #64 Undefined RHF application name. #65 Invalid user number. #66 No CRT entry for this application. #67 PID not found in Remote Host Facility mainframe table (T_RHFMT). #68 PID disabled. #69 No empty slot for T_CAT ordinal in the application entry in T_CRT. #6A No empty entry in connected application table (T_CAT). #6B RCD NAD is disabled. #6C SHD NAD is disabled.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 6 of 13)

Word	Field	Description
Beta(3)	remote lid	Remote logical identifier.
	remote pid	Remote physical identifier (returned by system).
Beta(4)	level 5 parameters	Level 5 connection to be sent to the LCN (returned by system).
Beta(5) through Beta(8)	level 6 parameters	Data from the NAD on the host incoming application request (returned by system).
Beta(9)	zip	Zip code of the NAD that received this connection (returned by system).
	n	Number of outstanding connections for this connection application to be sent to the remote host (returned by system).
	minacn	Minimum application connection number (returned by system).
	maxacn	Maximum application connection number (returned by system).
Beta(10)	ordinal	Index to entry in T_CAT (returned by system).

Function code:

c=7	Change maximum number of copies of an RHF application:

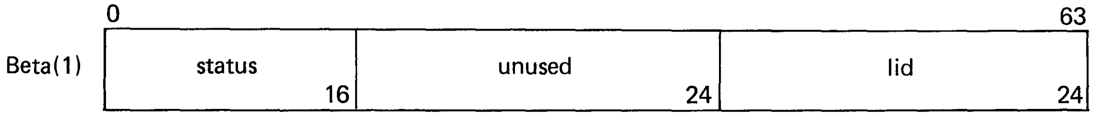
	0		63
Beta(1)	aname 56	unused 8	
Beta(2)	status 16	maxall 8	unused 40

Word	Field	Description
Beta(1)	aname	Application name or ALL. The option ALL is used to set all the application limits to the same value on one command.
Beta(2)	status	Error returned: #71 Application is not found. #72 Maximum specified is too large.
	maxall	Maximum number of currently running applications allowed (0 to 8).

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 7 of 13)

Function code:

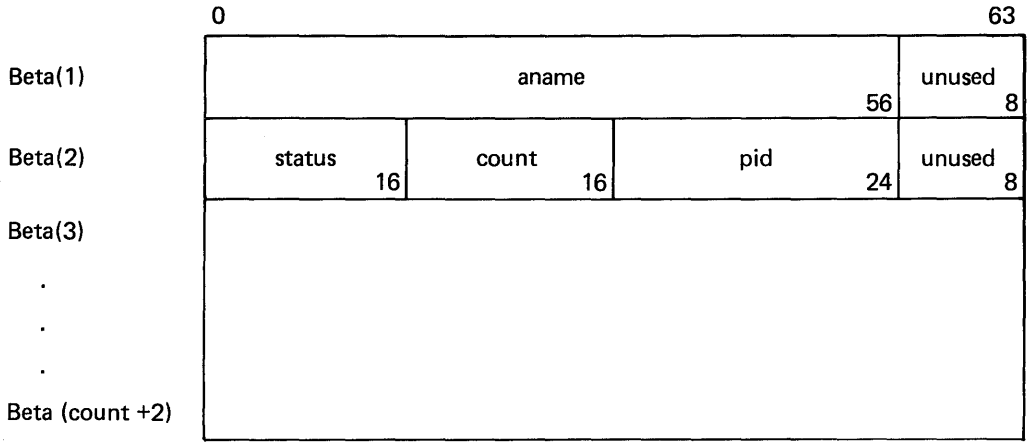
c=8   Change the default output LID:



Word	Field	Description
Beta(1)	status	Error code returned: 0 Successful completion, no error. #81 The specified LID is not defined. #82 The default output LID is already sent to the specified value.
	lid	Logical identifier of a remote host (three ASCII alphanumeric characters).

Function code:

c=#A   Return T_CAT table:



Word	Field	Description
Beta(1)	aname	Application name. If aname is zero, then the entire connected Application Table is returned. Otherwise, the application name and pid are used as qualifiers for the entries returned.
Beta(2)	status	Error message returned: #A1 Named application not found. #A2 Beta length is too small for qualified T_CAT entries.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 8 of 13)

Word	Field	Description
Beta(2)	count	Number of words returned [excluding Beta(1) and Beta(2)].
	pid	Remote pid. If pid is zero, then aname is the only qualifier for returned entries. Otherwise, the corrected remote pid and application name are used as qualifiers for the entries returned.

Function code:

c=#9 Save information about an error condition:

	0			63
Beta(1)	status 16	interval 16	ec 16	msgno 16
Beta(2)	time (yymmddhhmnsspppp) 64			

Word	Field	Description
Beta(1)	status	Error information returned: 0 No errors. The information from the Beta has been saved by the virtual system. #91 The time interval since the last occurrence of the ec/msgno error condition has not elapsed. The information from the Beta was not saved.
	interval	Time interval in minutes (binary). The time is compared with the last occurrence of ec/msgno and if the difference is greater than interval, the information in the Beta is saved.
	ec	Error condition category: 1 RHF application internal error. 2 RHF application SIL error.
	msgno	Error message number associated with the error condition.
Beta(2)	time	Time stamp of the occurrence of the error condition. The format used for the time stamp is the master clock time (refer to the ´MASTER=´ parameter of Q5TIME).

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 9 of 13)

Function code:

c=#B Return T_RHMFT table:

	0			63
Beta(1)	lid 24	pid 24	nad 8	unused 8
Beta(2)	status 16	count 16	unused 32	
Beta(3)	returned entries			
Beta(count+2)				

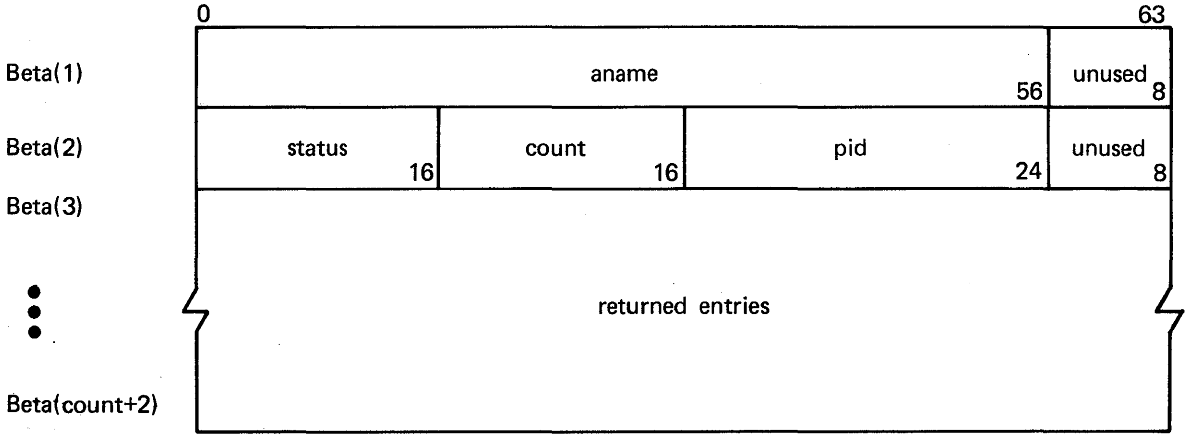
If Beta(1) is 0, the entire RHF mainframe table is returned. Otherwise, the nonzero field in Beta(1) is used to qualify the set of RHF mainframe table entries returned:

Word	Field	Description
Beta(1)	lid	Logical identifier.
	pid	Physical identifier.
	nad	Network access device number.
Beta(2)	status	Error message returned: #B1 No matching LID found. #B2 No matching NAD found. #B3 No matching PID found. #B4 Beta length too short for qualified T_RHMFT entries.
	count	Number of words returned [excluding Beta(1) and Beta(2)].

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 10 of 13)

Function code:

c=#C Return T_APPT table:

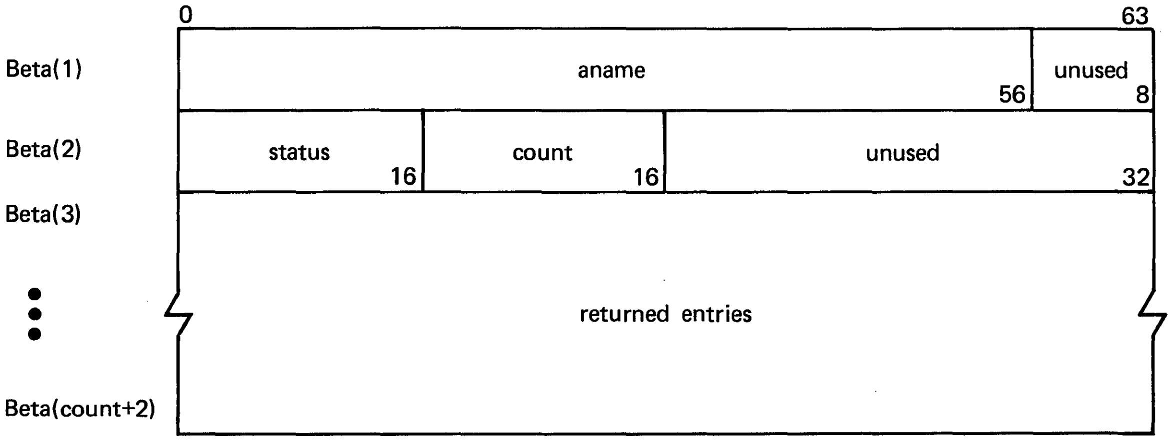


Word	Field	Description
Beta(1)	aname	Application name. If aname is 0, the entire application table is returned. Otherwise, the application and pid are used as qualifiers for the entry returned.
Beta(2)	status	Error message returned: #C1 Named application not found. #C2 Beta length too short for full application table.
	count	Number of words returned [excluding Beta(1) and Beta(2)].
	pid	If pid is 0, then pid is not used as a qualifier for returned entries; otherwise, the 3-character pid names and application names are used as qualifiers for the entry returned.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 11 of 13)

Function code:

c=#D Return T_CRT table:

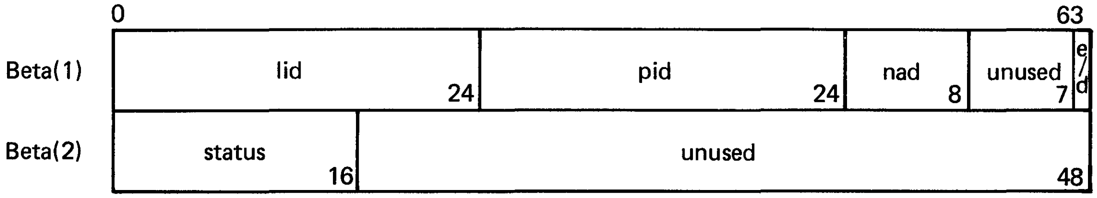


Word	Field	Description
Beta(1)	aname	Application name. If aname is 0, the entire T_CRT table is returned. Otherwise, the application is used as a qualifier for the entry returned.
Beta(2)	status	Error message returned: #D1 Named application not found. #D2 Beta length too short for qualified T_CRT entries.
	count	Number of words returned [excluding Beta(1) and Beta(2)].

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 12 of 13)

Function code:

c=#E Enable/disable PID/LID:

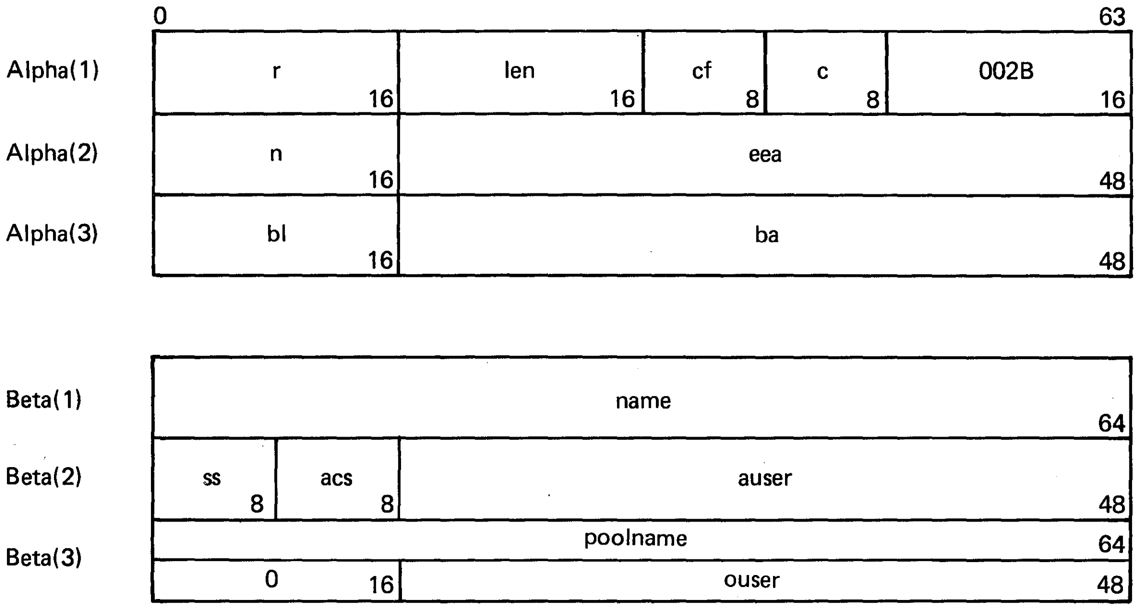


Word	Field	Description
Beta(1)	lid	Logical identifier.
	pid	Physical identifier.
	nad	Network access device.
	e/d	Enable/disable entry: 0 Disable entry. 1 Enable entry.
Beta(2)	status	Error message returned: #E1 LID not found. #E2 PID not found. #E3 LID specified is for a PID that is currently disabled. #E4 No LIDs for the specified PID. #E5 LID, PID, or NAD already in requested state. #E6 NAD was not found, or none of lid, pid, or nad were specified.

Figure 5-40. RHF_CALL (f=#002A) (Beta) Message Format (Sheet 13 of 13)

ACCESS CONTROL (f=#002B)

This message provides program level control of access permissions for private, public, and pool files. The format of this message is shown in figure 5-41.



Word	Field	Description
Alpha(1)	r	Response code. If no error occurs, the response code is 0; otherwise: 1 Error code is returned in the ss field of Beta. 2 Illegal c or cf option. 3 Caller not privileged. 4 Caller is not site security administrator. #211 Number of requests is illegal (the n field is 0 or greater than 16). #214 Beta buffer length error. Either the first word is greater than the maximum user virtual address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message. Otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion. The value of len must be a multiple of 3.

Figure 5-41. ACCESS CONTROL (f=#002B) Message Format (Sheet 1 of 3)

Word	Field	Description
Beta(2)	cf	Ownership option: 0 Private file. 1 File resides in pool specified by poolname (caller must be the pool boss). 2 Public file (caller must be privileged).
	c	Control field: 0 Grant access to user. 1 Grant production status to the file and remove all write permissions. 2 Remove production status from the file. Caller must be the site security administrator user number.
Alpha(2)	n	Number of requests in this message; maximum is 16.
	eea	Virtual bit address to receive control if an error occurs during the processing of this message (r≠0).
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first word of the Beta portion.
Beta(1)	name	File name, in ASCII, of the file whose access permission is to be changed.
Beta(2)	ss	Error response field. The values are: 0 Normal completion. 1 File not found. 2 Pool not found. 3 Illegal access permission. 4 FILEI is full, no entry made. 5 Access control list is full, no entry made. 6 User not pool boss. 7 MODPFI error. 8 User number is invalid for specified file. 9 Caller is not the file owner. #A User number is not defined. #B Illegal access for tape. #C Write access and there is no write ring. #D Write permissions are not valid for a production file. #E No user table entry is available (c=1 or 2). #F Write permissions are not valid for a drop file.
	acs	File access permissions. This 8-bit field specifies the access permissions to be granted. Five bits are currently defined: Bit / Hex. Value / Description 1-3 / - / Unused. 4 / 10 / Give execute access. 5 / 8 / Give modify access.

Figure 5-41. ACCESS CONTROL (f=#002B) Message Format (Sheet 2 of 3)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td></td><td>acs</td><td>
<table>
<tr><th>Bit</th><th>Hex. Value</th><th>Description</th></tr>
<tr><td>6</td><td>4</td><td>Give append access.</td></tr>
<tr><td>7</td><td>2</td><td>Give read access.</td></tr>
<tr><td>8</td><td>1</td><td>Give write access.</td></tr>
</table>
Users affected will have only those access permissions specified by these bits (a replacement operation).
</td></tr>
<tr><td>Beta(2)</td><td>auser</td><td>
This parameter identifies whose access permission is to be modified. Its definition is dependent on the cf (ownership) option:
<table>
<tr><th>File</th><th>Description</th></tr>
<tr><td>Private local</td><td>auser must be binary 0. Only the owner´s access permissions can be changed.</td></tr>
<tr><td>Private permanent</td><td>auser can have one of the following values:

The ASCII user number of the user whose access permissions are to be modified.
"GENRAL", which indicates that all access permissions are to be modified.
"*", left-justified, blank-filled, which indicates all access permissions are to be modified.
Binary 0, which indicates that the caller´s (file owner´s) access permission is to be changed.
</td></tr>
<tr><td>Public</td><td>auser is ignored. The general access permissions are to be changed.</td></tr>
<tr><td>Pool</td><td>auser can have one of the following values:

"GENRAL", which indicates that all pool members´ access permissions are to be modified.
"*", equivalent to "GENRAL".
Binary 0, which indicates that the caller´s (pool boss´) access permissions are to be changed.
</td></tr>
</table>
</td></tr>
<tr><td>Beta(3)</td><td>poolname</td><td>Name of pool in which poolfile resides.</td></tr>
<tr><td></td><td>ouser</td><td>File owner´s ASCII user number (c=1 or 2).</td></tr>
</table>

Figure 5-41. ACCESS CONTROL (f=#002B) Message Format (Sheet 3 of 3)

TAPE MANAGEMENT (f=#002C)

This message associates a logical file name with a magnetic tape unit. The logical file is a local file.

Message option 1 allows the user to specify a list of VSNs to be associated with this local tape file. The VSN list is maintained by the system until the tape file is returned. If a user attempts to assign VSNs to an existing file, an error is returned but the file is not returned.

Message option #2 checks user validations for interactive or batch tape access. If the user is not allowed tape access, an error will be returned. Message option 2 allows the user to specify density, conversion mode, tape format, noise size, and label type of the tape file. The user can also specify request processing options. Message option 2 causes the system to compare the VSN supplied by option 1 with the VSNs read from mounted tapes. If a match is found, the system automatically assigns the tape unit to the job. If the tape is not mounted, a request for assignment is displayed at the operator console and the job is suspended until the requested VSN is mounted. If the tape is unlabeled, the operator enters the VSN command that associates a VSN with a tape unit. The system can then assign the tape. Refer to the VSOS 2 Operator's Guide for more information.

Message option 3 is a combination of the first two options. An error is returned if a local file (tape or disk) with the same file name already exists.

Message option 4 is like option 3, but it also blank labels a new tape. The label buffer descriptor is used only for this option. Either the caller must be privileged or the installation option IP_TPVOL must be set to 1.

If the tape file is an ANSI standard labeled tape, the multifile set name is the same as the logical file name. This logical file is given an HDR1 label with a file sequence number equal to 1. That means that, by default, it is the first file on the tape unless this HDR1 label is replaced by a subsequent LABEL call. The file attributes in Beta(8) through Beta(A) are assigned to the logical file and the multifile set.

The format of the TAPE MANAGEMENT message is shown in figure 5-42.

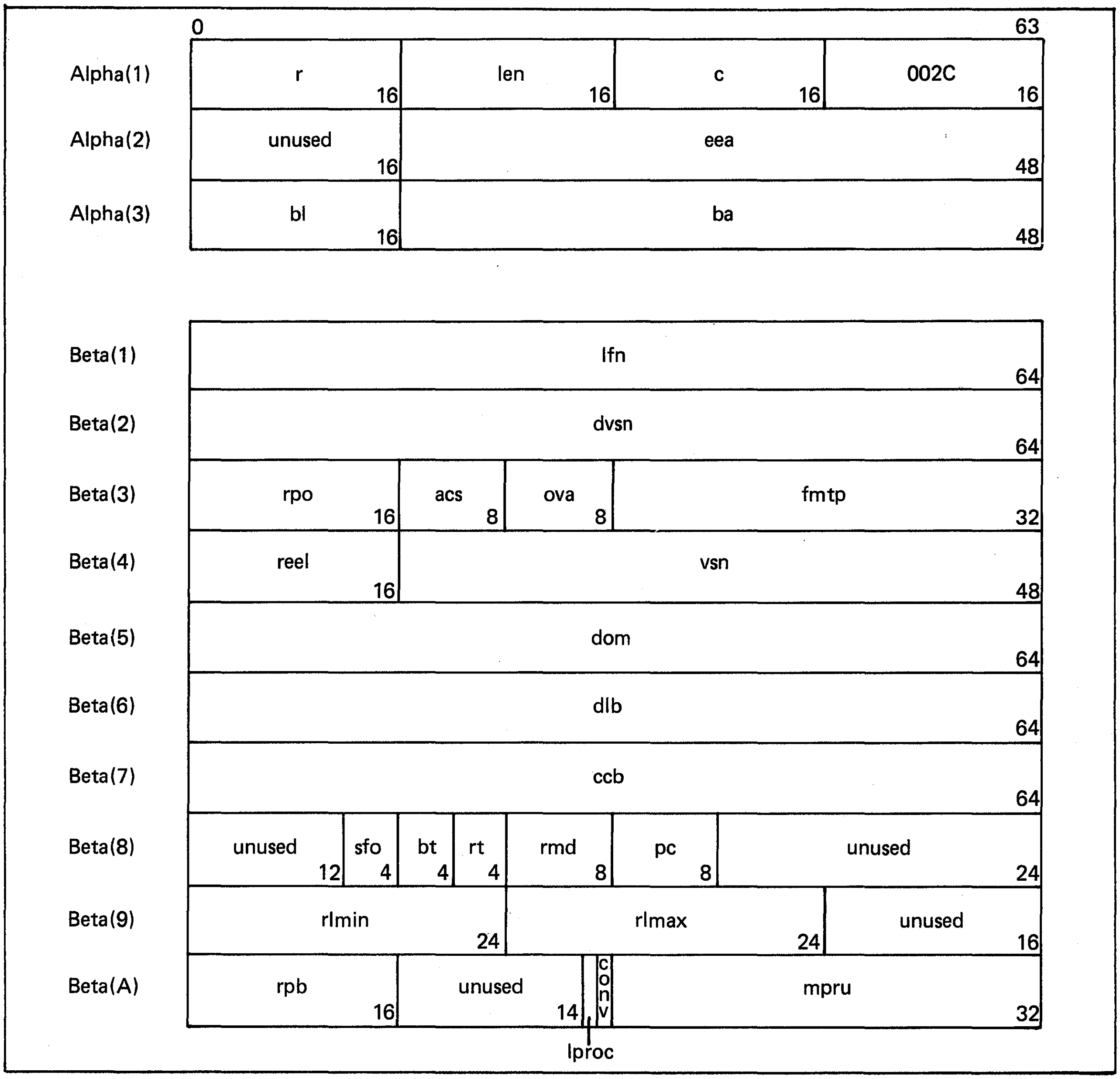


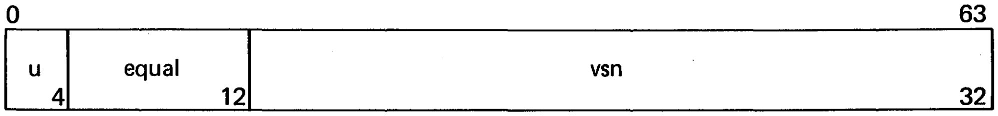
Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 1 of 9)

Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 File already exists. 2 Cannot blank label a tape. 3 Illegal c option field. 4 Interactive tape access requested when the installation parameter allowing interactive access is not appropriately set. 5 Nonstandard labeling is not allowed. 6 Illegal original volume accessibility. 7 No room in the file index. 8 Standby job cannot issue the call. 9 Invalid logical file name. #A Requested read unconditional flag and the installation parameter allowing this option is not set. #B Conversion mode does not match the label. #C Mismatch of density. #D No VSN list (c=1, 3). #E More than 255 VSNs (c=1, 3). #F Illegal conversion mode. #10 Illegal label type. #11 Illegal error correction mode. #12 Illegal tape format. #13 Illegal density. #14 Illegal access permission. #15 Mismatch of ova. #16 Illegal VSN (c=1, 3). #17 Illegal equal number in the virtual bit address of VSN list (c=1, 3). #18 Illegal label. #19 Number of tapes exceeds the number requested on the resource card. #1A Volume is not available. #1B ioc is not available. #1C Read-only access (c=4). #1D User not allowed tape access. #25 Illegal file organization. #26 Illegal block type. #27 Illegal record type.
	len	If this field is #FFFF, bl in Alpha(3) contains the length of the remote Beta buffer. ba contains the location of the remote Beta buffer. If this field is not #FFFF, Beta is assumed to begin at Alpha(3), and len is the length, in words, of the Beta portion.
	c	Message options: 1 Assign VSNs. 2 Request a tape file. 3 Assign VSNs and request a tape file. 4 Write VOL1 labels. Request a tape as in option 3. Beginning-of-volume labels (VOL1) in the label buffer are verified and written.

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 2 of 9)

Word	Field	Description
Alpha(2)	eea	Error exit address.
Alpha(3)	bl, ba	If the Beta and Alpha portion are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	lfn	Logical file name. If the tape file is an ANSI standard labeled tape, lfn is also the multifile set name.
Beta(2)	dvsn	Descriptor for the volume serial number list. The volume serial number is left-justified, with blank fill. This list contains the VSNs that are to be assigned to the file. This field is ignored for option 2:

0-15	lvsn	Length of the VSN list in 64-bit words (0 < lvsn < 256).
16-63	avsn	Virtual bit address of the VSN list. The list must begin on a word boundary:



u	Unused.
equal	If 0, this VSN is to be processed in sequential order. If 1, the next n VSNs are scanned and the first available VSN is assigned. All equal entries should have equal=n set.
vsn	Volume serial number. The VSN is a left-adjusted alphanumeric name, one to six letters or digits in length. If VSN is fewer than six characters, this field must be blank-filled.

Beta(3)	rpo	Request processing options. This field is ignored for option 1:

Bit	Name	Description
0-9		Unused.
10	try	Error retry parameter. This field applies only when reading the tape:

0	Standard error recovery processing takes place when a hardware read error occurs.
1	Error inhibit; all hardware read errors are ignored and processing continues.

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 3 of 9)

Word	Field	Description		
Beta(3)	rpo	Bit	Name	Description
		11		Unused.
		12	ru	Read unconditional processing option: 0 The user is not allowed to read past the end of information or the end of tape. 1 The user is allowed to read past the end of information or the end of tape. This could cause the tape to go off the reel.
		13	iu	Tape unload processing option (inhibit unload): 0 When the tape is released (refer to the DESTROY FILE system message, option 0), the tape is rewound to the load point and unloaded from the drive. 1 When the tape is released (refer to the DESTROY FILE message), the tape is rewound to the load point, but it is not unloaded from the drive.
		14		Unused.
		15	ring	0 Ring is not needed. (Read permission only.) 1 Ring is needed. (Write or read/write permission.)
	acs	Access permission for the logical tape file or multifile set. A ring must be in the tape for acs=2 or 3. The tape must not have a ring for acs=1. If the write enable status of the tape being assigned does not correspond with what was requested by this option, the job is suspended and the operator is sent a message requesting the tape be mounted correctly: 1 Write permission only. 2 Read permission only. 3 Read/write permission.		
	ova	Original volume accessibility character. This field must match the volume accessibility character in the tape VOL1 label if nonblank. This field applies for message options c=2, 3, and 4. Default is the installation parameter IP_TPVA.		

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 4 of 9)

Word	Field	Description
Beta(3)	fmtp	Format parameters. This field is ignored when c=1:

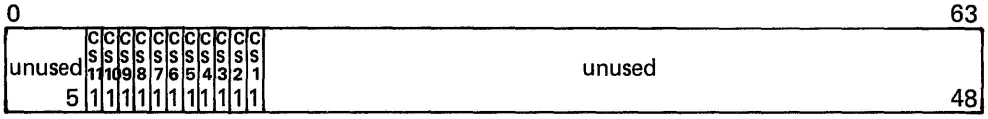
Bit	Name	Description
32-63	NS	Noise size in frames. This option applies only for V- or NV-formatted tape files when the tape is being read. Any PRU containing fewer than the specified number of frames is considered noise and is discarded by the system. A noise size of 0 causes the default noise size to be used. The default size is 0. The maximum NS is 31 decimal frames. NS is ignored for I, SI, and LB tape formats.
37	ECB	Hardware error correction mode for tapes being written in GCR mode. This field is set by the system from the EC field: 0 Enabled; the system allows more single-track errors to be written than can be corrected when the tape is read. 1 Disabled; a single-track error while writing a 6250-cpi tape results in standard error recovery processing.
38-44		Reserved.
45-47	DENS	Tape recording density. This parameter applies only to writing data on an unlabeled tape positioned at load point. Data is written on a labeled tape at the same density in which labels are written. Data is read from a tape at the same density at which it was written. The default density is an installation-defined option. The default for the released system is 6250 cpi. The density selected is returned in DENS by the system: 0 Default. 1 6250 cpi (GE). 2 1600 cpi (PE).
48-49	EC	Hardware error correction mode for GCR tapes. The mode selected is returned in ECB: 0 Installation default (IP_TPEC). 1 Enabled. 2 Disabled.
50-51		Unused.

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 5 of 9)

Word	Field	Description		
		Bit	Name	Description
Beta(3)	fmtp	52-55	CM	Conversion mode character set for the file data. Specifies the character set that data is converted from when it is read from the tape. The default character set for a labeled tape is the character set in which the labels are written. For an unlabeled tape, the default is an installation-defined option. The default for the released system is ASCII: 0 Default. 1 ASCII. 2 EBCDIC. Observe that the tape file must be requested in coded mode for the conversion to take place. Refer to the conv field in the CHANGE system message.
		56-59	LT	Label tape. Specifies the type of labels, if any, that are on the tape. The default label type is the type of labels on the tape being assigned. If LT=0, the label type selected is returned: 0 Default. 1 ANSI standard label. 2 Unlabeled. 3 Nonstandard; valid only if privileged caller or installation option IP_TPNSL=1.
		60-63	TF	Tape format. (Refer to appendix G for more detailed information.) The format of the data on the tape. The default is an installation-defined option. The TF selected will be returned if TF=0: 0 Default. 1 Large block format (LB). 2 SCOPE internal (SI). 3 NOS internal (I). 4 Variable length block (V). C Variable length block with embedded tape marks (NV).

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 6 of 9)

Word	Field	Description
Beta(4)	reel	The position in the VSN list of the tape to be assigned. If reel=0, it defaults to 1 or the first VSN. This field applies only when c=2 or 3.
	vsn	Volume serial number. The system returns the VSN of the volume assigned in option c=2 and 3.
Beta(5)	dom	Descriptor for operator message. If nonzero, this descriptor points to a message which is flashed on the 0 display after the MOUNT message. This field applies only for options c=2 and 3: 0-15 lom Length of operator message text, in bytes (1 ≤ lom ≤ 64). 16-63 aom Virtual bit address of the operator message. This address must begin on a byte boundary.
Beta(6)	dlb	Descriptor for the label buffer. This field is used for option c=4 only. The label buffer must contain a VOL1 and a HDR1 label.
Beta(7)	ccb	Change control bits:



Bit	Description
cs11	Maximum PRU size: 0 Do not change the maximum PRU size. 1 Change the maximum PRU size to that specified in the mpru field.
cs10	Tape mode: 0 Do not change the tape mode. 1 Change the tape mode to that specified in the tm field.
cs9	Label processing: 0 Do not change the label processing. 1 Change the label processing to that specified in the lp field.
cs8	Records per block: 0 Do not change the records per block. 1 Change the records per block to that specified in the rpb field.

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 7 of 9)

Word	Field	Description	
Beta(7)	ccb	Bit	Description
		cs7	Record mark: 0 Do not change the record mark. 1 Change the record mark to that specified in the rmd field.
		cs6	Padding character: 0 Do not change the padding character. 1 Change the padding character to that specified in the pc field.
		cs5	Record type: 0 Do not change the record type. 1 Change the record type to that specified in the rt field.
		cs4	Maximum record length: 0 Do not change the maximum record length. 1 Change the maximum record length to that specified in the rlmax field.
		cs3	Minimum record length: 0 Do not change the minimum record length. 1 Change the minimum record length to that specified in the rlmin field.
		cs2	Blocking type: 0 Do not change the blocking type. 1 Change the blocking type to that specified in the bt field.
		cs1	File organization: 0 Do not change the file organization. 1 Change the file organization to that specified in the sfo field.
Beta(8)	sfo	File organization: 0 Sequential.	

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 8 of 9)

Word	Field	Description
Beta(8)	bt	Blocking type: 0 SIL assumes the file was created before SIL was added to the system; therefore, it enters default values in the SIL fields of the file index entry. 1 Internal blocking (I). 2 C-type blocking. 4 Exact record count blocking (K).
	rt	Record type: 0 Control word (W). 1 ANSI fixed length (F). 2 Record mark (R). 4 Lower CYBER control word (L). 5 System block (B). 7 Undefined (U).
	rmd	Record mark; 8-bit ASCII character (any character is valid).
	pc	Padding character; 8-bit ASCII character (any character is valid).
Beta(9)	rlmin	Minimum record length; 24-bit length in number of bytes.
	rlmax	Maximum record length; 24-bit maximum length in number of bytes.
Beta(A)	rpb	Records per block.
	lproc	Label processing options.
	conv	Data conversion option: 0 There is no data conversion. 1 Convert data.
	mpru	Maximum length of the PRU.

Figure 5-42. TAPE MANAGEMENT (f=#002C) Message Format (Sheet 9 of 9)

TAPE SWITCH VOLUME (f=#002D)

This virtual system message causes the system to perform end-of-tape processing on the current volume and position to the beginning of volume on the next reel. The user is blocked until completion of the call. The logical tape file must be opened. This call will not position past a beginning-of-file or end-of-file label group.

The TAPE SWITCH VOLUME message enables the user to perform his own end-of-tape processing. When an I/O operation encounters end-of-tape, control is returned to the user if the user selected the end-of-tape processing option on the OPEN FILE message. An ioer=40 is returned in the TAPE FUNCTION message that encountered end of tape. An ioer=31 is returned in any other TAPE FUNCTION message outstanding at the time end of tape was encountered. The TAPE SWITCH VOLUME message automatically clears any existing ioer condition. If the user wishes to perform any tape function call before the TAPE SWITCH VOLUME system message, the user must issue the TAPE FUNCTION message, system function #30, to clear the ioer.

At the time the TAPE SWITCH VOLUME option is issued, the system performs end-of-tape processing on the current volume. If the last operation was a write, the system performs the following:

Tape Format	Unlabeled	Labeled
V	Writes two tape marks	Writes a tape mark, an EOV1, and two tape marks. If the user end-of-volume label buffer was supplied, the system may also write EOV2 through EOV9 and UTL labels.
I,SI,LB	Writes a tape mark, an EOV1 label, and two tape marks	Same as for labeled V tapes.

If the last operation was a read, the tape is labeled, a tape mark immediately follows, the user is supplied an end-of-volume label buffer, and all labels from this tape mark (beginning with EOV1) through the next tape mark are returned (as space permits).

The current tape is unloaded and the system requests the operator to mount the next VSN. After the tape has been assigned and if it is labeled, the system reads or writes beginning-of-volume labels, depending on whether the last operation was a read or write, respectively. The user may supply a beginning-of-volume label buffer. If the label is being read, any nonzero fields in the user HDR1 label supplied at LABEL time are compared with the HDR1 field on the tape. An error is returned if any nonzero field does not match. Then all labels from VOL1 through the first tape mark are returned to the user label buffer, as space permits. Verification of additional labels is the user's responsibility. If the label is being written, the system uses the previous VOL1 and HDR1 labels. The current VSN is placed in VOL1 and the chapter number is incremented by one in HDR1. UVL, HDR2 through HDR9, and UHL labels are written if present in the user buffer.

The format of the TAPE SWITCH VOLUME message is shown in figure 5-43.

	0			63
Alpha(1)	r 16	len 16	c 16	002D 16
Alpha(2)	unused 16	eea 48		
Alpha(3)	bl 16	ba 48		

Beta(1)	ioc 8	unused 40	ioer 16
Beta(2)	dtt 64		
Beta(3)	dvsn 64		
Beta(4)	duelb 64		
Beta(5)	delb 64		
Beta(6)	dublb 64		
Beta(7)	dblb 64		

Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 Illegal I/O connector. 2 Tapes table descriptor is too short. 3 Tape input/output error is returned. 4 Label buffers are too short. 5 Illegal labels. 6 Not at end-of-volume. 7 Volume not available. 8 Label is unexpired and IP_TPEXP=0.

Figure 5-43. TAPE SWITCH VOLUME (f=#002D) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	len	If this field is #FFFF, bl in Alpha(3) contains the length of the remote Beta buffer. ba contains the location of the remote Beta buffer. If this field is not #FFFF, Beta is assumed to begin at Alpha(3), and len is the length, in words, of the Beta portion.
	c	Message options: 0 If the last function was a write, switch the volume. If the last function was not a write, switch the volume if the tape is positioned at the trailer labels. 1 If at least one write operation was issued for this file, write trailer labels at the current tape position and switch the volume. Be aware that data may be lost. If there was no write operation for this file, switch the volume if the tape is positioned at the trailer labels.
Alpha(2)	eea	Error exit address.
Alpha(3)	bl,ba	If the Beta and Alpha portion are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	ioc	Input/output connector for this tape file.
	ioer	Input/output error number. Refer to appendix B for more a detailed description of the ioer codes.
Beta(2)	dtt	Tapes table descriptor. If nonzero, the tapes table is not returned: 0-15 ltt Word length of the tapes table buffer. This buffer must be 12 words long. 16-63 att Bit address of the tapes table buffer. The buffer must begin on a word boundary.
Beta(3)	dvsn	Descriptor for the VSN list. If nonzero, the VSN list is returned by the system: 0-15 lvsn Length of the VSN list, in words (0 < lvsn < 256). 16-63 avsn Virtual bit address of the VSN list. This field must be on a word boundary.
Beta(4)	duelb	Descriptor for user end-of-volume labels. If duelb is nonzero, then user end-of-volume labels are supplied by the user. This applies only when writing labels.

Figure 5-43. TAPE SWITCH VOLUME (f=#002D) Message Format (Sheet 2 of 3)

Word	Field	Description
Beta(5)	delb	End-of-volume label buffer descriptor. If delb is nonzero, the system returns all the end-of-volume labels here: 0-15 lelb Length of end-of-volume label buffer, in words. 16-63 aelb Bit address of the end-of-volume label buffer. The buffer must begin on a word boundary.
Beta(6)	dublb	Descriptor for user beginning-of-volume labels. If dublb is nonzero, the user beginning-of-volume labels are supplied by the user. This field applies only when writing labels: 0-15 lublb Length of user beginning-of-volume label buffer, in words. 16-63 aublb Virtual bit address of user beginning-of-volume label buffer. The buffer must begin on a word boundary.
Beta(7)	dblb	Beginning-of-volume label buffer descriptor. If dblb is nonzero, the system returns all of the beginning-of-volume labels here: 0-15 lublb Length of beginning-of-volume label buffer, in words. 16-63 aublb Bit address of beginning-of-volume label buffer. The buffer must begin on a word boundary.

Figure 5-43. TAPE SWITCH VOLUME (f=#002D) Message Format (Sheet 3 of 3)

LABEL (f=#002E)

This message is issued to request a logical file which belongs to an existing multifile set. The logical file is a local file. One or more LABEL calls can be issued for the same multifile set. One or more LABEL calls can be issued for the same logical file within the multifile set as long as the logical file is closed: that is, the LABEL message cannot be issued for an open file. Label processing is performed at OPEN time.

The file attributes of the multifile set are assigned to the logical file. The change control bits define which attributes are superceded by the LABEL message.

The format of the LABEL message is shown in figure 5-44.

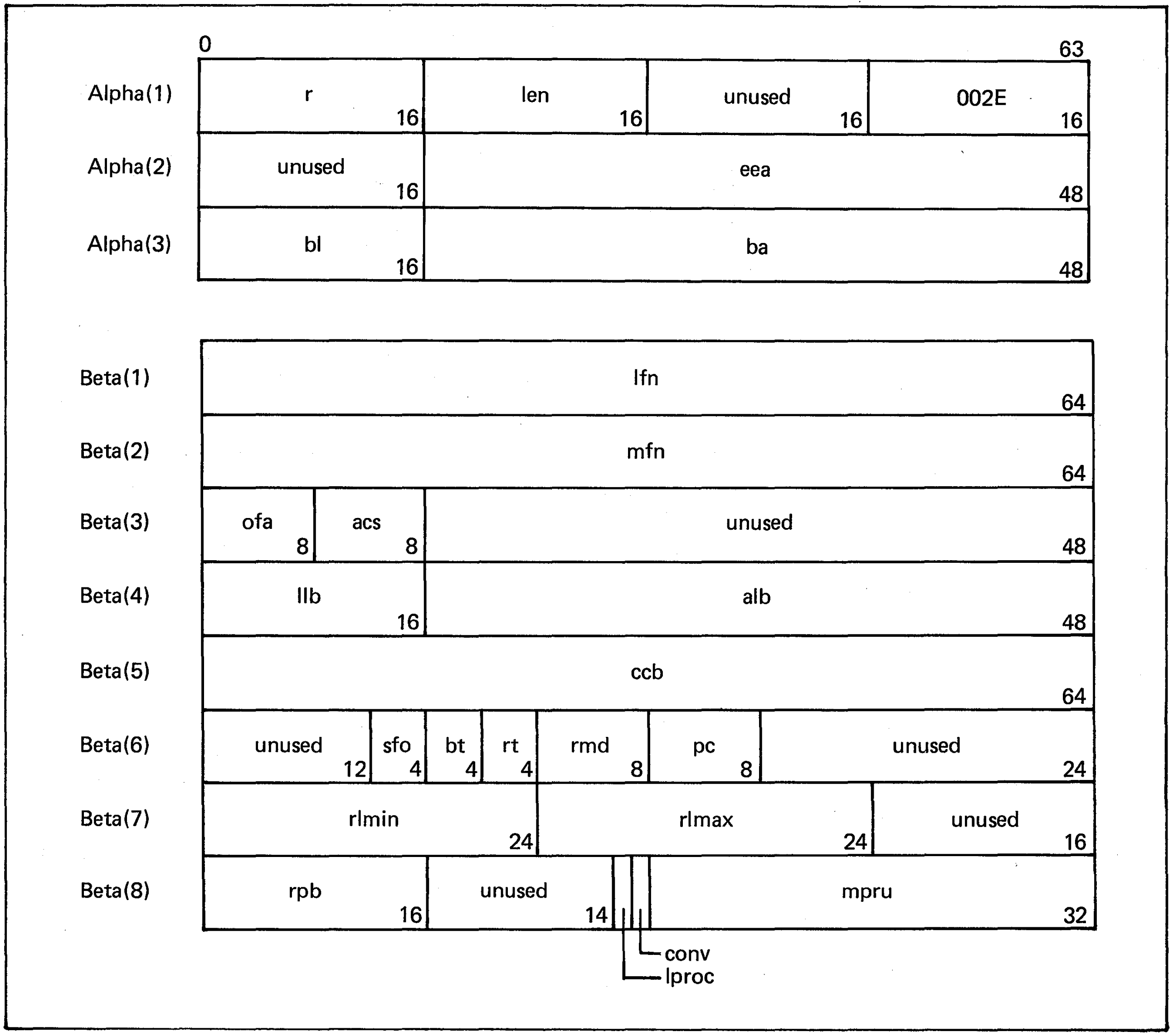


Figure 5-44. LABEL (f=#002E) Message Format (Sheet 1 of 5)

Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 Multifile set does not exist. 2 Logical file already exists and it does not belong to this multifile set. 4 Label buffer out of bounds. 5 lfn is open; ioc exists. 6 Illegal labels. 7 Illegal access. 8 Illegal original file accessibility character. 9 Duplicate FSN specified. #B Tape requested is unlabeled or nonstandard. #C No unit assigned. #D No room in FILEI for entry. #E Illegal logical file name.
	len	If this field is #FFFF, bl in Alpha(3) contains the length of the remote Beta buffer. ba contains the location of the remote Beta buffer. If this field is not #FFFF, Beta is assumed to begin at Alpha(3), and len is the length, in words, of the Beta portion.
Alpha(2)	eea	Error exit address.
Alpha(3)	bl, ba	If the Beta and Alpha portion are not contiguous (len=#FFFF), these parameters indicate the length in full words and virtual bit address of the first full word of the Beta portion.
Beta(1)	lfn	Logical file name. If the tape file is an ANSI standard labeled tape, lfn is also the multifile set name.
Beta(2)	mfn	Multifile set name. If this field is 0, mfn is the same as the logical file name.
Beta(3)	ofa	Original file accessibility character. This field must match the file accessibility character in the tape HDR1 label. This applies only when labels are being written. The default is the installation parameter IP_TPFA.
	acs	Access permissions for the logical file. acs must be a subset of the access permissions supplied at the time of the request: 1 Write permission only. 2 Read permission only. 3 Read/write permission.
Beta(4)	llb	Length of the label buffer, in words.
	alb	Virtual bit address of the label buffer. The buffer must begin on a word boundary.

Figure 5-44. LABEL (f=#002E) Message Format (Sheet 2 of 5)

Word	Field	Description
Beta(5)	ccb	Change control bits:

Bit	Description
cs11	Maximum PRU size: 0 Do not change the maximum PRU size. 1 Change the maximum PRU size to that specified in the mpru field.
cs10	Tape mode: 0 Do not change the tape mode. 1 Change the tape mode to that specified in the tm field.
cs9	Label processing: 0 Do not change the label processing. 1 Change the label processing to that specified in the lp field.
cs8	Records per block: 0 Do not change the records per block. 1 Change the records per block to that specified in the rpb field.
cs7	Record mark: 0 Do not change the record mark. 1 Change the record mark to that specified in the rmd field.
cs6	Padding character: 0 Do not change the padding character. 1 Change the padding character to that specified in the pc field.
cs5	Record type: 0 Do not change the record type. 1 Change the record type to that specified in the rt field.

Figure 5-44. LABEL (f=#002E) Message Format (Sheet 3 of 5)

Word	Field	Description	
Beta(5)	ccb	Bit	Description
		cs4	Maximum record length: 0 Do not change the maximum record length. 1 Change the maximum record length to that specified in the rlmax field.
		cs3	Minimum record length: 0 Do not change the minimum record length. 1 Change the minimum record length to that specified in the rlmin field.
		cs2	Blocking type: 0 Do not change the blocking type. 1 Change the blocking type to that specified in the bt field.
		cs1	File organization: 0 Do not change the file organization. 1 Change the file organization to that specified in the sfo field.
Beta(6)	sfo	File organization: 0 Sequential.	
	bt	Blocking type: 0 SIL assumes the file was created before SIL was added to the system; therefore, it enters default values in the SIL fields of the file index entry. 1 Internal blocking (I). 2 C type blocking. 4 Exact record count blocking (K).	
	rt	Record type: 0 Control word (W). 1 ANSI fixed length (F). 2 Record mark (R). 4 Lower CYBER control word (L). 5 System block (B). 7 Undefined (U).	
	rmd	Record mark; 8-bit ASCII character (any character is valid).	
	pc	Padding character; 8-bit ASCII character (any character is valid).	

Figure 5-44. LABEL (f=#002E) Message Format (Sheet 4 of 5)

Word	Field	Description
Beta(7)	rlmin	Minimum record length; 24-bit length in number of bytes.
	rlmax	Maximum record length; 24-bit maximum length in number of bytes.
Beta(8)	rpb	Records per block.
	lproc	Label processing options: 0 Read and verify the existing labels. 1 Write new labels.
	conv	Data conversion option: 0 There is no data conversion. 1 Convert data.
	mpru	Maximum length of the PRU.

Figure 5-44. LABEL (f=#002E) Message Format (Sheet 5 of 5)

USER REPRIEVE (f=#002F)

The USER REPRIEVE system message allows the user to have control returned to a specified address for processing during termination processing.

The aaf field is set when an application accounting routine makes this system message call. After the aaf flag is set, enable or disable of user reprieve is not allowed until an accounting routine with the appropriate password disables user reprieve.

The format of the USER REPRIEVE system message is shown in figure 5-45.

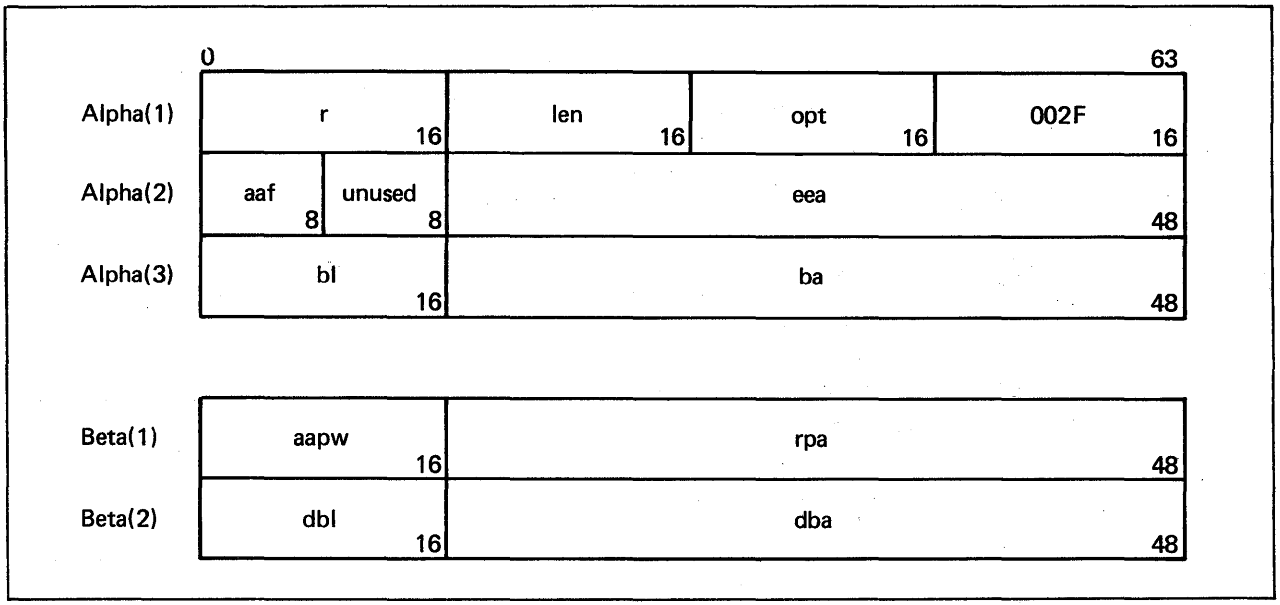


Figure 5-45. USER REPRIEVE (f=#002F) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	r	Response code: 0 No error. 1 Routine or data base address error. 2 Routine data base length error. 4 Reprieve enable or disable not allowed. #214 Beta buffer length error.
	len	Length of the Beta. If len=#FFFF, then Alpha(3) contains the length and starting address of the Beta; otherwise, Beta(1) is assumed to start in Alpha(3).
	opt	Option code: 0 Enable user reprieve. 1 Disable user reprieve.
Alpha(2)	aaf	Application accounting flag: 0 Off. 1 On.
	eea	Error exit address.
Alpha(3)	bl	Beta length.
	ba	Beta address.
Beta(1)	aapw	Accounting password.
	rpa	Reprieve address.
Beta(2)	dbl	Data base length.
	dba	Reprieve data base address.

Figure 5-45. USER REPRIEVE (f=#002F) Message Format (Sheet 2 of 2)

EXECUTE IQM REQUEST (f=#0030)

The EXECUTE IQM REQUEST system message processes IQM requests. Only privileged system tasks can issue this message. Only the IQM utility can issue c option 1.

The format of the EXECUTE IQM REQUEST system message is shown in figure 5-46.

	0			63
Alpha(1)	r 16	len 16	c 16	0030 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		

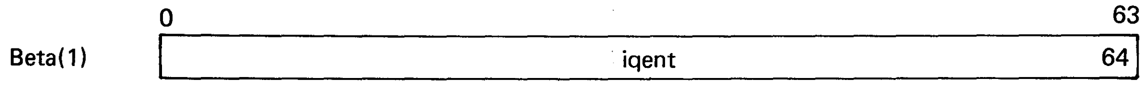
Word	Field	Description
Alpha(1)	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 Bad parameter; ss code contains the description. 2 User number of message issuer is not that of the IQM. 3 Job file was not added to the input queue because the queue is full. 4 Invalid message option (c field value). #12 Bad caller. #13 Input queue is full; resubmit job. #14 Bad c option. #15 Batch input file is not found. #16 Batch input file is open. #17 Batch user is a privileged system task. #18 IQM does not exist; consult an analyst. #19 Error was made in giving batch input file to IQM; consult an analyst. #1A Batch input file is of wrong type (tape of connected file). #1B Device on which batch input file resides is logically down. #1C Caller does not own batch input; cannot give it to IQM. #211 Either 0 or too many Beta entries were specified. #214 Beta buffer length error. Either the first word address of Beta plus its length is greater than the maximum virtual user address, or the Beta buffer is too small for the number of requests and length specified.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3) and len is the length in words of the Beta portion.

Figure 5-46. EXECUTE IQM REQUEST (f=#0030) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	c	Message options: 1 Remove an entry from the input queue. 2 Add an entry to the input queue.
Alpha(2)	n	Number of input queue entries to remove (also number of Beta entries; c=1 only).
	eea	Error exit address; virtual bit address to receive control if an error occurs during message processing (r≠0). If this field is 0 when an error occurs, the task is aborted.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length in words of Beta and the virtual bit address of its first full word.

Message option:

c=1 Remove an entry from the input queue and reorder the subpriorities of the remaining entries as needed:



Word	Field	Description
Beta(1)	iqent	Job descriptor number of entry to be deleted (0 through 2047).

Message option:

c=2 Add an entry to the input queue. This option changes file ownership from the caller to IQM. After the input queue entry is made, the job is scheduled as appropriate:

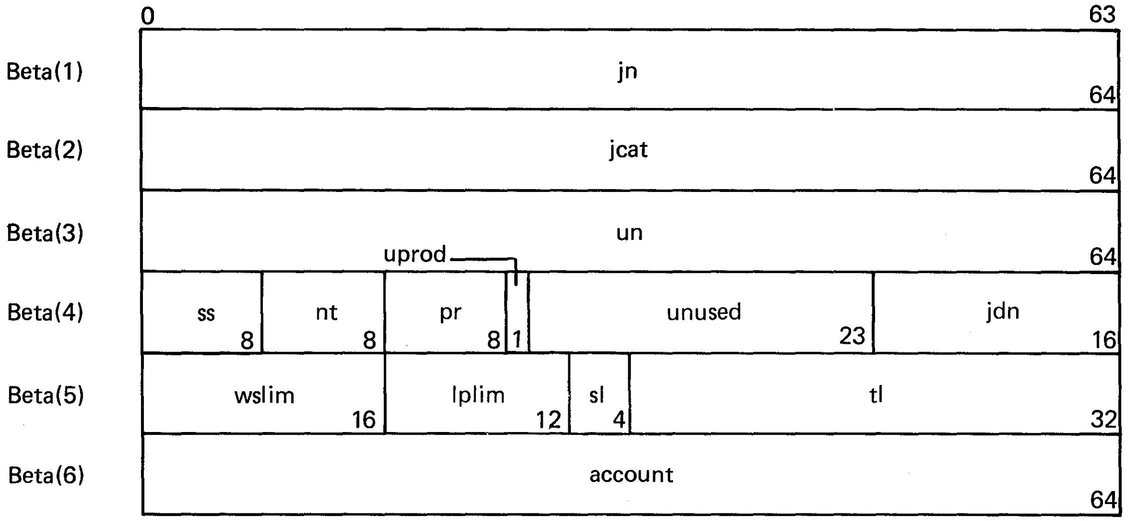


Figure 5-46. EXECUTE IQM REQUEST (f=#0030) Message Format (Sheet 2 of 3)

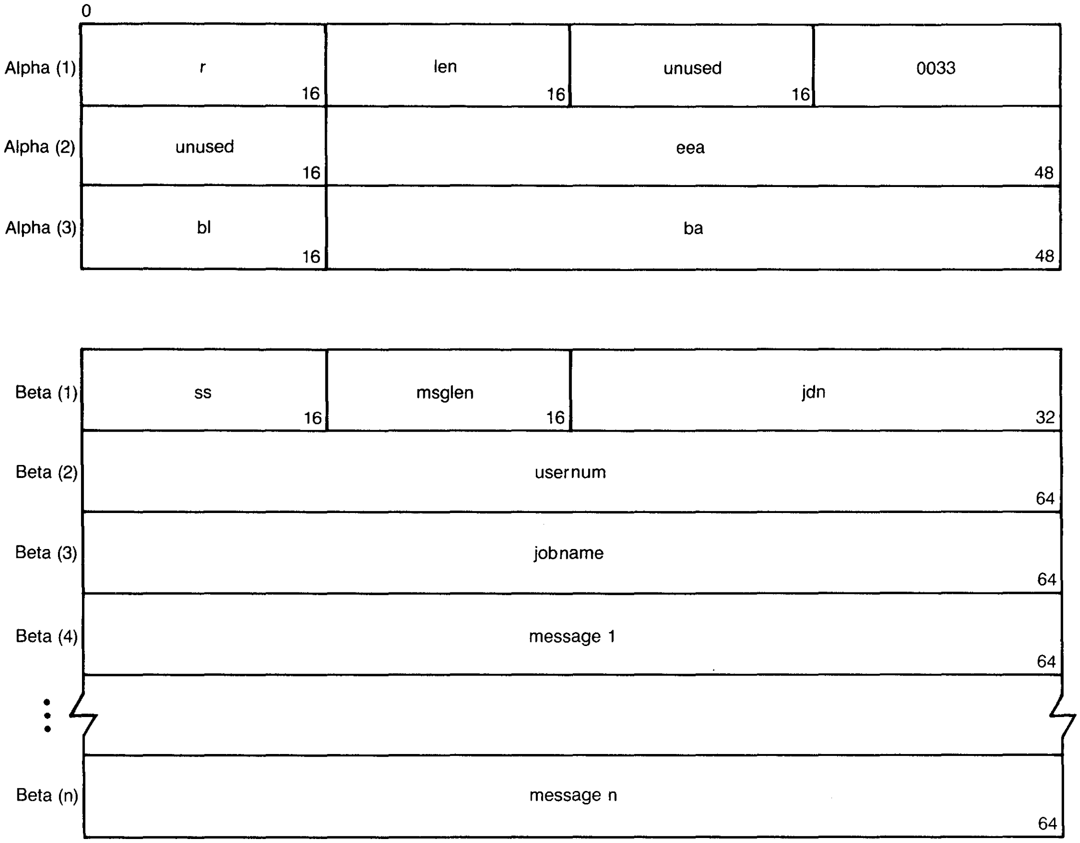
Word	Field	Description
Beta(1)	jn	Batch input file name (eight ASCII characters, blank-filled). IQM will modify jn, if necessary, to cause the batch name for this job to be unique on the system. The modified file name is returned in this field. If the call fails, jn is not modified.
Beta(2)	jcat	Job category (eight ASCII characters, blank-filled or 0).
Beta(3)	un	Binary user number of owner of the job.
Beta(4)	ss	Error response code: 0 No error. 1 Job category does not exist. 2 Maximum working set limit is exceeded. 3 Maximum large page limit is exceeded. 4 Invalid time limit. 5 Invalid priority. 6 User is locked out of specified job category. 7 Invalid user number. 8 Number of jobs per user exceeded. 9 No JDNs available to assign to input file. 15 Large page limit exceeded.
	nt	Number of tape drives required by the job.
	pr	Priority (1 through 15).
	uprod	0 is not a production user. 1 is a production user.
	jdn	Job descriptor number assigned to the job. Values for jdn are 1 through 2047.
Beta(5)	wslim	Working set limit. If 0, the job will be assigned a working set limit equal to the maximum for the job category.
	lplim	Large page limit.
	sl	Security level (1 through 8).
	tl	Time limit in system seconds.
Beta(6)	account	Account identifier under which the job will run.

Figure 5-46. EXECUTE IQM REQUEST (f=#0030) Message Format (Sheet 3 of 3)

SEND MESSAGE TO JOB SESSION (f=#0033)

This message allows the user to send a message to a batch job's dayfile or to an interactive user terminal. Only privileged tasks are allowed to use this system call. VSOS uses the job descriptor number, the user number, and the job name supplied in the Beta to locate the proper job session. If found, the message is queued up for delivery to the job session. If the job session is not found or other errors are encountered, an error response is returned in the Alpha.

The format of the SEND MESSAGE TO JOB SESSION is shown in figure 5-47.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when the message has been processed. If no error occurs, the error code is 0; otherwise: 1 Length of Beta in character bytes is either 0 or greater than 4120. 2 Caller is not a privileged task. 3 Job session is not active. 4 System error (ss code in Beta is error code returned by system routine TOCON).

Figure 5-47. SEND MESSAGE TO JOB SESSION (f=#0033) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(1)	len	If len=#FFFF, Alpha(3) contains the length and virtual bit address of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length of the Beta portion.
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during message processing (if r is different from 0). If this field is zero when the error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Alpha and Beta portions are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion. There is only one Beta per Alpha.
Beta(1)	ss	Error return code (integer) for r=4. 1 Message length is greater than 4096 characters. 2 Bad function code in Beta. 3 Dayfile is full. 4 Dayfile is not implicit. 5 Invalid base virtual address for dayfile. 6 Unable to find dayfile. 7 Problem program blocked while waiting for dayfile. 8-13 Unused. 14 No terminal ID number for interactive user. 15 Either teletype has been logged out or this message would overflow the MFline buffer.
	msglen	Message length in characters (integer)
	jdn	Job descriptor number (1 through 2047).
Beta(2)	usernum	User number, in ASCII, left-justified, blank-filled.
Beta(3)	jobname	Job name, in ASCII, left-justified, blank-filled.
Beta(4) through Beta(n)	message	Message text, in ASCII.

Figure 5-47. SEND MESSAGE TO JOB SESSION (f=#0033) Message Format (Sheet 2 of 2)

RETURN FROM INTERRUPT (f=#0051)

For control to return to the calling routine, an interrupt routine must issue this message when it has finished performing its tasks for either an input/output or program message interrupt. The message consists of an Alpha portion only, which is shown in figure 5-48.

Because the interrupt routine (level 1) cannot be interrupted by any other software interrupts, it will run until it issues a RETURN FROM INTERRUPT message. The current interrupt is then released and its invisible package is lost. The level-0 invisible package becomes current, and its register file image is restored by the operating system. All information from the level-1 register file is lost.

An option in this message allows level 1 to become the new level 0 after all additional interrupts stacked for this and any other level-1 routines have been processed. In this case, the register file image for level 0 is lost at the time level 1 becomes level 0. The new level 0 can have its own level-1 interrupt routines.

When interrupts occur and the interrupt routine is already in control, the operating system stacks the interrupt information in the interrupt address stack in the program's minus page. When the interrupt routine issues a RETURN FROM INTERRUPT message, level 0 is not restarted. Instead, the next interrupt on the stack is taken. This process is repeated until the stack is empty.

	0			63
Alpha(1)	r 16	len 16	unused 16	0051 16
Alpha(2)	n 16	eea 48		
Alpha(3)	bl 16	ba 48		

Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response code is 0; otherwise: 1 Response code was issued by routine other than interrupt routine. 2 Illegal Beta address (only for n=4). #214 Beta buffer length error (only for n=4).
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3), and len is the length in words of the Beta portion (two words). Used only if n=4.

Figure 5-48. RETURN FROM INTERRUPT (f=#0051) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	n	Message options. Release the current interrupt and take the next interrupt in the stack, if one exists. When all interrupts outstanding have been processed, or if no other interrupts existed, do one of the following: 0 Return control to the interrupted program at the point of interruption. 1 Return control to the point following this particular RETURN FROM INTERRUPT message; that is, make the interrupt routine that issued this message the new level-0 routine. 2 Return control to the interrupted program at the address in register 4. 3 Abnormal termination control interrupt only. Abort at the point of original interrupt. Normal user dumps and trace-back information are produced for the original fatal error. 4 Return control to the interrupted program at the address in Beta(1) using data base information in Beta(2).
	eea	Virtual bit address to receive control if an error occurs during processing of this message (r≠0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If Beta and Alpha portions of the message are not contiguous (len=#FFFF), these parameters indicate the length and virtual bit address of the first full word of the Beta portion.

Message option:

For n=4 only:

	0	63
Beta(1)	unused 16	vba 48
Beta(2)	dbl 16	dba 48

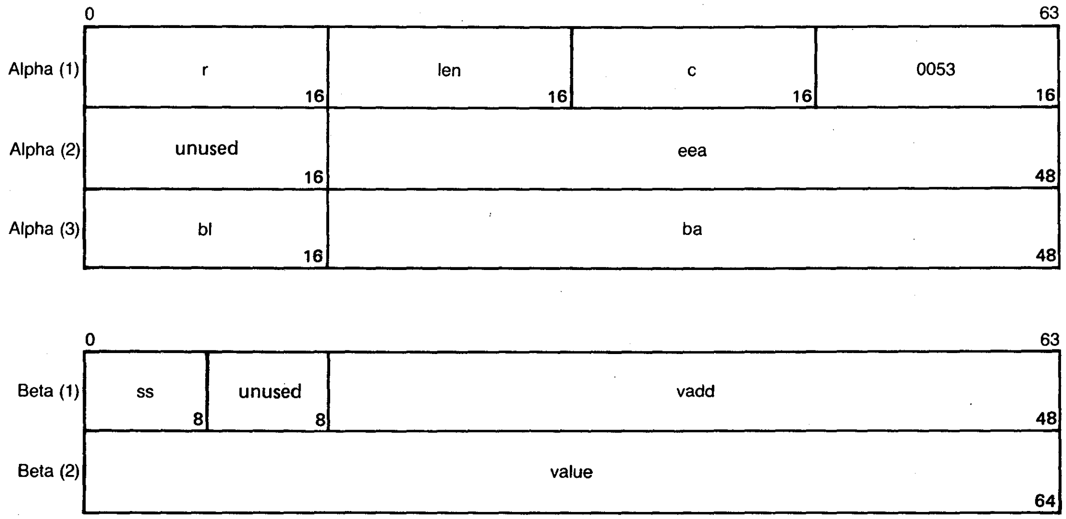
Word	Field	Description
Beta(1)	vba	Virtual bit address in the interrupted program to which control is returned.
Beta(2)	dbl	Length of data base to be reloaded.
	dba	Address of data base to be reloaded for return to interrupted program.

Figure 5-48. RETURN FROM INTERRUPT (f=#0051) Message Format (Sheet 2 of 2)

SHRLIB ALTER OR RESTORE (f=#0053)

A user program can issue this message for altering or restoring the contents of the system shared library file. The operating system keeps a record of alterations for each user program. Alteration for shared library in one user program does not affect the other user program. If a user program does not restore the shared library before it terminates, the operating system automatically restores the shared library file when this user program terminates.

The format of this message is shown in figure 5-49.



Word	Field	Description
Alpha(1)	r	Response code; returned by the operating system when this message has been processed. If no error occurs, the response is 0; otherwise: 1 ss contains error code. #214 Beta buffer error; either the first word address of Beta plus length is greater than the maximum user virtual address or the length of the Beta buffer is greater than 2.
	len	If this field is #FFFF, Alpha(3) contains the length and virtual bit address of the Beta portion of the message; otherwise, Beta is assumed to begin at Alpha(3) and len is the length in words of the Beta portion; len must be at least one word long.
	c	Message option: 1 Place value at vadd; if location is currently altered, it will be altered again. 2 Restore original value at vadd.

Figure 5-49. SHRLIB ALTER OR RESTORE (f=#0053) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	eea	Virtual bit address to receive control if an error occurs during processing of this message (r not equal to 0). If this field is 0 when an error occurs, the error is considered fatal.
Alpha(3)	bl, ba	If the Beta and Alpha portions of the message are not contiguous (len=#FFFF), the parameter indicates the length and virtual bit address of the first full word of the Beta portion.
Beta(1)	ss	Return error code: 0 No error. 1 vadd is not in SHRLIB range. 2 There is no room in the SHRALT table for another change to SHRLIB; C must equal 1. Each DB is allowed a maximum of 20 changes with a total of 127 for all DBs. 3 The page containing vadd is not in memory; the shared library working set needs to be increased. 4 vadd is not on a word boundary. 5 Location being restored was never changed; C must equal 2. 6 No value specified, len was 1; C must equal 1. 7 Illegal option; C must equal 1 or 2.
	vadd	Bit address on a word boundary of the location that is to be altered/restored.
Beta(2)	value	For C=1, this is the 64-bit value to be placed at vadd. For C=2, this is not used.

Figure 5-49. SHRLIB ALTER OR RESTORE (f=#0053) Message Format (Sheet 2 of 2)

TAPE FUNCTION (f=#F406)

The TAPE FUNCTION message is processed by the resident system. It is issued by user mode programs to initiate tape I/O and positioning function. The tape file must be open. This message does not read or position past a beginning-of-file or end-of-file label group. If a positioning function causes the tape to be positioned backwards and the last operation was a write, the system performs end-of-file processing. For V unlabeled tapes, it writes two tape marks. For I, SI, LB labeled or unlabeled tapes and V labeled tapes, it writes one tape mark and an EOF1 label followed by two tape marks.

This message allows the user to continue processing or give up the CPU until I/O completion. The resident give-up call (f=#FF02) is issued to check for I/O completion.

The maximum number of TAPE FUNCTION messages outstanding at any one time for a tape file is defined by the installation parameter IP_TPNOR. If a program issues one more TAPE FUNCTION message for a tape file than IP_TPNOR, the caller is blocked until one of the caller´s previous TAPE FUNCTION messages completes.

The TAPE FUNCTION system message can be broken down into four different chapters. The first chapter consists of user-supplied information: I/O connector number and function code. For I/O function (sfnc < #10), the user must set the buffer address, buffer length, and the length of the logical record unit array. The buffer cannot span more than 48 small or large pages. Depending on the word offset (wo=0/1), the buffer address begins on any half/full-word boundary. Each logical record unit begins on the next 32/64-bit boundary in case the preceding LRU is not a multiple of 32/64 bits. For some of the positioning functions, a skip count must be supplied.

For the read function, the buffer length must be at least large enough to hold the maximum PRU size supplied at open time. The system will not start tape motion to read the next PRU unless there is at least room in the buffer to hold mpru. Therefore, it is recommended that the user add mpru to the read buffer length. For the read skip function, mpru is ignored; however, the user can specify a maximum LRU that is to be returned. The buffer does not have to be as long as mlru. If the system has returned at least one LRU to the buffer, the system will not start tape motion to read the next LRU unless there is at least room in the buffer to hold mlru. If mlru=0, it is considered infinite and mlru=0 results in single LRUs.

The second chapter is filled in from system tables by the system. It consists of information supplied by the user at request, label, or open time: maximum PRU size and format parameters. It also contains the caller user number and job name.

The third chapter consists of information returned by the system at the competion of the call or a tape I/O error. For I/O functions, the number of processed LRUs is returned. For some of the positioning functions, a skip count is returned. The updated tapes table, including block IDs and PRU counts, is also returned at the completion of the request. This information can be saved.

The last chapter holds the LRU array. It is only used for I/O functions (sfnc < #10). Each entry holds the logical record size, tape, and record mark information. It is set by the caller for a write and returned by the system for a read.

The format of the TAPE FUNCTION message is shown in figure 5-50.

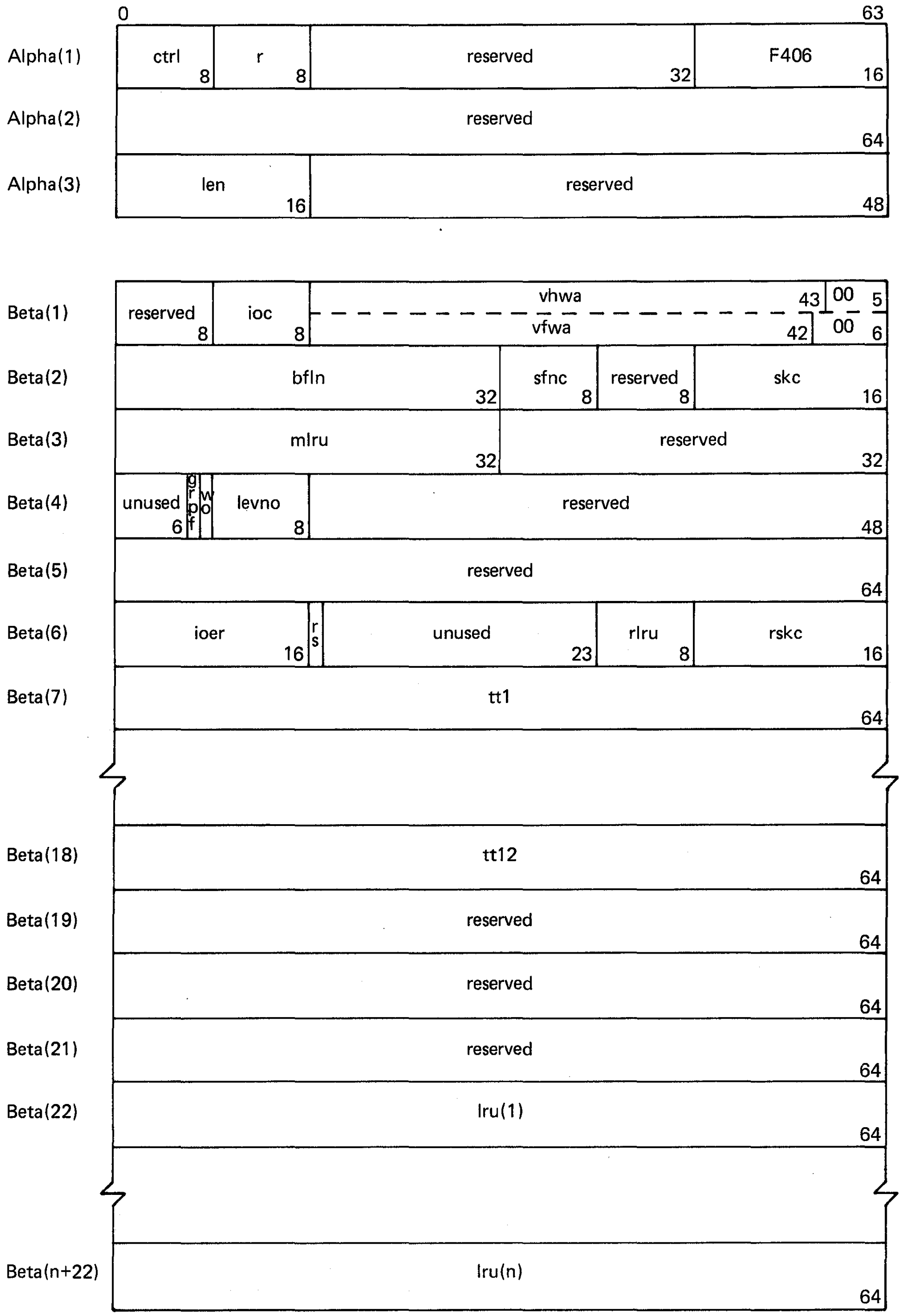
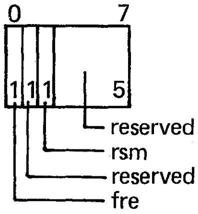


Figure 5-50. TAPE FUNCTION (f=#F406) Message Format (Sheet 1 of 5)

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>Alpha(1)</td><td>ctrl</td><td>Control bits:


<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>fre</td><td>Cleared when the call is complete. The caller must set the fre bit.</td></tr>
<tr><td>rsm</td><td>Set to 1 if the caller wants to be resumed immediately. rsm=0 if the caller wants to give up until the I/O completes. For backward positioning functions (#10 ≤ sfnc ≤ #1F), the system sets rsm=0.</td></tr>
</table></td></tr>
<tr><td></td><td>r</td><td>Response code returned by VSOS when message processing is complete:
<table>
<tr><td>0</td><td>No error.</td></tr>
<tr><td>1</td><td>Error occurred before the request was issued to the tape subsystem.</td></tr>
<tr><td>2</td><td>Error occurred and request was issued to the tape subsystem. The error number is returned in ioer.</td></tr>
</table></td></tr>
<tr><td>Alpha(3)</td><td>len</td><td>Length of Beta, in words. For positioning functions (sfnc ≥ #10), the user should set len=21. For I/O functions (sfnc < #10), the user should set len=21 plus the number of words in the LRU array.</td></tr>
<tr><td>Beta(1)</td><td>ioc</td><td>Input/output connector number of the tape, set by the caller.</td></tr>
<tr><td></td><td>vhwa</td><td>Virtual half-word address of the buffer, set by the caller. vhwa is used if wo=0.</td></tr>
<tr><td></td><td>vfwa</td><td>Virtual full-word address of the buffer, set by the caller. vfwa is used if wo=1.</td></tr>
<tr><td>Beta(2)</td><td>bfln</td><td>Overall buffer length in 8-bit bytes, set by the caller. bfln must be a multiple of 4/8 bytes based on the word offset (wo=0 or wo=1). On completion, bfln is set to the number of bytes left in the buffer.</td></tr>
</table>

Figure 5-50. TAPE FUNCTION (f=#F406) Message Format (Sheet 2 of 5)

Word	Field	Description
Beta(2)	sfnc	System function, set by the caller:
		1 Read data. The read data reads data from the tape and places it into the user´s buffer until the requested amount of data has been read. The read is stopped if the LRU array is full, the buffer does not contain sufficient space to hold mpru, a fatal error is encountered, or an end of group is encountered. If a PRU exceeds mpru, a DEVICE CAPACITY EXCEEDED I/O error is returned and the tape is positioned after the PRU. The EOR flag is only set in the last LRU entry if the read was stopped at the end of an LRU.
		2 Read skip. The read skip reads data from the tape and places it into the user´s buffer until the requested amount of data has been read. If a PRU exceeds mlru, only mlru bytes of data are returned to the buffer, the excess data flag is set, and the tape is positioned at the end of the LRU. The read skip is stopped if the LRU array is full, a fatal error is encountered, or an end of group is encountered. The EOR flag is always set in each LRU entry.
		8 Write data. The write data writes data on tape from the user´s buffer until the request has been completed. The write data is stopped if a fatal error is encountered. If the LRU size is not a multiple of the PRU size, an end of LRU is always written at the end of the LRU. If the LRU size is a multiple of the PRU size, an end of LRU is written if the EOR flag is set in the LRU array.
		#10 Skip backward PRUs. The skip backward PRUs backspaces physical records until the count in SC is completed or until end of LRU, end of group, or beginning of information is encountered.
		#11 Skip backward LRUs. The skip backward LRUs backspaces LRUs until the count in SC is completed or until an end of LRU with higher level, end of group, or beginning of information is encountered.
		#12 Skip backward groups. The skip backward groups backspaces groups until the count of SC is completed or until a beginning of information is encountered.
		#13 Rewind to beginning of information. The rewind to beginning of information rewinds the tape to the beginning of information.
		#14 Rewind volume. Rewinds the tape to the beginning of volume. The operation is stopped if a beginning of information is encountered on the current volume.

Figure 5-50. TAPE FUNCTION (f=#F406) Message Format (Sheet 3 of 5)

Word	Field	Description
Beta(2)	sfnc	#20 Skip forward PRUs. The skip forward PRUs forward spaces physical records until the count in SC is completed or until an end of LRU, end of group, or end of information is encountered.
		#21 Skip forward LRUs. The skip forward LRUs forward spaces LRUs until the count in SC is completed or until an end of LRU with a higher level, an end of group, an end of file, or end of information is encountered.
		#22 Skip forward groups. The skip forward groups forward spaces groups until the count in SC is completed or until an end of information is encountered.
		#23 Skip forward to end of information. The skip forward to end of information forward spaces the tape to the end of information.
		#30 Reset fatal error or group mark condition. After an ioer is returned or the group mark flag is set in the LRU array with grpf set in the call, the tape subsystem will signal an error to all subsequent calls for that unit with an ioer=31 until this system function is issued. A subsequent close of any backward positioning function also clears this error.
	skc	Skip count. The number of PRUs, LRUs, or logical files for sfnc. This field is ignored for all other options of sfnc.
Beta(3)	mlru	Maximum LRU size, in 8-bit bytes. For read skip only, if an LRU exceeds mlru, only mlru bytes of data are returned, the excess flag is set, and the tape is positioned at the end of the LRU. If mlru=0, mlru is considered infinite. This field is set by the system to mpru for all data functions except read skip.
Beta(4)	grpf	Group mark flag. If grpf=0, the tape subsystem will stop reading and terminate the message if a group mark is encountered. All subsequent calls for that unit will be returned with an ioer=#31. The user must issue sfnc=#30 to reset this condition. A subsequent CLOSE or backward positioning function on this unit will also clear this condition. If grpf=1, the tape subsystem will stop reading and terminate the call if a group mark is encountered. However, any subsequent calls for that unit will be issued.
	wo	Word offset. If wo=0, the next LRU begins on a 32-bit boundary, if the last did not end on one. If wo=1, the next LRU begins on a 64-bit boundary, if the last did not end on one.
	levno	Level number for the skip backward/forward LRU functions. This field applies only for I, SI, and LB tape formats ($0 \leq$ levno $\leq$ #E).

Figure 5-50. TAPE FUNCTION (f=#F406) Message Format (Sheet 4 of 5)

Word	Field	Description
Beta(6)	ioer	Error number returned by the system. For request type errors 1 through 100, control is returned to the caller. For tape I/O errors 101 through 200, control is returned to the caller only if user error processing was selected in the OPEN FILE message. Refer to appendix B for a complete description of these errors.
	rs	Reel swap, returned by the system: 0 No reel swap. 1 Reel swap occurred.
	rlru	Number of words in the LRU array completed. This field is returned by the system for I/O functions.
	rskc	Returned skip count. The number of PRUs, LRUs, or groups skipped.
Beta(7) through Beta(18)	tt	Tapes table entry, returned by the system. Refer to tapes table in job management tables for a complete description.
Beta 22 through Beta(n+22)	lru	Logical record unit array entry, set by the caller for a WRITE function and returned by the system for a READ or READSKIP function:

Bit	Description
00-03	Unused.
04-07	Level number. This field applies only for I, SI, and LB tape formats (0 ≤ lev ≤ #E).
08-10	Reserved.
11	Excess data flag. This can occur only on a read skip. It indicates that data was skipped.
12	Parity flag; set to 1 by the system when this LRU has an error. This applies only for read operations when no retry has been selected. The data is returned.
13	End-of-LRU flag; set to 1 by the caller to write an end of LRU. Returned by the system when an end of LRU was detected for a read.
14	End-of-group flag; set to 1 by the caller to write an end of group after this LRU. This is returned by the system when an end of group was detected for a read. This flag is set also if an end of information is encountered without encountering an end of group.
15	End-of-information flag; returned by the system when an end of information was detected for a read. This flag is ignored for a write.
16-31	Reserved.
32-63	LRU size, in 8-bit bytes. This field is set on a write and returned on a read.

Figure 5-50. TAPE FUNCTION (f=#F406) Message Format (Sheet 5 of 5)

EXPLICIT I/O (f=#F500)

The EXPLICIT I/O message is processed by the resident system. It is issued by user mode programs to initiate transfer of data to and from mass storage files to and from buffers defined by the message.

This message allows the user to continue processing and give up the CPU until I/O completion. The resident give-up call (f=#FF02) is issued to check for I/O completion.

The program´s minus page contains an I/O connector for each file the user program has opened; a file´s I/O connector number is issued to designate the file on which input/output is being performed. To perform explicit I/O on a file, the program must first open the file.

The Beta portion of the message contains the buffer definition. The user must set the buffer address and the buffer length. The buffer cannot scan more than 24 small or large pages. Therefore, to use the maximum buffer size, the buffer should be on a page boundary.

The Alpha and Beta words must be contiguous and not cross a page boundary. They should not be modified until all input and output described by the call is completed. The free bit in Alpha(1) has been defined to help the user determine when input/output is done; the user sets the bit before the message is issued, and the operating system clears the bit when the Alpha and Beta words are no longer in use. The resident give-up message (f=#FF02) can also be used to check for I/O completion.

When the central operating system detects an error before a request is sent to the peripheral operating system, the cerr field of the EXPLICIT I/O message is filled appropriately, control passes to the error exit address, and message processing terminates. A data transfer error detected by the peripheral operating system does not cause control to pass to the error exit address; however, the serr field of the EXPLICIT I/O message is filled appropriately.

The format of the Alpha portion of the EXPLICIT I/O message is shown in figure 5-51. The formats of the Beta portion are shown in figure 5-52.

<table>
<tr><td>Alpha(1)</td><td>ctrl
8</td><td>r
8</td><td colspan="2">reserved
32</td><td>#F500
16</td></tr>
<tr><td>Alpha(2)</td><td>01
8</td><td>00
8</td><td colspan="3">ca
48</td></tr>
<tr><td>Alpha(3)</td><td colspan="2">#0015
16</td><td colspan="3">eea
48</td></tr>
</table>

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>Alpha(1)</td><td>ctrl</td><td>Control bits:
<table>
<tr><th>Bit</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>free</td><td>Cleared when the call is completed. The caller must set the free bit.</td></tr>
<tr><td>2</td><td>rsm</td><td>Set to 1 if the caller wants to be resumed immediately. rsm=0 if the caller wants to give up until the I/O completes.</td></tr>
</table></td></tr>
<tr><td></td><td>r</td><td>Response code returned by VSOS when the message is complete:
<table>
<tr><td>0</td><td>No error.</td></tr>
<tr><td>1</td><td>Error occurred before the request was issued to the I/O device. The error number is returned in CERR.</td></tr>
<tr><td>2</td><td>Request was issued to the I/O device and an error occurred. The error number is returned in SERR.</td></tr>
</table></td></tr>
<tr><td>Alpha(2)</td><td>ca</td><td>Address at which execution continues following successful completion of the call. If 0, execution continues at the address following the call.</td></tr>
<tr><td>Alpha(3)</td><td>eea</td><td>Error exit address: virtual bit address to receive control if an error occurs during message processing (r≠0). If this field is 0 when an error occurs, the task is aborted.</td></tr>
</table>

Figure 5-51. EXPLICIT I/O (f=#F500) (Alpha) Message Format

Beta(1) through Beta (13)	• • •
Beta(14)	fc
Beta(15)	ioc
Beta(16)	fadd
Beta(17)	blen
Beta(18)	badd
Beta(19)	cerr
Beta(20)	serr
Beta(21)	pkno

Word	Field	Description
Beta(1) through Beta(13)		Used by the resident system to issue a C51x call.
Beta(14)	fc	Function code: 1 Read data from a disk file to a buffer. 2 Write data from a buffer to a disk file.
Beta(15)	ioc	Input/output connector number for the file on which input and output are being performed.
Beta(16)	fadd	Logical block address of the file where data transmission is to begin.
Beta(17)	blen	Length of the virtual range, in blocks, to be associated with this buffer. The maximum size is 24*n; n is the number of blocks in a page, large or small. If the maximum length is used, the buffer must be on a page boundary.
Beta(18)	badd	Starting virtual block address of the buffer where data transfer requests will deposit or obtain information.

Figure 5-52. EXPLICIT I/O (f=#F500) (Beta) Message Format (Sheet 1 of 2)

Word	Field	Description
Beta(19)	cerr	Errors detected by the central operating system before the request is sent to the I/O device. The values are: 1 Nonexistent I/O connector. 2 Buffer size is greater than 24 small pages, is 0, or is 24 pages and not on a page boundary. 3 This file is not open for explicit I/O. 4 Alpha/Beta crosses page boundary or is not contiguous. 5 Illegal function code. 7 No buffer address was given. 8 File address is out of bounds. 9 Illegal attempt to access a file. #B Buffer size is greater than 24 large pages, is 0, or is 24 pages and not on a page boundary for a large page buffer. #C Buffer lies on each large and small page. #F Buffer is already in use; previous I/O, which uses the same buffer, is not complete. #10 Attempt to reuse Alpha before the previous call, which uses the same Alpha address, is complete. #12 Attempt to read or write in an unassigned virtual space (buffer error). #14 File is not at end of information in append mode.
Beta(20)	serr	Errors detected by the I/O device. Bits are numbered from left to right, 0 to 23. They are: 0-17 No meaning. 18 No more disk space is available when extending the file. 19 Reached end of file. Indicates that the buffer extends past the end of the file. 20 Error encountered when extending the file. 21 Attempted to extend file beyond user´s or pool´s file limit. 22 Fatal device error detected by I/O device. 23 Illegal message detected by I/O device.
Beta(21)	pkno	Returned by operating system. Pack number of disk pack on which fatal device error was detected (SERR bit 22 set).

Figure 5-52. EXPLICIT I/O (f=#F500) (Beta) Message Format (Sheet 2 of 2)

ADVISE (f=#FF00)

The ADVISE message is used by a program to inform the operating system of an anticipated change in the need for virtual space. The ADVISE message has two purposes: either to provide execution and input/output overlap to reduce the number of page faults for a job, or to release pages of memory no longer required by that job. The ADVISE message is intended for use in improving job execution speed.

The ADVISE message indicates a virtual range (a range of virtual addresses). The ADVISE message can be used in one of these ways:

- To page in a virtual range.
- To page out a virtual range.
- To replace one virtual range with another.

The ADVISE in function is initiated without blocking the job from execution. Pages required to accommodate the ADVISE in request are obtained from the following categories in the specified order:

- Pages freed as the result of an accompanying ADVISE out function.
- Unallocated pages.
- Unmodified pages outside the working sets of connected tasks.
- Unmodified pages belonging to disconnected tasks.
- Modified pages belonging to the requestor but outside the requestor´s working set.

If insufficient memory is available to accommodate the ADVISE in function, as many pages as possible are ADVISEd in and the user is informed that only a partial function was performed. The maximum number of pages that are read into memory by PAGER for any single ADVISE in function is 16 small pages or 1 large page. If the requestor exceeds the limit, the maximum number of pages ADVISEd in will be the limit. The user is returned the highest virtual small page address plus one in the specified virtual range that is in memory after the ADVISE in function is complete.

The user should be aware of the following points:

- Only a single Beta is allowed for an ADVISE in function in an ADVISE message.
- If a requested page is already in memory, that page is ignored and the remaining pages are ADVISEd in.
- If a page fault occurs for an ADVISE in page prior to its arrival in memory, the system blocks the job from further execution until the page fault is satisfied.
- If a job has a machine-size working set, an ADVISE in function is accomplished by selecting the least recently used pages from within the job´s working set as replacement pages.
- A virtual bit address that is not defined in any virtual map is considered to be a definition of new free space. An appropriate entry is made in the drop file map, and memory space is allocated.

The ADVISE out function is used to remove a virtual range from memory and is initiated without the job being blocked from execution. All unlocked modified pages within the specified virtual range are written to mass storage. The pages are then deleted from the page table. The user should be aware of the following restrictions:

- Only a single Beta is allowed for an ADVISE out function in an ADVISE message.
- If a page within the specified virtual range is locked or not in memory, that page is ignored and the remainder of the request is processed. If a locked page is detected, the user is informed.
- If a write access occurs for a page being written to disk as the result of an ADVISE out, the job is page blocked until the input/output is complete.

An ADVISE replace function is the combination of an ADVISE in function and an ADVISE out function in a single ADVISE message. The function can replace in full, or in part, one virtual range with another. The system first performs the ADVISE out function and then initiates the ADVISE in function. At a minimum, the number of pages freed by the ADVISE out function are then available for the ADVISE in function. If fewer pages are specified in the ADVISE out function than in the ADVISE in function, additional pages are selected by the system and paged out to provide sufficient pages to satisfy the ADVISE in function. The additional pages can be obtained from the following categories in the specified order:

- Unallocated pages.
- Unmodified pages belonging to connected tasks but outside their working sets.
- Unmodified pages belonging to disconnected tasks.
- Modified pages outside the requestor's working set.

It should be observed that two Betas are specified for the ADVISE replace function: one for the ADVISE out specification, and one for the ADVISE in specification, in any order.

The ADVISE message format shown in figure 5-53 is used when an ADVISE is issued directly to the system.

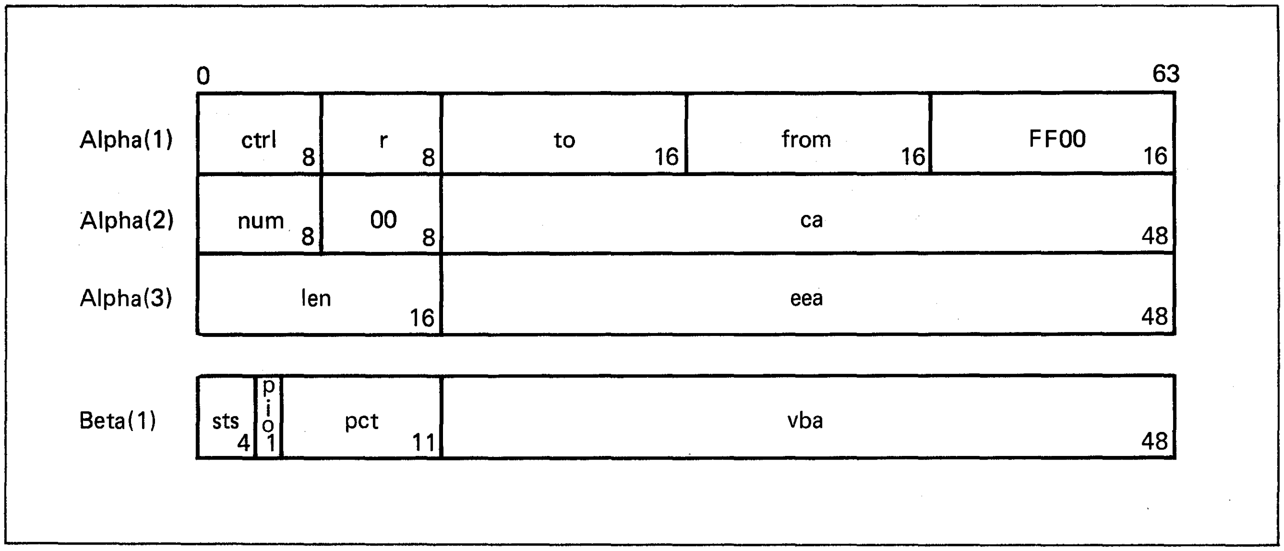


Figure 5-53. ADVISE (f=#FF00) Message Format (Sheet 1 of 3)

Word	Field	Description
Alpha(1)	ctrl	Should be set to 0. KERNEL will set this field to 0 on return to the caller.
	r	Response code returned by the operating system after the specified ADVISE function has been processed. If no error is detected and the full request is processed, r is set to 0; otherwise: 1 The sts field in the parameter set provides further status. 2 Illegal option specified for this message. Multiple ADVISE in or ADVISE out functions specified in a single message. 3 Address vba is out of the virtual address range for a user program. 4 Beta length error. Either the first word address of Beta plus length is out of the virtual address range for a user program or more than two parameter sets are specified.
	to	Should be set to 0. KERNEL will set this field to 0 on return to the caller.
	from	Should be set to 0. KERNEL will set this field to 0 on return to the caller.
Alpha(2)	num	Number of Betas for this call: 1 A single ADVISE in or a single ADVISE out message. 2 ADVISE replace message (that is, an ADVISE in message and an ADVISE out message with the order being immaterial).
	ca	Completion address for this call.
Alpha(3)	len	Number of full words for each parameter set. Should be set to 1.
	eea	Virtual bit address to receive control if an error is detected during message processing (r≠0).

Figure 5-53. ADVISE (f=#FF00) Message Format (Sheet 2 of 3)

Word	Field	Description
Beta(1)	sts	Status field, which is set to 0 unless r is set to 1. In this case, the ADVISE function was incomplete and the code in this field indicates the cause: 1 ADVISE function is not complete due to more pages specified than a single ADVISE function can accommodate. The system ADVISEd 16 small pages or a single large page but did not satisfy the total request. ADVISE in Beta returned to the requestor contains the first small page address in the specified virtual range not processed. ADVISE out virtual range specification is unaltered. The status code 1 can be combined with any other status. For example, a status of 3 would indicate that more than 16 small pages or a single large page was specified in addition to a locked page within the specified virtual range being encountered while doing an ADVISE out function. 2 Partial ADVISE out function performed due to pages within the specified virtual range being locked. 4 Partial ADVISE in function performed due to insufficient memory resources to accommodate the request. 8 Page within specified ADVISE in range was found to be already in core.
	pio	Function requested: 0 ADVISE in function requested. 1 ADVISE out function requested.
	pct	Page count, in blocks. PAGER determines what size page the requested virtual range is mapped into. If the request is for pages mapped into a large page, the entire large page is ADVISEd. If the request is for more than 16 small pages, PAGER will only ADVISE 16 pages and informs the requestor of this fact.
	vba	Virtual bit address of the start of the virtual range specified in the ADVISE function. vba is updated to the last page address plus one processed by an ADVISE in function (unaltered by an ADVISE out function).

Figure 5-53. ADVISE (f=#FF00) Message Format (Sheet 3 of 3)

PROCESS SYSTEM PARAMETER (f=#FF01)

The PROCESS SYSTEM PARAMETER system message can change or retrieve the value of one or more of the following system parameters in the file descriptor block.

Parameter	Description
C_RFL	Current working set size limit.
C_MLP	Maximum large page limit.
C_RLP	Current large page limit.

If the current large page limit is set less than the number of large pages currently assigned to the task, large pages are purged until the limit is reached.

The message format is shown in figure 5-54. The Alpha and Beta portions must be contiguous. Multiple Betas can be specified.

	0				63
Alpha(1)	ctrl 8	r 8	to 16	from 16	FF01 16
Alpha(2)	num 8	00 8	ca 48		
Alpha(3)	0001 16	eea 48			
Beta(1)	fc 16	opt 8	sts 8	val 32	

Word	Field	Description
Alpha(1)	ctrl	A value specified in this field is not used. KERNEL sets this field to 0 on return to the caller.
	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 An error occurred [refer to the sts field in the Beta(1)]. 2 Illegal function code or option specified in Beta(1). 4 Beta buffer length error. Either the first word address of Beta plus its length is greater than the maximum virtual user address, or the Beta buffer is too small for the number of requests and length specified.
	to	A value specified in this field is not used. KERNEL sets this field to 0 on return to the caller.
	from	A value specified in this field is not used. KERNEL sets this field to 0 on return to the caller.
Alpha(2)	num	Number of parameter sets (Beta portions) for this call.

Figure 5-54. PROCESS SYSTEM PARAMETER (f=#FF01) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	ca	Address at which execution continues following successful completion of the call. If 0, execution continues at the address following the call. If nonzero, execution continues at the specified address.
Alpha(3)	eea	Error exit address; virtual bit address to receive control if an error occurs during message processing (r≠0). If this field is 0 when an error occurs, the task is aborted.
Beta(1)	fc	Function code indicating the DB field value to be set or returned: 1 Maximum large page limit. 2 Current large page limit. 3 Current working set size limit.
	opt	Message option: 0 Change the DB field value specified by fc to the value in the val field. 1 Return the DB value specified by fc in the val field.
	sts	Beta status code: 0 No errors. 1 Requested current large page limit greater than maximum large page limit. 2 Requested current working set limit greater than maximum working set size limit. 3 Requested current working set limit too small to accommodate job´s current large page limit. 4 Illegal operation requested.
	val	New system parameter value if opt=0; field in which value is returned if opt=1.

Figure 5-54. PROCESS SYSTEM PARAMETER (f=#FF01) Message Format (Sheet 2 of 2)

GIVE UP CPU ON OUTSTANDING RESIDENT I/O OR TIME (f=#FF02)

The GIVE UP call is issued by a user or the virtual system when the caller wants to suspend execution waiting on completion of a resident I/O call. Control is not transferred to the ca or eea of the I/O call. On completion of the I/O, control is returned to the ca address of the GIVE UP call or to the next location after the GIVE UP exchange if ca is 0. If a time interval is specified and the call times out before I/O has completed, control is returned to the eea in the GIVE UP call and the r field is set to 1. The GIVE UP call may also be used when the caller wants to GIVE UP the CPU waiting on an elapsed time. At the end of a GIVE UP, control will be returned to the ca address or the address following the call.

The message (Alpha/Beta) must not cross a page boundary. The system returns an error if this occurs.

The message format is shown in figure 5-55.

	0				63
Alpha(1)	cntrl 8	r 8	to 16	from 16	FF02 16
Alpha(2)	num 8	00 8	ca 48		
Alpha(3)	len 16	eea 48			

Beta(1)	tc 64	
Beta(2)	dt 64	
Beta(3)	00 16	lc 48

Word	Field	Description
Alpha(1)	cntrl	Set to 0 or #80. KERNEL sets this field to 0 on return to the caller.
	r	Response code returned by VSOS when message processing is complete: 0 No errors. 1 Call has timed out. 2 Parameter call error or message crossed a page boundary.
	to	Set to 0.
	from	Set to 0.

Figure 5-55. GIVE UP CPU ON OUTSTANDING RESIDENT I/O OR TIME (f=#FF02) Message Format (Sheet 1 of 2)

Word	Field	Description
Alpha(2)	num	Number of parameter sets for this call is 1.
	ca	Completion address for this call. This will normally be 0.
Alpha(3)	len	Number of words in Beta. This should be set to 3.
	eea	Virtual bit address to receive control if time out is detected during message processing. If the address of the outstanding resident call is not located, it is assumed to have completed properly.
Beta(1)	tc	Time GIVE UP was issued, returned by the system.
Beta(2)	dt	Delay time in microseconds. Minimum value is 100,000 microseconds (0.1 second). Maximum value is 59,999,999 microseconds (1 minute). For GIVE UP on I/O with dt=0, the caller is blocked until the I/O call completes. When dt≠0 and the I/O call does not complete in dt microseconds, control is returned to the eea with r=1. For GIVE UP on time with dt=0, control is immediately returned with no error.
Beta(3)	lc	Virtual bit address of the resident I/O call if GIVE UP is for I/O. If lc=0, then GIVE UP CPU for dt microseconds. For a GIVE UP on time, the user may be disconnected from the alternator. If both dt and lc are 0, the call is ignored without warning.

Figure 5-55. GIVE UP CPU ON OUTSTANDING RESIDENT I/O OR TIME (f=#FF02) Message Format (Sheet 2 of 2)

VIRTUAL SYSTEM DEBUG TOOL

6

The virtual system debug tool provides a breakpoint capability to track the execution of the virtual portion of VSOS. This facility is designed as a tool for the systems programmer who wishes to stop execution of the operating system at desired virtual system locations. This debugging aid provides the systems programmer with the capability to temporarily stop execution of the virtual system by setting execute breakpoints, and then to reset those breakpoints and continue execution.

VSDT is built as part of the virtual system of VSOS. While the operating system is running, a debug command can be entered into a reserved shared table (T_VSD) by using the maintenance control unit (MCU) write command. This table is periodically checked by the resident portion of the operating system for commands. When a command is entered, control is passed to VSDT for handling. If an attempt is made to set a breakpoint at a paged-out address, that page will be paged-in by VSDT. All current breakpoint addresses can be found in the T_VSD table.

RESIDENT SYSTEM

The resident system periodically checks the contents of the VSDT input buffer word of the T_VSD table for a nonzero value. Upon detection of a nonzero value, control is passed to VSDT which examines the input buffer word. If it contains COMMAND, the debug command buffer is checked for a debug command. Any existing debug command is then processed by VSDT.

If the input buffer word contains CONTINUE (normally entered after a breakpoint has been hit and appropriate chapters of memory have been displayed), VSDT resumes execution of the virtual system. The status word of the T_VSD table (containing STOP) will be cleared. If CONTINUE is entered without a breakpoint having been hit, the error message word in the T_VSD table will contain the message ILL COMM (illegal command) and control will return to the resident system.

VIRTUAL SYSTEM

VSDT handles the setting and resetting of breakpoints, as well as the resumption of virtual system execution. When control is passed from the resident system to VSDT, VSDT checks the input buffer word in the T_VSD table for either COMMAND, in which case it will process debug commands, or, CONTINUE, for resumption of virtual system execution after a breakpoint has been hit.

If a breakpoint command has been entered, VSDT will put the breakpoint address into the T_VSD table; save the instruction at the breakpoint address; set up the breakpoint jump instruction; and clear the word COMMAND from the input buffer word. If a command to reset the breakpoint has been entered, VSDT will restore the original instruction at the breakpoint address, as well as clear the breakpoint address in the T_VSD table.

USER AND SYSTEM INTERFACES

The MCU is used as the input device for the VSDT. Debug commands are entered in ASCII format into the T_VSD table by the MCU AS command. To stop the operating system after system autoload, the user can enter a breakpoint command into the T_VSD debug command buffer. When the execution of the virtual system is stopped, MCU memory commands are used for system examination.

This discussion assumes that the user is familiar with the MCU and its commands. Refer to the VSOS 2 Operator's Guide.

SHARED TABLE

An additional table (T_VSD) appears in the shared table list. The main function of this table is communication between the user and the resident system. The structure of the T_VSD table is given in table 6-1.

Table 6-1. Structure of the T_VSD Table (Sheet 1 of 2)

Name	Word	Description
W_VSDNAME	0	Contains the name T_VSD (left-justified with blank fill) identifying the table.
W_VSDINPUT	1	System command input buffer. Cleared to zeros after command is processed. COMMAND. Process VSDT debug command which has been entered in the T_VSD debug command input buffer. The word COMMAND is left-justified with blank fill. CONTINUE. Continue execution of the virtual system after a breakpoint has been hit. The word CONTINUE is left-justified with blank fill.
W_VSDBSTAT	2	Virtual system status word. Contains the word STOP*A or STOP*X (left-justified with blank fill) when breakpoint is hit. STOP*A indicates that an access breakpoint had occurred and STOP*X indicates that an execute breakpoint had occurred. The status word is cleared to zeros upon continuation of virtual system execution.
W_VSDBADD †	3	Contains the last breakpoint address (right-justified with zero fill) hit by virtual system. For an execute breakpoint, it will be the address of the instruction causing the breakpoint. For an access breakpoint, it will be the address of the last executed instruction which can be up to 35 instructions past the instruction causing the breakpoint.

† The upper 16 bits of W_VSDBADD contain a value equal to the number of times the breakpoint has been hit. When the value reaches the count specified by the n option, the system will halt and the appropriate message will be placed in W_VSDBSTAT.

Table 6-1. Structure of the T_VSD Table (Sheet 2 of 2)

Name	Word	Description
W_VSDERRMES	4	Error messages returned by VSDT, left-justified with blank fill. Cleared to zeros when next command is entered.
W_VSDC1	5	Start of debug command buffer. Commands are entered in ASCII mode (left-justified with blank fill). Cleared to zeros after a command is processed. If an error has occurred, the buffer is not cleared to zeros. The command can then be reentered in its entirety, or edited in place. In either event, if any extraneous characters remain after the corrected command has been placed in the command buffer, they must be blanked out by use of the MCU AS command.
W_VSDC2	6	Continuation of debug command buffer.
W_VSDC3	7	Continuation of debug command buffer.
W_VSDC4	8	Continuation of debug command buffer.
W_VSDC5	9	Continuation of debug command buffer.
W_VSDC6	A	Continuation of debug command buffer.
W_VSDC7	B	End of debug command buffer.
W_VSDAB1 †	C	Bits 16-63 contain the address of breakpoint which has been set in the virtual system.
W_VSDAB2	D	Bits 16-63 contain the address of breakpoint which has been set in the virtual system.
W_VSDAB3	E	Bits 16-63 contain the address of breakpoint which has been set in the virtual system.
W_VSDAB4	F	Bits 16-63 contain the address of breakpoint which has been set in the virtual system.
W_VSDAB5	10	Bits 16-63 contain the address of breakpoint which has been set in the virtual system.
W_VSDAB6	11	Bits 16-63 contain the address of breakpoint which has been set in the virtual system.
W_VSDABA	12	Bits 16-63 contain the address of the read/write access breakpoint.
W_VSDREG	13	Bits 16-63 contain the address of the virtual system register file.

† The upper 16 bits of W_VSDAB1 through W_VSDAB6 and W_VSDABA contain the value specified by the n option, which is the number of times a breakpoint is to be executed before the system is stopped.

COMMANDS

Since all debug breakpoint commands are entered by MCU memory alteration commands, systems programmers are responsible for entering debug commands into the appropriate locations in the T_VSD table.

The commands supported by VSDT are shown in table 6-2.

Table 6-2. VSDT Command Summary

Command	Description
BKPT †	Set execute breakpoint.
COMMAND	Process virtual system debug commands.
CONTINUE	Continue execution from last user breakpoint.
RBKP	Remove all or selected breakpoints.
PAGEIN	Bring page containing given virtual address into memory.

† Underscored letters indicate the short form of the command.

COMMAND FORMAT

The following conventions must be observed when entering the breakpoint commands:

- A blank must be used to separate the command and its first parameter.
- Brackets are used to indicate an optional parameter. Defaults are assumed for omitted parameters and are defined in the command descriptions. Either no parameters or any single parameter can be selected.
- Underscored letters of each command or keyword parameter indicate the minimum character string used to specify the command or parameter. Any number of characters, from the minimum string to the entire word, can be used. For example, BK, BKP, and BKPT will all result in execution of the BKPT command.
- All address, count, and number parameters are specified in hexadecimal notation.
- Commands are entered into T_VSD starting with the leftmost byte in the command buffer.
- Blanks are not used between subsequent parameters (with the exception of the blank used to separate the command and its first parameter). Commas are used to separate all subsequent parameters.

The virtual address parameter for debug commands is assumed to have an offset bit address C00000000000 in hexadecimal notation; that is, C00000000000 will be added to the value entered by the systems programmer to derive the virtual system address.

DEBUG COMMANDS

Debug commands are entered into W_VSDC1 through W_VSDC7.

Up to six execute breakpoints are allowed in the virtual system at any time. These are software breakpoints and cause the virtual system to stop execution prior to executing the instruction at the specified virtual address.

The user can specify the number of times a breakpoint is reached before execution of the virtual system is stopped.

The format of the command to be entered at the MCU is:

```
AS,(address of T_VSD + 5),"command (excluding COMM/CONT) parameters"

AS,(address of T_VSD + 1), "command (COMM/CONT)"
```

The command buffer and input word are cleared upon acceptance of the command.

VSDT commands for setting and resetting breakpoints are shown in table 6-3. The breakpoints can be removed individually or all together.

Table 6-3. VSDT Commands for Setting and Resetting Breakpoints

Command	Parameters	Description
BKPT	virtual address (,n)	BKPT sets an execute breakpoint which causes the virtual system to stop execution at the specified address. Up to six breakpoints can be set at one time. Default is n=1.
RBKP	ALL	RBKP allows the user to remove all breakpoints that have been virtual address set (ALL) or to remove a single breakpoint set by virtual address. Default is ALL.

CONTROL COMMANDS

Control commands are entered into W_VSDINPUT which allow the user to input debug commands into VSDT, and continue execution after a breakpoint has been reached.

Table 6-4 shows the command for accessing paged-out addresses.

Debug commands are processed by VSDT by using the MCU command AS to write the word COMMAND into W_VSDINPUT following entry of the debug command into the command buffer. The debug commands are entered initially into the T_VSD table or after a breakpoint has been hit.

The CONTINUE command allows execution to continue after a breakpoint has been reached. Execution can be continued by using the MCU A command to write the word CONTINUE into W_VSDINPUT in the T_VSD table.

The user can also obtain breakpoint status information. No specific command is needed. Since breakpoint addresses are stored in the T_VSD table, the MCU D addr command can be used to check for breakpoints set.

Table 6-4. VSDT Command for Accessing Paged-Out Addresses

Command	Parameter	Description
PAGEIN	virtual address	PAGEIN causes the virtual system to access the specified virtual address causing, in some cases, a page fault. This command is used to bring a page containing the specified virtual address into memory.

ERROR MESSAGES

Error messages which are placed in W_VSDERRMES of the T_VSD table are shown in table 6-5. The user can input the proper command in its entirety after diagnosing the error message or edit it in place. W_VSDERRMES is cleared after the acceptance of subsequent commands.

Table 6-5. VSDT Error Messages

Error Message	Description
ILL COMM	Illegal command.
ILL PARM	Illegal parameter.
ILL NUM	Illegal number.
TAB FULL	Breakpoint table is full.
ILL ADDR	Illegal address specified in RBKP command.
NO ADDR	No address has been specified in a breakpoint instruction.

ANALYZER 7

ANALYZER is a tool for extracting information from a system dump. It generates an output file containing the following information, in the order listed:

- Time, date, version information, and system options.
- A listing of TABST, VTABST, the system tables map, and the system dayfile buffer.
- A hexadecimal dump with ASCII interpretation of resident tables and boat area.
- Hexadecimal dumps with ASCII interpretation of performance data area, test mode buffers, and SPY buffers, if defined, and the last location of the resident operating system.
- A hexadecimal dump with ASCII interpretation of shared tables.
- A listing of PAGER internal information, page table dumps, and the lock table. Four listings of the page table are provided, sorted by:

  Page table bit address
  Physical page number
  Virtual address and key
  Key and virtual address

- A dump of the history table with entries listed in chronological order. Time is reported as an installation option.
- A dump of the resident operating system dynamic stack.
- A dump of the virtual operating system minus page, register file, and dynamic stack.
- A dump of the user´s minus page, second minus page, third minus page, and dynamic stack, for any active user.
- A hexadecimal dump of any core areas specified with the CORE parameter.
- A hexadecimal dump with ASCII interpretation of virtual space for a given user as specified via the VRANGE parameter.
- A hexadecimal dump with ASCII interpretation of the FILEI.
- A hexadecimal dump with ASCII interpretation of selected virtual tables as specified via the VTABLE parameter.

Input for the ANALYZER is constructed by the following steps:

1. A dump file the size of memory is created.
2. With the system stopped and the resident operating registers saved, the WCMF MCU command is used to dump memory to the existing file.

ANALYZER can be used to examine the file after the system has been recovered or autoloaded.

The format of the ANALYZER execute line is shown in figure 7-1. All parameters are optional and order independent.

```
ANALYZER     (INPUT=ifn,LIST=ofn,
              CORE=l1,h1,...,l4,h4,
              VRANGE=db,vbitadd1,n1,...,vbitadd32,n32,
              VTABLE=vtable1,...,vtable32,
              LO= ⎧ * ⎫ )
                  ⎨ S ⎬
                  ⎩ V ⎭
```

INPUT=ifn	ifn is the name of the input file containing a dump of memory. The default is INPUT=DUMPFILE.
LIST=ofn	ofn is the name of the file to contain ANALYZER output. The default is LIST=OUTPUT. Any existing file with the same name as the output file is destroyed, and a new file with that name is created.
CORE=l1,h1,...,l4,h4	Additional areas of memory can be dumped. As many as four li,hi pairs can be specified. Words li through hi are dumped as follows: li Low hexadecimal bit address of physical memory to be dumped. hi High hexadecimal bit address of physical memory to be dumped.
VRANGE=db,vbitadd1,n1, ..., vbitadd32,n32	Additional areas of virtual space to be dumped. As many as 32 ranges of virtual space may be specified. db Specifies the DB number associated with the virtual space to be dumped. vbitadd1 Specifies the virtual bit address of the first word of a range to be dumped. n1 Specifies the number of words to be dumped.

Figure 7-1. ANALYZER Execute Line Format (Sheet 1 of 2)

VTABLE=vtable1,..., vtable32	Areas of the virtual operating system to be dumped. As many as 32 table names may be specified. vtablei Specifies the virtual system table by name to be dumped.
LO=$\left\{\begin{matrix}*\\S\\V\end{matrix}\right\}$	It may be desirable to suppress certain portions of ANALYZER output. The default is LO=*. This condition generates all of the standard output. Specifying LO=S will suppress dumping of the user's registers and dynamic stack information. Specifying LO=V will cause only the information requested by the VRANGE and/or the VTABLE parameters to be dumped. The options *, S, and V are mutually exclusive.

Figure 7-1. ANALYZER Execute Line Format (Sheet 2 of 2)

ACCOUNTING

8

The accounting system is that part of the virtual system that gathers system resource usage statistics and records them at two locations: in a cumulative accounting buffer and in an active accounting file. The statistics recorded in the cumulative accounting buffer are a subset of those recorded in the active accounting file. Statistics recorded in the cumulative accounting buffer can be retrieved by a user or utility program only when the job for which the statistics are being gathered is running; statistics recorded in the active accounting file can be retrieved by an installation-defined user at any time after the file is no longer active.

VSOS provides two units with which to measure resource usage, system time units and system billing units. As described later in this chapter, the system billing units (SBU) calculation uses a variable rate accounting factor and a service level index while the system time units (STU) calculation does not.

The information recorded in the accounting file is designed to be accessed by an installation-defined accounting program. No VSOS utilities currently exist to use these statistics.

CALCULATION OF STUs

A system routine called TCHARGE calculates the number of STUs consumed by a task. To calculate the STUs consumed, it first multiplies each system resource usage quantity by the weighting factor for that resource and then adds the products.

The system resources and their units of measure are listed in table 8-4. The weighting factors are specified by installation parameters. A weighting factor of 0 eliminates the corresponding system resource from the STU calculation.

A system second is 1 million STUs.

STATISTICS ACCUMULATION

Having gained access to the operating system, a user can then execute tasks and jobs. Statistics are gathered on an event basis and are accumulated over each task or batch job in both the cumulative accounting buffer and the active accounting file. The buffer statistics are also provided as information to the user via the USER/ACCOUNTING COMMUNICATION (f=#000E) and MISCELLANEOUS (f=#0024) messages.

CUMULATIVE ACCOUNTING BUFFER

The cumulative accounting buffer is provided so that resource usage statistics for a task or job are available during the time that the task or job is actually running. The buffer contains one 16-word entry for each active task (each descriptor block). The fields in each entry are updated as the tasks use up various system resources. When a job has finished, the entries for that job no longer exist in the cumulative accounting buffer; the entries exist only for the duration of the job.

A user is given access to the statistics in the cumulative accounting buffer for his or her running tasks alone via the USER/ACCOUNTING COMMUNICATION message. The virtual system returns the current accumulation of all statistics in the buffer to a user that issues this message with the c field set to 3. Based on these statistics, it might be possible to modify user programs to reduce system resource usage and perhaps reduce charges. This reduction would be dependent on the installation-defined charge algorithm.

In addition to individual task statistics, total job statistics for all controllee tasks of a batch job are available from the cumulative accounting buffer. These cumulative job totals are stored in the batch processor´s entry in the buffer. By issuing the USER/ACCOUNTING COMMUNICATION message with c=3, the batch processor can access the total raw statistics for the duration of a batch job. These unfactored values can be used as input to an installation-provided routine that computes the charges for the job. Any of these charges, raw statistics, or factored charges might then be printed on the job´s dayfile.

ACCOUNTING FILE

The accounting file is a history file of the operating system resource usage, including tape and disk file usage, task and job execution times, terminal usage, and autoload and recovery events. Records are written into the file as resources are used. Accounting record types are listed in table 8-1.

Table 8-1. Accounting Record Type and Subtype Codes

Subtype	Record Type Code									
	Task 1	Terminal 2	Disk File 3	Magnetic Tape 4	System 5	Job 6	Network Usage 7	Channel Usage Statistics 9	Periodic System A	Periodic Job B
0	Begin task	Login	Open	Assignment	Autoload	Job start	Input queue file	-	Periodic system	Periodic job
1	End task	Logout	Close	Release	Recovery	Job end	Output queue file	Configuration	Periodic system	-
2	Field overflow	-	Create	-	New date entered	Job change	Output data file	Global channel usage data	-	-
3	Task information	notmsg/ notwds field overflow	Destroy	-	New time entered	-	Input data file	-	-	-
4	Project information	-	Extend	-	-	-	Teletype entry	-	-	-
5	-	-	Reduce	-	-	-	Purge file	-	-	-
6	-	-	Change user number	-	-	-	Interactive session	-	-	-
7	-	-	Change file name	-	-	-	-	-	-	-
8	-	-	Change account number	-	-	-	-	-	-	-

More than one accounting file can exist; all such files are unattached permanent files belonging to the system user number; however, only one file, named Q5AF, can be active at any one time. If during system initialization no active accounting file Q5AF exists, one is created at that time. A second file, Q5AF2, is also created. When the current active accounting file becomes full, it is renamed, file Q5AF2 becomes the active accounting file Q5AF, and a new Q5AF2 is created.

The naming convention used when renaming the *AF file is: Ayydddnn (yyddd is the Julian date and nn is a number ranging from 00 to 99). File names will be used in sequence if they do not already exist. The next file name in sequence after Ayyddd99 is Ayyddd00.

The processing sequence of accounting files is determined by information recorded in the accounting file headers. Ayyddd00 is normally the start of the sequence for a day and should be verified by checking that the accounting file header information for the previous file´s name indicates the previous day´s date.

After 90 files are recorded for a given day, the system sends the operator a warning message indicating the accounting system is nearing the maximum number of accounting files. When 100 files exist for a given day, the system sends another warning message to the operator, indicating that no new accounting files will be written. Recording of accounting information then stops. The warning message that the information is being lost continues to be issued periodically.

The size of all accounting files is determined by the system parameter AFSIZE.

The accounting file can be used as follows:

- For each task, the accounting system writes a begin task accounting record to the accounting file. A unique task number is assigned to each task and is placed in all accounting records associated with the task. Each accounting period, the system writes a new task field overflow record for the task or updates an existing one. When the task completes, the system writes an end task accounting record.

- To accumulate raw statistics for the duration of a task, the installation´s program must locate on the accounting file all task field overflow records for a unique task number or job name, extract the desired statistics, and sum them for each overflow record for the task.

- At the start of each batch job, the batch processor issues an option of the USER/ACCOUNTING COMMUNICATION message (f=#000E), and a job start accounting record is written to the accounting file. The header of this record contains the user number and job descriptor number (JDN) under which the batch processor is running. Any controllee of this batch processor runs under this user number and JDN. The task record headers for these controllee tasks contain these identifiers also. At the end of a batch job, the batch processor issues another option of the USER/ACCOUNTING COMMUNICATION message, and a job end accounting record is written to the accounting file.

- To accumulate raw statistics for the duration of a batch job, the installation´s accounting program must locate the job start and job end records on the accounting file, and record the user number and JDN. A search can also be done using the job/session name. Any task overflow records (between the job start and job end records) containing this user number and jdn are for tasks of this batch job. An additional task overflow record is generated for the batch processor clean-up (for example, close dayfile, change file ownership). This record may not be charged to the user job depending on the site´s accounting program. Statistics can be extracted from each of these overflow records and summed to get the cumulative batch job statistics. These totals are equivalent to those extracted from the batch processor´s entry in the cumulative accounting buffer.

- At the option of the user site, the task information record allows information to be entered into the accounting file (such as billing type, or queueing time). This information is entered via message 7 for USER/ACCOUNTING COMMUNICATION (f=#000E).

Active Accounting File Blocks

Accounting records are accumulated in a 512-word memory buffer (this buffer is not the cumulative accounting buffer). The buffer is written to the accounting file when the buffer is full, or when an entry is made in the buffer after more than 5 minutes have elapsed since the previous buffer was written to the accounting file. The format of the buffer and, therefore, of the accounting file record blocks, is shown in table 8-2.

Table 8-2. Active Accounting File Block Format

Word	Contents
0	Master clock time and date information at the time that the first entry is written to the buffer.
1	Microsecond central processor clock reading at the time that the first entry is written to the buffer.
2	Reserved for system use.
3	Master clock time and date information at the time that the last entry is written to the buffer.
4	Microsecond central processor clock reading at the time that the last entry is written to the buffer.
5	Cumulative system CPU overhead in microseconds. The sum of KERNEL, PAGER, and virtual system CPU times.
6	Cumulative USER CPU time in microseconds.
7	Cumulative WAIT time in microseconds. The CPU is available for user execution but all user tables are waiting for I/O completion.
8	Cumulative IDLE time in microseconds. The CPU is available for user execution but no user tasks are waiting for the CPU.
9-#1FE	Filled sequentially with accounting records. Unused words contain binary 0.
#1FF	This word is set to contain all hexadecimal 2´s when the buffer is full.

The master clock entries in words 0 and 3 of an active accounting file block are eight decimal numbers indicating the year, month, day, hour, minute, second, millisecond, and an installation-defined machine designator. The entries are represented as 16 digits in hexadecimal form within one word, as shown in figure 8-1. For example, the date November 6, 1985, the time 9:10:20.623, and the machine designator 1 would appear as shown in figure 8-2.

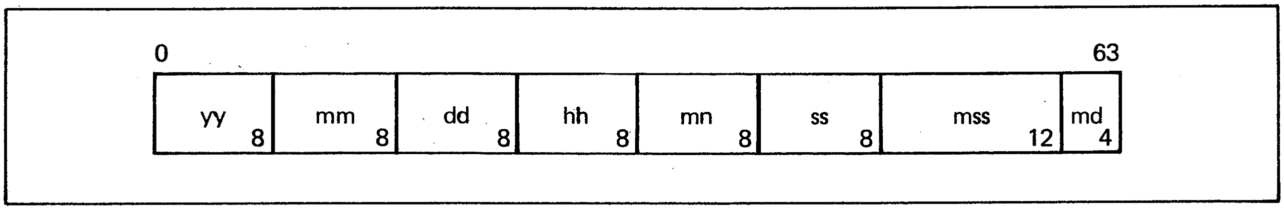


Figure 8-1. Master Clock Format

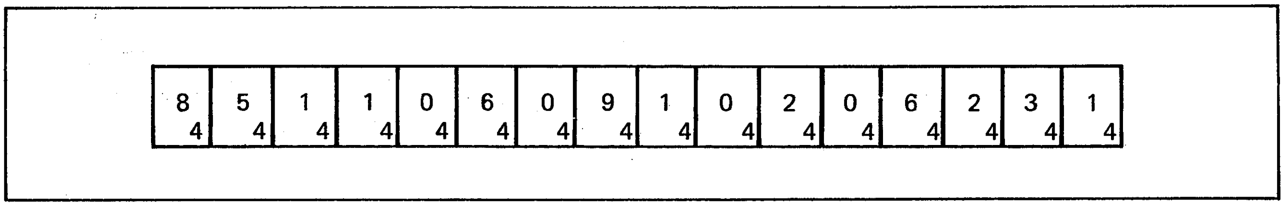
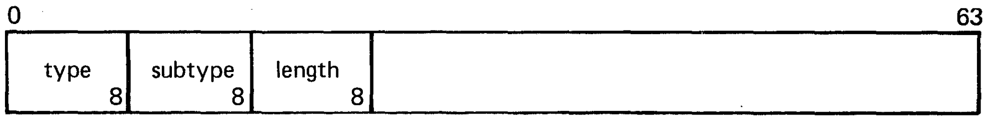


Figure 8-2. Master Clock Example

Table 8-3 lists the information kept in the first block of *AF, including the name to which the name *AF is to be changed when the file becomes full, and the address of the block in which space is currently available. Accounting records are written sequentially into *AF beginning with the second block. The record type and length are indicated by the first word of each record, as shown in figure 8-3.

Table 8-3. Active Accounting File Format (First Block)

Word	Contents
0	Ayydddnn, the name to be given to this file when it is deactivated. yyddd is the year and day number computed when the file is activated; nn is a sequence number, modulo 100.
1	Name of the last accounting file, in the form Ayydddnn. At autoload, this field contains binary 0 if no *AF file exists at that time.
2	Name of the next accounting file, Ayydddnn.
3	Relative block address of the most recently created block in the file, where the header block is block 0.
4	Reserved for system use.
5	Reserved for system use.
6	Microsecond central processor clock value when this file is deactivated.
7	ASCII date when this file is deactivated, in the form mm/dd/yy (month/day/year).
8	ASCII time when this file is deactivated, in the form hh:mm:ss (hours:minutes:seconds).
9-1FF	Unused.



Field	Description
type	Accounting record type: 1 Task record. 2 Terminal record. 3 Disk file management record. 4 Tape record. 5 System record. 6 Job record. 7 Network usage record. 9 Channel usage statistics record. #A Periodic system record. #B Periodic job record.
subtype	Subtype code for the accounting record type indicated in the type field (refer to table 8-1).
length	Total word length of this record.

Figure 8-3. Accounting Record Format (First Word)

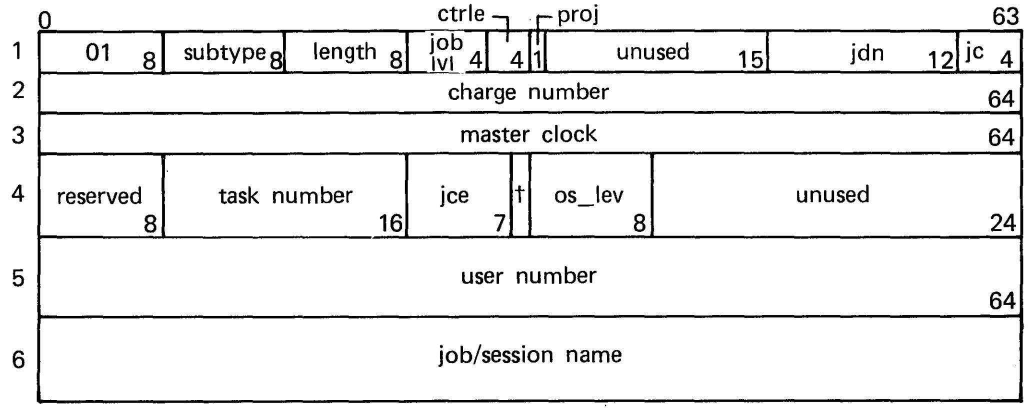
Task Records

Task accounting (type 1) records are written in the accounting file whenever a task is started or ended, or when task accounting information is available and no task field overflow record exists in the current accounting file block. A task record consists of a six-word header plus additional words, depending on the record subtype. The header format is shown in figure 8-4.

When the subtype code is 0 (begin task), the additional record words are as shown in figure 8-5. When the subtype code is 1 (end task), only the record header is used. All task accounting information is accumulated in task field overflow records before an end task record is written.

A task field overflow (subtype 2) record contains raw accounting statistics accumulated for the duration of a task. When the operating system calls the accounting system with task accounting information, the accounting system searches for a task field overflow record within the current 512 word buffer pertaining to the specified task. If such a record is found, its fields are updated with the new information. If no task field overflow record exists for the specified task, such a record is created and filled with the current information. The record words are shown in figure 8-6.

Each task field overflow record field (except for the ferr and wssiz fields) is cumulative over the accounting periods for the task until potential overflow is detected. For cumulative values for an entire task, the appropriate field should be summed over all of the task's task field overflow records in the accounting file. The values for the ferr and wssiz fields for the entire task are in the last record for the task.



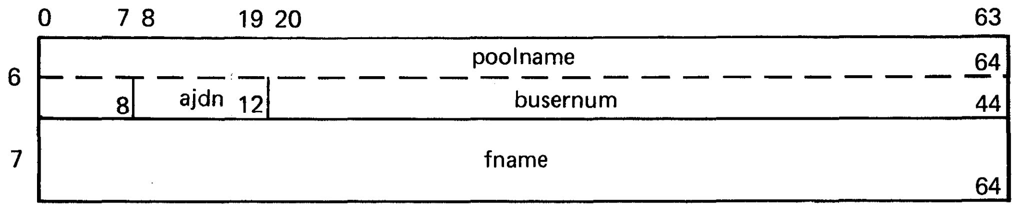
† Reserved.

Word	Field	Description
1	subtype	Subtype code: 0 Begin task. 1 End task. 2 Field overflow. 3 Task information. 4 Project information.
	length	Total word length of this record.
	job lvl	Job level (1 through 15).
	ctrle	Controllee level (1 through 9).
	proj	Project accounting flag.
	jdn	Job descriptor number.
	jc	Job class: 1 Standby. 2 Batch. 3 Interactive. 4 High priority. 5 System.
2	charge number	Account identifier from login line or job statement; identifies account to which the task is being charged.
3	master clock	Value of the master clock when this record was created.

Figure 8-4. Task Record Header Format (Sheet 1 of 2)

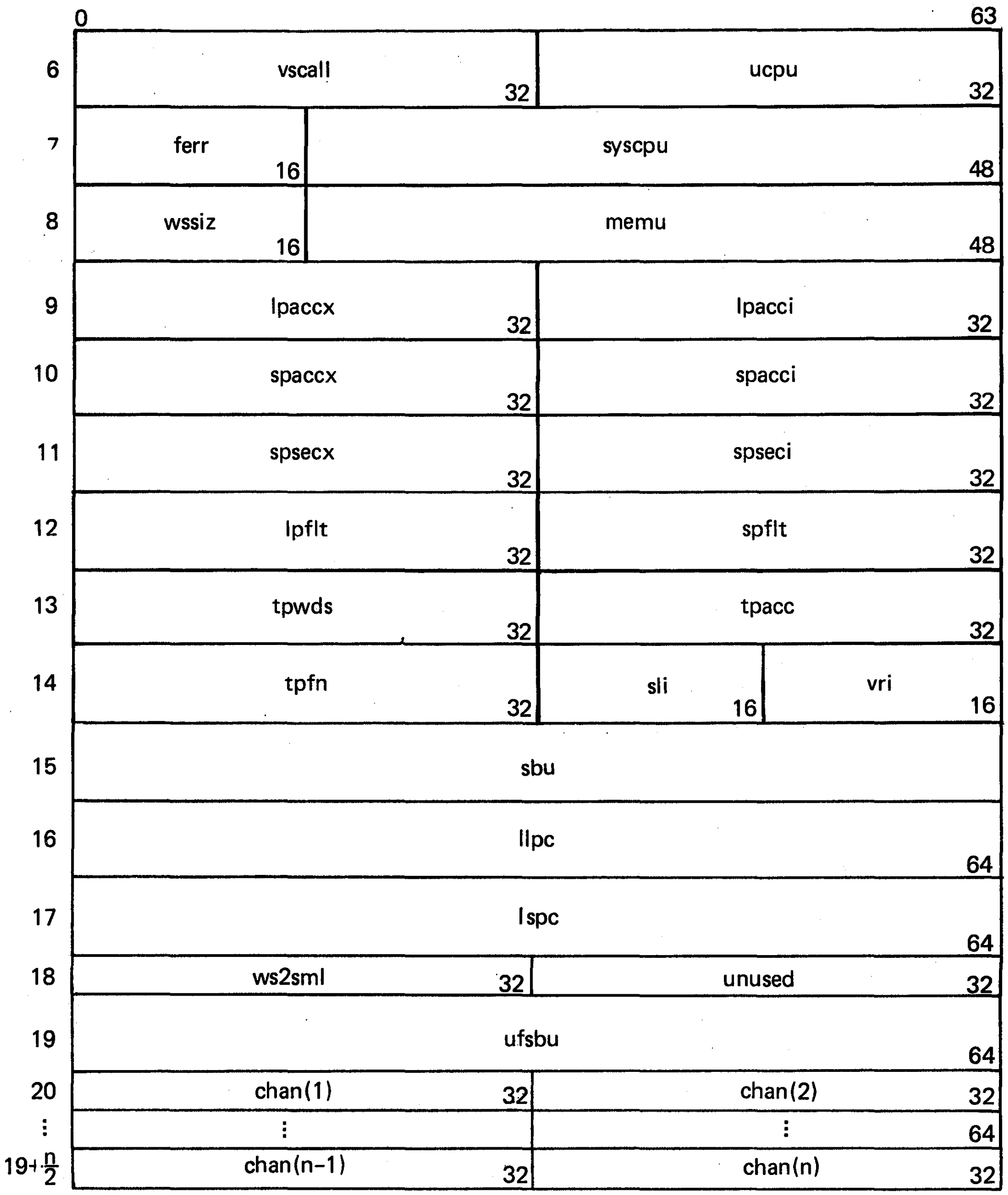
Word	Field	Description
4	task number	Number assigned when the begin task record is written. All subsequent accounting records pertaining to this task contain this number. (The accounting system assigns task numbers sequentially.)
	jce	Job category entry number (0 through 65).
	os_lev	Operating system version level. 0 through 7, where 0 2.0 or earlier. 1 2.1, 2.1.5, or 2.1.6. 2 2.2 or later. 3-7 Undefined.
5	user number	Binary user number.
6	job/ session name	Job or interactive session name associated with this task.

Figure 8-4. Task Record Header Format (Sheet 2 of 2)



Word	Field	Description
6	poolname,	Name of the pool in ASCII, if this file is a pool file member.
	ajdn	Job description number of user which has private file attached.
	busernum	Binary user number of the task file owner, if this is a private or public file.
7	fname	Name of the task file in ASCII, left-justified with blank fill.

Figure 8-5. Task Record Format, Subtype 0



Word	Field	Description
6	vscall	Number of virtual system user calls made.
	ucpu	User execution CPU time, in microseconds.
7	ferr	Error number of the fatal error condition, transferred from word 139 of the user's minus page. This field is 0 if there is no error.

Figure 8-6. Task Record Format, Subtype 2 (Sheet 1 of 2)

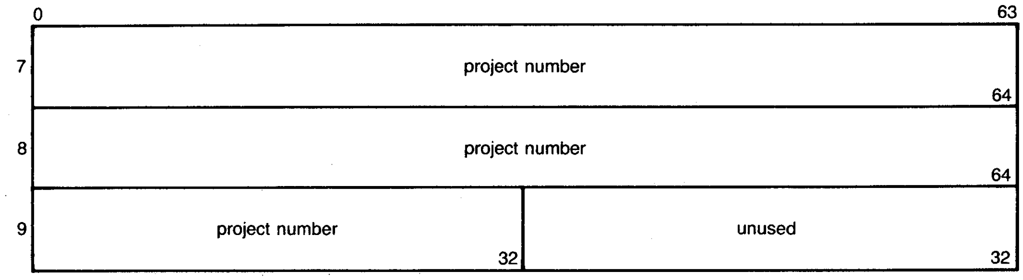
Word	Field	Description
7	syscpu	Virtual CPU time, in microseconds, for user program execution.
8	wssiz	Average working set size, in small pages.
	memu	Memory usage [at the end of each accounting period, (current working set size)]*[user CPU time for current accounting period is computed.]
9	lpaccx	Number of disk accesses (I/O requests issued) for large page explicit reads and writes.
	lpacci	Number of disk accesses (output requests issued) for large page implicit writes.
10	spaccx	Number of disk accesses (I/O requests issued) for small page explicit reads and writes.
	spacci	Number of disk accesses (output requests issued) for small page implicit writes.
11	spsecx	Number of disk sectors transferred for explicit reads and writes.
	spseci	Number of disk sectors transferred for implicit writes.
12	lpflt	Number of disk accesses (input requests issued) that resulted from large page faults (large page implicit reads).
	spflt	Number of disk accesses (input requests issued) that resulted from small page faults (small page implicit reads).
13	tpwds	Number of 16-bit bytes transferred to or from tape files.
	tpacc	Number of tape accesses (I/O requests issued) for reads and writes.
14	tpfn	Number of nonread and nonwrite tape functions, such as read hardware status.
	sli	Service level multiplying index used for variable rate accounting.
	vri	Variable rate index used for variable rate accounting.
15	sbu	Floating point summation of task´s system billing units (summation of MVAL´s returned from MCHARGE), or floating point summation of task´s system time units (summation of TVALs returned from TCHARGE).
16	llpc	Number of large pages lost to other tasks.
17	lspc	Number of small pages lost to other tasks.
18	ws2sml	CPU time, in microseconds, that the task´s working set size appeared to be too small.
19	ufsbu	Number of nonfactored standard billing units or system time units.
20	chan(i)	Task channel usage for channel i.

Figure 8-6. Task Record Format, Subtype 2 (Sheet 2 of 2)

The number of disk sectors transferred for large page implicit reads can be readily computed from the value of the lpflt field.

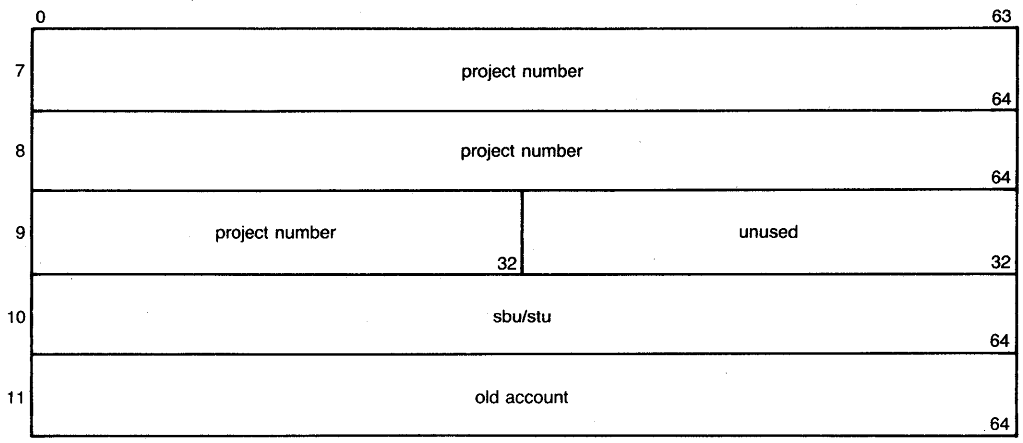
A task information field (subtype 3) allows a task, privileged or nonprivileged, to send records to the account file that contains billing information.

A project information record (subtype 4) allows a project number to be written to the account file. There are three formats for the project information record. The first format is for an interactive job (figure 8-7). The other two formats are for batch jobs. The first of the two batch formats occurs when the CHARGE statement is the first CHARGE executed in the job stream. The SBU/STU amount accumulated since the beginning of the job is put in word 10. If the CHARGE statement is the first executable statement in the job stream, the SBU/STU amount will be zero (figure 8-8). The second of the two batch formats occurs when the CHARGE statement is the second through the last CHARGE statement executed in the job stream (figure 8-9).



Word	Field	Description
7-9	project number	1-to 20-character project number, in ASCII, left-justified with blank fill.

Figure 8-7. Task Record Format, Subtype 4 Interactive Job

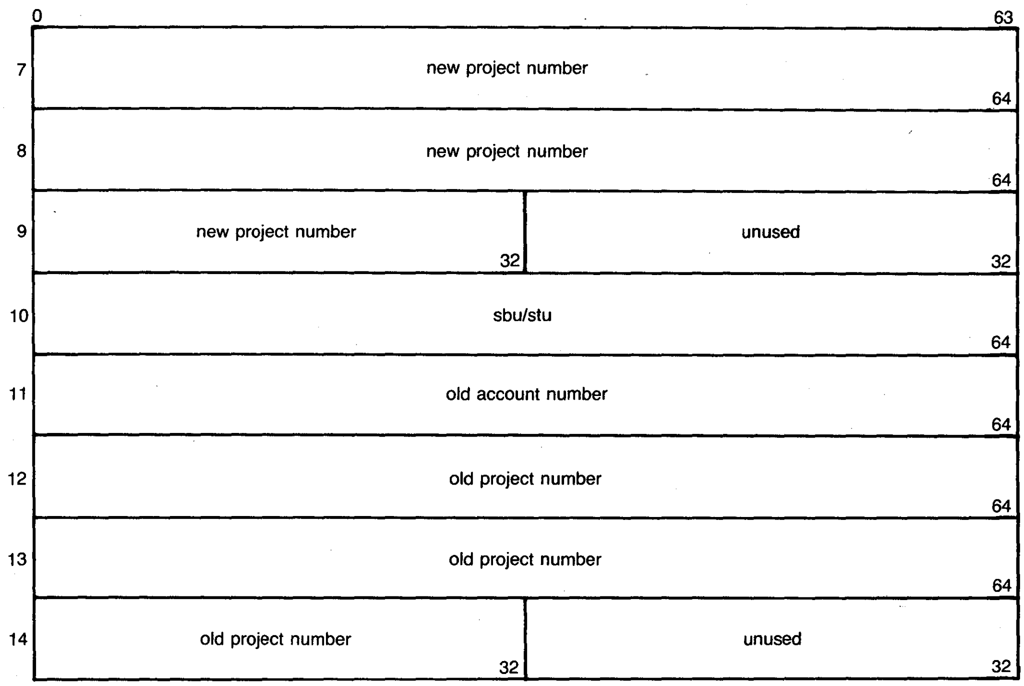


Word	Field	Description
7-9	project number	1-to 20-character project number, in ASCII, left-justified with blank fill.
10	sbu/stu	SBU/STU amount accumulated since the beginning of the job. This field is set to zero if the CHARGE statement is the first executable statement.

Figure 8-8. Task Record Format, Subtype 4 Batch Job
With First CHARGE Statement (Sheet 1 of 2)

Word	Field	Description
11	old account	1-to 8-character old account identifier, left-justified with blank fill.

Figure 8-8. Task Record Format, Subtype 4 Batch Job With First CHARGE Statement (Sheet 2 of 2)



Word	Field	Description
7-9	new project number	1-to 20-character new project number, in ASCII, left-justified with blank fill.
10	sbu/stu	SBU/STU amount generated by the old project number/old account combination (real).
11	old account number	1-to 8-character old account identifier, left-justified with blank fill.
12-14	old project number	1-to 20-character old project number in ASCII, left-justified with blank fill.

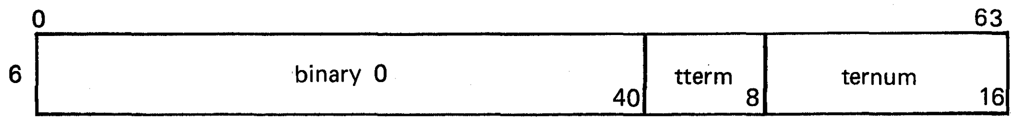
Figure 8-9. Task Record Format, Subtype 4 Batch Job With Second Through Last CHARGE Statement

Terminal Records

Terminal accounting (type 2) records are entered into the accounting file whenever a user logs on or off a terminal, or when the notmsg or notwds fields in the user table are about to overflow. A terminal record consists of a six-word header plus one additional word. The format of the header is the same as for the task record header, except that the type field is 2, the subtype field can be 0 (login), 1 (logout), or 3 (notmsg or notwds field overflow), and the controllee level, jdn, job category, job descriptor number, and task number fields are unused.

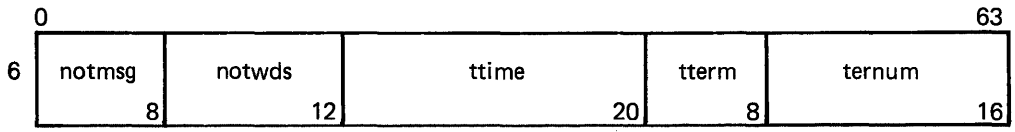
When the subtype code in the record header is 0 (login), the sixth word in the record is as shown in figure 8-10.

When the subtype codes are 1 (logout) and 3 (notmsg/notwds field overflow), the sixth word in the record is as shown in figure 8-11.



Word	Field	Description
6	tterm	Reserved for operating system use.
	ternum	Reserved for operating system use.

Figure 8-10. Terminal Record Format, Subtype 0



Word	Field	Description
6	notmsg	Number of messages to or from the terminal since login or since a previous overflow record.
	notwds	Number of words transferred to or from the terminal since login or since a previous overflow record.
	ttime	Time the terminal has been in use, in seconds, since login or since a previous overflow record.
	tterm	Reserved for operating system use.
	ternum	Reserved for operating system use.

Figure 8-11. Terminal Record Format, Subtypes 1 and 3

Disk File Management Records

Disk file accounting (type 3) records are issued whenever there is nondata transfer activity for a file. Disk file records consist of a six-word header plus additional words, depending on the record subtype. The format of the header is the same as for the task record header (figure 8-4), except that the type field is 3, and the subtype field can be 0 (open), 1 (close), 2 (create), 3 (destroy), 4 (extend), 5 (reduce), 6 (change user number), 7 (change file name), or 8 (change account number).

The format of the additional words (words 6 and 7) when the subtype codes are 0 (open), 1 (close), and 2 (create) is the same as the format of a task record of subtype 0 (figure 8-5), except that the file named can be a data file as well as a code (executable) file.

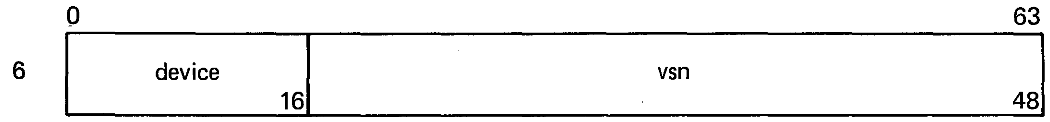
Three additional words are required (words 6, 7, and 8) when the subtype codes are 3 (destroy), 4 (extend), and 5 (reduce). The first two words are the same as words 6 and 7 for subtypes 0, 1, and 2. The third word consists of two 32-bit fields. The leftmost field gives the number of 512-word blocks in the file at the time that the destroy, extend, or reduce record is written. The rightmost field gives the number of seconds that this file existed under the user number, file name, account number, and size specified.

The format of the additional words when the subtype codes are 6 (change user number), 7 (change file name), and 8 (change file account number), is the same as words 6, 7, and 8 for subtypes 3, 4, and 5. Word 9 contains the new binary user number for subtype 6, the new file name for subtype 7, and the new account number for subtype 8.

Tape Records

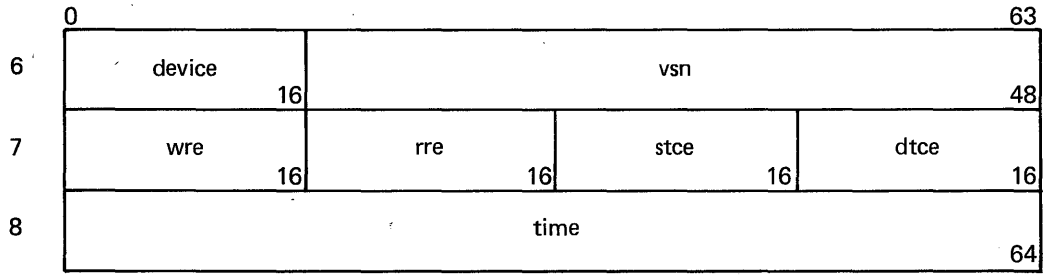
Tape accounting (type 4) records are entered into the accounting file whenever a tape is assigned to, or released from, a task. Tape records consist of a six-word header, plus additional words depending on the record subtype. The format of the header is the same as for the task record header (figure 8-4), except that the type field is 4, and the subtype field can be 0 (assignment), or 1 (release).

When the subtype code is 0 (assignment records), the record word is shown in figure 8-12. When the subtype code is 1 (release records), the record words are shown in figure 8-13.



Word	Field	Description
6	device	Device number of the tape unit.
	vsn	Volume serial number.

Figure 8-12. Tape Record Format, Subtype 0



Word	Field	Description
6	device	Device number of the tape unit.
	vsn	Volume serial number.
7	wre	Total accumulation of write recoverable errors.
	rre	Total accumulation of read recoverable errors.
	stce	Total accumulation of single-track, hardware-corrected errors.
	dtce	Total accumulation of double-track, hardware-corrected errors.
8	time	Tape hook-up time, in seconds.

Figure 8-13. Tape Record Format, Subtype 1

System Records

System accounting (type 5) records are entered into the accounting file when the system is autoloaded or recovered, or when new date or time information is entered. A system record consists of a six-word header, plus additional words depending on the record subtype. The header consists of:

- The first is as shown in figure 8-3, where the rightmost 40 bits are unused, and the subtype codes are 0 (autoload), 1 (recovery), 2 (new date entered), and 3 (new time entered).
- The second word is unused.
- The third word is the value of the master clock at the time this record was created.
- The fourth, fifth, and sixth words are unused.

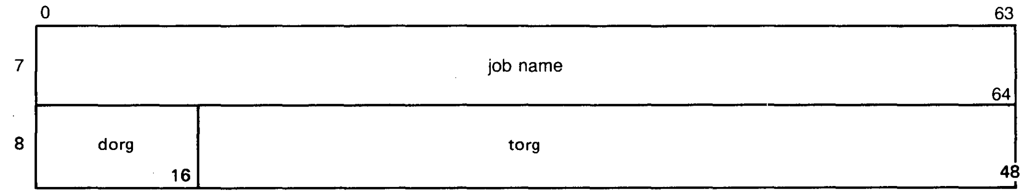
When the subtype code is 0 (autoload records), four additional words are used:

- The seventh word contains the current date, eight ASCII characters in the format mm/dd/yy, where mm, dd, and yy are decimal numbers signifying the month, day, and year.
- The eighth word contains the current time, eight ASCII characters in the format hh:mm:ss, where hh, mm, and ss are decimal numbers signifying the hour, minute, and second.
- The ninth word contains the current central processor clock time in microseconds.
- The tenth word contains the number of small pages currently available for subtypes 1, 2, or 3, and is unused for subtype 0.

Job Records

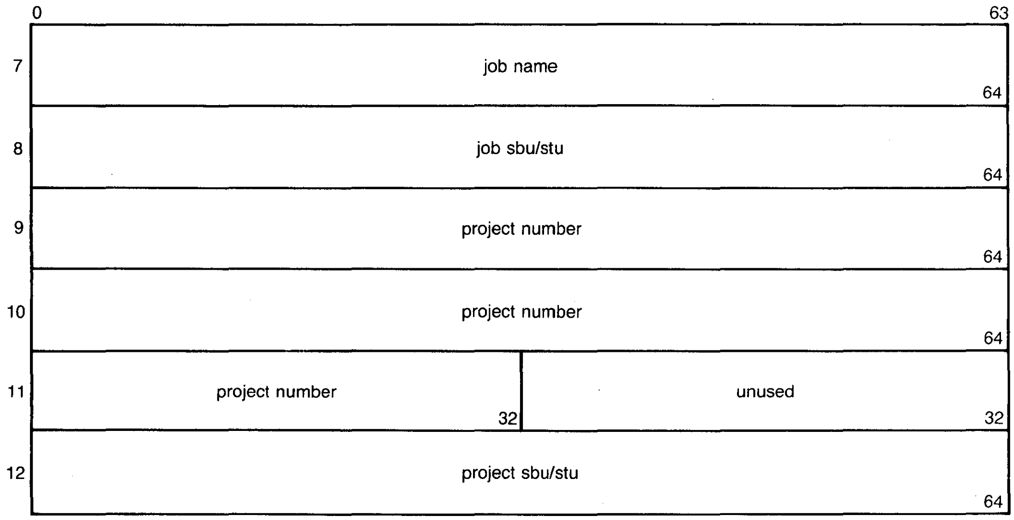
Job accounting (type 6) records are entered into the accounting file for the start and end of jobs. Job records consist of a six-word header plus two additional words for subtype 0, and two to six additional words for subtype 1 and 2. The format of the header is the same as for the task record header (figure 8-4), except that the type field is 6, and the subtype field can be 0 (job start), 1 (job end), or 2 (change job).

The format of the job records is shown in figures 8-14, 8-15, and 8-15.1. The last word for subtype 0 (job start) will be zeroes for any jobs not generated by BATCHPRO. If there is no project number active for the job, the last four words of the subtype 1 (job end) record will not be present.



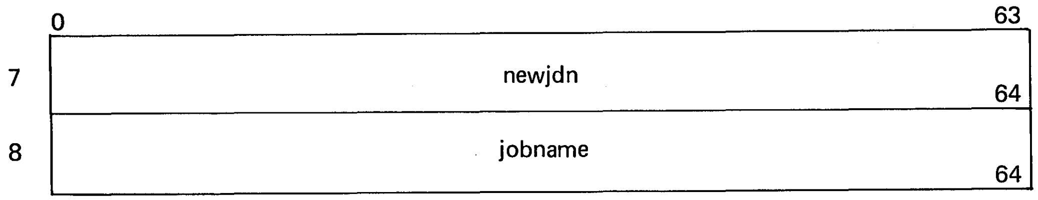
Word	Field	Description
7	job name	Job or interactive session name associated with this job (in ASCII, left-justified with blank fill).
8	dorg	Date when the job entered the input queue. Format is yyddd: yy The last two digits of the year (left 7 bits). ddd The number of days since the beginning of the year (right 9 bits).
	torg	Time when the job entered the input queue. Format is an integer representing system clock, in seconds, since midnight.

Figure 8-14. Job Record Format, Subtype 0



Word	Field	Description
7	job name	Job or interactive session name associated with this job (in ASCII, left-justified with blank fill).
8	job sbu/ stu	SBU/STU amount accumulated for the total job.
9-11	project number	1- to 20-character project number active for this job, left-justified with blank fill. These three words will not be present if a project number is not active for this job.
12	project sbu/stu	SBU/STU amount accumulated for the listed project number. This word word will not be present if a project number is not active for this job.

Figure 8-15. Job Record Format, Subtype 1



Word	Field	Description
7	newjdn	The changed new value of jdn for the job.
8	jobname	Job name (ASCII, left-justified).

Figure 8-15.1. Job Record Format, Subtype 2

Network Usage Records

Network usage (type 7) records are recorded in the system accounting file whenever RHF receives or sends a file. The format is shown in figure 8-16.

	0					63
0	07 8	subtype 8	length 8	dd 16	unused 20	nbs 4
1	account number 64					
2	master clock 64					
3	st 24	nblks 24	unused 16			
4	user number 64					
5	job/session name 64					
6	nchars 36	netblks 24	fiic 4			
7	file name 64					
8	unused 32	cpu 32				

Word	Field	Description
0	subtype	File: 0 Input queue file. 1 Output queue file. 2 Output data file. 3 Input data file. 4 Purge file.
	length	Number of words in the record.
	dd	Data declaration type: C6 File character set ≤ 64. C8 ASCII file and ASCII separators (file character set > 64). US Structured binary. UU Unstructured binary.
	nbs	Network block size indicator: 0 Null transfer. 1 2880 bytes. 2 3840 bytes. 3 4064 bytes.

Figure 8-16. Network Usage Record Format (Sheet 1 of 2)

Word	Field	Description
1	account number	Account to be charged. For input files, the account is from the job; for output files, the account from the file.
2	master clock	Value of the master clock at the time the record was created.
3	st	Logical ID which identifies the CYBER front-end.
	nblks	Number of 512-word blocks transferred to and from disk.
4	user number	Binary user number.
5	job/ session name	Job or interactive session name associated with this task.
6	nchars	Number of characters transferred on the network.
	netblks	Number of network blocks transferred (number of C700/C701/C702 messages).
	fiic	Internal format, subtype=1: 0 ASCII carriage control. 1 ANSI carriage control.
7	file name	Name of file on the CYBER 200 (name of job if QTFS).
8	cpu	CPU time, in microseconds.

Figure 8-16. Network Usage Record Format (Sheet 2 of 2)

Channel Usage Statistics Records

Configuration (type 9, subtype 1) records are written to the account file at system initialization time and whenever a new account file is started. The format is shown in figure 8-17.

Global channel usage data (type 9, subtype 2) records are recorded at a periodic rate which is set at system autoload time. The format is shown in figure 8-18.

IOLOG turns on and off the writing of these statistics to the accounting file. LOGINT sets the time interval of collection. Refer to the VSOS 2 Operator´s Guide for more information about these parameters.

	0								63
0	09 8	01 8	length 8	recf 4	unused 28	nzips 8			
1	master clock 64								
2	devid 12	pu 4	chan 3	port 5	reserved 8	pzip 8	szip 8	pinad 8	sinad 8
3	ponad 8	sonad 8	status bits 16	type 8	unused 8	reserved 16			
4	devid 12	pu 4	chan 3	port 5	reserved 8	pzip 8	szip 8	pinad 8	sinad 8
5	ponad 8	sonad 8	status bits 16	type 8	unused 8	reserved 16			
• •	• •								
6	devid 12	pu 4	chan 3	port 5	reserved 8	pzip 8	szip 8	pinad 8	sinad 8
7	ponad 8	sonad 8	status bits 16	type 8	unused 8	reserved 16			

Word	Field	Description
0	length	Total word length of this record.
	recf	Multiple record flag. If this field is set to 1, a continuation record follows.
	nzips	Number of zip codes.
1	master clock	Value of the master clock at the time the record was created.

Figure 8-17. Channel Usage Statistics Record Format, Subtype 1 (Sheet 1 of 3)

Word	Field	Description
2, 4, 6	devid	Unique three-digit device number (hexadecimal): 001-#0FF Disk. #100-#1FF Tape. #300-#3FF RHF. #500 Maintenance control unit (MCU).
	pu	Physical unit number. This field is used for disk only.
	chan	Channel number: 0 If no SCEX. 1-4 If on SCEX.
	port	Port number for pzip (1 to #10).
	channel	Channel number.
	pzip	Primary zip code for this device.
	szip	Secondary zip code for this device.
	pinad	Primary inboard (C200) network access device (NAD) number.
	sinad	Secondary inboard (C200) NAD number.
3, 5, 7	ponad	Primary outboard (device) NAD number.
	sonad	Secondary outboard (device) NAD number.
	status bits	Bits from SCTFILE describing the device status such as up/down or on/off. This field is used by disk and tape only. Disk status bits:

Bit	Description
16-19	Type installed on disk drive: 0 System pack. 1 Private pack. 2 System/drop pack. 3 Private/drop pack.
20	Indicate whether disk drive is up or down (specified by the operator) for use by the system: 0 Disk drive is down. 1 Disk drive is up.
21	Indicates whether or not disk contains a track fault map: 0 Disk does not have a track fault map. 1 Disk has a track fault map.

Figure 8-17. Channel Usage Statistics Record Format, Subtype 1
(Sheet 2 of 3)

Word	Field	Description
3, 5, 7	status bits	Tape status bits:

Bit	Description
16	Primary NAD down bit.
17	Backup NAD down bit.
18	TAD1 down bit.
19	TAD2 down bit.
22	Single-access bit; set to 1 if TAD2 does not exist.
28	Unit assigned to the user.
29	Unit is read only.
30	Status down bit.
31	Use off bit.

Bits 16 through 19 are normally set to 0, unless the respective TAD NAD is configured through the USE,NUM,NO command. If all TCD or TAD NADs are down, the unit should be down and off.

Word	Field	Description
	type	NAD type:

1	MCU interface NAD (MID).
2	Disk I/O channel NAD (DCD).
3	Tape I/O channel NAD (TCD).
4	RHF I/O channel NAD (RCD).
8	Disk controller NAD (DAD).
9	Tape controller NAD (TAD).
A	RHF remote system NAD (SHD).
B-F	Reserved.

Figure 8-17. Channel Usage Statistics Record Format, Subtype 1 (Sheet 3 of 3)

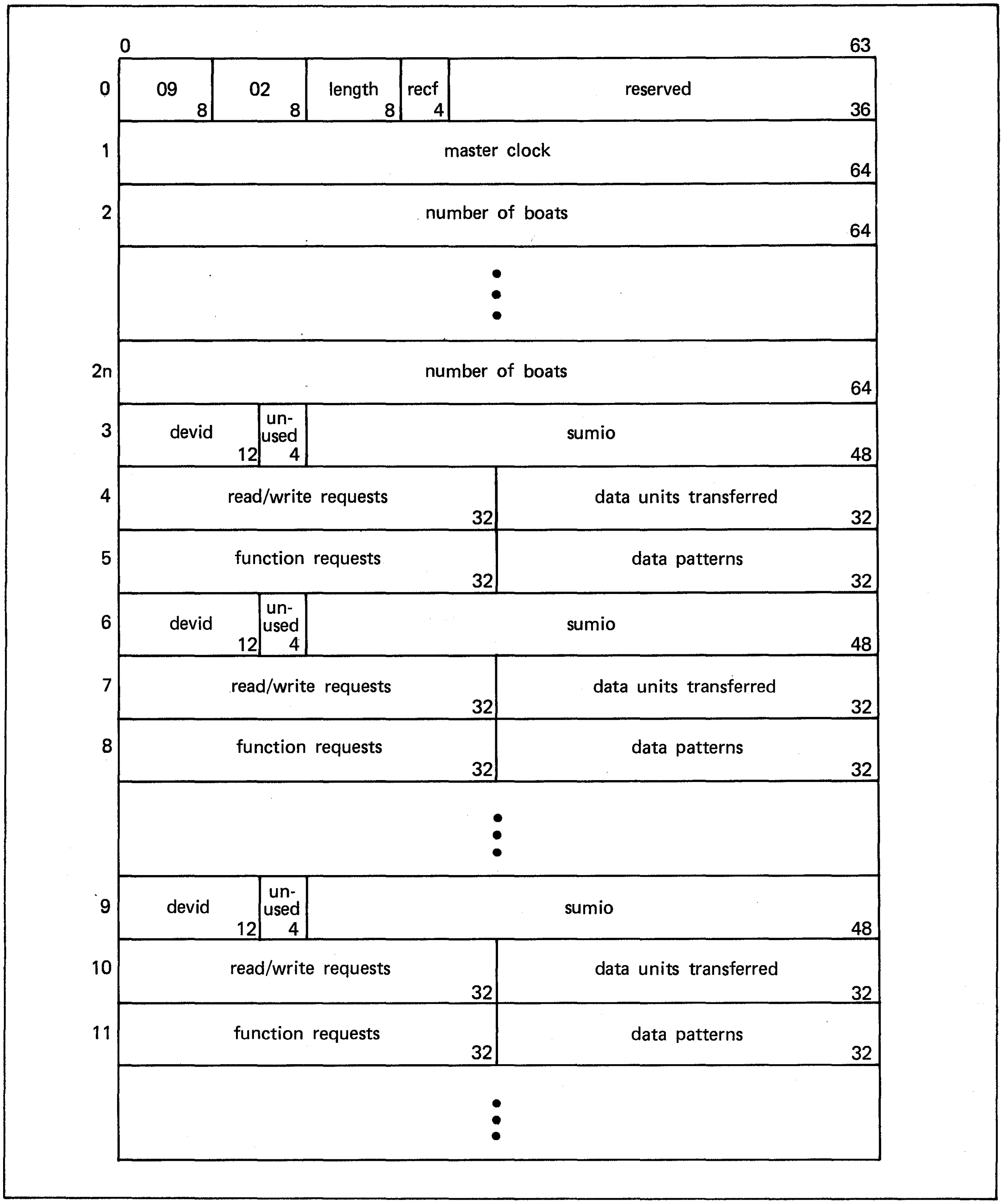


Figure 8-18. Channel Usage Statistics Record Format, Subtype 2 (Sheet 1 of 2)

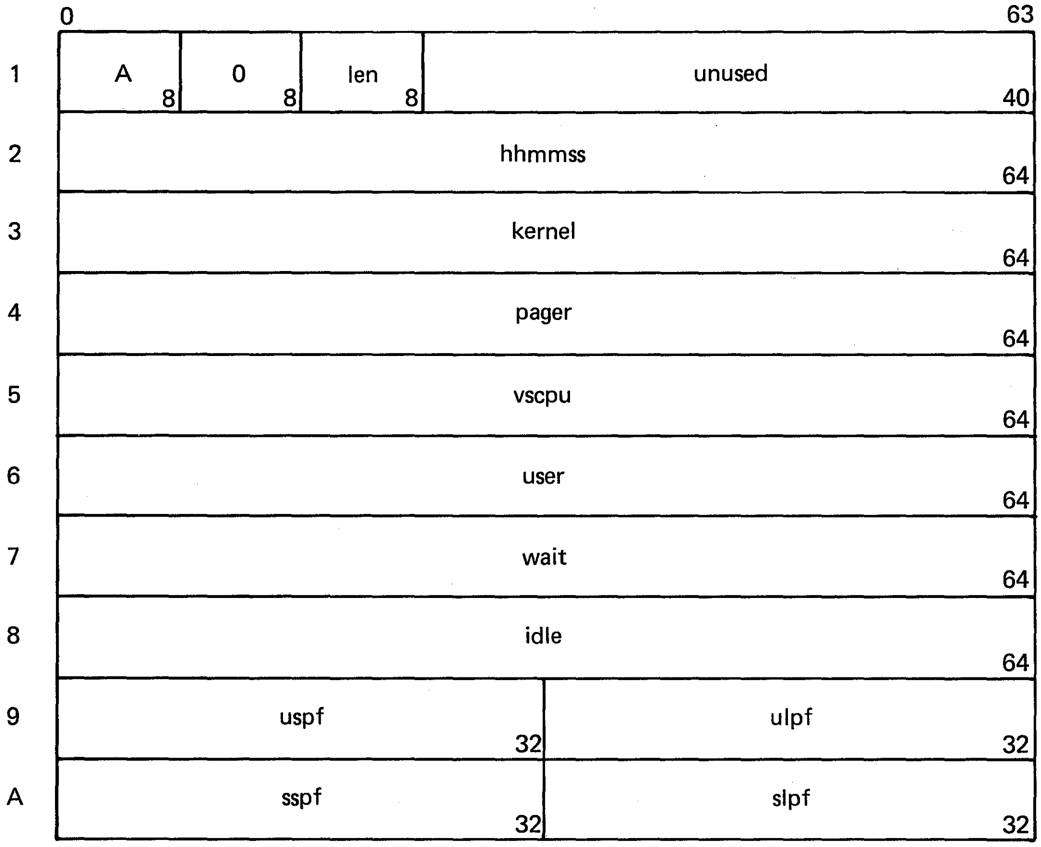
Word	Field	Description
0	length	Variable length is dependent on the number of devid entries containing data.
	recf	Multiple record flag. If this field is set to 1, a continuation record follows.
1	master clock	Value of the master clock at the time the record was created.
2-2n	number of boats	Number of boats on the corresponding positional zip code.
3, 6, 9	devid	Unique three-digit device number (hexadecimal): #001-#0FF Disk. #100-#1FF Tape. #300-#3FF RHF. #500 MCU.
	sumio	Summation of I/O requests to completion times, in microseconds, for this device.
4, 7, 10	read/write requests	Number of read/write requests on this device.
	data units transferred	Number of data units transferred for this device unit. 32 768 bits for 819 disk. 32 bits for tape. 32 bits for RHF. 16 bits for MCU.
5, 8, 11	function requests	Number of nonread/nonwrite requests for this device.
	data patterns	Number of data pattern written (819 disk only).

Figure 8-18. Channel Usage Statistics Record Format, Subtype 2 (Sheet 2 of 2)

Periodic System Records

Periodic system (type A) records are recorded at a periodic rate that is set at system autoload time. This is done at the same periodic rate as that of channel usage records.

The time and counters in the periodic system record are accumulative since the last autoload. The format is shown in figure 8-19.



Word	Field	Description
1	len	Length of the periodic system record (which is a minimum of 10, plus the number of disk drives divided by two, plus the number of tapes that are currently configured).
2	hhmmss	Value of the master clock at the time the record was created.
3	kernel	KERNEL time, in microseconds.
4	pager	PAGER time, in microseconds.

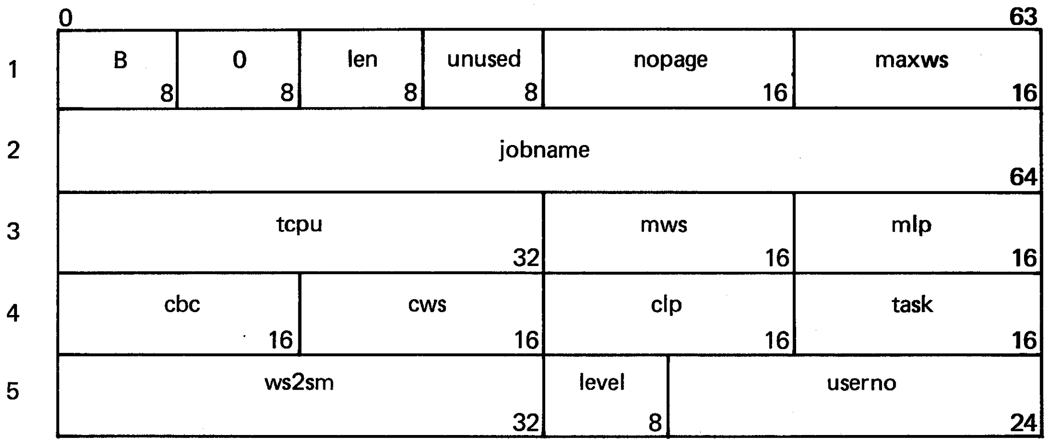
Figure 8-19. Periodic System Record Format (Sheet 1 of 2)

Word	Field	Description
5	vscpu	Virtual system time, in microseconds.
6	user	User time, in microseconds.
7	wait	I/O wait time, in microseconds.
8	idle	Idle time, in microseconds.
9	uspf	User small page faults.
	ulpf	User large page faults.
A	sspf	System small page faults.
	slpf	System large page faults.

Figure 8-19. Periodic System Record Format (Sheet 2 of 2)

Periodic Job Records

Periodic job (type B) records are recorded at a periodic rate which is set at system autoload time. This is done at the same periodic rate as that of channel usage records. This record is not written to the accounting file if there are no jobs currently in the system. The format is shown in figure 8-20.



Word	Field	Description
1	len	Total word length of this record. This value is 1 + 4 * (number of jobs currently in the system).
	nopage	Count of committed memory blocks.
	maxws	Maximum allowable working set.
2	jobname	Job name.
3	tcpu	User execution CPU time, in microseconds.
	mws	Maximum working set.
	mlp	Maximum large page.
4	cbc	Current block count.
	cws	Current working set.
	clp	Current large page count.
	task	Task number.
5	ws2sm	CPU time the task was confined to its maximum working set limit.
	level	Level of task. If level=1, this task entry is the batch processor.
	userno	User number.

Figure 8-20. Periodic Job Record Format

STANDARDIZED ACCOUNTING ENHANCEMENTS

This chapter describes the set of standardized accounting enhancements (SAE) for use on VSOS. It includes the algorithm for computing SBUs and also describes the variable rate accounting (VRA) feature.

Accounting calculations are a part of the virtual system. At the end of each accounting period on VSOS, the SBUs used for a task are computed and entered into accounting records.

CALCULATION OF SBUs

The formula for calculating SBUs is based on usage, system resource variable rate factor (VRF), and service level factor. The algorithm used in standardized accounting enhancements is shown in figure 8-21.

(W(I),I=1,19) is the set of weighting factors associated with the set of system resources (SR(I),I=1,19). A set of installation parameters representing these factors is maintained by the system.

```
SBU=PF(SL)*(VRF(VRI)*W(1)*SR(1)+SUM(W(I)*SR(I),
 I=2,19))
```

PF	Weighting factor for priority or service level.
SL	Service level index.
VRF	Variable rate factor.
VRI	Variable rate index.

Figure 8-21. Algorithm for SBU Calculation

System resources are described as shown in table 8-4. The value for each system resource is determined by the system, dependent on the user task activity in the current accounting period.

Table 8-4. System Resources

I	SR(I)	Description
1	UCPU	User CPU execution time (microseconds) used during this accounting period.
2	SCPU	System CPU time (microseconds) used during this accounting period.
3	MEMU	Memory usage during this accounting period: working set size * CPU time.
4	LPACCX	Number of disk accesses (I/O requests issued) for large page explicit reads and writes during this accounting period.
5	LPACCI	Number of disk accesses (I/O requests issued) for large page implicit writes during this accounting period.
6	SPACCX	Number of disk accesses (I/O requests issued) for small page explicit reads and writes during this accounting period.
7	SPACCI	Number of disk accesses (I/O requests issued) for small page implicit writes during this accounting period.
8	SPSECX	Number of disk sectors transferred for explicit reads and writes during this accounting period.
9	SPSECI	Number of disk sectors transferred for implicit writes during this accounting period.
10	LPGFLT	Number of large page faults (accesses, I/O requests issued) for faults during this accounting period.
11	SPGFLT	Number of small page faults (accesses for faults) during this accounting period.
12	TAPWDS	Number of 16-bit bytes transferred to or from tape during this accounting period.
13	TAPACC	Number of tape accesses, one for each read or write during this accounting period.
14	TAPFNT	Number of tape functions (other than read or write) during this accounting period.
15	AVWSS	Average working set size during this accounting period.
16	VSCALLS	Number of virtual system user calls made during this accounting period.
17	LLPC	Number of large pages lost.
18	LSPC	Number of small pages lost.
19	WS2SML	CPU time (microseconds) that task´s working set size limit appeared to be too small.

VARIABLE RATE ACCOUNTING

Private controllee files running on the system have a standard rate at which the SBUs they consume during execution are charged. Certain other controllees, such as public utilities or applications packages, can be charged at a different rate whose relation to the standard rate is determined by the variable rate factor and service level factor.

The set of defined VRFs is maintained in a virtual system table known as the variable rate table, T_VRF. The set of service level factors, which control job cost dependent on job class (high priority, priority, interactive, batch, or standby), are located in the virtual system table T_PF. SAE makes provisions for the maintenance and use of these two tables.

Variable Rate/Service Level Tables

The variable rate table (T_VRF) is the image of the variable rate chapter of the Q5VRF file as it existed at autoload time. Similarly, the service level table T_PF is the image of the service level chapter of the Q5VRF file at autoload time.

The variable rate index assigned to an executing controllee (refer to EDITPUB and Dynamic Variable Rate Accounting Call in Chapter 4, Volume 1 of the VSOS Reference Manual) provides an offset into T_VRF. The user CPU component of SBUs for this controllee is directly proportional to the variable rate factor pointed to by the VRI. A variable rate factor of 1.0 represents a rate equal to the standard installation charge. It is suggested that the installation enter some default value (such as 1.0) in the first entry of the variable rate chapter of Q5VRF, because all system files will be initially created with a default VRI setting of 0.

The service level or priority of a job provides an offset into T_PF. The SBU calculation for this job is directly proportional to the service level factor pointed to by the SL.

Variable Rate File

The variable rate file is partitioned into two chapters, a set of variable rate entries and a set of service level entries. The format of the file is shown in figure 8-22.

A variable rate entry consists of two fields, a variable rate factor and a password. The variable rate factor is one word containing a 64-bit, floating-point number. The password is one word containing eight ASCII characters, left-justified and blank-filled. If no password is desired for this entry, the field may be set to binary 0.

A service level entry is one word containing a service level factor. This is a 64-bit, floating-point number.

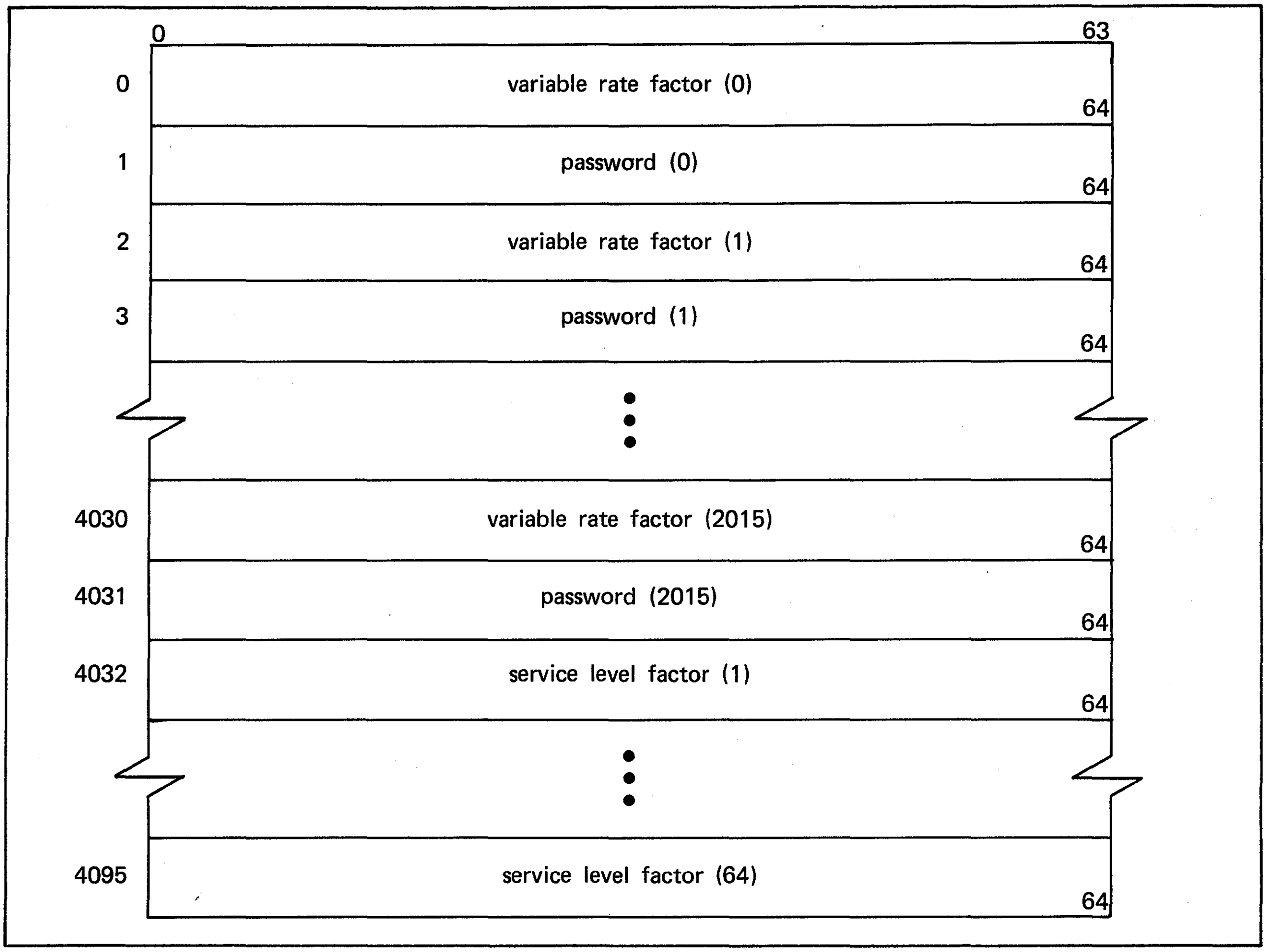


Figure 8-22. Q5VRF File Format

Virtual System Table Definition

If an installation allows variable rate processing (determined by installation parameter IP F_VR), the following steps are taken at autoload time. If the installation parameter IP_F VR is set to 0, meaning that variable rates are not valid at this installation, the variable rate table is never allocated, and no variable rate processing is done. Also, the service level table is not allocated, and no service level factoring of SBUs will be done.

AUTOCON locates the PFI entry for the mass storage file which contains the set of entries for the variable rate table and the service level table.

The file name Q5VRF and the system user number are used as defaults by AUTOCON. Refer to the configuration table display for AUTOCON in the VSOS 2 Operator's Guide for more information. An alternate file can be specified by entering the line:

```
VR = file name, user number
```

at AUTOCON execution time. If the file is not found, AUTOCON displays the message:

```
VARIABLE RATE FILE NOT FOUND
```

and allows the operator to either correct the file name or continue without retrying.

File Maintenance

Q5VRF is maintained as a private file under the system user number 999998. The LOOK utility can modify or display the contents of Q5VRF. If the file entries are changed, an autoload must be done to load the new values into the virtual system variable rate table.

SYSTEM DAYFILE

The system dayfile is a record of all significant system events, including job dayfile entries. (A job dayfile records the processing of a single batch job.) The following paragraphs describe the general dayfile entry format and the specific formats of each entry type.

The system dayfile is a printable file, owned by the system user number, that contains the following types of entries:

- Dayfile entries
- System entries
- Label entries
- Diagnostic entries

Dayfile entries include user dayfile entries, all interactive commands, errors from the batch processor, privileged system task errors, and errors from the terminal input processor.

System entries include all operator commands, all messages sent to the operator, and all illegal terminal logins.

A label entry is written when the file is created. It includes the name of the current system dayfile, the name of the previous system dayfile, and the name of the next system dayfile.

Diagnostic entries are written by customer engineering diagnostics.

The current active dayfile has the name specified by installation parameter IP_SDF_¢NAME. The value used at system release is Q5SDF. If no active dayfile exists during system initialization, the system creates one. The system also creates a second file named according to installation parameter IP_SDF2_¢NAME. The resulting release value is Q5SDF.

When the current active dayfile becomes full, the system performs the following steps.

1. Changes the name of the current primary file to Q5Ddddnn, where ddd is the day number computed when the file was activated and nn is a sequence number, modulo 100. The system detaches the file.
2. Changes the name of the current secondary file to the primary file name (determined by installation parameter IP_SDF_¢NAME).
3. Creates a new secondary file. (The size of this file is determined by the installation parameter IP_SDFSIZE.)

All users can make system dayfile entries. Only privileged/authorized users can do so without also putting the message into the job dayfile. All users are able to make job dayfile entries without putting the message into the system dayfile.

GENERAL FORMAT OF SYSTEM DAYFILE ENTRY

The general format of the system dayfile entry is as follows:

```
hh.mm.ss xxxx text
     or
   *        text
```

hh.mm.ss	Master clock at the time the entry was made. This field starts in column 2 and is preceded by a space.
xxxx	Entry type. This field starts in column 11, is preceded by a space, and is four characters long. This field can have one of the following values: USER SYST LABL DIAG
text	String of characters as defined for the entry type (refer to the entry type format description). This field starts in column 16, is preceded by a space, and is up to 2020 characters long.
*	An asterisk in column 6 indicates that this line is a continuation of the previous line.

The dayfile is an SIL R format file where each entry line is terminated by a #1F character code. The last entry line is terminated by character codes #1F and #1C. All fields in the text are separated by one or more blanks.

USER ENTRIES

A USER type system dayfile entry is written as a result of the following system events:

- The system writes an entry in a job dayfile.
- The user enters a control statement at a terminal.
- The terminal input processor returns an error to a terminal.
- The batch processor returns an error to a job dayfile.
- A privileged system task returns an error to the user.
- A task calls QSSNDMDF with either BOTH or SDFUSER specified.

The format of the system dayfile entry resulting from a job dayfile entry is as follows:

```
hh.mm.ss USER un jdn jn message
            or
  *                message
```

un	User number for this entry (six decimal digits).
jdn	Job descriptor number of the job (four decimal digits).
jn	Batch job file name.
message	First 2000 characters of the job dayfile entry, starting in column 34.

The format of the system dayfile entry when the user enters a control statement from the terminal is as follows:

```
hh.mm.ss USER un jdn message
            or
  *             message
```

un	User number for this entry.
jdn	Job descriptor number of the job (four decimal digits).
message	First 2000 characters of control statement, starting in column 34.

The format of the system dayfile entry for privileged system task errors, errors from the terminal input processor, and errors from the batch processor is as follows:

```
hh.mm.ss USER un jdn message
            or
  *             message
```

un	User number for this entry.
jdn	Job descriptor number of the job (four decimal digits).
message	First 2000 characters of error message, starting in column 34.

SYSTEM ENTRIES

A SYST type system dayfile entry is written as a result of the follwing three events:

- The operator enters a command.
- The user task sends a message to the operator.
- The user enters an illegal login.

The format of the system dayfile entry when the operator enters a command:

```
hh.mm.ss SYST message
        or
     *       message
```

message	First 2000 characters of the operator command, starting in column 34.

A system dayfile entry is written when a user task sends a message to the operator. It is not written when the message to the operator is a system error condition and the dayfile has been turned off (refer to the TMSF operator command description in the VSOS 2 Operator's Guide). The format of this type of system dayfile entry is:

```
hh.mm.ss SYST un jdn tn message
              or
     *                 message
```

un	Task user number for this entry.
jdn	Job descriptor number.
tn	Task name.
message	First 2000 characters of the message being sent to the operator. The message starts in column 34.

The format of the system dayfile entry when a user enters an illegal login command is as follows:

```
hh.mm.ss SYST message
         or
     *       message
```

message	First 2000 characters of the illegal login command, starting in column 34.

LABEL ENTRIES

The header label is written when a new system dayfile is started. The format is:

```
hh.mm.ss LABL fn lf nf dd/mm/yy rs vs sysid mid ps
```

fn	Name of this file will be given when it is deactivated; eight characters in the format Q5Ddddnn.
lf	Name of last file; eight characters in the format Q5Ddddnn. This is blank if there was no previous file.
nf	Name of the next file to be started; eight characters in the format Q5Ddddnn.
dd/mm/yy	Current date.
rs	Resident system version; eight characters in the format RSxxxxxx.
vs	Virtual system version; eight characters in the format VSxxxxxx.
sysid	System ID/pool; eight characters.
mid	Machine ID; one character.
ps	Page size; two characters.

DIAGNOSTIC ENTRIES

The format of diagnostic entries written by customer engineering diagnostics is:

```
hh.mm.ss DIAG message
         or
    *         message
```

message	First 2000 characters of the diagnostics message, starting in column 34.

COMMON EXECUTE LINE SUPPORTING ROUTINES 9

This chapter contains information for a system programmer who is interested in developing an application or utility.

CONVENTIONS

A controllee execute line is entered for processing by VSOS either as a batch processor control statement or as an interactive terminal type-in. An execute line can occur as one or more physical records representing card images or terminal lines. From the point of view of the common execute line supporting routines, an exact correspondence exists between batch commands and terminal commands, including continuation of the command text to more than one card or terminal line. (From the point of view of the user, however, this correspondence does not exist.)

Standard processing is done on five types of linguistic expressions called tokens. The tokens are:

- Alphanumeric identifiers.
- Decimal numeric constants.
- Hexadecimal numeric constants prefixed by the character #.
- Character or string constants delimited by the character ".
- The special characters, which are / # " & .) , = and blank.

The # character is referred to in text as a hash mark. The & character is an ampersand.

Execute line options are defined by means of positional or keyword-identified values. Standard diagnostics are issued if abnormal syntax or conditions are encountered.

A set of four system library routines are to be used to guarantee adherence to the conventions previously stated. The routines are:

Routine	Description
Q7ENVIRN	Determines the program environment.
Q7KEYWRD	Processes the text of an execute line.
Q7MODE	Determines if the task's controller is a terminal.
Q7PROMPT	Provides interaction with the controller; collects parts from several input records, and builds the complete character string for processing by Q7KEYWRD.

When a controllee execute line requires more than one terminal line, an ampersand must be used to designate continuation to the subsequent line. Card image continuation is performed automatically during batch processing if a terminator character has not been encountered. The ampersand signals a logical end of record and can be followed by comments. The text of the execute line consists of two or more tokens: the first is alphanumeric and identifies the task name, while the last is a special character called a terminator. The terminator characters are a period and right parenthesis. An implicit terminator occurs at the end of a terminal line that does not contain an ampersand. Comments can be placed immediately following a terminator character or an ampersand. The following execute lines are equivalent:

```
    SAMPLE,A.     optional comments

    SAMPLE&,A.    optional comments

    SAMPLE A
```

A parameter list can follow the task name but must precede the terminator character. Order-dependent parameters must be in the order specified; key-dependent parameters can appear in any order. Parameter formats depend on the control statement specified, but they always follow the same general guidelines.

Consecutive parameter list items are separated by level-1 separator characters comma and blank. In addition, the left parenthesis acts as a level-1 separator between the task name and the parameter list. A parameter list item can be defined by a list of user numbers or file names. These values are also separated by the level-1 parameter separators. A file name can be followed by attributes of disposition code or length, with attributes separated from each other by the level-2 separator character slash.

Blank is a special character and only performs a separator function when not used with other separators or terminators. Any level of separator can be preceded or followed by blanks, which serve only to highlight the separator; in a similar fashion, the terminator characters can be preceded by highlighting blanks.

System utilities or tasks provide default settings for all on/off options. In addition, the input, output, and binary file options have the default names INPUT, OUTPUT, and BINARY. Where tasks create files for the user, the task can determine the necessary file size or the user is allowed to submit an estimate of an adequate size. Tasks that create files also determine the disposition of the file upon task completion. The user has the opportunity to specify file disposition.

The task name is constructed of one to eight letters and digits. Except where reference is made to a drop file, the first character must be a letter. The task name is bound on the left by the start of the command and on the right by a level-1 separator or a terminator.

Order-dependent parameters are strings of nonseparator, nonterminator characters. Their interpretation is strictly a function of the particular product. An order-dependent parameter list is ended by a terminator or by the occurrence of a key-dependent parameter.

The following are examples of execute lines using order-dependent parameter lists:

```
COPY(FILEA,FILEB)

PURGE,FILE1,FILE2,FILE3.
```

A key-dependent parameter has the general structure shown in figure 9-1. The following is an example of an execute line using a key-dependent parameter list:

```
FTN(I=COMPILE,L=OUTPUT,B=BINARY/PU/#240)
```

key=defns

key	A string of letters and numbers, 1 to 255 characters, delimited to the left by a level-1 separator and on the right by an = character, a separator, or a terminator.
defns	Strings of nonseparator, nonterminator characters whose interpretation is strictly a function of the particular product and the key identifier.

Figure 9-1. Key-Dependent Parameter Format

Examples of the use of both parameter forms are:

```
WXYZ(FILE1,FILE2,OU=MAPFILE)
```

To ensure that ambiguities do not arise, the programmer calling the keyword word processors must not allow the following:

- A parameter resembling a file name to follow a file name list unless that parameter has a key.
- A parameter resembling a user number to follow a user number list unless that parameter has a key.
- A parameter resembling a text string to follow a text string unless that parameter has a key.

Parameter values can be strings of letters and digits, decimal digit strings, hexadecimal digit strings, and character strings delimited by quotation marks. In some cases, the alphanumeric string can occur as two decimal digits followed by one to six letters and numbers. This exception is provided to accommodate drop file names. Decimal digit strings are normally interpreted as decimal constants; a hexadecimal constant is normally preceded by a hash mark. Values that must be virtual bit addresses are always hexadecimal values even if the hash mark is not present. In some cases, such as the GROS option of the loader, a hash mark is required to distinguish the address from identifier data in the same list. Some examples are:

```
WXYZ(EN="!FILE!",OU=MAP/#10)

WXYZ(FILE,OU=MAP/16,LI=SYSLIB,MYLIB)

PURGE,12DROP,JUNK.

COPY,42DROP,SAVEDROP.
```

Lists of values are as order-flexible as the values permit; the user is normally given maximum flexibility consistent with the task requirements. The following equivalent parameter strings illustrate this flexibility:

```
B=FILE/10/PR

B=FILE/PR/10
```

All key-dependent parameters have on and off settings where appropriate, and can be turned on and off. Turning on keys can be accomplished by means of a key=1 parameter, or by use of the key name only; these keys can also be turned off by means of a key=0 parameter. File identification keys should be turned off with key=0. Where the option is normally off, a parameter of the form key=filename turns the option on for a specific file, while use of the key name only turns the option on for a default file.

The following execute lines are equivalent, and illustrate the on/off ability:

```
IMPL,X.

IMPL,I,X=1.

IMPL,X,I=INPUT,B=BINARY/#40.
```

SUPPORTING ROUTINES

Common execute line standards are supported by four subroutines from the system library. The subroutines, which are callable from FORTRAN, META, and IMPL, are:

Subroutine	Description
Q7ENVIRN	Determines the program environment of the calling task. The task may be in one of three environments: batch, interactive, or no level-1 controller.
Q7MODE	Determines whether the parent controller of the calling task is a terminal or another task. A batch job falls into the latter category.
Q7PROMPT	Inputs parameters to be passed to Q7KEYWRD for syntax checking. It prompts terminal users for input if no parameters are specified in the execute line. It also strips the trailing period or matching outside parenthesis characters from the parameter text before calling Q7KEYWRD.
Q7KEYWRD	Examines a character string, checks its syntax, and converts data to internal format. In the case of a detected error, it prints error messages and requests; in interactive mode, it permits error correction by accepting reinput of an execute line parameter. Also, in interactive mode, Q7KEYWRD can be set to request and input each parametric keyword through the use of an appropriate prompting message.

Assembly language routines call any of these subroutines by using the FORTRAN or IMPL type of calling sequence, while FORTRAN and IMPL programs access the routines using CALL statements.

If the main program is coded in FORTRAN, the original execute line is processed by FORTRAN initialization for run-time file substitution. In interactive mode, the program may subsequently call Q7PROMPT or Q7KEYWRD for other lines.

Q7ENVIRN

The function of this subroutine is to determine a task's program environment and to return the information in a full word whose variable name is supplied as the only parameter to the Q7ENVIRN routine. A full word is defined as a 64-bit word that is aligned on a word boundary. The call statement format of Q7ENVIRN is shown in figure 9-1.1.

Q7ENVIRN (environ)

environ   A full-word variable in which one of the following values is returned:

0	The task is executing from within a batch job.
1	The task is executing from within an interactive session.
2	The task does not have a level-1 controller (for example, QTF, PTFS, or QTFS).

Figure 9-1.1. Q7ENVIRN Call Statement Format

Q7MODE

The function of this subroutine is to determine if a task's controller is a terminal and to return the information in a full word whose variable name is supplied as the only parameter to the Q7MODE routine. A full word is defined as a 64-bit word that is aligned on a word boundary. The call statement format of Q7MODE is shown in figure 9-2.

```
CALL Q7MODE (mode)

    mode     A full-word variable in which one of the following values is returned:

                0   Controller is not a terminal.

                1   Controller is a terminal.
```

Figure 9-2. Q7MODE Call Statement Format

Q7PROMPT

This routine serves as an interface between a calling routine and the Q7KEYWRD subroutine. It inputs user parameters into an input buffer, then passes the text to Q7KEYWRD for syntax checking. If no text is specified on the execute line, Q7PROMPT can prompt the interactive user for parameters, using the message PLEASE SPECIFY PARAMETERS or a message provided by the calling routine; otherwise, if no text is specified on the execute line, it can proceed with a call to Q7KEYWRD, optionally setting bit 60 in the options parameter, which causes Q7KEYWRD to prompt for individual keywords.

An input buffer can either be supplied by the calling routine or allocated by Q7PROMPT. Delineator characters, such as matching outside parentheses or a trailing period, are deleted from the input text prior to the call to Q7KEYWRD.

Input text can be continued on succeeding lines in interactive mode, provided that an ampersand (continuation character) is appended to each line.

The call statement format of Q7PROMPT is shown in figure 9-3. The opt, r, rbuf, rlen, and t_i parameters are not used by Q7PROMPT, but are passed to Q7KEYWRD for use in syntax checking. If a text string is not specified in the controllee execute line and the p parameter is not negative, Q7PROMPT prompts for parameters and saves them in a buffer with the name specified as the buf parameter.

CALL Q7PROMPT (txt,p,opt,r,buf,blen,rbuf,rlen,t_1,...,t_n)†

txt	Text string to be passed to Q7KEYWRD. The string must contain any desired carriage control characters.
p	Indicates whether prompting is desired:

>0	Number of character bytes in txt. Use txt to prompt for parameters.
0	Use the text string PLEASE SPECIFY PARAMETERS to prompt for parameters.
-1	Do not prompt for parameters. The value of the variable blen is 0.
-2	Do not prompt for parameters. Options bit 60 should be 0.
-3	Do not prompt for parameters. Wait for message.

buf	Name of buffer file into which the parameters are to be read. If the blen field is 0, the buf field is the name given to a buffer provided by Q7PROMPT.
blen	Name of a full-word variable whose nonzero value indicates the number of character bytes in buf. If the value of blen is 0, no buffer is provided by the caller; in this case, Q7PROMPT allocates a 4096-character buffer named buf. A count of the number of characters actually read is returned by the system into the blen field.

†The opt, r, rbuf, rlen, and t_i fields are described under the Q7KEYWRD call statement.

Figure 9-3. Q7PROMPT Call Statement Format

Q7KEYWRD

The keyword subroutine scans a line of text, checks syntax, and converts data to internal formats. It prints error messages and inputs replacement expressions as required. Q7KEYWRD processes text containing both positional and keyword type parameters. The calling routine provides Q7KEYWRD with syntax tables that completely describe the general format of the input parameters. Q7KEYWRD uses the tables to interpret the specific parameters in the execute line test. These input parameters, called keyword expressions, are written as follows:

key_1 key_2 key_3 . . .

Each key_i is separated from other keyword expressions by one or more blanks or by commas, and has one of the following formats:

lhs = rhs

lhs

rhs

The syntax tables for each key_i keyword relate the valid left-hand sides (lhs) of the expression to valid right-hand sides (rhs). This includes specifying whether the degenerate cases, lhs and rhs, are to be treated as having no left-hand side or no right-hand side. Each key_i, then, can be any one of the following possibilities:

lhs(1) = rhs(1,1)
.
.
.
lhs(1) = rhs(1,n1)
lhs(2) = rhs(2,1)
.
.
.
lhs(m) = rhs(m,nm)

m is the number of possible left-hand sides for the expression, left-hand side k having nk possible right-hand sides.

The syntax tables also specify positional relationships among the keyword expressions. A given expression, key_i, can be flagged as positional, meaning that it must appear after expressions key_1, . . . , $key_{(i-1)}$ but before the expressions $key_{(i+1)}$ . . . If an expression (key_i) is not flagged as positional, it can appear in any order with preceding or succeeding nonpositional expressions; so, if key_i, $key_{(i+1)}$, and $key_{(i+2)}$ are nonpositional, any of the following are valid:

key(i), key(i+1), key(i+2)
key(i), key(i+2), key(i+1)
key(i+1), key(i), key(i+2)
key(i+1), key(i+2), key(i)
key(i+2), key(i), key(i+1)
key(i+2), key(i+1), key(i)

The syntax tables also indicate which key_i parameters are required in the execute line text. If a parameter flagged as required is not encountered in its required location, an error message is issued.

Left-hand sides for an expression include:

- None (the degenerate case, lhs).
- A literal character string, 1 to 255 characters long.

Right-hand sides for an expression include:

- None (the degenerate case, rhs).
- A literal character string, 1 to 255 characters long.
- An arbitrary character string, 1 to 255 characters long.
- Any remaining unscanned text, up to 255 characters maximum.
- A number in the range 0 to $2^{47}-1$ (table setting indicates whether the number can be decimal, hexadecimal with a leading # character, or an address in hexadecimal with no leading # sign required; table settings can also indicate the range of the number if the default range is not sufficiently restrictive).
- A user number.
- A file name (table settings indicate whether a drop file name can be specified and whether the length, print, and punch attributes can be specified).

Lists of numbers, user numbers, and file names separated by slashes, blanks, or commas can be allowed as right-hand sides. A field in the syntax tables indicates that lists are to be allowed and specifies the maximum number of elements permissible.

The entry point Q7KEYWRD is used for both FORTRAN and IMPL calling sequences. The call statement format of Q7KEYWRD is shown in figure 9-4. The lhs table pointers are illustrated in figure 9-5. Each entry in an lhs table points to an rhs table (also full-word-aligned) that describes valid right-hand sides for the given left-hand side and specifies the format in which information is returned in the return buffer to the calling routine.

CALL Q7KEYWRD(opt,r,buf,blen,rbuf,rlen,t_1,...,t_n)

opt Name of a full-word variable, the rightmost 5 bits of whose value indicate the following options:

Bit	Description
59	If 0, send error message to the terminal. If set to 1, return error message to the caller.
60	If 0, scan the input for keyword expressions. If set to 1, prompt for each keyword listed in the tables t_1,...,t_n.
61	If 0, or if user enters "cancel" in response to interactive prompt, abort on syntax error. If set to 1, return to caller on either condition.
62	If 0, prompt for replacement on syntax error. If set to 1, do not prompt for replacement.
63	If 0, send error messages to program controller for output. If set to 1, do not output error messages.

r Name of full-word variable to contain return codes. Return codes are:

0	Text scanned successfully.
1	Internal error; or parameters processed did not match any left-hand side or right-hand side tables; or user entered "cancel" in response to interactive prompt.
2	Return buffer too small.
3	Incorrect number of parameters in Q7 PROMPT/Q7 KEYWORD call line.
4	Invalid type field in lhs table entry.
5	Invalid type field in rhs table entry.
6	Invalid flags field in rhs table entry.
7	Words field for return buffer entry exceeds 255.
8	Options field bit 60 is 1, and prompt message length or address in lhs table header is 0.

Code 1 is returned only if bit 61 of opt field is 1.

Figure 9-4. Q7KEYWRD Call Statement Format (Sheet 1 of 2)

buf	Virtual bit address of string to be scanned for keyword expressions. This field is not used if prompting is requested (options bit 60 is 1).
blen	Name of full-word variable whose value specifies the number of characters in the string indicated by buf. This field is not used if options bit 60 is 1.
rbuf	Virtual bit address of the full-word-aligned buffer (the return buffer) in which reformatted keyword information is to be returned.
rlen	Name of full-word variable whose value specifies the number of characters in the return buffer.
t_i	Virtual bit address of full-word-aligned lhs table (figure 9-5) that describes acceptable syntax constructs and specifies formats for the returned information. The number of addresses varies with the syntax of the line being scanned.

Figure 9-4. Q7KEYWRD Call Statement Format (Sheet 2 of 2)

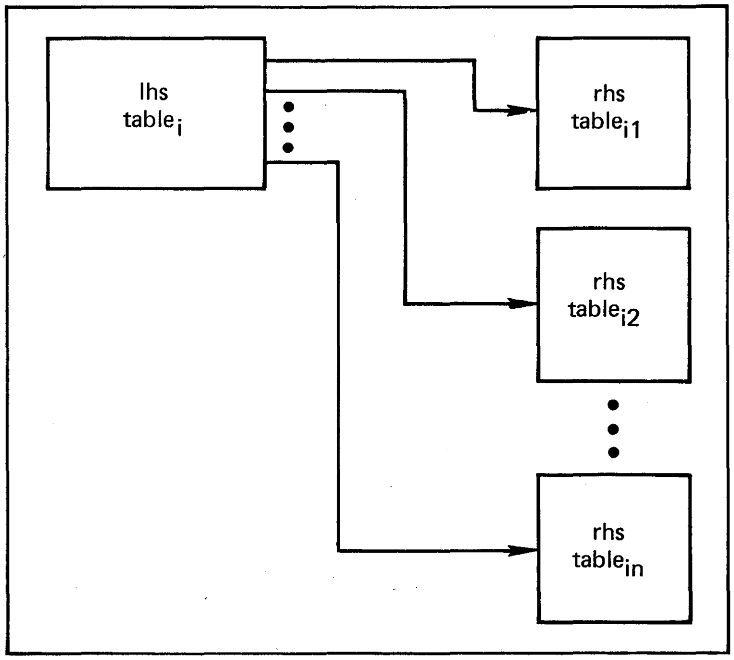


Figure 9-5. lhs Table Pointer Configuration

lhs Table

An lhs table consists of contiguous, variable-length, full-word-aligned entries describing valid keyword expressions. The entries describe the left-hand sides of expressions and, in turn, point to tables whose entries describe valid right-hand sides. A header relates positional and existence requirements of the keywords described by this table.

The lhs table format is shown in figure 9-6. The table header contains two words in the format shown in figure 9-7. Each lhs entry has the format shown in figure 9-8.

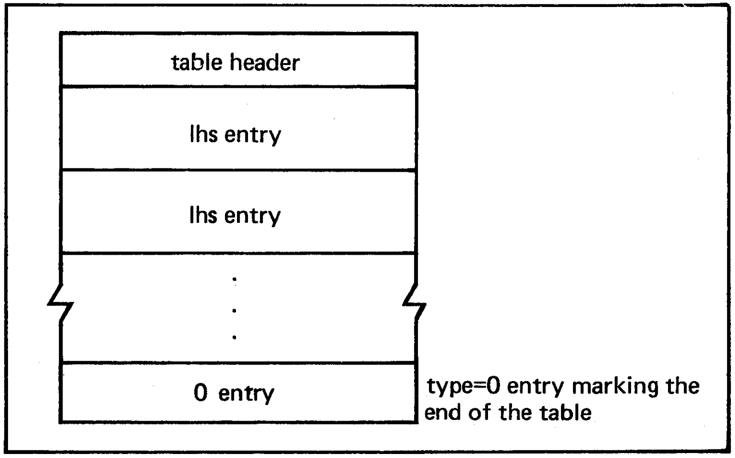
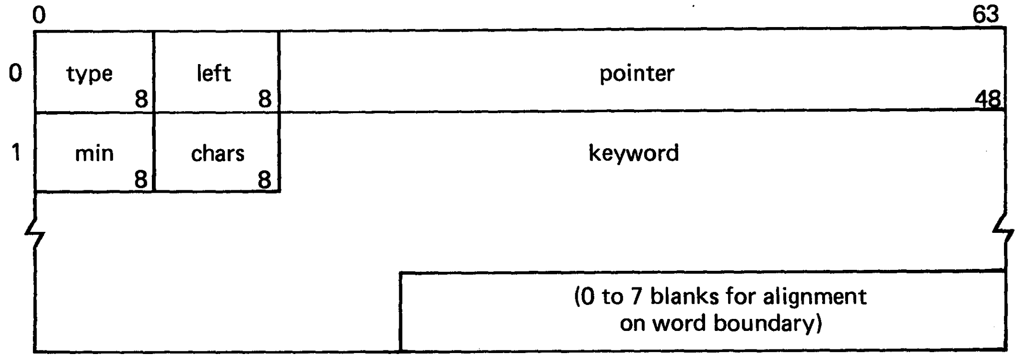


Figure 9-6. lhs Table Format

<table>
<tr><td></td><td>0</td><td></td><td></td><td>63</td></tr>
<tr><td>0</td><td>flags 8</td><td>count 8</td><td>table 8</td><td>unused 40</td></tr>
<tr><td>1</td><td colspan="2">prompt_len 16</td><td colspan="2">prompt 48</td></tr>
</table>

<table>
<tr><th>Word</th><th>Field</th><th>Description</th></tr>
<tr><td>0</td><td>flags</td><td>Bits that are set to describe keywords:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>6</td><td>If 0, entries describe a keyword that is not positional (that is, the keyword described can appear in any order with preceding or succeeding nonpositional keywords); if set to 1, entries describe a positional keyword.</td></tr>
<tr><td>7</td><td>If 0, entries describe an optional keyword; if set to 1, entries describe a required keyword (if no match is found, an error message is issued).</td></tr>
</table></td></tr>
<tr><td></td><td>count</td><td>A value that specifies the maximum number of times this table can be used to effect a keyword match.</td></tr>
<tr><td></td><td>table</td><td>A value set by the caller and returned in a return buffer entry on a successful lhs and rhs match. (The return buffer is described later in this chapter.)</td></tr>
<tr><td>1</td><td>prompt_len</td><td>A value that specifies the number of characters in a message whose address is given in the prompt field; valid only when the options bit 60 is set to 1.</td></tr>
<tr><td></td><td>prompt</td><td>Address of the text to be output as a prompt to request keywords associated with this table; valid only when the options bit 60 is set to 1. Any ASCII carriage control characters desired must be embedded in the text of the prompting message.</td></tr>
</table>

Figure 9-7. lhs Table Header Format



Word	Field	Description
0	type	Entry type. The values are: 0 End of the table. 1 There are no left-hand sides. 2 The keyword expression contains a left-hand side. **NOTE** Where both a literal character string and an arbitrary character string may be used as parameters, the arbitrary character string must follow the literal character string for the parameters to be interpreted correctly.
	left	A value set by the caller and returned in the return buffer entry upon a successful left-hand side and right-hand side match.
	pointer	Address of the table describing right-hand sides that are valid with this particular left-hand side.
1	min	Minimum number of characters needed in the left-hand side before attempting a substring match against the keyword (type=2).
	chars	The number of characters in the keyword if the type field is 2; must be 0 if the type is 1.
	keyword	Text to be used in validating the left-hand side of the expression (type=2).

Figure 9-8. lhs Table Entry Format

rhs Table

The rhs table contains contiguous, variable-length, full-word-aligned entries that describe valid right-hand side expressions. The table format is shown in figure 9-9. The first word of each rhs entry has the format shown in figure 9-10.

When the type field is 0, the rhs table entry is one word having the format shown in figure 9-10, but with the right, flags, and count fields unused. When the type field is 1, the rhs table entry is one word having the format shown in figure 9-10, but with the flags and count fields unused. When the type field is 2, the format of the rhs table entry is as shown in figure 9-11. When the type field is 3, the format of the rhs table entry is as shown in figure 9-12.

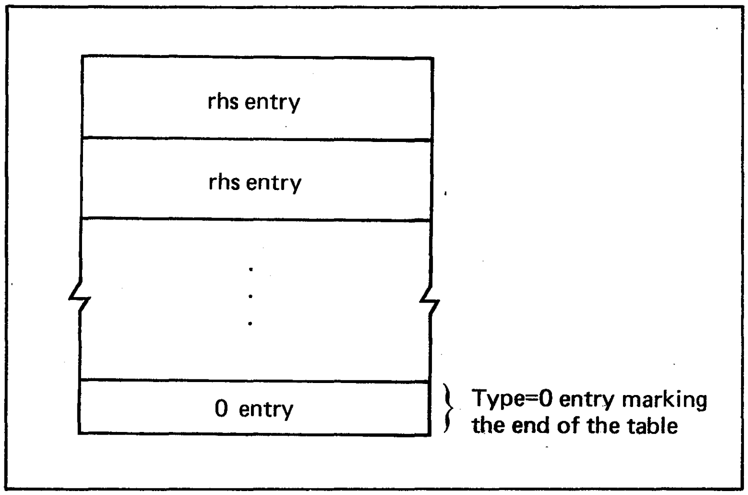
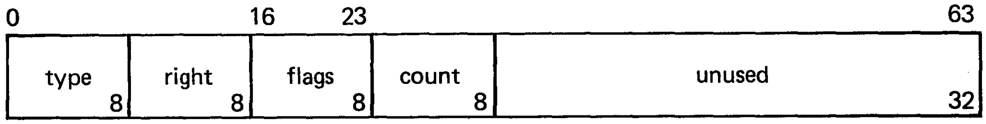


Figure 9-9. rhs Table Format



Field	Description
type	Entry type number. The format of each entry and the meaning of its flags and count fields vary according to the entry type. The types are:

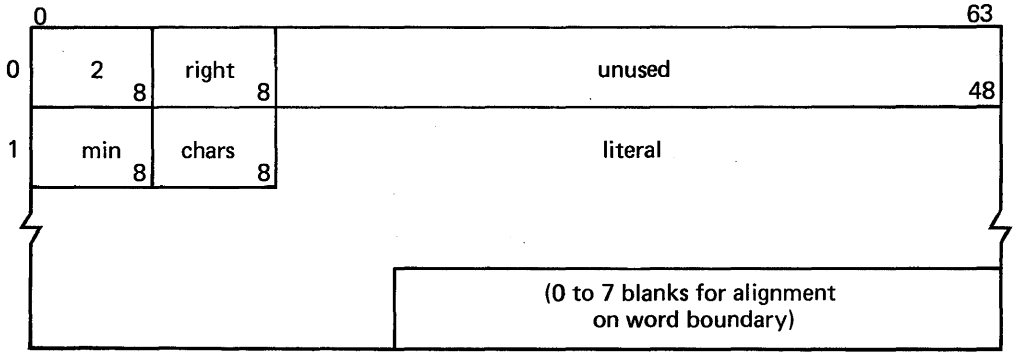
0	End of the table.
1	No right-hand side in the expression.
2	Literal; the right-hand side of the expression must match the initial substring of the literal.
3	Element list.
4	Arbitrary character string is returned in the return buffer.
5	All remaining text (255 characters maximum) is returned in the return buffer.
6	Numbers in the range of $-2^{47}-1$ to $2^{47}-1$ are returned in the return buffer.
7	File names are returned in the return buffer.
8	User number.
9	Ignore the keyword.

NOTE

Where both a literal character string and an arbitrary character string may be used as parameters, the arbitrary character string must follow the literal character string for the parameters to be interpreted correctly.

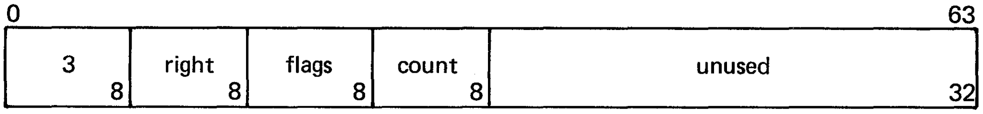
right	A value set by the caller and returned in the return buffer entry on a successful left-hand side and right-hand side match.
flags	Flag bits, which are set to describe valid right-hand sides of expressions. Bit meanings depend on entry type.
count	Maximum number of elements in the right-hand side for those entry types that allow lists.

Figure 9-10. rhs Table Entry Format (First Word)



Word	Field	Description
0	right	A value set by the caller and returned in the return buffer entry on a successful left-hand side and right-hand side match.
1	min	Minimum number of characters needed in the right-hand side before attempting a substring match against the literal.
	chars	Number of characters in the literal.
	literal	Text to be used in validating the right-hand side of the expression.

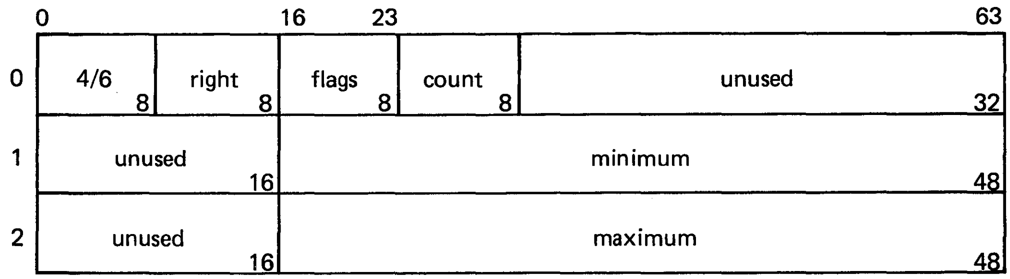
Figure 9-11. rhs Table Entry Format, Type 2



Field	Description
right	Value set by the caller and returned in the return buffer entry on a successful left- and right-hand side match.
flags	Maximum length of a list element. The length of the this field is in the range of one to eight bytes.
count	Maximum number of elements in the list.

Figure 9-12. rhs Table Entry Format, Type 3

When the type field is 4, the format of the rhs table entry is as shown in figure 9-13, except that the flags field is not used. The right-hand side of the expression contains 1 or more literal character strings (255 characters maximum per literal string are returned in the return buffer). Quotes may be embedded within the literal string by using the double quotation mark character to indicate the presence of a quote. During processing, the string will be appropriately edited. Enclosing quotes are required only if special characters defined in table 9-1 are part of the text.



Word	Field	Description
0	right	A value set by the caller and returned in the return buffer entry on a successful left-hand side and right-hand side match.
	flags	Flag bits. If bit 20 is 0, the second and third words are not present. When set to 1, bits 20 through 23 select the following options: 20 Range check is desired; the number specified must be greater than or equal to minimum and less than or equal to maximum. 21 Hexadecimal number valid with a leading hash mark. 22 Address valid (hexadecimal number without a leading hash mark). 23 Decimal number valid. Two or more of bits 21, 22, and 23 can be set at one time; however, a potential identification problem exists if bits 22 and 23 are both set: if the field contains digits 0 to 9 only, the number is treated as decimal; otherwise, it is assumed to be hexadecimal.
	count	Maximum number of values that can appear in the right-hand side.
1	minimum	Lower boundary for a valid number (integer).
2	maximum	Upper boundary for a valid number (integer).

Figure 9-13. rhs Table Entry Format, Type 4/6

When the type field is 5, the format of the rhs table entry is as shown in figure 9-10, except that the flags and count fields are not used. When the type field is 6, the format of the rhs table entry is as shown in figure 9-13.

When the type field is 7, the format of the rhs table entry is as shown in figure 9-10. Four of the individual bits in the flags field can be set to 1, in which case they have the following meanings:

Bit	Description
20	Punch attribute (PU) is valid.
21	Print attribute (PR) is valid.
22	Length attribute can be specified.
23	Drop file name can be specified.

Two or more of the bits can be set at one time. The count field contains the number of file names that appear in the associated return buffer entry.

When the type field is 8, the format of the rhs table entry is as shown in figure 9-10. Two of the individual bits in the flags field can be set to 1, in which case they have the following meanings:

Bit	Description
22	Return an ASCII value.
23	Return a binary value.

One or both of the bits can be set at one time.

When the type field is 9, the format of the rhs table entry is as shown in figure 9-10, except that the right, flags, and count fields are not used. For this type, no entry is made in the return buffer and processing continues with the next keyword expression.

Return Buffer

This buffer is used to contain reformatted keyword information that is returned. The end of the returned information is indicated by a full-word binary 0. The return buffer format is shown in figure 9-14.

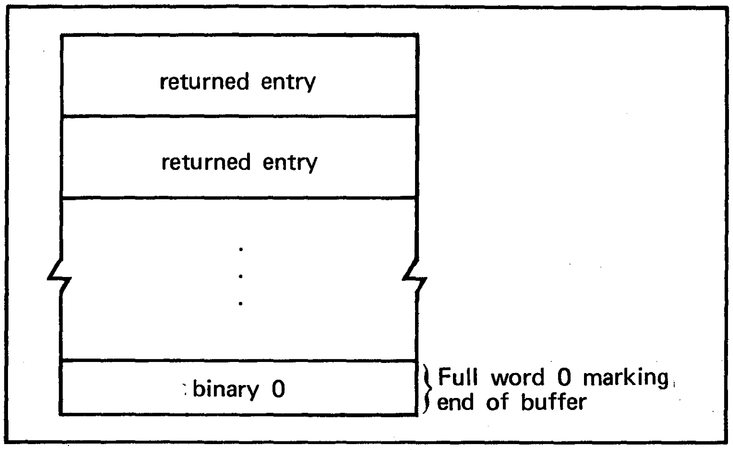
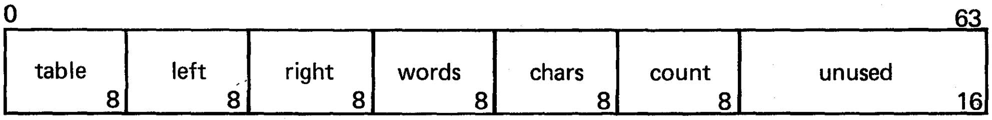


Figure 9-14. Return Buffer Format

The format and length of each return buffer entry depends on the type field of the right-hand side table entry that successfully matched the right-hand side of the keyword expression. Common to all entries is the first full word, whose format is shown in figure 9-15.



Field	Description
table	Value from the header of the lhs table (t_i) that provided the left- and right-hand side entries affecting the keyword match.
left, right	Values from the left field of the particular lhs table entry and the right field of the rhs table entry affecting the left and right-hand side matches.
words	Total number of words in this particular entry.
chars	The meaning of this field varies with the type of the right-hand side. It is the length of the returned information or the length of an element of a returned list, such as a list of file names.
count	Number of returned elements if returned data consists of a list of items.

Figure 9-15. Return Buffer Entry Format (First Word)

When the rhs table entry types are 1 and 2, the format of the return buffer entry is as shown in figure 9-16. The words field is always 1.

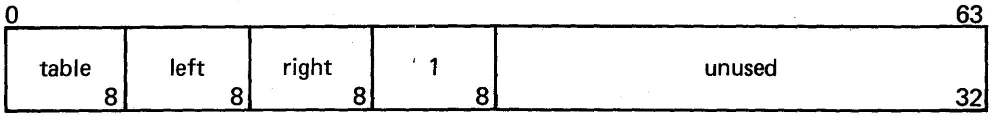
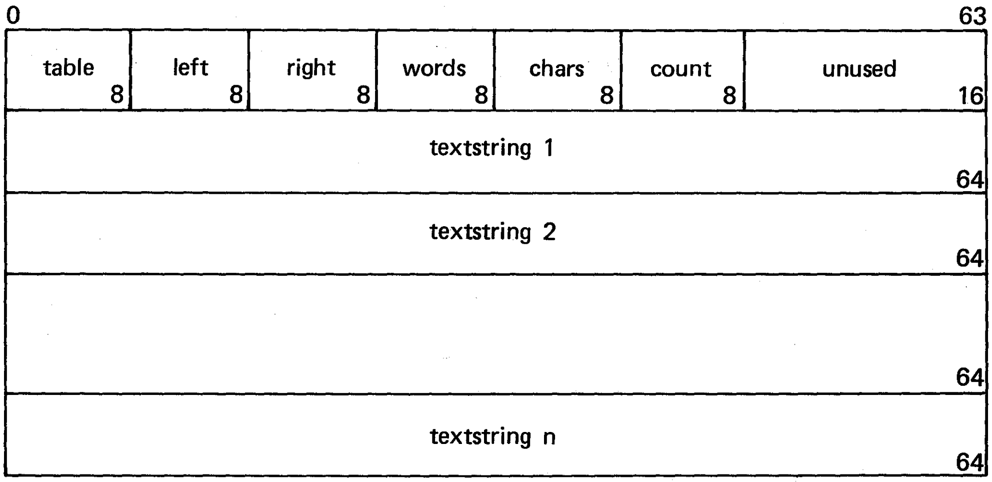


Figure 9-16. Return Buffer Entry Format, Types 1 and 2

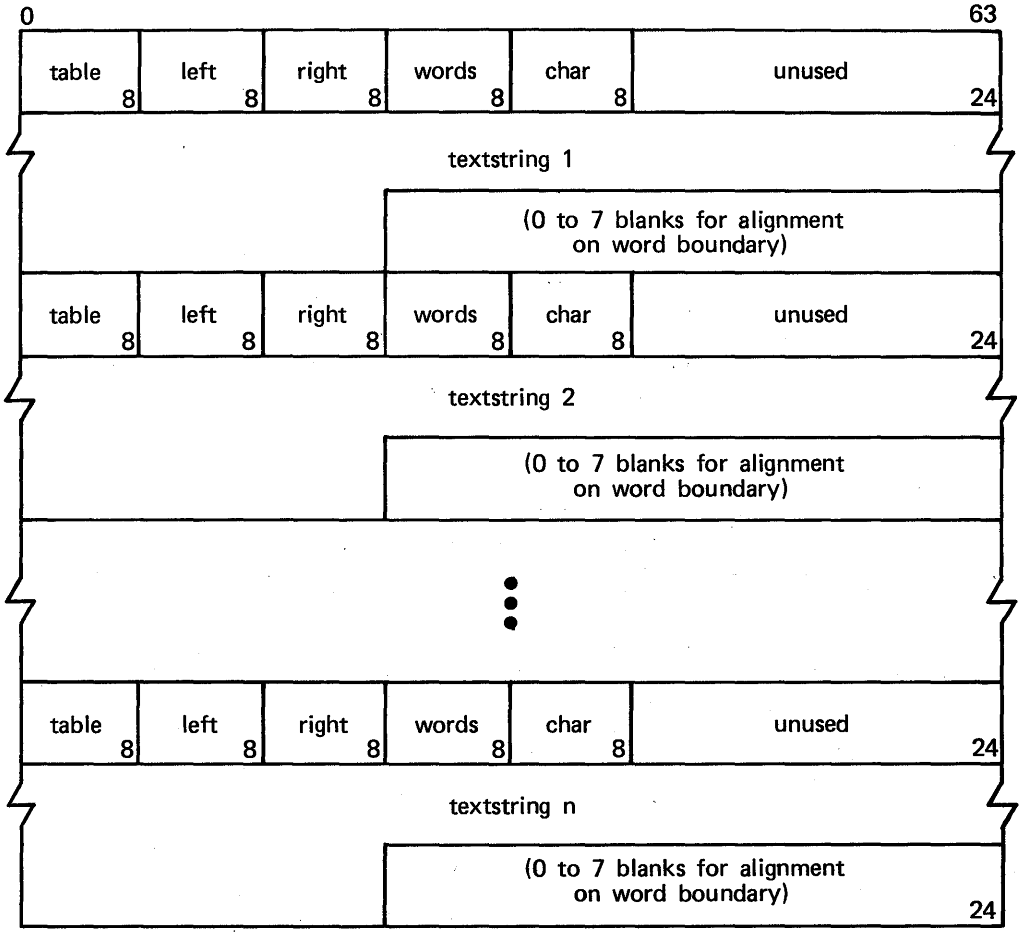
When the rhs table entry type is 3, the flags field is used to specify the maximum allowable length of a list element. The allowable range of values is 1 through 8. The format of the return buffer is shown in figure 9-17.



Field	Description
table	Value from the header of the lhs table (t_i) that provided the left- and right-hand side entries affecting the keyword match.
left, right	Values from the left field of the particular lhs table entry and the right field of the rhs table entry affecting the left and right-hand side matches.
words	Total number of words in this particular entry.
chars	The meaning of this field varies with the type of the right-hand side. It is the length of the returned information or the length of an element of a returned list, such as list of file names.
count	Number of elements returned.
text string i	Returned elements, ASCII left-justified and blank-filled.

Figure 9-17. Return Buffer Entry Format, Type 3

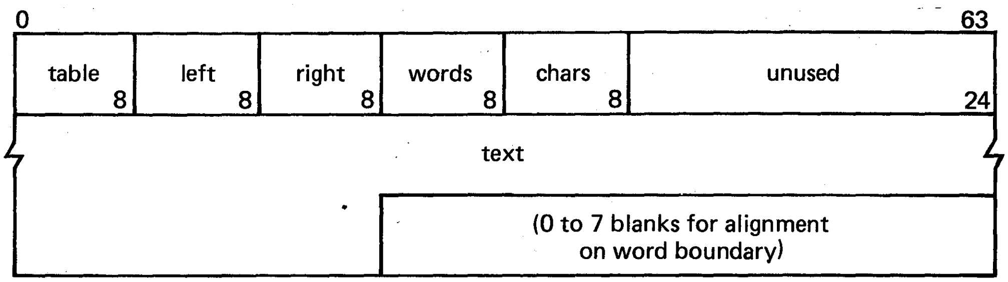
When the rhs table entry type is 4, the format of the return buffer entry is shown in figure 9-18. Since the multiple literal strings will likely be variable in length, the format of the return buffer returned for type 4 differs from the format of all other return buffers. A header word will precede each literal string returned.



Field	Description
char	Number of characters returned in text.
text string	The right-hand side of the expression.

Figure 9-18. Return Buffer Entry Format, Type 4

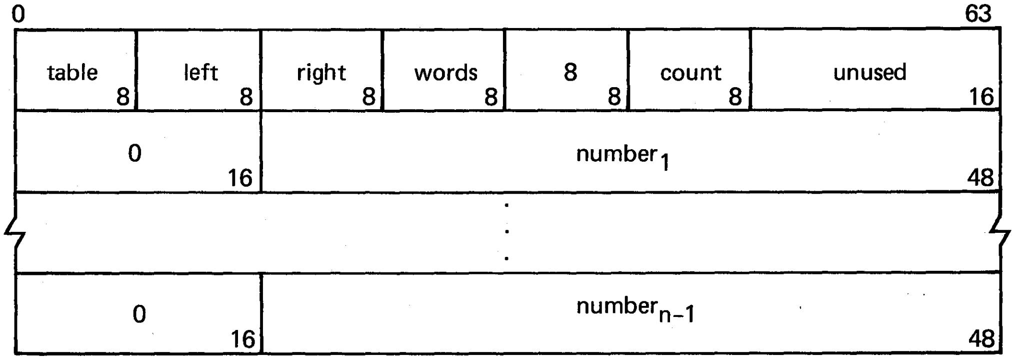
When the rhs table entry is 5, the format of the return buffer is as shown in figure 9-19.



Field	Description
chars	Number of characters returned in text.
text	The remaining text (type=5).

Figure 9-19. Return Buffer Entry Format, Type 5

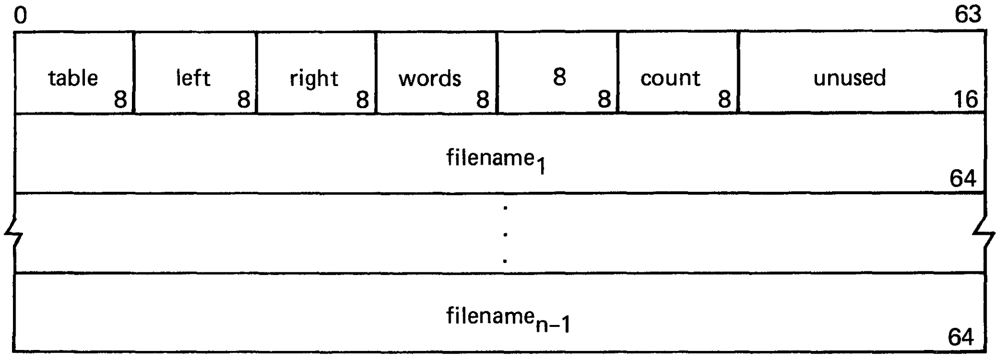
When the rhs table entry type is 6, the format of the return buffer entry is as shown in figure 9-20. The chars field is always 8.



Field	Description
count	The number of digits returned.
$number_i$	Binary form of the number specified.

Figure 9-20. Return Buffer Entry Format, Type 6

When the rhs table entry type is 7 with flag bits 20, 21, and 22 all set to 0, the format of the return buffer entry is as shown in figure 9-21. The chars field is always 8.



Field	Description
count	Number of files listed, equal to n-1.
filename$_i$	**Logical file name, in ASCII.**

Figure 9-21. Return Buffer Entry Format, Type 7 with Zeroed Flags

When the rhs table entry type is 7 with flag bits 20, 21, or 22 set to 1, the format of the return buffer entry is as shown in figure 9-22. The chars field is always 16.

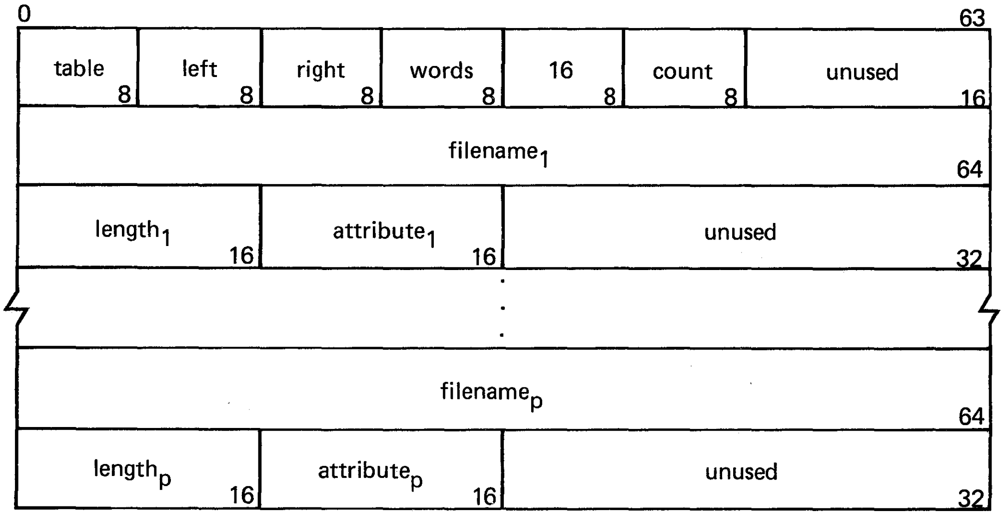
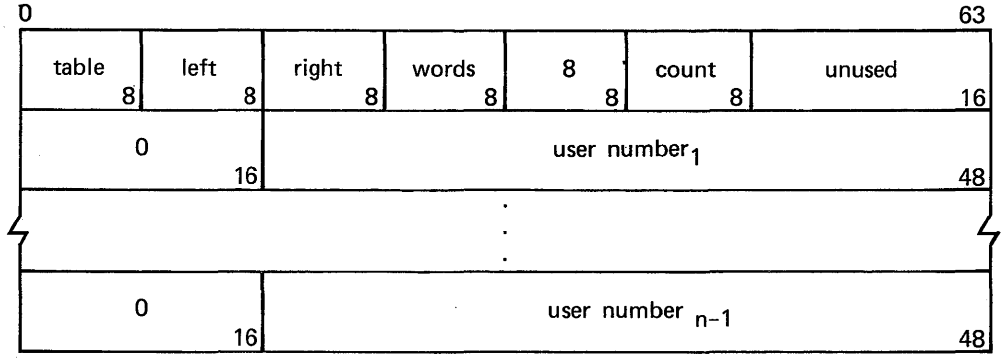


Figure 9-22. Return Buffer Entry Format, Type 7 with Set Flags (Sheet 1 of 2)

Field	Description
count	Number of files listed; has the value (n-1)/2.
filename$_i$	Name of the file specified, left-justified with blank fill.
length$_i$	Length of the file in small pages. If not specified, binary 0 is returned.
attribute$_i$	File attribute: ASCII punch (PU) or print (PR). If not specified, blanks are returned.

Figure 9-22. Return Buffer Entry Format, Type 7 with Set Flags (Sheet 2 of 2)

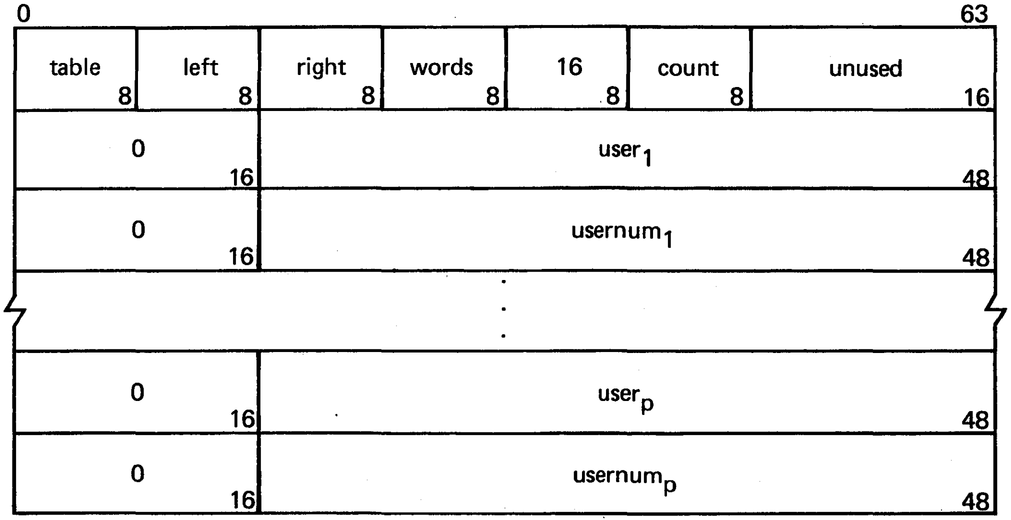
When the rhs table entry type is 8 with only one of flag bits 22 and 23 set, the format of the return buffer entry is as shown in figure 9-23. The chars field is always 8.



Field	Description
count	Number of user numbers listed.
user number	ASCII or binary user number, depending on the flags set.

Figure 9-23. Return Buffer Entry Format, Type 8 with One Set Flag

When the rhs table entry type is 8 with both flag bits 22 and 23 set, the format of the return buffer entry is as shown in figure 9-24. The chars field is always 16.



Field	Description
count	Number of user numbers returned, equal to (n-1)/2.
user$_i$	Binary user number.
usernum$_i$	User number of the user; interpreted as ASCII characters.

Figure 9-24. Return Buffer Entry Format, Type 8 with Two Set Flags

Special Characters

The Q7KEYWRD subroutine scans for special characters in the execute line text to extract keyword expressions. These characters and their meanings (under given conditions) are described in table 9-1.

Table 9-1. Execute Line Special Characters

Character	Description
"	Delimits a literal character string on the right-hand side of the expression. An embedded quote within a literal character string must be represented by the double quotation mark character; for example, "AB""C""DE" would be the representation of the literal string AB"C"DE. Each string of the right-hand side must be enclosed in quotes if it includes a special character of table 9-1. Q7KEYWRD will not perform concatenation of a literal in quotes and other character strings.
blank	Delimits a keyword expression unless it occurs within a literal character string.
,	Delimits a keyword expression unless it occurs within a literal character string.
=	Separates the left- and right-hand sides of keyword expressions unless they occur within a literal character string.

LOADER CONVENTIONS

10

This chapter contains formats for the following loader tables:

- Module header table
- Code block table
- Code relocation table
- External/entry table
- Interpretive data initialization table
- Interpretive relocation initialization table
- Transfer symbol table
- Debug symbol table
- Symbol definition table
- Pseudoaddress vector table

The following loader tables are used by the system during error processing:

- Module header table
- Code block table
- External/entry table
- Debug symbol table
- Symbol definition table
- Pseudoaddress vector table

Error processing information is provided for every object module loaded to produce a controllee file. This includes object modules of user-specified files and required object modules for system library files.

The loader initializes the following registers in the 0 (zero) page of the controllee:

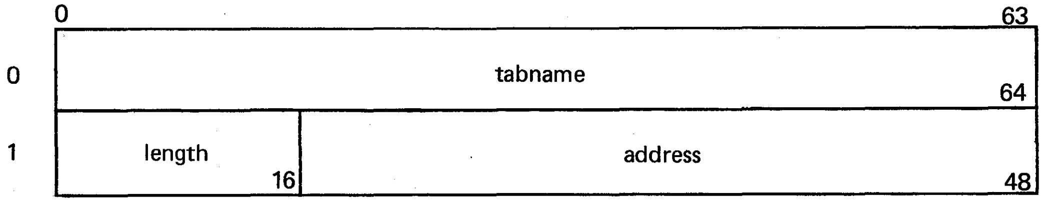
Register	Description
#05	SHRLIB version.
#06	Entry address (origin+8000).
#07	USERLIB owner.
#08	Origin (for C runtime).
#09	Length of error processing information.
#0A	Version (from VR parameter).
#0B	Date.
#0C	Time.
#0D	Address of the error processing information.
#0E	Contains SHRLIB if controllee requires the system shared library; otherwise, #0E = 0.
#0F	dorg and torg (from PFI) for SHRLIB.

Register	Description
#10	The rightmost 48 bits contain the bit address of the system shared library if the controllee requires the system shared library; otherwise, #10 = 0.
#11	ULIB name.
#12	dorg and torg for ULIB.
#13	Origin of ULIB.
#14	Constant #20.
#15	Constant #1A.
#16	Constant 1.
#1B	Dynamic stack address.
#1C	Current register save area descriptor (length=6, address=dynamic stack address - #180).
#1E	Length and address of main data base.
#1F	Entry address.

Other registers are initialized to 0, but can be initialized to other values, as necessary.

GENERAL TABLE STRUCTURE

The loader works with files that are composed of one or more object modules. Each object module consists of a number of standard tables; each table begins with a standard two-word header. The format of the table header is shown in figure 10-1.



Word	Field	Description
0	tabname	Name of the table, in ASCII. The table names are: △MODULE△ INT△RELO △△CODE△△ XFER△SYM REL△CODE SYMB△TAB EXT△ENTR SYMB△DEF INT△DATA △△PAV△△
1	length	Length of the table, in full words. For the debug symbol table, this field includes the length of the symbol definition table.
	address	Bit difference between the first word of the respective table and word 1 of the module header table; that is, back pointer (bits) + address of the first word of the respective table (bits) = address of word 1 of the header table (bits).

Figure 10-1. Loader Table Header Format

MODULE TABLES

The module tables described here are the header, code block, code relocation, and external/entry tables.

MODULE HEADER TABLE

The module header table contains general information concerning the object module and provides a linkage to all the other tables in the module. The format of the module header table is shown in figure 10-2.

Words 7 through n of the module header each contain a table type and an address pointer to a table of that type. The pointer contains a bit address relative to the first word address of the header. By convention, the first table described is the code block table and the second is the external/entry table.

Table types are listed in table 10-1. Only types 1, 2, 6, and 301 appear in the error processing information area of an object module.

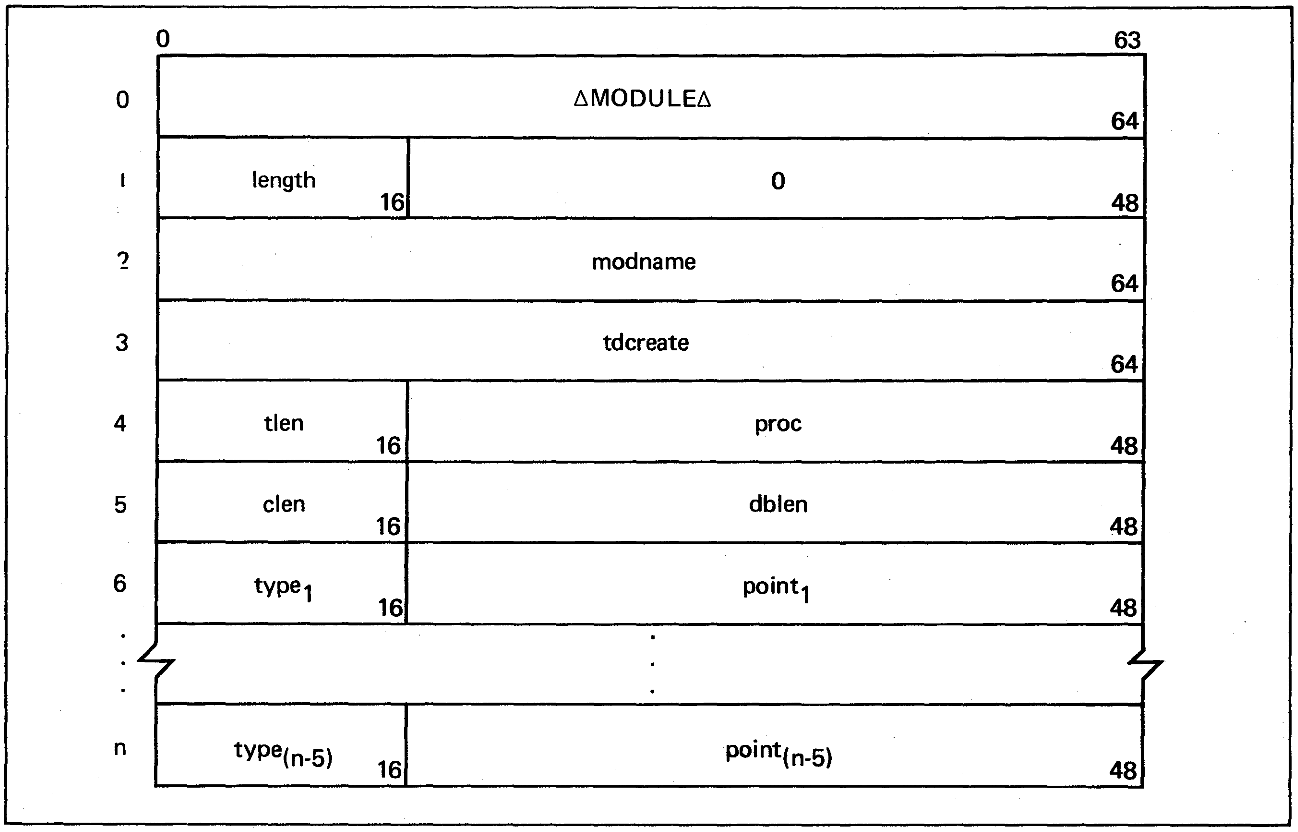


Figure 10-2. Module Header Table Format (Sheet 1 of 2)

Word	Field	Description
1	length	Length of the table, in full words.
2	modname	Name of the module, in ASCII; eight characters, left-justified with blank fill.
3	tdcreate	Date and time the module was created; 15 digits (in hexadecimal form) and a positive sign. The format is +yymmddhhttssccc, where yy expresses the year, mm the month dd the day, hh the hour, tt the minute, ss the second, and ccc the millisecond.
4	tlen	Word length of tables, excluding the code.
	proc	ASCII name of the processor that created the module.
5	clen	Length of the code, in words.
	dblen	Length of the data base area, in bits.
6 to n	$type_i$	Table type (refer to table 10-1).
	$point_i$	Address pointer to a table of the type indicated in the type field. If the hexadecimal type is 4, the pointer contains the bit address of the next module header table.

Figure 10-2. Module Header Table Format (Sheet 2 of 2)

Table 10-1. Module Header Table Types

Type †	Module Name	Description
1	△△CODE△△	Code block table.
2	EXT△ENTR	External/entry table.
3	REL△CODE	Code relocation table.
5	XFER△SYM	Transfer symbol table.
6	SYMB△TAB	Debug symbol table.
101	INT△DATA	Interpretive data initialization table.
201	INT△RELO	Interpretive relocation initialization table.
301	△△PAV△△△	Pseudoaddress vector table.

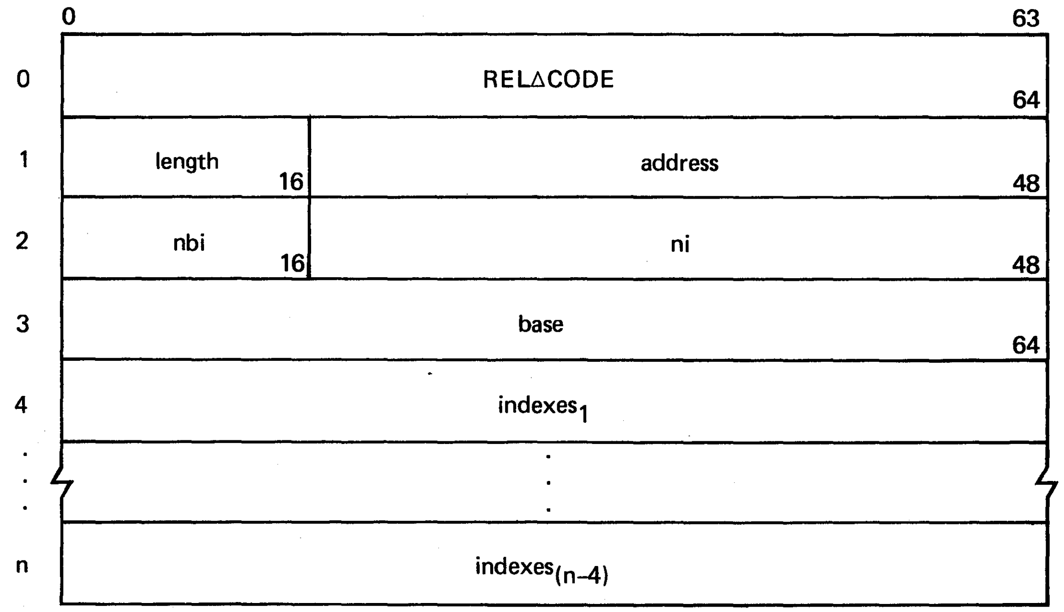
† These types appear in the error processing information area of an object module.

CODE BLOCK TABLE

The code block table contains the executable code. The table consists of the two-word loader table header (figure 10-1), followed immediately by one or more words of executable code. The table name is CODE. When the code block table is loaded in the controllee, the code block table has a pointer in the error processing information area. In this capacity, the table contains the program name (in ASCII) in word 1 rather than the character string △△CODE△△.

CODE RELOCATION TABLE

This table describes relocation in the code. The format of the code relocation table is shown in figure 10-3. When this table is processed, the bit base address of the code is added to the 48-bit fields pointed to by the indexes in the bit string. If this table has a type of 8003; it means that SLGEN has preprocessed this table by adding the addresses at which this library is to be placed in the 48-bit field.



Word	Field	Description
1	length	Length of the table, in full words.
	address	Bit difference between the first word of the respective table and word 1 of the module header table.
2	nbi	Number of bits per index in the bit strings that start in word 5.
	ni	Number of indexes in the string.
3	base	Current bit address to which this module is relocated.
4 to n	indexes$_i$	Bit string of indexes, each nbi bits long. Each index references a half-word of code to be relocated relative to the base address of the code.

Figure 10-3. Code Relocation Table Format

EXTERNAL/ENTRY TABLE

The external/entry table contains definitions for all entry points, external symbols, and common blocks. These definitions consist of lists of entry point names, external names, entry point descriptors, and external descriptors. The format of the table is shown in figure 10-4. In words 3+n and 3+2n, the quantity n-m is the number of external names in the table.

Each descriptor in the external/entry table has the form shown in figure 10-5.

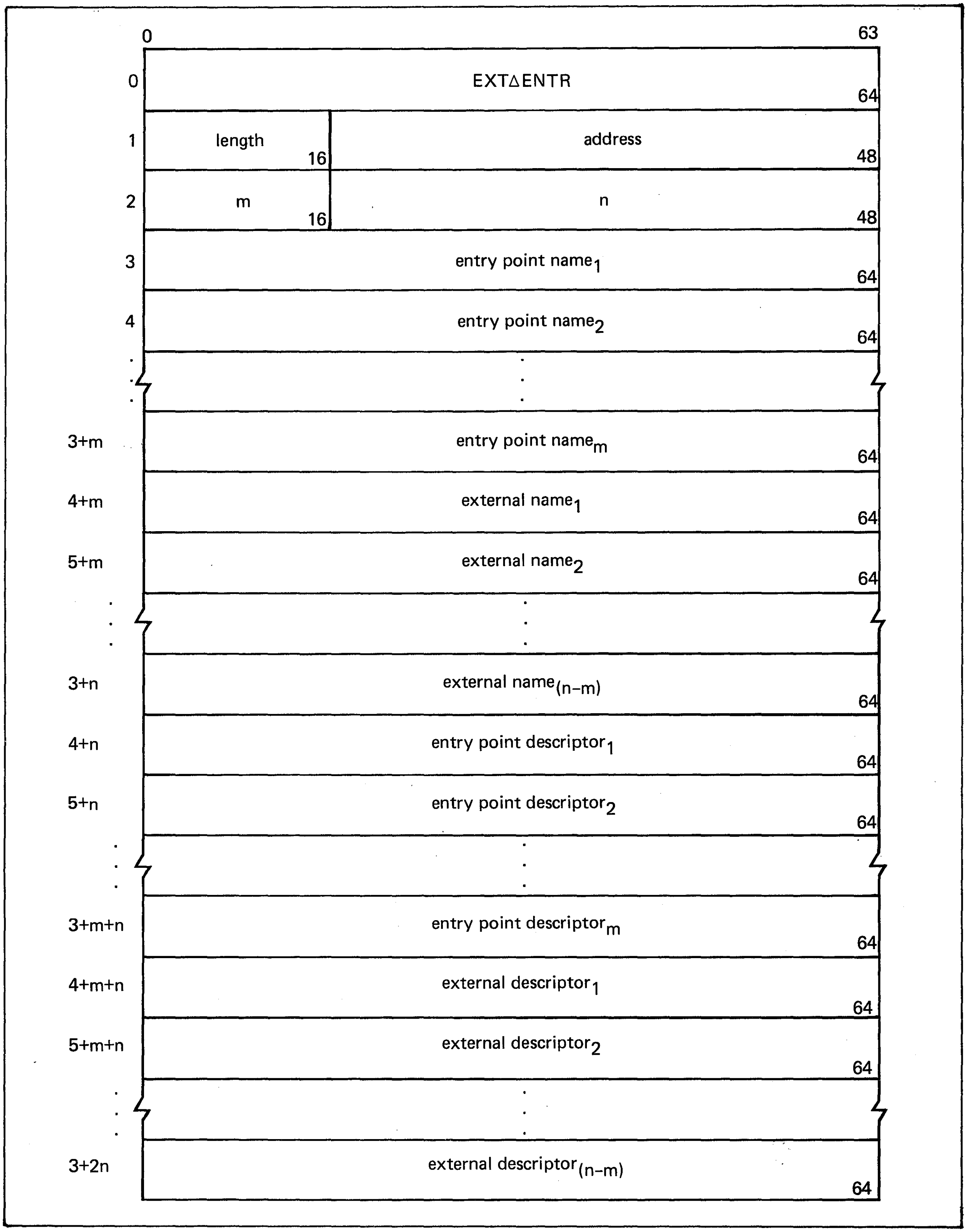
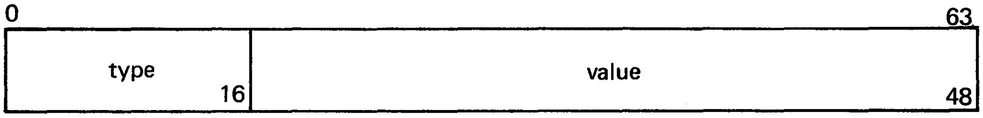


Figure 10-4. External/Entry Table Format (Sheet 1 of 2)

Word	Field	Description
1	length	Length of the table, in full words.
	address	Bit difference between the first word of the respective table and word 1 of the module header table.
2	m	Number of entry point names in the table.
	n	Total number of names in the table.

Figure 10-4. External/Entry Table Format (Sheet 2 of 2)



Field	Description
type	Symbol type: 1 Entry point in code. 2 Entry point in data. 3 Constant entry point. #14 External procedure. #15 External data. #16 Common block.
value	Depends on the type field. If type is 14 or 15, value is 0; otherwise, if type is 1 to 2, value is the entry point's relative bit address in the code or data block; if type is 3, value is a 48-bit constant; and if type is 16, value is the bit length of the common block.

Figure 10-5. Descriptor Format for Externals and Entry Points

The symbol types are defined as follows:

Entry point

A named value defined in the procedure; it is to be referenced as an external by an external procedure. It can be an address in the code block, an address in the data base, or a constant value.

Common block

A named alterable space referenced by one or more procedures. A common block can be initialized with relocatable data. A blank common is a common block with a name of eight blanks.

External procedure

An external that is referenced in a call. Having a symbol doubly defined as a common block and external procedure is specifically allowed. All external procedure names are eight characters, left-justified with blank fill.

External data

An external that is referenced by a method other than a procedure call.

INTERPRETIVE DATA INITIALIZATION TABLE

When the loader processes information in the interpretive data initialization table, areas of static space are initialized. The table consists of the two-word loader table header (figure 10-1), followed immediately by one or more variable-length entries. The table name is INTΔDATA. Each entry contains a one-word descriptor and a two-, three-, four-, or six-word data item.

Data item and item descriptor pairs in the interpretive data initialization table are formatted as shown in figures 10-6 through 10-9. The first word in each figure is the data item descriptor. The remainder of each figure describes the data item proper, which is stored in the formats shown.

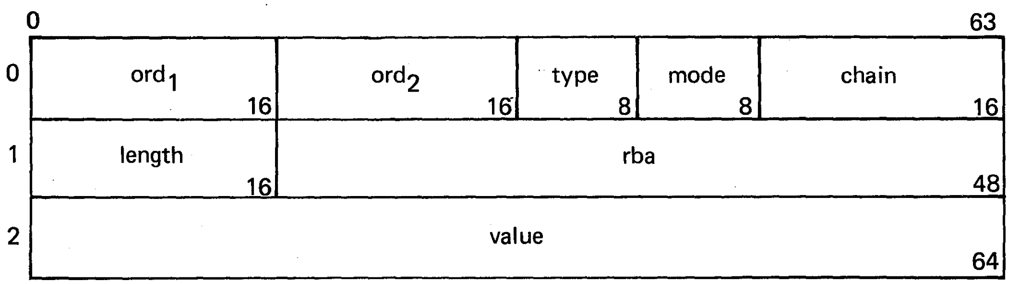
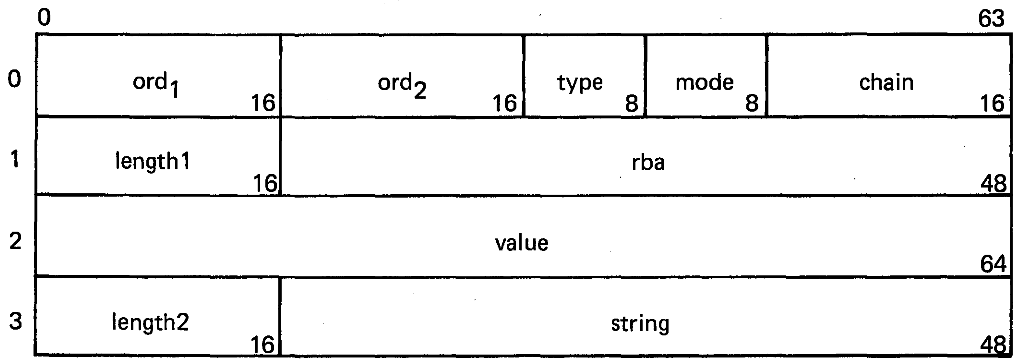


Figure 10-6. Data Item Format 1 (Sheet 1 of 2)

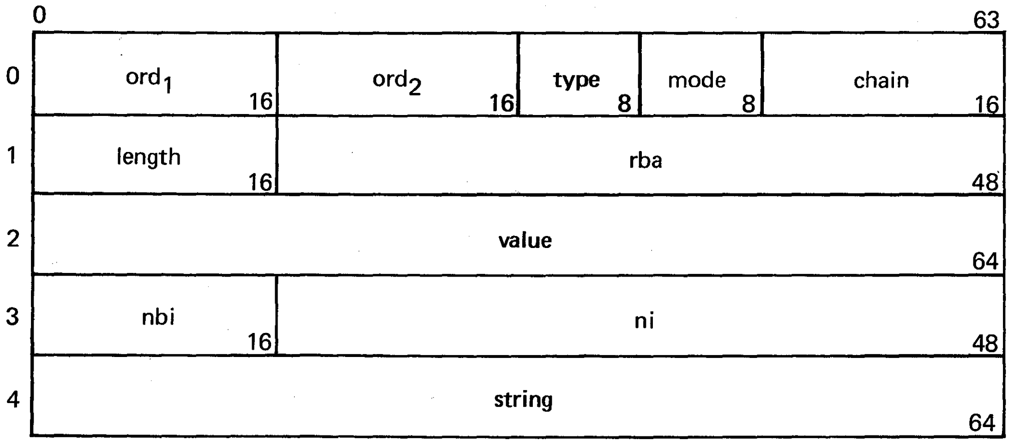
Word	Field	Description
0	ord_1	Pseudoaddress vector table ordinal of static space to be initialized.
	ord_2	Pseudoaddress vector table ordinal of space relative to which relocation is to be performed (relocation base).
	type	Type of data item that follows: 1 Full-word broadcast. 2 Half-word broadcast (not defined if the mode field is 1). 3 Full-word vector transmit. 4 Half-word vector transmit (not defined if the mode field is 1). 9 Byte string. A Bit string.
	mode	Mode flag: 0 Value to destination. 1 Value plus relocation base to destination. 2 Destination plus relocation base to destination. When the mode flag is 0, the values in the item are stored directly into the destination field (specified by the ord_1 field), and the ord_2 field is ignored. When the mode flag is 1, the relocation base is added to the values before they are stored in the destination field; for this case, the result is always on a word boundary. When the mode flag is 2, the relocation base is added to the destination field; in this case, the value field is absent in the data item.
	chain	Full-word count to the next data item descriptor in the table (same as a count of the number of full words in the data item).
1	length	The length of the vector in words for data item types 1, 2, 3, and 4; the number of bytes of information in the value field, for data item type 9; and the number of bits of information in the value field, for data item type A.
	rba	Relative bit address.
2	value	A string or a vector value, depending on the data item type, as follows: • A full word to be stored in consecutive full words, starting at the relative bit address in the rba field (type 1). • A left-justified half-word to be stored in consecutive half-word locations, starting at the relative bit address in the rba field (type 2). • A full-word vector to be transmitted to the relative bit address in the rba field (type 3). • A half-word vector to be transmitted to the relative bit address in the rba field (type 4). • A left-justified byte string to be stored at the address in the rba field (type 9). • A left-justified bit string to be stored at the address in the rba field (type A).

Figure 10-6. Data Item Format 1 (Sheet 2 of 2)



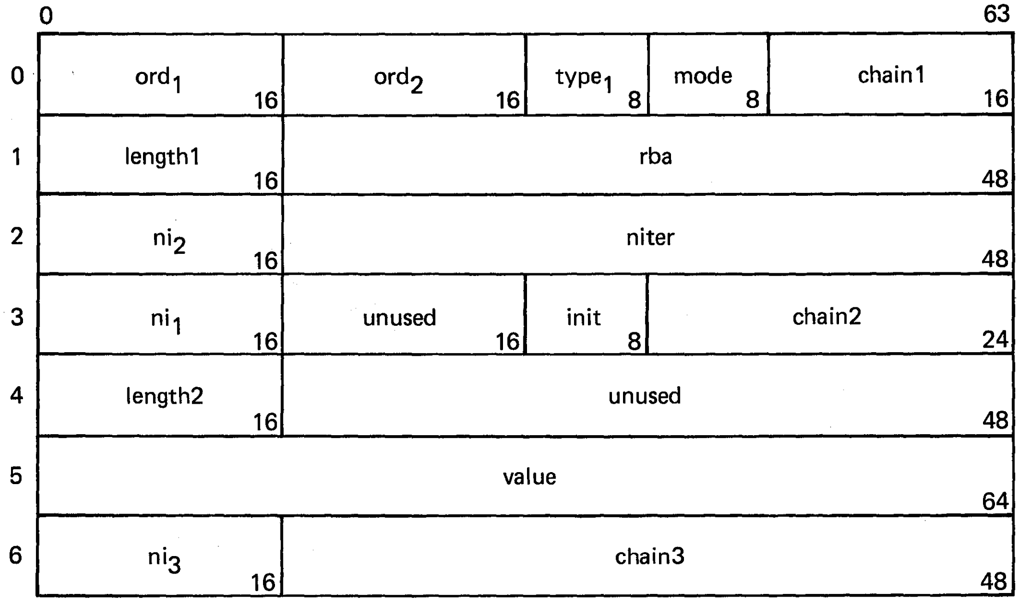
Word	Field	Description
0	ord_1	Pseudoaddress vector table ordinal of static space to be initialized.
	ord_2	Pseudoaddress vector table ordinal of space relative to which relocation is to be performed (relocation base).
	type	Type of data item that follows: 5 Full-word sparse vector. 6 Half-word sparse vector (not defined if the mode field is 1).
	mode	Mode flag: 0 Value to destination. 1 Value plus relocation base to destination. 2 Destination plus relocation base to destination. The meaning of this field is the same as that of the mode field in figure 10-6.
	chain	Full-word count to the next data item descriptor in the table (same as a count of the number of full words in the data item).
1	length1	Number of 1-bits in the order vector specified in the string field.
	rba	Relative bit address of the location to which the sparse vector is to be transmitted.
2	value	Value part of the vector to be transmitted; contains a full-word of values (type 5), or a left-justified, half-word of values (type 6).
3	length2	Length of the control vector specified in the string field.
	string	A left-justified bit control vector (an order vector).

Figure 10-7. Data Item Format 2



Word	Field	Description
0	ordl	Pseudoaddress vector table ordinal of static space to be initialized.
	ord$_2$	Pseudoaddress vector table ordinal of space relative to which relocation is to be performed (relocation base).
	type	Type of data item that follows: 7 Full-word index list. 8 Half-word index list (not defined if the mode field is 1).
	mode	Mode flag: 0 Value to destination. 1 Value plus relocation base to destination. 2 Destination plus relocation base to destination. The meaning of this field is the same as that of the mode field in figure 10-6.
	chain	Full-word count to the next data item descriptor in the table.
1	length	Number of values in the item.
	rba	Relative bit address of the location to which the indexed elements of the vector are to be transmitted.
2	value	A vector; contains a full-word of values (type 7), or a left-justified, half-word of values (type 8).
3	nbi	Number of bits per index.
	ni	Number of indexes.
4	string	A bit string of ni indexes. Each index is nbi bits long and contains a full-word count (for type 7), or a half-word count (for type 8).

Figure 10-8. Data Item Format 3



Word	Field	Description
0	ord_1	Pseudoaddress vector table ordinal relative to the data area to be initialized.
	ord_2	Pseudoaddress vector table ordinal of space relative to which relocation is to be performed (relocation base).
	$type_1$	D (nested list).
	mode	Mode flag: 0 Value to destination. 1 Value plus relocation base to destination. 2 Destination plus relocation base to destination. The meaning of this field is the same as that of the mode field in figure 10-6.
	chainl	Full-word count to the next data item in the nested list.
1	lengthl	Number of nested item types that follow.
	rba	Relative bit address of the vector.
2	ni_2	Nested iteration start item.
	niter	Number of times the data item and items associated with this iteration start item are to be repeated.

Figure 10-9. Data Item Format D (Sheet 1 of 2)

Word	Field	Description
3	ni_1	Nested data item identifier.
	init	Any initialization data type. If there is more than one data item in an iteration, types cannot be mixed.
	chain2	Length of the data item in number of words.
4	length2	Half-word vector length.
5	value	A left-justified half-word to be stored in consecutive half-word locations, starting at the relative bit address in the rba field.
6	ni_3	Nested iteration end item.
	chain3	Nested item designator: 0 No nested item types follow. 1 More nested item types follow.

Figure 10-9. Data Item Format D (Sheet 2 of 2)

INTERPRETIVE RELOCATION INITIALIZATION TABLE

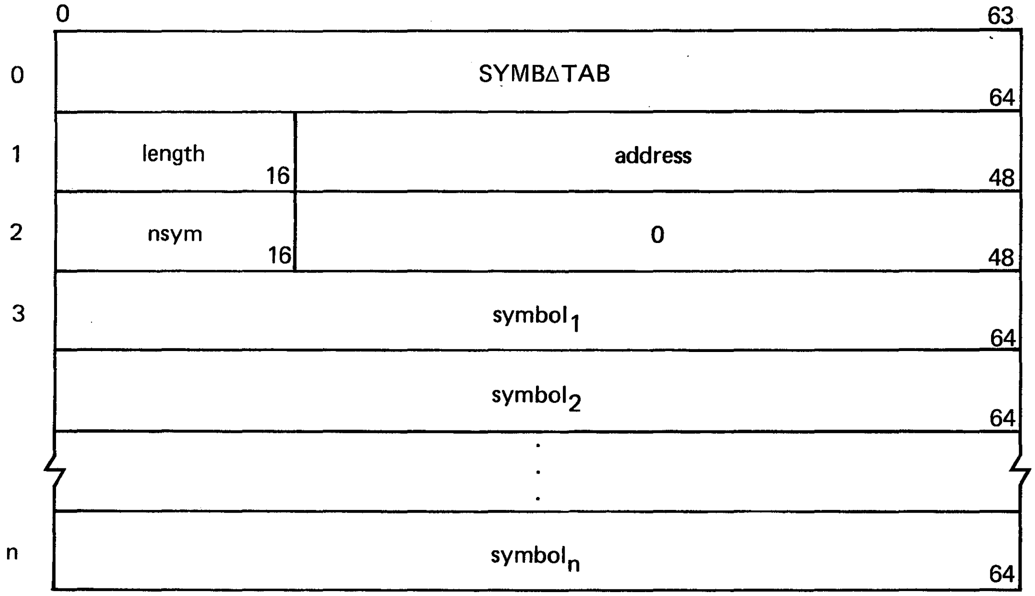
The interpretive relocation initialization table consists of the two-word loader table header (figure 10-1), followed immediately by one or more relocation items, one word per item. Item formats are similar to data initialization table formats but do not contain values. The name of the table is INT Δ RELO.

TRANSFER SYMBOL TABLE

The transfer symbol table consists of the two-word loader table header (figure 10-1), followed immediately by one word containing the transfer symbol. The table name is XFER SYM. The transfer symbol is the symbolic name of the entry point to which control is to be transferred at the start of execution; the name is left-justified with blank fill.

DEBUG SYMBOL TABLE

The debug symbol table, which contains the ASCII representation of symbols that appear in a program, allows a symbol to be referenced by name rather than by address. This table appears in the error processing information area if the compiler or assembler used is capable of generating the table, and if the appropriate option is selected and used during compilation or assembly. The format of the table is shown in figure 10-10. The length field in word 1 is the total length of the debug symbol table and the symbol definition table.

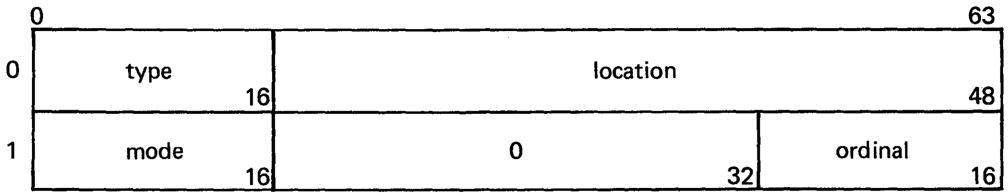


Word	Field	Description
2	nsym	Number of symbols in this table.
3	symbol$_i$	A symbol, which can be any of the following: • Variable or array name, in ASCII; must be left-justified with blank fill. • Statement line number, in ASCII; must be a hexadecimal value, right-justified with binary zero fill. • Statement label, in ASCII. Labels that are symbolic names are stored left-justified with blank fill; labels that are statement numbers are stored right-justified with ASCII zero fill. A statement line number of #FFFF is used to indicate code moved out of logical position by extended basic block optimization (EBBO).

Figure 10-10. Debug Symbol Table Format

SYMBOL DEFINITION TABLE

The symbol definition table is an extension to the debug symbol table. It provides further definition to the debugging symbols, including the type of symbol, address, and mode. The table consists of the two-word loader table header (figure 10-1), followed immediately by one or more two-word entries in the format shown in figure 10-11. The table name is SYMB△DEF.



Word	Field	Description
0	type	Symbol type: type=0 Unknown. type=1 Half-word register variable name. type=2 Variable or array name. type=3 Variable or array name. type=4 Line number. type=5 Label.
	location	Location field for the symbol type: 1 Half-word address within the register file; because half-word values can be stored in full-word registers, the location can range up to hexadecimal 1FF. 2 Full-word register number. 3 Bit address relative to the start of the data base. 4 Bit address relative to the start of the code base. 5 Bit address relative to the start of the code base.
1	mode	Symbol mode, consisting of three parts as shown below; in the case of a descriptor, p and dtype describe the contents of the reference vector:

Subfield	Description
p	Precision base indicator: 0 Precision base is 32-bit or irrelevant. 1 Precision base is 64-bit.

Figure 10-11. Symbol Definition Table Entry Format (Sheet 1 of 2)

Word	Field	Description	
1	mode	Subfield	Description
		desc	Descriptor indicator: 0 Not a descriptor. 1 Vector descriptor. 2 Vector descriptor array. 4 Sparse vector descriptor. 5 Sparse vector descriptor array.
		dtype	Type of the referenced vector: 0 Unknown. 1 Logical. 2 Integer. 3 Real. 4 Double precision. 5 Complex. 6 Character. 10 Bit.
	ordinal	Ordinal of the pseudoaddress vector table of the data base or common block (described under Pseudoaddress Vector Table, next).	

Figure 10-11. Symbol Definition Table Entry Format (Sheet 2 of 2)

PSEUDOADDRESS VECTOR TABLE

The table pointed to by the ordinal in the symbol definition table is the pseudoaddress vector table of the data base or common block. The table consists of the two-word loader table header (figure 10-1), followed immediately by two words giving a code address and data base address, and one or more two word entries in any of the formats shown in figure 10-12. The table name is△△ PAV△△△.

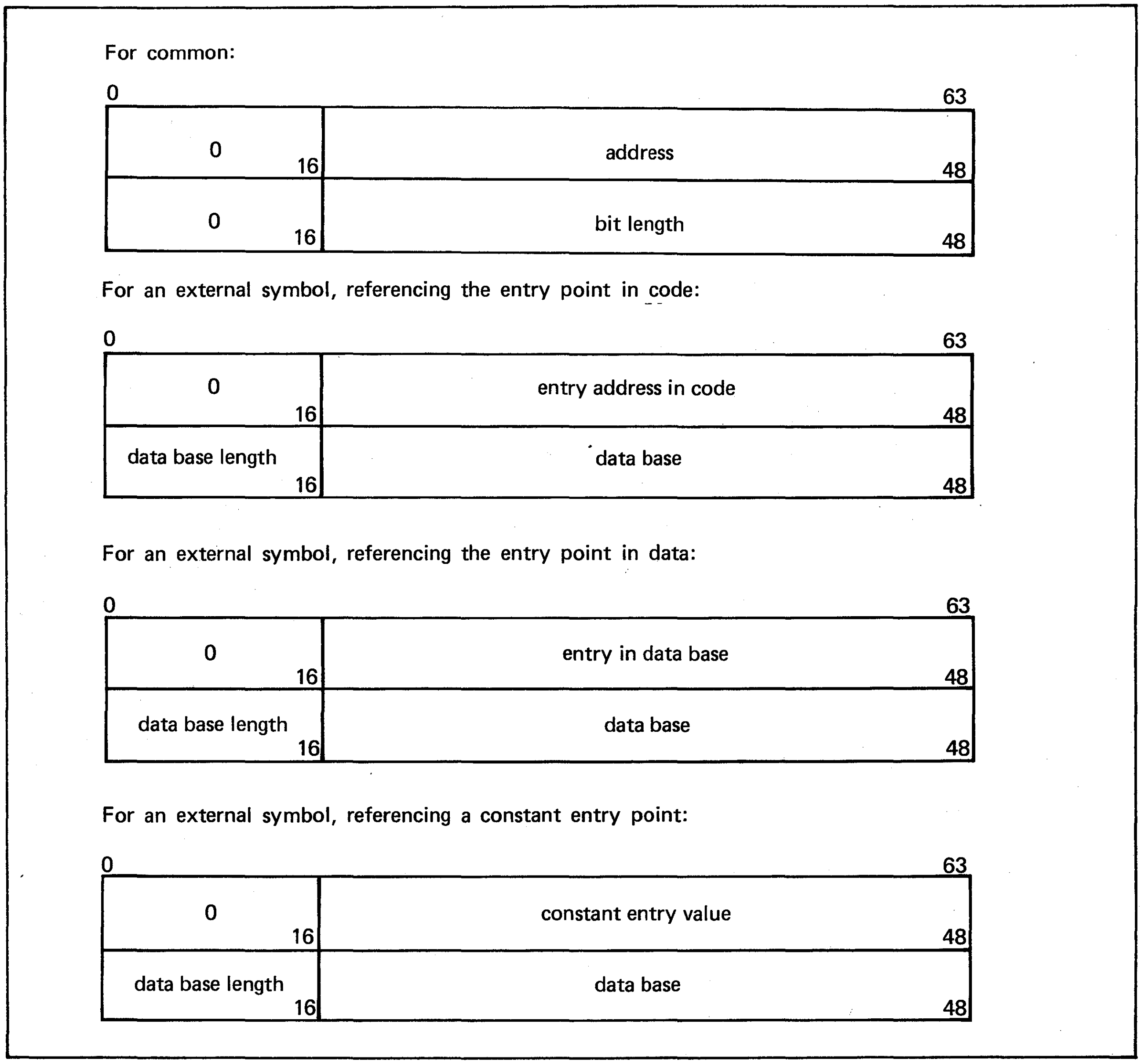


Figure 10-12. Pseudoaddress Vector Table Entry Formats

CHARACTER SET A

The ASCII character set is shown in table A-1. Aids for hexadecimal-to-octal and hexadecimal-to-decimal conversion are given in tables A-2 and A-3.

Table A-1. ASCII Character Set with Punched Card Codes and EBCDIC Translation

	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
b4 b3 b2 b1	COL ROW	0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	SP no-punch SP 40	0 0 0 F0	@ 8-4 @ 7C	P 11-7 P D7	` 8-1 ` 79	p 12-11-7 p 97	11-0-9-8-1 DS 20	12-11-0-9-8-1 30	12-0-9-1 41	12-11-9-8 58	12-11-0-9-6 76	12-11-8-7 9F	12-11-0-8 B8	12-11-9-8-4 DC
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	! 12-8-7 | 4F	1 1 1 F1	A 12-1 A C1	Q 11-8 Q D8	a 12-0-1 a 81	q 12-11-8 q 98	0-9-1 SOS 21	9-1 31	12-0-9-2 42	11-8-1 59	12-11-0-9-7 77	11-0-8-1 A0	12-11-0-9 B9	12-11-9-8-5 DD
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	" 8-7 " 7F	2 2 2 F2	B 12-2 B C2	R 11-9 R D9	b 12-0-2 b 82	r 12-11-9 r 99	0-9-2 FS 22	11-9-8-2 CC 1A	12-0-9-3 43	11-0-9-2 62	12-11-0-9-8 78	11-0-8-2 AA	12-11-0-8-2 BA	12-11-9-8-6 DE
0 0 1 1	3	ETX 12-9-3 ETX 03	DC3 11-9-3 TM 13	# 8-3 # 7B	3 3 3 F3	C 12-3 C C3	S 0-2 S E2	c 12-0-3 c 83	s 11-0-2 s A2	0-9-3 23	9-3 33	12-0-9-4 44	11-0-9-3 63	12-0-8-1 80	11-0-8-3 AB	12-11-0-8-3 BB	12-11-9-8-7 DF
0 1 0 0	4	EOT 9-7 EOT 37	DC4 9-8-4 DC4 3C	$ 11-8-3 $ 5B	4 4 4 F4	D 12-4 D C4	T 0-3 T E3	d 12-0-4 d 84	t 11-0-3 t A3	0-9-4 BYP 24	9-4 PN 34	12-0-9-5 45	11-0-9-4 64	12-0-8-2 8A	11-0-8-4 AC	12-11-0-8-4 BC	11-0-9-8-2 EA
0 1 0 1	5	ENQ 0-9-8-5 ENQ 2D	NAK 9-8-5 NAK 3D	% 0-8-4 % 6C	5 5 5 F5	E 12-5 E C5	U 0-4 U E4	e 12-0-5 e 85	u 11-0-4 u A4	11-9-5 NL 15	9-5 RS 35	12-0-9-6 46	11-0-9-5 65	12-0-8-3 8B	11-0-8-5 AD	12-11-0-8-5 BD	11-0-9-8-3 EB
0 1 1 0	6	ACK 0-9-8-6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 6 F6	F 12-6 F C6	V 0-5 V E5	f 12-0-6 f 86	v 11-0-5 v A5	12-9-6 LC 06	9-6 UC 36	12-0-9-7 47	11-0-9-6 66	12-0-8-4 8C	11-0-8-6 AE	12-11-0-8-6 BE	11-0-9-8-4 ⑁ EC
0 1 1 1	7	BEL 0-9-8-7 BEL 2F	ETB 0-9-6 ETB 26	' 8-5 ' 7D	7 7 7 F7	G 12-7 G C7	W 0-6 W E6	g 12-0-7 g 87	w 11-0-6 w A6	11-9-7 IL 17	12-9-8 GE 08	12-0-9-8 48	11-0-9-7 67	12-0-8-5 8D	11-0-8-7 AF	12-11-0-8-7 BF	11-0-9-8-5 ED
1 0 0 0	8	BS 11-9-6 BS 16	CAN 11-9-8 CAN 18	(12-8-5 (4D	8 8 8 F8	H 12-8 H C8	X 0-7 X E7	h 12-0-8 h 88	x 11-0-7 x A7	0-9-8 28	9-8 38	12-8-1 49	11-0-9-8 68	12-0-8-6 8E	12-11-0-8-1 B0	12-0-9-8-2 CA	11-0-9-8-6 EE
1 0 0 1	9	HT 12-9-5 HT 05	EM 11-9-8-1 EM 19	) 11-8-5) 5D	9 9 9 F9	I 12-9 I C9	Y 0-8 Y E8	i 12-0-9 i 89	y 11-0-8 y A8	0-9-8-1 29	9-8-1 39	12-11-9-1 51	0-8-1 69	12-0-8-7 8F	12-11-0-1 B1	12-0-9-8-3 CB	11-0-9-8-7 EF
1 0 1 0	10 (A)	LF 0-9-5 LF 25	SUB 9-8-7 SUB 3F	* 11-8-4 * 5C	: 8-2 : 7A	J 11-1 J D1	Z 0-9 Z E9	j 12-11-1 j 91	z 11-0-9 z A9	0-9-8-2 SM 2A	9-8-2 3A	12-11-9-2 52	12-11-0 70	12-11-8-1 90	12-11-0-2 B2	12-0-9-8-4 ⑀ CC	12-11-0-9-8-2 |(LVM) FA
1 0 1 1	11 (B)	VT 12-9-8-3 VT 0B	ESC 0-9-7 ESC 27	+ 12-8-6 + 4E	; 11-8-6 ; 5E	K 11-2 K D2	[12-8-2 ¢ 4A	k 12-11-2 k 92	{ 12-0 C0	0-9-8-3 CU2 2B	9-8-3 CU3 3B	12-11-9-3 53	12-11-0-9-1 71	12-11-8-2 9A	12-11-0-3 B3	12-0-9-8-5 CD	12-11-0-9-8-3 FB
1 1 0 0	12 (C)	FF 12-9-8-4 FF 0C	FS 11-9-8-4 IFS 1C	, 0-8-3 , 6B	< 12-8-4 < 4C	L 11-3 L D3	\ 0-8-2 \ E0	l 12-11-3 l 93	¦ 12-11 ¦ 6A	0-9-8-4 2C	12-9-4 PF 04	12-11-9-4 54	12-11-0-9-2 72	12-11-8-3 9B	12-11-0-4 B4	12-0-9-8-6 ⑂ CE	12-11-0-9-8-4 FC
1 1 0 1	13 (D)	CR 12-9-8-5 CR 0D	GS 11-9-8-5 IGS 1D	– 11 – 60	= 8-6 = 7E	M 11-4 M D4	] 11-8-2 ! 5A	m 12-11-4 m 94	} 11-0 D0	12-9-8-1 RLF 09	11-9-4 RES 14	12-11-9-5 55	12-11-0-9-3 73	12-11-8-4 9C	12-11-0-5 B5	12-0-9-8-7 CF	12-11-0-9-8-5 FD
1 1 1 0	14 (E)	SO 12-9-8-6 SO 0E	RS 11-9-8-6 IRS 1E	. 12-8-3 . 4B	> 0-8-6 > 6E	N 11-5 N D5	^ 11-8-7 ¬ 5F	n 12-11-5 n 95	~ 11-0-1 ~ A1	12-9-8-2 SMM 0A	9-8-6 3E	12-11-9-6 56	12-11-0-9-4 74	12-11-8-5 9D	12-11-0-6 B6	12-11-9-8-2 DA	12-11-0-9-8-6 FE
1 1 1 1	15 (F)	SI 12-9-8-7 SI 0F	US 11-9-8-7 IUS 1F	/ 0-1 / 61	? 0-8-7 ? 6F	O 11-6 O D6	_ 0-8-5 _ 6D	o 12-11-6 o 96	DEL 12-9-7 DEL 07	11-9-8-3 CU1 1B	11-0-9-1 E1	12-11-9-7 57	12-11-0-9-5 75	12-11-8-6 9E	12-11-0-7 B7	12-11-9-8-3 DB	EO 12-11-0-9-8-7 FF

64-Character ASCII Subset

96-Character ASCII Subset

LEGEND

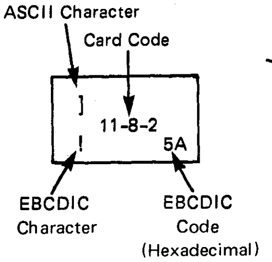


Table A-2. Hexadecimal-to-Octal Conversion Aids

		First Hexadecimal Digit															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Second Hexadecimal Digit	0	000	020	040	060	100	120	140	160	200	220	240	260	300	320	340	360
	1	001	021	041	061	101	121	141	161	201	221	241	261	301	321	341	361
	2	002	022	042	062	102	122	142	162	202	222	242	262	302	322	342	362
	3	003	023	043	063	103	123	143	163	203	223	243	263	303	323	343	363
	4	004	024	044	064	104	124	144	164	204	224	244	264	304	324	344	364
	5	005	025	045	065	105	125	145	165	205	225	245	265	305	325	345	365
	6	006	026	046	066	106	126	146	166	206	226	246	266	306	326	346	366
	7	007	027	047	067	107	127	147	167	207	227	247	267	307	327	347	367
	8	010	030	050	070	110	130	150	170	210	230	250	270	310	330	350	370
	9	011	031	051	071	111	131	151	171	211	231	251	271	311	331	351	371
	A	012	032	052	072	112	132	152	172	212	232	252	272	312	332	352	372
	B	013	033	053	073	113	133	153	173	213	233	253	273	313	333	353	373
	C	014	034	054	074	114	134	154	174	214	234	254	274	314	334	354	374
	D	015	035	055	075	115	135	155	175	215	235	255	275	315	335	355	375
	E	016	036	056	076	116	136	156	176	216	236	256	276	316	336	356	376
	F	017	037	057	077	117	137	157	177	217	237	257	277	317	337	357	377
Octal		000 –	037	040 –	077	100 –	137	140 –	177	200 –	237	240 –	277	300 –	337	340 –	377

Table A-3. Hexadecimal-to-Decimal Conversion Aids

		Exponent for Base 16					
		5	4	3	2	1	0
Hexadecimal Number	0	0	0	0	0	0	0
	1	1048576	65536	4096	256	16	1
	2	2097152	131072	8192	512	32	2
	3	3145728	196608	12288	768	48	3
	4	4194304	262144	16384	1024	64	4
	5	5242880	327680	20480	1280	80	5
	6	6291456	393216	24576	1536	96	6
	7	7340032	458752	28672	1792	112	7
	8	8388608	524288	32768	2048	128	8
	9	9437184	589824	36864	2304	144	9
	A	10485760	655360	40960	2560	160	10
	B	11534336	720896	45056	2816	176	11
	C	12582912	786432	49152	3072	192	12
	D	13631488	851968	53248	3328	208	13
	E	14680064	917504	57344	3584	224	14
	F	15728640	983040	61440	3840	240	15

$$\begin{array}{c|c} & i \\ \hline j & m \end{array}$$

$j_{16} \times 16^{i} = m_{10}$

To find $E_{16} \times 16^{3}$; look at row E, column 3 and find 57344

DIAGNOSTICS B

This appendix describes the meanings of the system error codes and tape error codes. Privileged system task error codes are documented in the VSOS 2 Operator's Guide. System dead codes and NAD disaster codes are now documented in the VSOS Troubleshooting Guide. RHF connect reject codes are documented in the RHF Application-to-Application Interface Specification.

SYSTEM ERROR CODES

The error codes listed in table B-1 are returned in word #8B of the minus page. The errors signified by the codes terminate the task that generated the error. The KERNEL, RESTART, and RECOVERY tasks are part of the resident operating system; the AOK and SCAT tasks are part of the virtual system.

Table B-1. System Error Codes (Sheet 1 of 2)

Hexadecimal Code	Significance	Issued By
5	Illegal instruction; the instruction is not in the CYBER 200 instruction set.	KERNEL
6	The exit force instruction does not have a pointer to a system message to be executed.	KERNEL
7	Illegal request.	KERNEL
8	Parity error in data transfer between the CPU and central memory.	RESTART
A	A C50x request did not contain a file segment table ordinal.	Resident system
B	Illegal C504 request.	Resident system
22	Disk I/O error occurred for read/write of a drop file.	PAGER
24	Large page limit exceeded.	PAGER
25	Page size conflict in drop file.	PAGER
26	Virtual address duplicate direct fault.	PAGER
27	Write violation in system call.	PAGER

Table B-1. System Error Codes (Sheet 2 of 2)

Hexadecimal Code	Significance	Issued By
28	Write violation occurred while the system was swapping in a page referenced by the job.	PAGER
29	The job referenced a page within the virtual system address range.	PAGER
2A	The drop file map is full; the job can define no more vitual regions.	PAGER
2B	This job class is not allowed large pages.	Virtual system
2C	The job referenced a page in the shared library reserved area.	PAGER
2D	Drop file space overflow; no more virtual space can be mapped into the drop file.	PAGER
2E	Page was not mapped because the drop file map is full.	PAGER
2F	Drop file overflow was caused by a call to the virtual system.	PAGER
30	No time available for this task.	PAGER
31	The paging routine received an I/O error.	PAGER
40	Bound implicit map anomaly.	Virtual system
51	File segment table is full.	GETSEG
209	No source file exists.	AOK
210	No drop file exists.	AOK
212	The pointer to the system message Alpha does not exist.	SCAT
213	The pointer to the system message Alpha was out of bounds.	SCAT
215	No error exit address exists.	SCAT
Bxx	File is already extended to maximum. xx is an I/O connector number.	GETSEG
Cxx	Attempted to read past the end of file on a file. xx is an I/O connector number.	REXTEND
Dxx	No segment space in FILEI left for extension. xx is an I/O connector number.	GETSEG
Exx	No space left on the disk for extension. xx is an I/O connector number.	GETSEG

TAPE ERROR CODES

The system returns a tape error to the caller in the ioer field of the call. The errors that range from 1 to 100 return control to the caller when one of these errors is detected. The errors that range from 101 to 200 are tape I/O errors. These errors can be fatal or require operator action unless the caller selected user error processing in the OPEN system message. The codes listed in table B-2 are in decimal notation.

Table B-2. Tape Error Codes (Sheet 1 of 4)

Tape Errors	
Code	Significance
001	Call not in user range.
002	Illegal subfunction code (sfnc).
003	Nonexistent I/O connector (ioc).
004	Buffer size greater than 48 pages.
005	Tried to write zero-length logical tape record (V tape format).
007	PRU read is longer than MPRU. Device capacity exceeded.
008	LRU is greater than MPRU.
009	WRITE attempted a zero-length PRU.
010	User WRITE buffer went minus.
011	HDR1 label not in label buffer.
012	Non-numeric file sequence number.
013	Section 1 is not in VSN list.
014	Cannot swap backwards, no previous VSN.
016	File accessibility characters do not match.
017	Position not found in multifile set.
019	Next VSN was not given.
020	Tape file does not have proper access.
021	Read or skip forward after write (illogical sequence).
030	Attempt to reuse call before previous call is complete.
031	Previous call for this unit had a fatal error.
032	Call crosses page boundary.
033	IOC is not for a tape file.
034	Tape not assigned to this user.
037	For write operation, sum of LRU sizes is greater than buffer length.

Table B-2. Tape Error Codes (Sheet 2 of 4)

Tape Errors	
Code	Significance
039	Forward motion attempted when end of information has been detected on this file.
040	End of tape encountered. (This number is returned to the user only if the user selected end-of-tape processing in the Q5OPEN call.)
041	Load point encountered on tape from backward motion.
042	Tape format mismatch.
043	EOI encountered while positioning to HDR1.
044	Illegal user labels in label buffer.
045	Small and large pages exist in the buffer.
046	System tables full, try again.
047	I/O request currently outstanding for this buffer.
048	Length of LRU array less than or greater than 255.
049	Attempted to write over unexpired label.
050	Buffer size smaller than MPRU for read data function.
051	Tried to write two consecutive tape marks.
052	Data in LRU array after end of group.
053	Buffer length is less than MPRU.
054	Tape unit not assigned to any user.
055	All hardware paths to tape unit are down.

Tape I/O Errors	
Code	Significance
101	Tape that is unlabeled should be labeled.
102	Tape that is labeled should be unlabeled.
105	Write parity error irrecoverable.
106	Unrecognizable label group.
107	Header label fields do not match.
108	Record fragment encountered.
109	ATS software error.
110	Unexpected load point detected.
111	Read parity error unrecoverable.

Table B-2. Tape Error Codes (Sheet 3 of 4)

Tape I/O Errors	
Code	**Significance**
112	Unrecognizable trailer label.
113	Cannot read label group.
115	ATS hardware error, see hardware status.
116	Position uncertain, ready dropped.
117	Unrecoverable erase parity error.
118	Unrecoverable tape mark parity error.
120	Unit reserved by other controller.
122	Tape mark write verify failure.
123	Blank tape encountered during read.
125	Tape repositioning error; block ID mismatch.
126	Tape repositioning error; invalid block ID.
128	Channel malfunction I/O suspended by driver.
129	Multifile position uncertain.
131	Dev ID burst fault. Remount on any unit.
132	Dev tape cleaner fault. Remount on any unit.
135	Tape unit switched offline.
136	No write enable ring in reel.
137	Controller not capable of requested density.
138	Unexpected error returned by ATS controller.
139	Software interface error between NADs.
140	TAD hardware error.
141	Write verify error.
142	Unit remained busy after rewind.
143	Unit dropped ready during rewind.
145	Illegal user level number.

Table B-2. Tape Error Codes (Sheet 4 of 4)

Tape I/O Errors	
Code	Significance
146	Label reposition error.
147	Unit reset status active, position uncertain.
148	Tape label not multiple of 80 characters.
149	Unit is not ready at reserve time.
150	No file mark after EOF1.
151	Missed file mark.
152	No label block after file mark.
153	VOL1 not detected after load point.
160	Encountered two tape marks in reverse.
170	No current block count given for fund position.
171	No file mark or load point on latest block ID.
172	No label found on labeled tape.
173	Cannot find position in find position.
174	Tape mark encountered, position found.
175	Compare count is over block ID count.

GLOSSARY C

Access

A parameter that specifies the read, write, append, modify, and/or execute access desired for a file at the time the file is opened or created. The system grants access only if the appropriate field of the file index table allows such access.

Account Block

The amount of system resources accumulated per charge number.

Account Identifier

One to eight characters indicating who is to be charged for system resource usage attributable to a user number.

ATC

Abnormal termination control.

ATS

Advanced tape system.

Batch Dayfile

A file produced by the batch processor for a batch job that gives a history of the job. Information on the file includes the time various control statements began execution and any error or status information produced by system utilities. The dayfile is printed as the last part of job output.

Batch Job

A series of tasks that is executed as controllees of the batch processor.

Batch Processor

A system utility that initiates and controls batch jobs. Control statements that are file names cause the files named to be executed as controllees of the batch processor. Other control statements result in actions taken by the batch processor alone.

Block

A contiguous 512-word quantity starting on an even 512-word boundary. The block is the unit used for expressing file and memory lengths.

Bound Implicit Map

Part of the minus page of an executing file that relates virtual addresses with physical mass storage addresses.

Byte

A sequence of 8 bits that is a subdivision of a word and represents a single character.

CAT

Currently active table (T_CAT) used by RHF processing. This is a virtual system table.

Central Processing Unit (CPU)

The computational facility of VSOS.

Charge Number

Combination of the account identifier and project number that is to be charged for system resources.

Checkpoint

A system feature that captures a task and any of its controllees at some point in execution so that the task can be restarted from that point. In a FORTRAN program, checkpoint is called by the file name CHKPNT.

Controllee

A task called into execution by a controller.

Controllee Chain

A linked series of tasks that results when one task brings another task into execution. That task can, in turn, initiate another task. As many as nine levels of tasks can be involved. The highest level is level 1; the lowest is level 9.

The tasks in the chain are not run concurrently. When a controller starts a controllee, the controller is suspended until the controllee returns control to it.

Controllee File

Refer to Virtual Code File.

Controller

A task that produces another task.

A relative term that indicates that a member of a controllee chain that has a controllee task attached. A controller might be a controllee of another task. The batch processor is one controller that has no controller (that is, a level-1 task).

CPU

See Central Processing Unit.

CRT

Currently running table (T_CRT) used by RHF processing. This is a virtual system table.

DB

Descriptor block table.

Data Base

The constants and variables used by a routine, not including entities declared to be in common.

Default Project Number

A project number that is assigned to a user number as default. Whenever a user executes a job or interactive session, the system resources accumulated will be charged to the default project number if in existence, unless the user supplies a charge number within the job or interactive session.

Descriptor Block Number

A unique number associated with the program until it terminates or is disconnected. This number is the key link between the operating system and an executing program.

DFBM

Data flag branch manager.

Drop File

A file created by the system for modified pages of an executing task, free space, and write-temporary files.

Drop file names are formed by the system shifting the controllee file name right one character and prefixing it with a digit that identifies the level (1 through 9) in a controllee chain.

Drop File Map

Part of the minus page of an executing file that relates virtual addresses with physical mass storage addresses. An entry is made in the drop file map every time a free-space reference is made by the executing code.

Dynamic Stack

The stack that resides in free space. All registers are saved on subroutine calls in the dynamic stack.

EBCDIC

Extended binary coded decimal interchange code.

EOF

End of file.

EOG

End of group.

EOI

End of information.

Epilogue

A set of instructions executed at the exit of a subroutine that restores registers and resets conditions.

ERS

Efficient run size.

Explicit Input/Output

A means of accessing a mass storage or tape file in which data is buffered under program control. Contrast with Implicit Input/Output.

FADE

File access directory entries.

File

A collection of data that can be accessed by file name. In the absence of an adjective such as terminal or tape, all references to files in this manual imply mass storage files.

File Index Table

A system table that holds all information relating to active user´s files and their characteristics.

File Type

A category that defines file structure from a system standpoint. File types are physical, virtual data, and virtual code.

Free Space

Space in memory available for use that gets paged to and from the drop file. The range for free space is #4000 up to #7FFFFFFFFFFF.

FST

File segment table.

Implicit Input/Output

A means of accessing a mass storage file in which the system brings a page of the file into main memory in response to a reference on that page. Contrast with Explicit Input/Output.

Input/Output Connector (IOC)

An entry in a minus page that links a file with a task for input/output purposes.

Invisible Package

A hardware feature that contains the current address and control information for a task.

IOC

See Input/Output Connector.

IQM

Input queue manager.

JDN

Job descriptor number.

JDT

Job descriptor table.

Job

Refer to Batch Job.

Job Block

The amount of resources accumulated for the duration of the job.

Labeled Tape

A magnetic tape with labels conforming to American National Standard X3.27-1978, Magnetic Tape Labels for Information Interchange.

Large Page

128 small pages; 65,536 contiguous words of 64 bits.

Last-Group-File

Identifies the member of an output-file-family which contains disposition information for QTF.

LCN

Loosely coupled network.

Level

Depending on context, can refer to the security level of a file, the level of a file in the controllee chain, the level of a routine involved in an interrrupt, or the level of protocal in RHF. For the first and second meanings, refer to Security Level and Controllee Chain.

With respect to interrupt processing, level 0 refers to the normally executing routine. Level 1 refers to the interrupt routine when it is in execution.

Library

A file of modules, in a format produced by the system utility OLE, that can be used to satisfy external references during loading.

LID

Logical identifier. The name specified by a user to designate a remote host to be accessed through the Remote Host Facility.

Local File

A private file that is destroyed by the system after termination of the batch job or interactive terminal session that creates it.

LRU

Logical record unit.

Main Memory

Memory associated with the central processing unit from which instructions can be executed. Also called MCS.

Map

Refer to Bound Implicit Map or Drop File Map in chapter 2; also MAP system message (chapter 5).

Mass Storage File

A file management category that indicates no special processing after task termination. In a general sense, mass storage indicates disk-resident files, as opposed to magnetic tape or terminal files.

Master Project Number

One to three characters (the first three nonspecial characters of a project number) to be assigned to a mass storage file.

Master User

A user who has been designated to be able to audit any user files with a specific account identifier.

MCS

See Main Memory.

MCU

Maintenance control unit.

MDI

Marginal drive indicator.

Message

Refer to System Message.

Minus Page

The first page of a virtual file used by the system to hold items such as the invisible package, input/output connector information, and maps of defined virtual space. Drop files can also have a second minus page containing overflow input/output connector and map information.

MODDROP

A management category for implicitly opened files that indicates a file is read-only on mass storage. Modifications to the file are retained in the drop file (write-temporary) and do not alter the file image.

NAD

Network access device.

Nonprivileged

User number which does not have the privileged attribute. Refer to Privilege.

Object Code File

A file generated by compilation or assembly of a source language program that can be used by the loader to produce an executable file. Contrast with Virtual Code File.

OLE

System utility that creates and modifies a file in library format or modmerge file format.

Output File

A file management category that indicates a file is destined for print or punch equipment.

Also, a generic term for a file being written, as opposed to an input file being read.

Output-File-Family

A set of files residing on User-6 that was generated as the output of a batch job or as the output of an MFQUEUE.

Ownership

The term for the type of permanent file catalog to which a file belongs. Ownership indicates whether a file belongs to a private user, a pool, or the system (public).

Pack File Index (PFI)

A table of 16-word file index table entries which exists on each pack to control and describe the files located on that pack.

PAD

Pool access directory.

Page

The unit by which main memory is managed; a block of contiguous 512 64-bit words. Can be a large page of 128 blocks or a small page of 1, 4, or 16 blocks.

Page Fault

Reference by virtual address to a page not currently in main memory, causing a program interrupt and paging in.

Paging In

Operation to move a page from auxiliary memory to main memory.

Paging Out

Operation to move a page from main memory to auxiliary memory.

Permanent File

A private file that remains in the system after termination of the batch or interactive session that creates it.

Physical Data File

A file type that indicates a file containing nonexecutable data only.

Physical Memory Address

Address of a page in main memory. Also called physical address.

PID

Physical identifier. The unique name used by the Remote Host Facility to designate an individual host system.

Pool

One mechanism for file sharing on VSOS. A pool is a file set created and maintained by a pool boss. More than one user number can access a pool as determined by the PACCESS request for the pool.

Pool File

An ownership category that indicates a file can be accessed by any privileged task and, after PATTACH, by any task running under a user number the pool boss authorizes by using PACCESS.

PP

Peripheral processor.

Private File

An ownership category that indicates a file can be accessed either by a task running under the user number under which the file is stored, by a privileged user, or by another user who has been given permission by the owner.

Privilege (User)

An attribute granted a user number which allows access to all permanent files in the system and to some operating system functions.

Project Number

1 to 20 alphanumeric characters (including the special characters * and -) indicating to which project, within the account identifier, the system resources are to be charged.

Prologue

A set of instructions executed at the entry to a subroutine that swaps registers and sets initial conditions.

PRU

Physical record unit.

Public Files

Files considered to be system owned. They belong to user number 000000. Public files are accessible to all users.

Register File Block

The second block of a virtual code file which contains register contents when a task is not executing in the CPU.

RHF

Remote Host Facility.

RHFMT

Remote Host Facility mainframe table (T_RHFMT).

RHFT

Remote Host Facility table (T_RHFT).

SAE

Standardized accounting enhancements.

Scratch File

A management category that indicates a file is to be destroyed upon termination of the task that created it.

Security Level

Attribute of a file, task, job, or user number used to prevent unauthorized data access. The eight security levels are numbered 1 through 8, from least to greatest security.

SHRLIB

The area of point F virtual memory reserved for shared library routines (virtual bit address #800000000000 - #BFFFFFFFFFFF).

SIT

System intialization table.

Small Page

One, four, or sixteen blocks, where a block is 512 contiguous 64-bit words.

Source File

A generic term for a file containing information used by a utility or other task whose specific meaning depends on the context of its use: the controllee file associated with a drop file, for instance, is termed the source file.

In an UPDATE utility context, a file produced by UPDATE that would allow recreation of a new program library on a subsequent creation run. In the FORTRAN context, the input program text is called the source.

SPT

System processor table. This is a virtual system table.

System Billing Unit (SBU)

An installation-defined unit used for charging of system resource usage. The unit may incorporate tape use access, number of tape functions, number of disk accesses, number of pages transferred to or from disk, and CPU usage in microseconds, depending on installation parameter settings. An example of SBU is the time in microseconds of CPU use. Refer to System Time Unit.

System Dayfile

A file of all significant events in the system, including user dayfile entries, interactive commands, batch processor errors, privileged system task errors, and login errors.

System Interface Language (SIL)

Set of subroutines callable by user programmers. Each subroutine formats and issues one or more system messages.

System Message

The means by which the operating system and user tasks communicate with each other. System messages are calls to the virtual and resident systems.

System Time Unit (STU)

An installation-defined unit used for allocating system resources. The unit might incorporate tape use/access, number of tape functions, number of disk accesses, number of pages transferred to or from disk, and CPU usage in microseconds. An example of STU is time in microseconds of CPU use. Refer to System Billing Unit.

Task

An executable program.

TTY

Teletypewriter terminal unit.

UAT

User activity table.

UEP

User error processing.

User Number

Six digits that identify a file owner or user of system resources. One task can be in execution for a given user number for each suffix at one time.

User Project Control

A user attribute, if set for a user number, the charge number must be specified for the executing job or the user must have a default project number assigned.

Virtual Address

Address that refers to virtual memory and is translated, through the page table, into a physical address.

Virtual Address Space

The set of virtual addresses that belong to a specific active task.

Virtual Code File

A file type that indicates an executable file having a minus page as its first page and a page 0 as its second page. The file must be created by the loader. A virtual code file is also called a controllee file. Contrast with Object Code File.

Virtual Memory

A concept by which memory can be addressed as if it were as large as needed. The system manages correspondence between the user memory addresses and physical main memory.

Virtual Range

Range of virtual addresses. Same as Virtual Address Space.

VRA

Variable rate accounting.

VRF

Variable rate factor.

VRI

Variable rate index.

VSDT

Virtual system debug tool.

VSN

Volume serial number.

VSOS

Virtual Storage Operating System.

Word

A 64-bit division of main memory or mass storage. Bits are numbered 0 through 63, from left to right (most significant to least significant).

Working Set

Basis for managing the amount of physical memory available to a task. It is the portion of a task´s virtual address space that is referenced most frequently during a window of the execution of the task.

Working Set Size

Number of 512-word blocks in the working set.

Write-Temporary

Refer to MODDROP.

REGISTER FILE CONVENTIONS D

The VSOS assumes some conventions regarding the handling of the register file, an area containing 256 registers numbered from #0 to #FF. Some of the registers are used by the operating system for specific purposes, and others are available solely for the purposes of the user. One register file area of particular importance is the register save area, which is saved and restored each time an external procedure call is made.

REGISTERS

The register file is subdivided into five major areas, as shown in figure D-1. The environment register area and the working register area are jointly referred to as the register save area, registers that are saved on calls to external procedures.

MACHINE REGISTERS

These registers include registers #0, #1, and #2. Register #0 contains machine zero (machine zero is described in the CYBER 200 Computer System Hardware Reference Manual). Registers #1 and #2 are used by Data Flag Branch Manager. When a data flag branch occurs, the hardware sets register #1 to contain the address of the next instruction that would have executed had the data flag branch not occurred. The data flag branch causes transfer of control to the address contained in register #2. This address is set by the user (most likely, a software product such as DEBUG or FORTRAN runtime).

TEMPORARY REGISTERS

A user program can utilize two areas for temporary storage, addresses, or data. The two areas are from register #3 to #13, and from #20 to the end of the register save area.

The lower area (#3 to #13) is large enough for execution of short subroutines that do not call other subroutines (such as SIN and COS) completely within the temporary space, eliminating the need for saving and restoring the register save area when short modules are needed by a program. The upper area (#20 to an upper limit specified by the caller), which is large enough to hold a variety of user procedures, cannot be modified by the callee. If the callee needs to use registers in the range of #20-#FF, it must save and restore the caller´s copies of those registers.

GLOBAL REGISTERS

The contents of the global registers are universal to all programs including VSOS. The contents can be assumed by all modules.

The global registers contents are as follows:

Register	Contents
#14	The constant #20.
#15	The constant #1A.
#16	The constant #1.
#17	The parameter descriptor. The number of parameters being passed during a call is contained in the leftmost 16 bits; the virtual bit address of the parameter list is stored in the rightmost 48 bits. Figure D-2 illustrates how parameters are passed to routines. The parameters are passed by address.
#18, #19	Function results obtained from a called function. For example, the result of a trigonometric or exponential function would be placed in register #18. Register #19 could be used when a result has two components (for example, the imaginary part of a complex number whose real part is returned to register #18).

Registers #14 and #15 are used to swap the register file in/out at prologue/epilogue time.

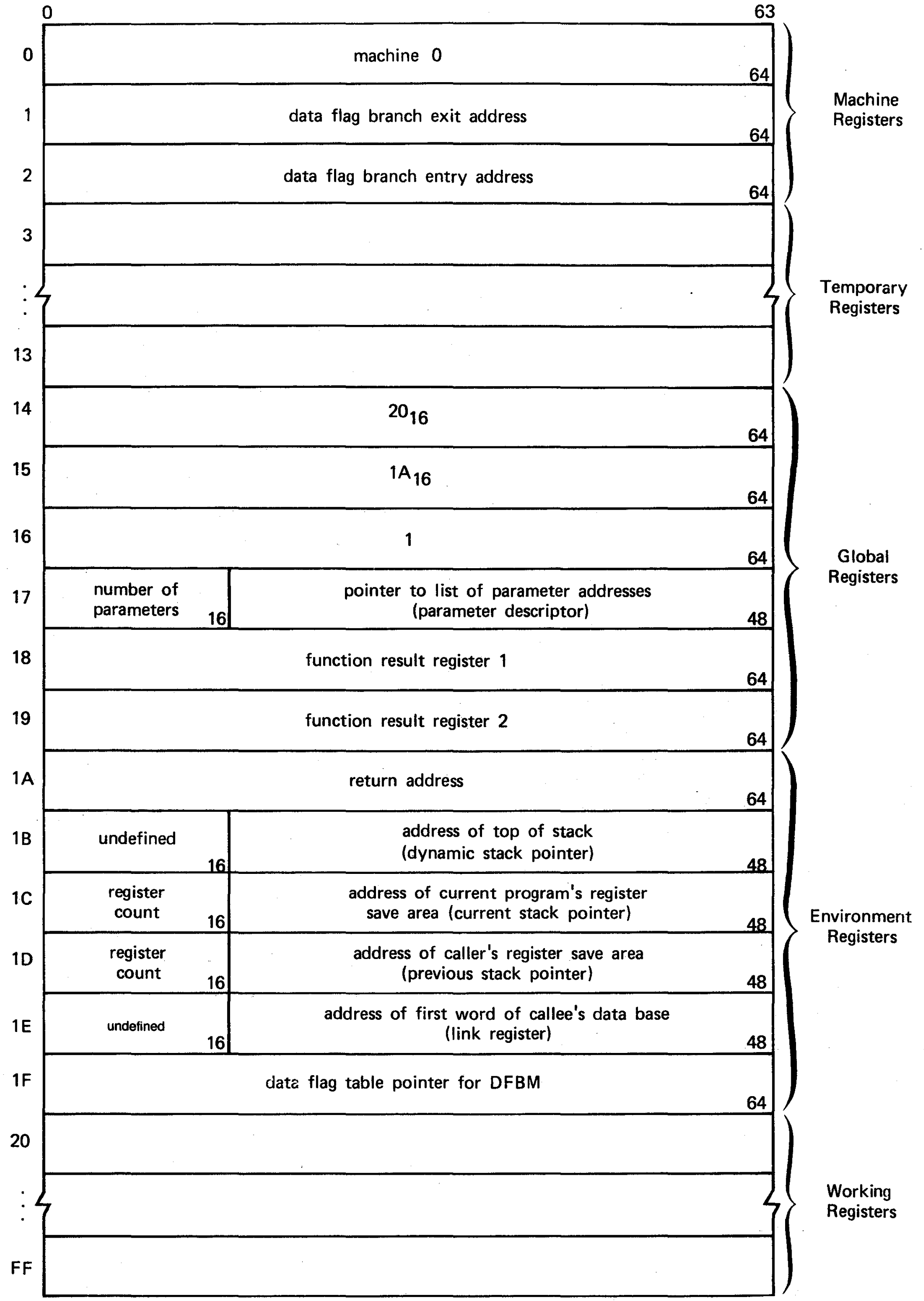


Figure D-1. Register File

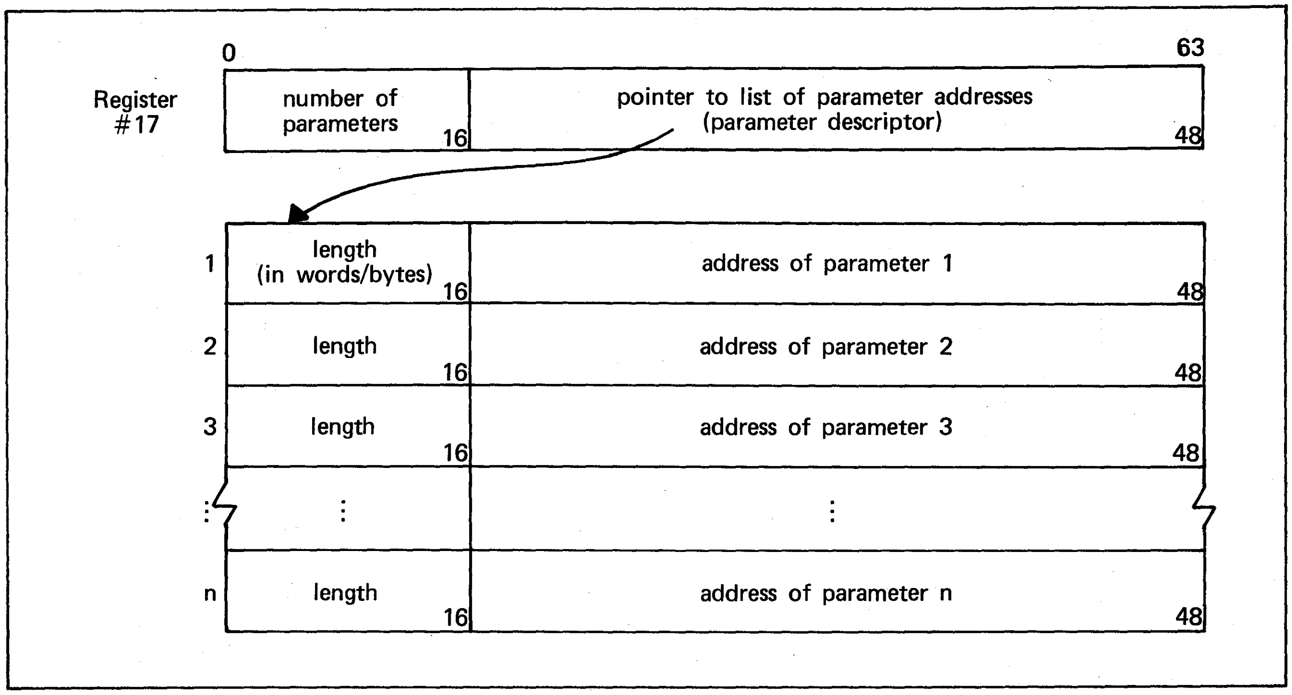


Figure D-2. List of Parameter Addresses

NOTE

Only descriptors or type character parameters have meaningful values in the length field in the parameter list.

If the parameter is an address, then this address is in the parameter list; for example, an array.

ENVIRONMENT REGISTERS

The environment registers consist of the minimum set needed to support the sharing of code in a virtual system and the general requirements of recursive, reentrant execution. These registers, along with a number of working registers, are the register save area. Each time a caller releases control to a callee, a new environment register/working area is established for the callee. A stack structure is used for this. The environment registers include:

Register	Description
#1A	Return register. Contains the virtual bit address of the location in the caller´s program to which the callee´s program normally returns.
#1B	Dynamic stack pointer. Contains the relative bit address of the next available free location in the dynamic stack. It is the caller´s responsibility to leave the address of the dynamic stack pointer on a double-word boundary. The dynamic stack pointer is always advanced prior to storing data into that region or before addresses pointing to that region are calculated.
#1C	Current stack pointer. Contains the length and relative bit base address of the region (the stack frame) in the dynamic stack where a caller wants its registers to be saved. The length of that region is the number of environment registers (6) plus the number of work registers (possibly none) needed for dynamic working storage for the program. Before making an external call, the caller must set the length portion of the current stack pointer to the number of registers to be saved by the callee. The current stack pointer is set by the caller, but it is the callee that establishes the new stack frame. A minimum of six registers must be saved (the number of environment registers).
#1D	Previous stack pointer. Contains the number of registers and the relative bit base address in the register file where the caller´s register save area has been saved. The callee´s previous stack pointer is an exact copy of the caller´s current stack pointer.
#1E	Link register. At subroutine entry contains the virtual bit address of the data base allocated to the module by the loader. The caller passes to the callee the address of the callee´s data base in the link register.
#1F	Pointer to the data flag table for the data flag branch manager (DFBM), for further information refer to the FORTRAN reference manual.

The environment registers are used and modified by program prologues and epilogues. An assembly language programmer must write an appropriate prologue/epilogue. Compilers will automatically generate the necessary prologue/epilogue. Compilers will automatically generate the necessary prologue/epilogue to ensure that the caller´s register save area is saved when an external routine is called.

REGISTER SAVE AREA

The register save area is only those register resident variables that are saved/restored. Many permanent variables/addresses are not register resident, but are memory resident. Nothing must be done to preserve these. When an executing program has called an external program, the instructions of the conventional prologue of the called program save the caller's register save area. (See discussion below about prologues.) The register save area is stored and saved as an element of a conventional chained stack in the register file. A stack element, called a stack frame, is diagrammed in figure D-3.

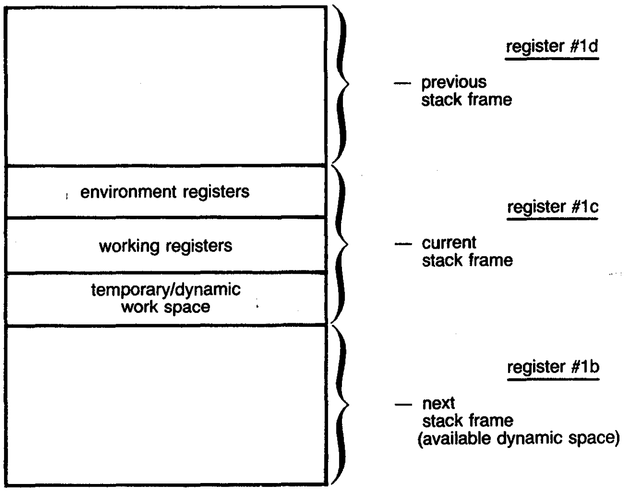


Figure D-3. Stack Frame

The initial size of a frame, defined by the difference of the values of the current stack pointer (the stack frame base) and the dynamic stack pointer, does not include temporary work space. Any time temporary work space is needed, the program can increment the dynamic stack pointer and in this way obtain space. Dynamic space use increases (frames are pushed onto the stack) until the lowest level called program has been executed; then, as the returns are encountered, the space is made available again in reverse order to the calls (frames are popped from the stack).

EXTERNAL PROCEDURE CALL SEQUENCE

The standard sequence of an external procedure call is one of the following:

```
    RTOR          ZZ,#1E               Load data base address.
    BSAVE         #1A,YY               Jump to subroutine.

    LOD           [XX,JJ],YY           Load subroutine address.
    IS            JJ,1                 Enter length
    LOD           [XX,JJ],#1E          Load database address.
    BSAVE         #1A,YY               Jump to subroutine.

    LOD           [XX,JJ],#1E          Load subroutine address.
    BSAVE         #1A,YY               Jump to subroutine.
    STO           [XX,QQ],YY           Store address of subroutine.

    LOD           [XX,JJ],#1E          Load database address.
    ELEN          #1C,PP               Enter length.
    BSAVE         #1A,YY               Jump to subroutine.
    STO           [XX,QQ],YY           Store address of subroutine.
```

Where the register number for YY is one less than the register number for ZZ. YY is the external subroutine address register, ZZ is the external subroutine data base register, and XX is a register containing an address within a data base or common block.

PROLOGUE SEQUENCE

There are basically at least three types of prologues:

1) The traditional prologue approaches the one shown earlier in this chapter in figure D-3 in that it does swap out/in the register file. However, only one swap is used.

2) A zero swap sequence (the FORTRAN compiler may generate such a prologue if optimization is selected) can be used if no registers in the range #1A to #FF are to be used by the callee.

3) An in between sequence whereby stores are used to save a few registers and loads to initialize them for the callee. This is a special case of 1), but may be used for performance reasons instead of using the SWAP instruction.

The prologue of the called procedure includes the following instructions: (type 1)

Instruction	Description
2A1E00xx	Set the number of registers to be loaded in register #1E.
781B0010	Save callers dynamic stack pointer.
781F0012	Save callers data flag table pointer.
781C0011	Save callers current stack pointer.
7D1E151C	Swap: saving caller´s registers, loading callee´s registers.
3E1Bxxxx	Set new dynamic space required.
7811001D	Move callers current stackpointer to callee´s previous stackpointer.
7810001C	Move callers dynamic stackpointer to callee´s current stackpointer.
631B101B	Update dynamic stack pointer.
7812001F	Restore contents of data flag pointer.
2A1C00xx	Set number of registers to be saved on subroutine call.

Another example of a prologue follows:

Instruction	Description
781A0005	Save return address.
781B0006	Save dynamic stack pointer.
781C0007	Save current stack pointer.
781E0008	Save address of callee´s data base.
3E09XXXX	Number of words (xxxx) to be reserved.
63091E0A	Reserved xxxx words to callee´s data base.
2A0A00YY	Set number of registers to be saved (yy).
781F000B	Save data flag table pointer.
7D0A151C	Swap restored yy registers from the callee´s current stack starting with register #1A.
7805001A	Update the return address.
3E0500YY	Set number of registers saved (yy).
7B05061C	Update current stack pointer yy in the length field dynamic stack; pointer´s address becomes current stack pointer´s address.
30050605	Change words (yy) to bits.

Instruction	Description
6305061B	Reserved (yy) bits to dynamic stack pointer.
7808001E	Update callee´s data base.
7807001D	Put current stack pointer to previous stack pointer.
7B000B1F	Update data flag table pointer.
781B0024	Save dynamic stack pointer (temporary register).
78170021	Save parameter descriptor address.

Some programs can perform their subroutines entirely within the temporary registers, and do not make external calls. Such routines need not contain a prologue and can be assembled or compiled to omit it.

EPILOGUE SEQUENCE

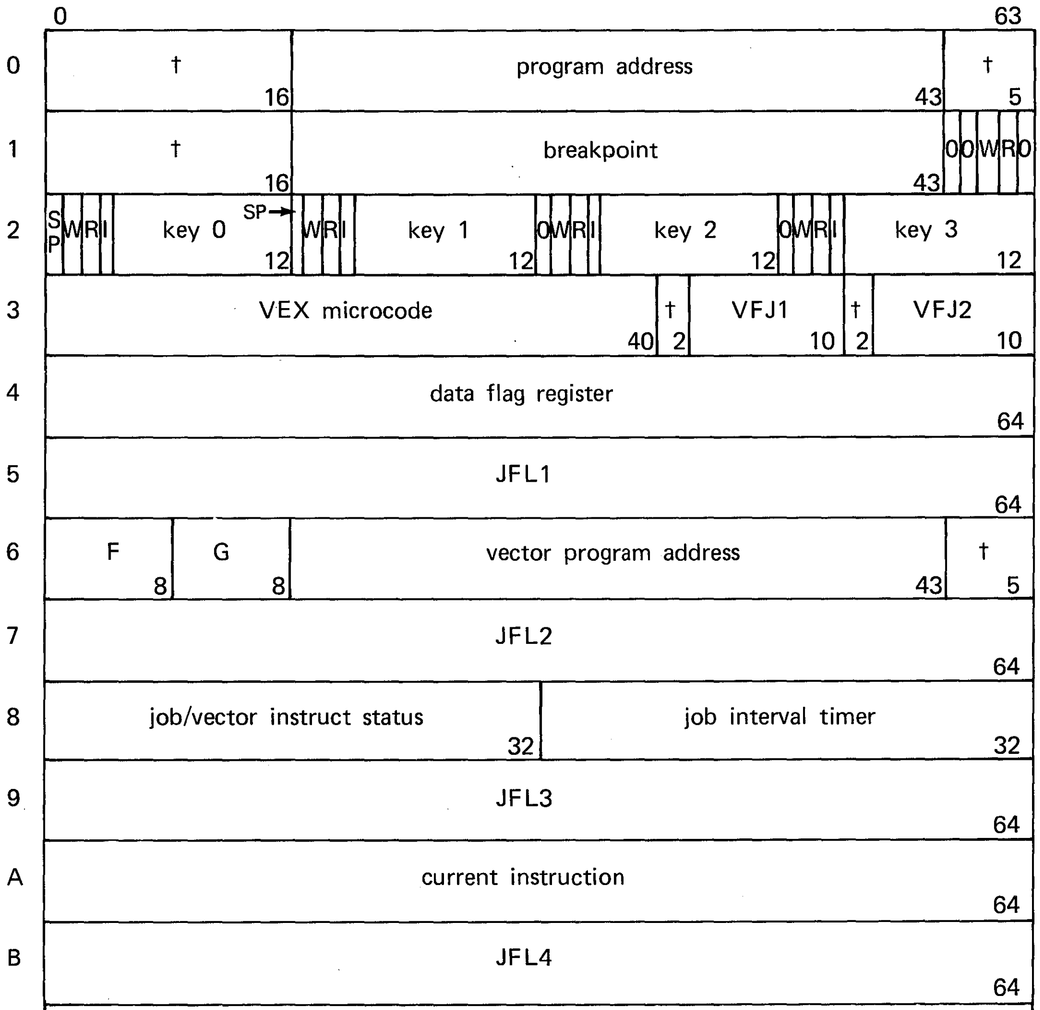
The epilogue of the called procedure should be as follows; however, instructions 7E1F0005 through 3B060000 are required only when using the DFBM:

Instruction	Description
7D1D1500	Using the length and address of the previous stack pointer, restore the register file from the callee´s current stack, starting with register #1A (the environment registers).
7E1F0005	Load word 0 from the data flag table to which register #1F points (ON_UNIT).
BE03180100000800	Enter the data flag register constant with the SFT, JIT, BKP, and enable bits set.
3B030004	Load and store the data flag branch register.
BE03180100000FE0	Enter the data flag register constant for an AND operation, which ensures that previously set free and monitor flags remain set.
2D030406	Perform a logical conjunction (AND) of the current data flag register with the constant for free flags and monitor flags.
2E060506	Perform a logical disjunction (Inclusive OR) of the current data flag register plus any free or monitor flags with word 0 from the data flag table containing the data flag settings for the caller.
3B060000	Load the data flag register with the setting.
3340001A	Jump to the return address specified in register #1A.

CYBER 205 INVISIBLE PACKAGE

The invisible package is a hardware convention that contains the address and control information required to begin a new job or to continue a job that was interrupted during execution. Each job is associated with an invisible package. When the CPU switches from monitor mode to job mode, the invisible package for the corresponding job is automatically loaded from main memory, beginning at the address assigned by the monitor. The invisible package data is loaded into the appropriate registers in the CPU. When the CPU switches from the job mode back to the monitor mode, as in the case of an interrupt, the contents of the corresponding registers are automatically stored in main memory as the invisible package for that job.

The contents of the invisible package are shown in figure E-1. For a description of fields not described in figure E-1, refer to the CYBER 200 Model 205 Computer System Hardware Reference Manual. Because the fixed portion of the absolute word address is divided within the hexadecimal character, bits 52 through 55 are shown as their binary equivalents.



†Undefined (must be set to 0).

Figure E-1. Invisible Package Contents (Sheet 1 of 4)

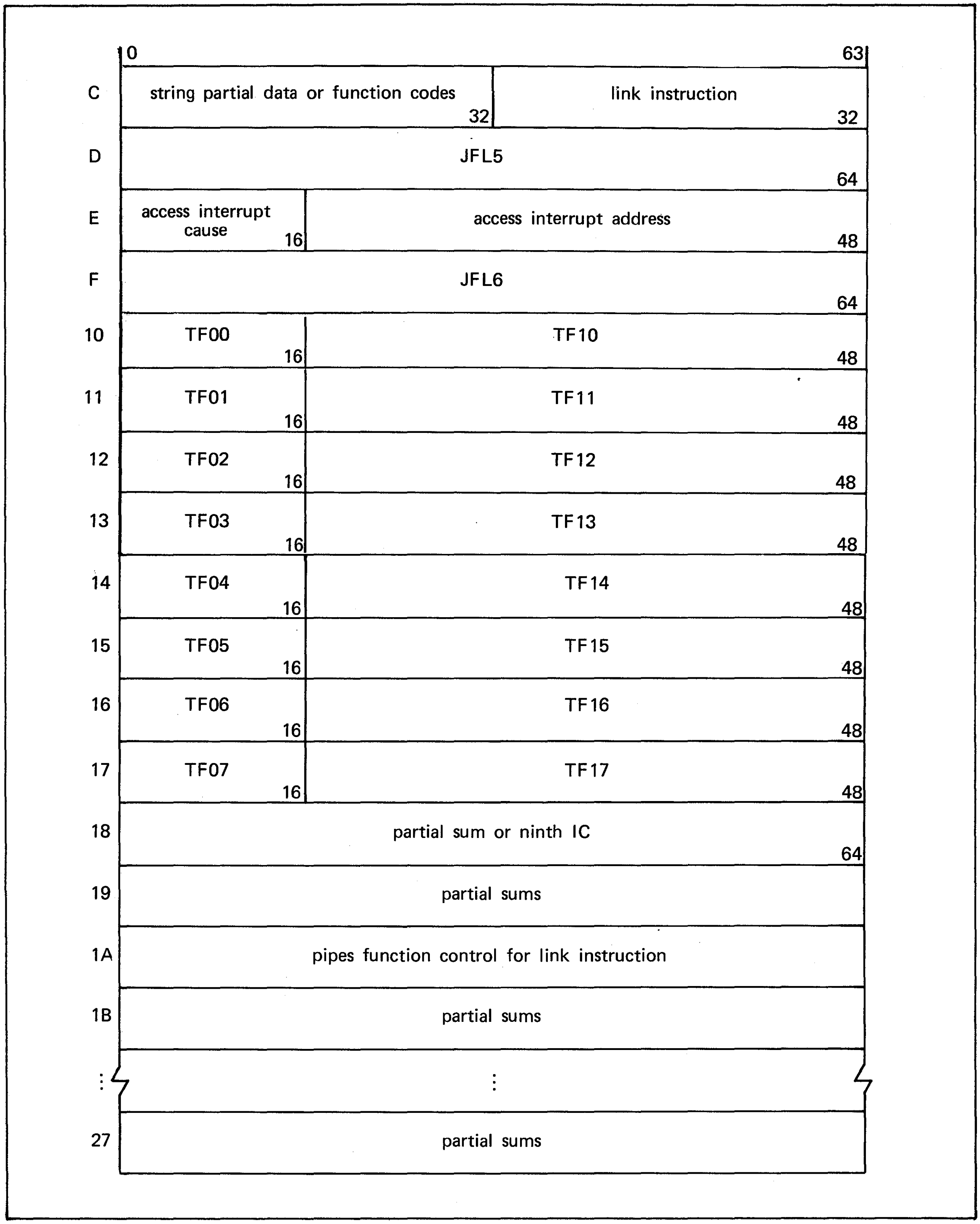


Figure E-1. Invisible Package Contents (Sheet 2 of 4)

Word		Description
1	Breakpoint usage bits:	
	0	Not used and must be 0.
	W	Check for breakpoint compare on write operands.
	R	Check for breakpoint compare on read operands.
2	Usage lockout bits for each key:	
	SP	Bits 0 and 16 together define a small page size for all small pages; bits 32 and 48 are not used and are set to 0:

Bit 0	Bit 16	Description
0	0	All small pages are 512 words.
0	1	Undefined.
1	0	All small pages are 2048 words.
1	1	All small pages are 8192 words.

	W	Lockout CPU write operations.
	R	Lockout CPU read operations.
	I	Lockout CPU instruction references.
3	Vector execution microcode conditions:	
	0-31	Not used and must be set to 0.
	32	Interrupt FF (signal to pipes).
	33	Link instruction in execution.
	34	Link instruction R bit 3.
	35	Link instruction R bit 4.
	36	CC instruction in execution.
	37	Not used and must be set to 0.
	38	Vector block scalar use of load/store registers.
	39	Flag 1.

Figure E-1. Invisible Package Contents (Sheet 3 of 4)

Word		Description
8		Job/vector instruct status bits:
	0	Vector restart.
	1	Not parallel operation.
	2-11	Undefined and must be set to 0.
	12	Stall bit (set for no data processed).
	13	D8 or D9 execution started.
	14	Undefined and must be set to 0.
	15	EBCDIC when set, ASCII when clear.
	16	SCR code bit 3 (exit at vector instruction termination).
	17	Select force of extension field length.
	18-19	Vector instruction register file update disable bits.
	20	D8 and D9 multiple match flag.
	21	String restart bit (old data flag).
	22-25	Undefined and must be set to 0.
	26	R-record FF.
	27	DA-DC toggle code bit 0.
	28	DA-DC toggle code bit 1.
	29	DA-DC toggle code bit 2.
	30,31	Undefined and must be set to 0.
C		Link instruction codes:
	0-15	Link (56) instruction F and R codes.
	16-31	Link F1 instruction F and G codes.
E		Access interrupt cause bits:
	0-11	Not used and must be set to 0.
	12	Associative work not in page table.
	13	Write operand violation attempted.
	14	Read operand violation attempted.
	15	Read instruction violation attempted.
18		Partial sum or ninth IC:†
	0-63	Partial sum for DX instruction or special broadcast quantity for link or CC instruction.
	0-15	Output item count for AX or C8 to CB instruction.
	16-63	C base address for AX instructions.

† These bits are undefined in all other applications.

Figure E-1. Invisible Package Contents (Sheet 4 of 4)

PROGRAM STATES F

The current disposition of a program is indicated by a number that is carried in the descriptor block for the program. This code can be gained by the privileged EXECUTE OPERATOR COMMAND message (f=#0021).

Codes having specific definitions are shown in table F-1. Codes not defined in the table have these general descriptions:

Code	Description
#1 - #9	Task is in the alternator.
#A - #F	Task is not in the alternator, but is partially in memory.
#10 - #1F	Task is not processing a message and is waiting.
#20 - #2F	Task is processing a message and is waiting.
#30 - #3F	System is performing functions for a program.
#40 - #4F	Miscellaneous.
#B9 - #BF	State is indicated by subtracting #80. Tasks in a terminal or nonterminal dump state have #80 added to their original state when they are being dumped to disk; for example, #3D + #80 = #BD.

Table F-1. Program State Codes (Sheet 1 of 2)

Code	Description
1	Task put in an alternator slot from the descriptor block load queue.
5	Task alternator unblocked after new slot time.
7	Outstanding explicit I/O requests and interrupts have completed.
11	Waiting for an alternator slot.
13	Waiting for entry in the explicit I/O buffer.
14	Waiting for the mainframe.
16	Waiting for nine-track tape assignment.
17	Waiting for system call completion.

Table F-1. Program State Codes (Sheet 2 of 2)

Code	Description
18	Waiting for I/O completion.
1E	Waiting for the controllee to get on disk.
1F	Waiting for the controllee to get on disk.
20	Waiting for a message from the controller.
21	Waiting for a message from the controllee.
22	Reserved for installation use.
23	Waiting for a message from the operator.
24	Waiting for an operator type in.
26	Waiting to send a message to the controllee.
27	Waiting to send a message to the controllee.
28	Reserved for installation use.
29	Waiting to send a message to the operator.
2A	Waiting to send a message to the teletypewriter.
30	Execute line in, descriptor block and keys assigned; message sent to load file management.
38	Waiting for termination of controllees at lower chain levels.
39	Terminate and kill all pages.
3A	Nonterminal dump. Suspend in state = #41 after completion.
3B	Program dump, accounting finished; cleanup done; code + 80_{16} accounting done; dump I/O.
3C	Dump finished, clean up to go.
3D	Terminal dump error.
3E	Terminal dump scheduled, no error.
3F	Nonterminal dump scheduled.
40	Suspend for a time period.
41	Suspended by the operator or the system.

TAPE FORMATS

G

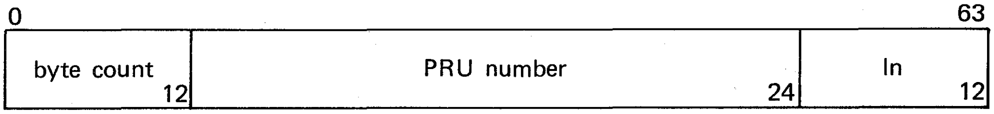
The online tape subsystem supports NOS and NOS/BE internal tape formats I and SI, respectively. It supports V, a variable PRU tape format, and NV, which is equivalent to lower CYBER S or L. In addition, a new tape format, large block format LB, is supported. For tape formats I, SI, and LB, a physical structure is superimposed over the user-declared SIL logical file structure (RT) by the operating system.

When the user issues a write, the user supplies a logical record unit array. Each entry in the array specifies the length of the logical tape record (LRU) and whether an end of file should also be written. After a read operation, the system returns information to the logical record unit array: number of bytes read, logical tape record status, end-of-group status, and end-of-information status. Observe that end of LRU, end of group, and end of information may have different meanings, based on the different tape formats. The characteristics of each of the tape formats follow.

I (INTERNAL) FORMAT

Figure G-1 shows the characteristics of the I tape format.

Characteristic	Description
Header	Labeled or unlabeled.
PRU size	Actual data block size that can range from 8 to 3840 bytes. All PRUs except those containing labels include a 48-bit PRU terminator formatted as follows:



Field	Description
byte count	Total number of bytes in the PRU, including the PRU terminator.
PRU number	Number of PRUs since the last HDR1 label.
ln	Level number: 0 End of LRU. F End of group.

User-specified maximum PRU size (MPRU) has no meaning.

Figure G-1. I Tape Format (Sheet 1 of 2)

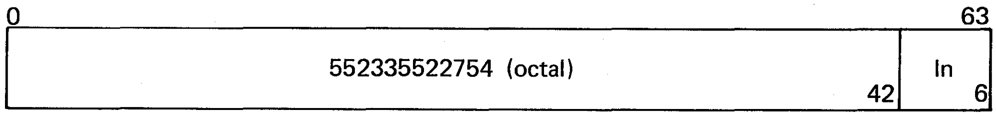
Characteristic	Description
Beginning of Information	For labeled tapes, a tape mark preceded by a beginning-of-volume or beginning-of-file label group is considered the beginning of information. For unlabeled tapes, load point is considered the beginning of information.
End of LRU	Any PRU with fewer than 512 CYBER 170 central memory words of data is considered an end of LRU. During a write operation, the level number field of the PRU terminator contains the level number obtained from the logical unit array; during read operations, the system returns an end-of-LRU status and the contents of the PRU terminator level number field. If the level number is 17 (octal), the system also returns an end-of-file status. Some PRUs may consist only of a PRU terminator.
End of group	Any PRU consisting of a PRU terminator only, with a level number of 17 (octal) is considered an end of group. The system ensures that an end of LRU always precedes an end of group by writing, if necessary, a PRU terminator with a level number of 0 prior to the end of group.
End of information	A tape mark followed by an EOF1 label is considered the end of information. This trailer sequence is generated by the system on labeled and unlabeled I, SI, and LB format tapes. The system issues a label content error if it encounters a tape mark without a valid label following it.
End of reel	If, during a write operation, the system senses the end of tape, it writes a trailer sequence following the PRU on which the EOT was sensed. This trailer sequence consists of a tape mark followed by an EOV1 label followed by three tape marks. The next PRU is written on the next reel. During a read operation, the EOT is observed and the system transfers to the user the PRU on which the EOT was sensed plus all following PRUs until a trailer sequence is recognized. Reading resumes on the next reel.
Noise	Not applicable.

Figure G-1. I Tape Format (Sheet 2 of 2)

SI (SYSTEM INTERNAL) FORMAT

Figure G-2 shows the characteristics of the SI tape format.

Characteristic	Description
Header	Labeled or unlabeled.
PRU size	The PRU size can range from 8 to 3840 bytes. Any PRU smaller than the maximum size except those containing labels contain a 48-bit special terminator. This terminator has the following format:



Field	Description
ln	Level number: 0-E End of LRU. F End of group.

Characteristic	Description
Beginning of information	For labeled tapes, a tape mark preceded by a beginning-of-volume or beginning-of-file label group is considered the beginning of information. For unlabeled tapes, load point is considered the beginning of information.
End of LRU	Any PRU containing fewer than 512 CYBER 170 central memory words represents an end of LRU. If an LRU consists of an exact multiple of 512 central memory words, the PRU that denotes the end of LRU consists solely of a special terminator. During write operations, the level number field of the special terminator contains the level number obtained from the logical record unit array; during read operations, the system returns an end-of-LRU status and the contents of the special terminator level number field. If the level number is 17 (octal), the system returns an end-of-file status.
End of group	Any PRU consisting of only a PRU terminator with a level number of 17 (octal) is considered an end of group. The system ensures that an end of LRU always precedes an end of group by writing, if necessary, a PRU terminator with a level number of 0 prior to the end of group.

Figure G-2. SI Tape Format (Sheet 1 of 2)

Characteristic	Description
End of information	A tape mark followed by an EOF1 label is considered the end of information. This trailer sequence is generated by the system on labeled and unlabeled I, SI, and LB format tapes. The system issues a label content error if it encounters a tape mark without a valid label following it.
End of reel	If, during a write operation, the system senses the end of tape, the system writes a trailer sequence following the PRU on which the EOT was sensed. This trailer sequence consists of a tape mark followed by an EOV1 label followed by three tape marks. The next PRU is written on the next reel. During a read operation, the EOT is observed and the system transfers to the user the PRU on which the EOT was sensed plus all following PRUs until a trailer sequence is recognized. Reading resumes on the next reel.
Noise	Not applicable.

Figure G-2. SI Tape Format (Sheet 2 of 2)

LB (LARGE BLOCK) FORMAT

Figure G-3 shows the characteristics of the LB tape format.

Characteristic	Description
Header	Labeled or unlabeled.
PRU size	Actual data PRU size that can range from 0 to 32768 bytes. All PRUs except those containing labels include a 48-bit PRU terminator formatted exactly like the I tape format terminator. User-specified maximum PRU size has no meaning.
Beginning of information	For labeled tapes, a tape mark preceded by a beginning-of-volume or beginning-of-file label group is considered the beginning of information. For unlabeled tapes, load point is considered the beginning of information.
End of LRU	Any PRU with fewer than 4096 central memory words of data is considered an end of LRU. During a write operation, the level number field of the PRU terminator contains the level number obtained from the logical record unit array; during read operation, the system returns end-of-LRU status and the contents of the PRU terminator level number field. If the level number is 17 (octal), the system also returns end-of-file status. Some PRUs may consist only of a PRU terminator.
End of group	Any PRU consisting of only a PRU terminator with a level number of 17 (octal) is considered an end of group. The system ensures that an end of LRU always precedes an end of group by writing, if necessary, a PRU terminator with a level number of 0 prior to the end of group.
End of information	A tape mark followed by an EOF1 label is considered the end of information. This trailer sequence is generated by the system on labeled and unlabeled I, SI, and LB format tapes. The system issues a label content error if it encounters a tape mark without a valid label following it.
End of reel	If, during a write operation, the system senses the end of tape, the system writes a trailer sequence following the PRU on which the EOT was sensed. This trailer sequence consists of a tape mark followed by an EOV1 label followed by three tape marks. The next PRU is written on the next reel. During a read operation, the EOT is observed and the system transfers to the user the PRU on which the EOT was sensed plus all following PRUs until a trailer sequence is recognized. Reading resumes on the next reel.
Noise	Not applicable.

Figure G-3. LB Tape Format

V AND NV (VARIABLE) FORMAT

Figure G-4 shows the characteristics of the V and NV tape formats.

Characteristic	Description
Header	Labeled or unlabeled.
PRU size	No explicit multiple of frames is required. The maximum PRU size may be specified in the MPRU field of the CHANGE FILE system message. If no PRU size is specified in the MPRU field, it is assumed to be 32 768 bytes. The maximum PRU size for V format is 48 pages.
Beginning of information	For labeled tapes, a tape mark preceded by a beginning-of-volume or beginning-of-file label group is considered the beginning of information. For unlabeled tapes, load point is considered the beginning of information.
End of LRU	On a READ or READSKIP request, each PRU is considered an end of LRU.
End of group	Tape mark. Observe that this is valid only for the NV format since it is non-ANSI.
End of information	If the tape is unlabeled, double tape marks located before the end-of-tape reflective marker denote end of information. If the tape is labeled, the end of information is a tape mark followed by an EOF1 label.
End of reel	If, during a write operation, the system senses the end of tape, the system writes a trailer sequence following the PRU on which the EOT was sensed. This trailer sequence consists of a tape mark followed by an EOV1 label for labeled tapes and tape marks for unlabeled tapes. The next PRU is written on the next reel. During a read operation, the EOT is observed and the system transfers to the user the PRU on which the EOT was sensed plus all following PRUs until a trailer sequence is recognized. Reading resumes on the next reel.
Noise	Any PRU containing fewer than the specified number of noise size frames is considered noise and is therefore ignored.

Figure G-4. V and NV Tape Formats

INDEX

COMMENT SHEET

MANUAL TITLE: CDC VSOS Version 2 Reference Manual, Volume 2 of 2

PUBLICATION NO.: 60459420 **REVISION:** H

NAME: ____________________

COMPANY: ____________________

STREET ADDRESS: ____________________

CITY: __________ **STATE:** __________ **ZIP CODE:** __________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND TAPE

CUT ALONG LINE

REV. 5/86 PRINTED IN U.S.A.

LD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA

Technology and Publications Division
ARH219
4201 North Lexington Avenue
Saint Paul, MN 55126-6198

D FOLD

CUT ALONG LINE

CORPORATE HEADQUARTERS P.O. BOX 0 MINNEAPOLIS, MINNESOTA 55440

CONTROL
DATA